Leakage resilience and black-box impossibility results in cryptography

by

Ali Juma

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Leakage resilience and black-box impossibility results in cryptography

Ali Juma

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2011

In this thesis, we present constructions of leakage-resilient cryptographic primitives, and we give black-box impossibility results for certain classes of constructions of pseudo-random number generators.

The traditional approach for preventing side-channel attacks has been primarily hardware-based. Recently, there has been significant progress in developing algorithmic approaches for preventing such attacks. These algorithmic approaches involve modeling side-channel attacks as *leakage* on the internal state of a device; constructions secure against such leakage are *leakage-resilient*.

We first consider the problem of storing a key and computing on it repeatedly in a leakage-resilient manner. For this purpose, we define a new primitive called a *key proxy*. Using a fully-homomorphic public-key encryption scheme, we construct a leakage-resilient key proxy. We work in the "only computation leaks" leakage model, tolerating a logarithmic number of bits of polynomial-time computable leakage per computation and an unbounded total amount of leakage.

We next consider the problem of verifying that a message sent over a public channel has not been modified, in a setting where the sender and the receiver have previously shared a key, and where the adversary controls the public channel and is simultaneously mounting side-channel attacks on both parties. Using only the assumption that pseudo-random generators exist, we construct a leakage-resilient shared-private-key authenticated session protocol. This construction tolerates a logarithmic number of bits of polynomial-time computable leakage per computation, and an unbounded total amount of leakage. This leakage occurs on the entire state, input, and randomness of the party performing the computation.

Finally, we consider the problem of constructing a large-stretch pseudo-random generator given a one-way permutation or given a smaller-stretch pseudo-random generator. The standard approach for doing this involves repeatedly composing the given object with itself. We provide evidence that this approach is necessary. Specifically, we consider three classes of constructions of pseudo-random generators from pseudo-random generators of smaller stretch or from one-way permutations, and for each class, we give a black-box impossibility result that demonstrates a contrast between the stretch that can be achieved by adaptive and non-adaptive black-box constructions.

Acknowledgements

First, I would like to thank my supervisor, Charlie Rackoff, with whom I spent countless hours discussing the results in this thesis and all the work leading up to them. Working with Charlie over the past eight years has been a truly outstanding experience. I have greatly benefited from Charlie's vast knowledge, his exceptional intuition, and his constant insistence on clarity in all forms of reasoning.

I would like to thank the members of my committee – Allan Borodin, Steve Cook, and Toni Pitassi – for all their advice and encouragement.

I would like to thank my co-authors (and fellow cryptography grad students) Josh Bronson, Periklis Papakonstantinou, and Yevgeniy Vahlis for their invaluable contributions to the results that appear in this thesis. I would particularly like to thank Yevgeniy for the many months we spent discussing and working on leakage-resilient cryptography.

I would like to thank my parents and my sister for their constant encouragement and support.

Finally, I would like to thank the Department of Computer Science and the Natural Sciences and Engineering Research Council of Canada for providing financial support.

Contents

Chapter 1

Introduction

Cryptography is the study of achieving clearly-defined security properties in a variety of settings. This includes giving constructions that achieve specific security properties under certain computational hardness assumptions, and proving that certain classes of constructions *cannot* achieve specific security properties. In this thesis, we present both kinds of results. We present constructions that achieve leakage resilience, and we give black-box impossibility results for certain classes of pseudo-random number generator constructions.

1.1 Leakage resilience

In a *side-channel attack*, an adversary obtains information about the internal state of a device by measuring such things as power consumption, computation time, and emitted sound and radiation, and then uses this information to break the security of a cryptographic primitive that is being computed by the device. Many such attacks have been developed (e.g. [QS01, BB03, Kuh03, Ber05, OST06]). The traditional approach for preventing such attacks has been hardware-focused. For instance, to prevent an attack based on emitted sound, one would place the device inside a sound-proof enclosure.

More recently, attention has turned to algorithmic approaches for preventing side-channel attacks. This involves modifying classical security models – where internal state information is perfectly hidden from the adversary – so that the adversary is allowed to obtain some internal state information, and then proving that security is achieved with respect to such models. The internal state information obtained by the adversary is referred to as *leakage* on the internal state, and constructions that are secure against adversaries obtaining leakage are known as *leakage-resilient* constructions.

When modeling the manner in which that adversary obtains leakage, the goal is to capture every "reasonable" physical measurement that a side-channel attacker might make. Several security models that allow the adversary to gain information through leakage have been developed. Of course, restrictions must be placed on this leakage, since if the entire internal state is leaked to the adversary then security is impossible. If the use of the device can be viewed as a sequence of discrete invocations, a natural restriction on leakage is to bound the *number of bits* leaked per invocation. But even this restriction is insufficient for devices whose computation is deterministic, since an adversary that gets b bits of leakage per invocation can learn mb bits about the state at invocation m, simply by leaking bits of the state at round m in each of the previous invocations, and using the fact that the state at invocation m can be computed from each of the previous states. To overcome this *future pre-computation attack*, even stronger restrictions on leakage have been considered. These restrictions include strongly bounding the *total amount* of leakage, or insisting that the leakage be computed by a strongly uninvertible function of the state, or requiring that the leakage be computed by shallow circuits. These approaches all allow the leakage to be a function of the entire state.

Another approach is motivated by the idea that *only computation leaks* [MR04]. That is, parts of the state only leak when they are involved in a computation. Dziembowski and Pietrzak [DP08] follow this approach by splitting the state into two halves, where computation alternates between the two halves and never takes place on both halves simultaneously. This thwarts the future pre-computation attack, and has the advantage of potentially being secure even when the total leakage is unbounded, as long as the leakage per round is bounded, and the leakage is computed by efficient (polynomial-size or polynomial-time) functions. We use this idea in our construction of leakage-resilient key proxies in Chapter 2.

The future pre-computation attack can also be thwarted using randomness. Specifically, if the state is updated each round in a manner that depends on an unpredictable string, the adversary will be unable to compute bits of a state when leaking on previous states. We use this idea in our construction of leakage-resilient authenticated sessions in Chapter 3. This idea is also used in the independent recent work of Brakerski *et al* [BKKV10] and Dodis *et al* [DHLAW10], and in the subsequent work of Malkin *et al* [MTVY11].

Leakage-resilient constructions of primitives including stream ciphers [DP08, Pie09], signature schemes [FKPR10, ADW09, KV09, BKKV10, DHLAW10, MTVY11], and public-key encryption [AGV09, NS09, DGK$^+$10, BKKV10] have been given. Some of these constructions [DP08, Pie09, FKPR10] use the "only computation leaks" model, while others bound the total amount of leakage [ADW09, KV09, AGV09, NS09] or require that the leakage functions be such that it is hard to find the secret key given all the leakage [DGK$^+$10]. Finally, very recent constructions [BKKV10, DHLAW10, MTVY11], which, as mentioned above, depend on randomness, allow unbounded total leakage without requiring the "only computation leaks" restriction.

1.1.1 Leakage-resilient key proxies

Storing a key and repeatedly computing on it is a common task in cryptography. In Chapter 2, we consider the problem of performing this task securely in a setting where the adversary is able to obtain leakage on internal state information. We define a new primitive called a *leakage-resilient key proxy*, which stores a key and allows arbitrary polynomial-time computation to be performed on this key while ensuring that the adversary gains no useful information from internal state leakage. We construct a leakage-resilient key proxy in the "only computation leaks" model. We allow leakage to be computed by adversarially-chosen polynomial-size circuits whose output length is restricted but where the total amount of leakage (as the adversary repeatedly obtains leakage from each computation) is unbounded.

The previous work most closely related to ours is that of Faust *et al* [FRR$^+$10], who show how to transform any stateful circuit so that it can be securely computed in a setting where the entire internal state is subject to leakage computed by adversarially-chosen AC^0 circuits whose output length is restricted (but, as in our setting, the total amount of leakage is unbounded).

Concurrent to our work, Goldwasser and Rothblum [GR10] address essentially the same problem. That is, their result can be viewed as a construction of a leakage-resilient key proxy in the "only computation leaks" model. They rely on a weaker assumption than we do (they use the Decisional Diffie Hellman assumption while we use fully-homomorphic public-key encryption) and tolerate more leakage per round than we do. On the other hand, they rely on the "only computation leaks" assumption more strongly than we do. Specifically, while our construction splits the state into two parts, with computation alternating between the two parts, they split the state into a number of parts that is linear in the size of the circuit being computed. Furthermore, while our construction requires only a singe *leak-free component* (a piece of hardware that is assumed not to leak at all), they require a number of leak-free components that is linear in the size of the circuit being computed.

Our definition

We begin by defining a *key proxy*, an object that stores a key and allows computation to be performed on this key.

Definition 1 (Key proxy) A *key proxy* is a pair of PPT algorithms $(\mathsf{KPInit}, \mathsf{KPEval})$. For fixed $c \in \mathbb{N}$ and for all $n \in \mathbb{N}$, $K \in \{0,1\}^{n^c}$, $\mathsf{KPInit}(1^n, K)$ outputs an initial state S. For every circuit $F : \{0,1\}^{|K|} \to \{0,1\}^n$, $\mathsf{KPEval}(1^n, F, S)$ updates state S and outputs $F(K)$.

We now discuss informally what it means for a key proxy to be *leakage-resilient*. Intuitively, we would like it to be the case that an adversary "learns nothing useful" from leakage, even when the adversary chooses key K himself and adaptively chooses polynomially-many functions F to be evaluated on K and leakage queries to be evaluated on KPEval's state and randomness. Of course, in most applications, the adversary will not choose K himself, but by giving him this power in the definition, we enforce the requirement that the adversary "learns nothing useful" from leakage *no matter how much a-priori information he has about* K. We formalize this intuition by requiring the existence of a simulator producing "simulated leakage" such that no adversary can distinguish actual leakage and simulated leakage. For each query to KPEval, the simulator is given the query F, the response $F(K)$, and the adversary's leakage queries.

Recall that we are using the "only computation leaks" leakage model. This means that the adversary's leakage queries are applied only to the portion of the state of KPEval that is actually in use. Following Dziembowski and Pietrzak [DP08], we model the state using two pieces of memory (called "memory A" and "memory B") that communicate via a public channel, where computation never simultaneously involves both pieces of memory.

Looking ahead to our construction, each call to KPEval will involve computation on memory A, then on memory B, and finally on memory A again. We allow the adversary to adaptively choose leakage queries for each of these three computations (that is, for each query F to KPEval, the adversary chooses a leakage query ℓ_1 for the initial computation on memory A, sees the result of this query along with any message sent over the public channel by memory A, chooses a leakage query ℓ_2 for the computation on memory B, and so on). Each leakage query must have output length $\log n$. We require the simulator to produce not only "simulated leakage" but also "simulated public channel communication". The simulator receives queries and produces simulated leakage and communication in the same order as an actual construction (that is, the simulator receives query F, response $F(K)$, and the first leakage query ℓ_1, outputs a simulated response to ℓ_1 along with simulated communication from memory A to memory B, then receives leakage query ℓ_2, and so on).

Applications Applications of leakage-resilient key proxies include leakage-resilient versions of signature schemes, public-key encryption schemes, and private-key encryption schemes. Note that definitions of security for public-key encryption and private-key encryption involve the computation of a challenge that is given to the adversary. When giving leakage-resilient versions of these definitions, we must disallow leakage on the computation generating the challenge in order to prevent the adversary from trivially succeeding; however, we do allow leakage on all other computation, both before and after the computation of the challenge.

Our construction

Our construction uses a fully-homomorphic public-key encryption scheme. Such schemes allow arbitrary computation to be performed on encrypted data, producing an encryption of the result.

Definition 2 (Fully-homomorphic public-key encryption scheme) *A fully-homomorphic public-key encryption scheme* is a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{EncEval})$ that satisfy the following conditions:

1. The triple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically-secure public-key encryption scheme.

2. The algorithm $\mathsf{EncEval}(pub, \boldsymbol{C}, F)$, where pub is a public key, $\boldsymbol{C} = (C_1, \ldots, C_n)$ is a vector of ciphertexts, and F is a circuit on n inputs, outputs a string C' which is a valid encryption under pub of
$$F(\mathsf{Dec}_{pri}(C_1), \ldots, \mathsf{Dec}_{pri}(C_n)).$$

Fully-homomorphic public-key encryption schemes have recently been constructed by Gentry [Gen09] and by van Dijk *et al* [vDGHV10].

We now informally describe our construction of a leakage-resilient key proxy. We use a fully-homomorphic public-key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{EncEval})$. On input K, our initialization procedure KPInit uses KeyGen to produce key pair (pri, pub), and uses Enc to produce an encryption C of K under pub. It outputs pri to memory A and outputs C to memory B. Given a circuit F as input, our evaluation procedure KPEval follows the approach shown in Figure 1.1.

Observe that our construction refreshes the contents of both pieces of memory on every call to KPEval. The purpose of such refreshing is to prevent the adversary from eventually leaking the entire contents of memory. However, it is not clear that this is sufficient to obtain leakage resilience. For example, it is possible that a ciphertext C output by $\mathsf{EncEval}$ will retain information about the corresponding inputs to $\mathsf{EncEval}$ (beyond whatever is implied by the plaintext corresponding to C), and reveal this information when it is decrypted. In particular, the decryption of C_{res} in memory A may reveal information about K (beyond whatever is implied by $F(K)$). To address this problem, the fully-homomorphic encryption schemes of Gentry and of van Dijk *et al* have *randomization* procedures that have the effect of removing extra information from ciphertexts. Roughly speaking, these procedures involve adding a random encryption of $\bar{0}$ to the given ciphertext. We were unable to prove that these randomization procedures have the desired effect when there is leakage on the randomness used to produce the encryption of $\bar{0}$. Consequently, we use a leak-free component to sample the two random encryptions of $\bar{0}$ needed by memory B to randomize C_{res} and C_{key}. We note that Faust *et al* [FRR$^+$10] and Goldwasser

Memory A		Memory B
Contents of memory: pri		Contents of memory: $C = \mathsf{Enc}_{pub}(K)$, Input: circuit F
Compute $(pri', pub') \leftarrow \mathsf{KeyGen}(1^n)$ Compute $C_{pri} \leftarrow \mathsf{Enc}_{pub'}(pri)$ Set memory to pri'		
	$\xrightarrow{pub', C_{pri}}$	
		Homomorphically compute using C, C_{pri}: $C_{\mathsf{res}} \leftarrow \mathsf{Enc}_{pub'}(F(K))$ $C_{\mathsf{key}} \leftarrow \mathsf{Enc}_{pub'}(K)$ Set memory to C_{key}
	$\xleftarrow{C_{\mathsf{res}}}$	
Compute $Y \leftarrow \mathsf{Dec}_{pri'}(C_{\mathsf{res}})$ Return Y		

Figure 1.1: Informal description of our key proxy, showing one call to KPEval

et al [GR10] also use leak-free components in their construction. While the components used by Faust *et al* sample from a simpler distribution than ours, they use linearly-many components (we use just one) and obtain security only with respect to leakage computed by AC^0 circuits (but they allow leakage on the entire state, not just on the portion involved in computation). The components used by Goldwasser *et al* sample from a distribution of complexity similar to ours (they also sample encryptions under a public key); as noted previously, Goldwasser *et al* use linearly-many components.

Proving our construction is secure We briefly describe the approach we use in our proof of security. We define a simulator that instantiates our construction using key $\bar{0}$, and uses this instantiation to respond to the adversary's leakage queries. We use a non-trivial hybrid argument to show that an adversary that distinguishes real leakage from simulated leakage (after polynomially-many calls to KPEval) yields an adversary that, roughly speaking, distinguishes real leakage from simulated leakage after only two calls to KPEval. Then we show how pairs of the new adversary's leakage queries can be combined into a single query (of twice the output length) using a guess-and-check approach: when the adversary would normally make the first leakage query it instead guesses an output, and then verifies this guess when it makes the second leakage query (and uses a coin flip as output when it turns out that its guess was wrong). Repeatedly combining pairs of leakage queries in this fashion yields an adversary that just makes a single leakage query and (essentially) distinguishes an encryption of $\bar{0}$ (used by the simulator) from an encryption of some key K (used by the real construction). To finish the proof, we use an observation of Akavia *et al* [AGV09] that every semantically-secure public-key encryption scheme remains secure when the adversary gets $O(\log n)$ bits of leakage on KeyGen.

Chapter 2 is joint work with Yevgeniy Vahlis [JV10].

1.1.2 Leakage-resilient authentication

In Chapter 3, we consider the problem of *leakage-resilient shared-private-key authenticated sessions*. Two parties, A and B, have shared an n-bit key, and A now wishes to send message pieces to B over public channel in a manner that allows B to verify that the message pieces he is receiving are indeed those sent by A, in the correct order. The adversary controls the public channel and adaptively obtains leakage from both parties. Assuming only the existence of pseudo-random generators, we construct a shared-private-key authenticated session protocol that is secure even when the adversary obtains $O(\log n)$ bits of leakage per computation. The leakage obtained by the adversary for each computation is computed based on the entire state, inputs, and randomness of the party performing the computation; that is, we do *not* use the "only computation leaks" assumption, nor do we use any leakage-free hardware. Our protocol also has the feature that all randomness used by each party is made public, and, in fact, this randomness can be chosen according to a high min-entropy distribution (that is, a distribution with at least $\log^2 n$ bits of min-entropy).

Our construction

Our construction uses a modified version of Pietrzak's leakage-resilient *stream cipher* [Pie09]. A stream cipher produces pseudo-random sequence of strings. Leakage-resilient stream ciphers have been constructed in the "only computation leaks" model by Dziembowski and Pietrzak [DP08] and Pietrzak [Pie09]. The definition of security for leakage-resilient stream ciphers involves only a single party computing the stream cipher. An important challenge that needs to be overcome when using leakage-resilient stream ciphers for leakage-resilient authentication is handling the issue of two parties each running a stream cipher using the same seed, where both parties are subject to leakage.

We modify the stream cipher of Pietrzak [Pie09] so that it uses public source of high min-entropy strings but allows the entire state to leak. Like Pietrzak's stream cipher, our stream cipher can be built from a pseudo-random function generator $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$.

Sender A		Receiver B
Contents of memory: K_{i-1} Input: message piece m_i		Contents of memory: K_{i-1}
		Choose high min-entropy r_i
	$\xleftarrow{r_i}$	
Compute $K_i \vert\vert X_i \leftarrow F_{K_{i-1}}(r_i)$ Set memory to K_i Compute $\alpha_i \leftarrow F'_{X_i}(m_i)$		
	$\xrightarrow{m_i, \alpha_i}$	

		Compute $K_i \vert\vert X_i \leftarrow F_{K_{i-1}}(r_i)$
		Set memory to K_i
		If $\alpha_i = F'_{X_i}(m_i)$:
		    Output m_i
		Else:
		    Output `Fail`

Figure 1.2: Informal overview of our authenticated session protocol, showing a single round i

In our construction, the two parties each run our stream cipher, using their shared key as the stream cipher's seed. The receiver chooses the required high-min entropy strings and sends them to the sender over a public channel. Each string that is output by the stream cipher is used by the sender to sign a message piece (specifically, the string output by the stream cipher is used as the seed of a pseudo-random function generator $F' : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ which is evaluated on the message piece to produce a signature), and used by the receiver to verify this signature. An informal overview of our construction is given in Figure 1.2.

Definition of security The adversary controls the public channel, and also chooses the message pieces m_i that are provided as input to the sender A. Note that since the adversary controls the public channel, he effectively has the ability to schedule the computation performed by A and B. For example, he can choose to run A for several rounds before B has run at all. Whenever a party performs computation, the adversary obtains $O(\log n)$ bits of leakage.

The receiver B is allowed to output `Fail` and halt (the idea is that he does so when he detects tampering by the adversary); when this happens, the security experiment ends immediately. The adversary's goal is to induce the receiver B to output, for some i, a message piece $m'_i \neq m_i$. The construction is secure if every polynomial-size adversary succeeds with at most negligible probability.

Proving our construction is secure Intuitively, the string r_i sent by B to A forces the adversary to "properly" interleave the computation of A and B. That is, we show that an adversary that breaks our construction yields an adversary that, roughly speaking, breaks our construction in manner that involves behaving passively (simply providing the output of B to A and vice-versa) until the round j in which B is induced to output an $m'_j \neq m_j$. We then argue that the leakage on A and B in each round can be combined into a single leakage query (whose output length is equal to the sum of output lengths of the all the leakage queries on A and B in a single round) for each round. The main idea is that since the adversary is behaving passively, the states of A and B are identical each round; hence it suffices for the adversary to leak on just one of these parties, since the leakage function can simulate the adversary receiving leakage from one of the parties, choosing the next leakage function, and obtaining leakage from the other party, and the leakage function can concatenate the responses to these queries in order to form its own output. This yields an adversary that is essentially interacting with just a single instance of the stream cipher underlying our construction. We show that this adversary can be used to break the stream cipher.

Separately, we show that our modified version of Pietrzak's leakage-resilient stream cipher is secure. We use ideas that are similar to those of Pietrzak, but there are significant differences in the details.

Chapter 3 is joint work with Yevgeniy Vahlis [JV11].

1.2 Black-box impossibility results for pseudo-random number generator constructions

Positive results in cryptography – results that assert that a particular definition of security can be satisfied – are typically conditional: "If an object O_1 with security property A exists, then an object O_2 with security property B exists." Most results of this form involve giving a *black-box construction* of an object with security property B from an object with security property A. Informally, this means using an *oracle* O_1 with security property A in order to construct an object with security property B, where the construction's proof of security shows how to use an *oracle* breaking the B-ness of the construction in order to break the A-ness of the construction's oracle O_1. Showing that an object satisfying property B *cannot* be obtained in a black-box way from an object satisfying property A can be taken as evidence that obtaining property B from property A is difficult and requires non-standard techniques. Impagliazzo and

Rudich [IR89] gave the first such black-box impossibility results.

The black-box setting has also been used to investigate *how efficiently* a particular property B can be obtained from a particular property A. For example, Gennaro *et al* [GGKT05] consider black-box constructions of pseudo-random number generators from one-way permutations, and give a bound on the number of bits of stretch per oracle query that such constructions can achieve.

Non-adaptive constructions of pseudo-random number generators The standard approach for constructing a large-stretch pseudo-random generator given a one-way permutation or given a smaller-stretch pseudo-random generator involves repeatedly composing the given primitive with itself. In Chapter 4, we consider whether this approach is necessary, that is, whether there are constructions that do not involve composition. More formally, we consider black-box constructions of pseudo-random generators from pseudo-random generators of smaller stretch or from one-way permutations, where the constructions make only non-adaptive queries to the given object. Viola [Vio05] and Lu [Lu06] consider a similar problem – constructing a pseudo-random generator making only non-adaptive queries to a given one-way *function* – and give black-box impossibility results for certain classes of such constructions. Miles and Viola [MV11] consider constructions of linear-stretch pseudo-random generators making only non-adaptive queries to a pseudo-random generator of 1-bit stretch, and give a black impossibility result for such constructions whose output must consist only of query response bits (that is, constructions where no computation can be performed on query responses). The classes of constructions considered by Viola, Lu, and Miles *et al* are, in general, incomparable to the classes of constructions we consider; their constructions are more general in terms of the number of oracle queries allowed and the manner in which oracle queries are chosen, but more restrictive in the computational power allowed after responses to the oracle queries are received.

Our results We consider three classes of constructions of pseudo-random number generators, and for each class, we give a black-box impossibility result that demonstrates a contrast between the stretch that can be achieved by adaptive and non-adaptive black-box constructions. Our classes are defined by specifying restrictions on the manner in which oracle queries are chosen or used; beyond these restrictions, we do not place any computational bounds on the constructions we consider.

- *Class 1: Constructions with short seeds*

  We begin by considering constructions whose seed length is not too much longer than the length of each oracle query. Suppose we have a pseudo-random generator $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$ and we wish to obtain a pseudo-random generator with larger stretch, say stretch $2 \cdot s(n)$. We can easily define such a generator $G^f : \{0,1\}^n \to \{0,1\}^{n+2\cdot s(n)}$ as follows: on input $x \in \{0,1\}^n$, G^f computes $y_0||y_1 = f(x)$ (where $|y_0| = s(n)$ and $|y_1| = n$), and outputs $y_0||f(y_1)$. Observe that this construction makes two *adaptive* oracle queries. This idea can easily be extended to obtain, for every $k \in \mathbb{N}$, a black-box construction making k adaptive oracle queries and achieving stretch $k \cdot s(n)$.

We show that black-box constructions making constantly-many *non-adaptive* queries, each of the same length as their seed length n, cannot even achieve stretch $s(n) + 1$, that is, such constructions cannot even achieve a one-bit increase in stretch. We show that this also holds for constructions whose seed length is at most $O(\log n)$ bits longer than the length n of each oracle query.

- *Class 2: Constructions with long seeds*

  We next consider constructions with arbitrarily long seeds, but where oracle queries are collectively chosen in a manner that depends only on a portion of the seed whose length is at most $O(\log n)$ bits longer than the length n of each query. While this setting may seem unnatural at first, it *is* possible in this setting to obtain a construction that makes constantly-many non-adaptive oracle queries to a pseudo-random generator and achieves more stretch than its oracle; indeed, even a single query suffices. For example, if $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$ is pseudo-random, then by the Goldreich-Levin theorem [GL89] we have that for all functions $m(n) \in O(\log n)$, the number generator $G^f : \{0,1\}^{n\cdot m(n)+n} \to \{0,1\}^{n\cdot m(n)+n+s(n)+m(n)}$ defined for all $r_1, r_2, \ldots, r_{m(n)}, x \in \{0,1\}^n$ as

$$G^f\left(r_1||r_2||\ldots||r_{m(n)}||x\right) = r_1||r_2||\ldots||r_{m(n)}||f(x)||\langle r_1, x\rangle||\langle r_2, x\rangle||\ldots||\langle r_{m(n)}, x\rangle$$

  is pseudo-random; the stretch of G^f is $m(n)$ bits greater than the stretch of f. Also observe that the query made by $G^{(\cdot)}$ depends only on a portion of the seed of $G^{(\cdot)}$ whose length is the same as the length of the query. Using this Goldreich-Levin-based approach, it is easy to see that *adaptive* black-box constructions whose input length is much longer than the length n of each oracle query can obtain stretch $k \cdot s(n) + O(\log n)$ by making k queries to an oracle of stretch $s(n)$, even when the portion of the seed that is used to choose oracle queries has length n.

  We show that black-box constructions $G^{(\cdot)}$ making constantly-many *non-adaptive* queries of length n to a pseudo-random generator $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$, such that only the rightmost $n + O(\log n)$ bits of the seed of $G^{(\cdot)}$ are used to choose oracle queries, cannot achieve stretch $s(n) + \omega(\log n)$. That is, such constructions making *constantly*-many non-adaptive queries cannot achieve greater stretch than the stretch provided by

Goldreich-Levin with just a *single* query. This holds no matter how long a seed is used by the construction $G^{(\cdot)}$.

- *Class 3: Goldreich-Levin-like constructions*

  Finally, we consider a class of constructions motivated by the streaming computation of pseudo-random generators. Specifically, we consider a class of constructions where the seed has a public portion that is always included in the output, the choice of each oracle query does not depend on the public portion of the seed, and the computation of each individual output bit depends only on the seed and on the response to a *single* oracle query. We refer to such constructions making non-adaptive oracle queries as *bitwise-nonadaptive* constructions. It is not hard to see that such constructions making polynomially-many *adaptive* queries to a one-way permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ can achieve arbitrary polynomial stretch; the idea is to repeatedly compose π with itself, outputting a hardcore bit of π on each composition. For example, using the Goldreich-Levin hardcore bit [GL89], a standard way of constructing a pseudo-random generator G^π of polynomial stretch $p(n)$ is the following: On input $r, x \in \{0,1\}^n$,

  $$G^\pi(r||x) = r||\langle r, x\rangle||\langle r, \pi(x)\rangle||\langle r, \pi^2(x)\rangle|| \ldots ||\langle r, \pi^{p(n)+n}(x)\rangle$$

  where $\pi^i := \underbrace{\pi \circ \pi \circ \ldots \circ \pi}_{i \text{ times}}$. Observe that the leftmost n bits of the seed of G are public in the sense that they are included in the output. Also observe that each of the remaining output bits of G is computed using only a single output of π along with the input bits of G. Finally, observe that the queries made to π do not depend on the public input bits of G, and the number of non-public input bits is no greater than the length n of each oracle query. Is the *adaptive* use of π in a construction of this form necessary? This question is particularly interesting if we wish to compute G in a streaming setting where we have small workspace and produce the output of G bit-by-bit. In such a setting, it seems difficult to use π in an adaptive manner, since we lack sufficient space to store query responses.

  We show that black-box bitwise-nonadaptive constructions $G^{(\cdot)}$ making queries of length n to a one-way permutation, such that the non-public portion of the seed of $G^{(\cdot)}$ is of length at most $n + O(\log n)$, cannot achieve linear stretch. This holds no matter the length of the public portion of the seed of $G^{(\cdot)}$.

Chapter 4 is joint work with Josh Bronson and Periklis Papakonstantinou [BJP11]. More specifically, the impossibility result for Class 1 builds on the Master's thesis of Bronson [Bro08], who gives a partial result for the case of constructions that make only a single oracle query, where this query must be the same as the seed. The impossibility results for Class 2 and Class 3 are joint work with Papakonstantinou.

Chapter 2

Leakage-resilient key proxies

Leakage-resilient cryptographic constructions – constructions that remain secure even when internal state information leaks to the adversary – have received much recent interest. Traditionally, security models have treated such internal state information as perfectly hidden from the adversary. However, the development of various side-channel attacks has made it clear that this traditional view is inconsistent with physical reality. In a side-channel attack, an adversary obtains information about the internal state of a device by measuring such things as power consumption, computation time, and emitted radiation.

Cryptographic primitives with long term keys, such as encryption and signature schemes, are often targeted by such attacks. An adversary observing information leakage from computation on the key can potentially accumulate enough data over time to compromise the security of the scheme. Consequently, storing keys and computing on them in adversarial environments has been an important goal both in theory and practice. Indeed, many operating systems provide cryptographic facilities that allow programs to access keys only through designated functions, such as signing and encrypting. Smart cards provide a similar interface in hardware. In both cases, the goal is to limit any adversary to interacting with the scheme through a specified interface. Nevertheless, information leakage through physical side-channels is often sufficient to overcome such barriers and break the scheme.

In this chapter, we propose an approach for protecting cryptographic keys and computing on them repeatedly in a manner that preserves the secrecy of the key even when information about the state of the device continuously leaks to the adversary. Towards this goal, we define a new primitive called a *key proxy*, which encapsulates a key K and provides a structured way of evaluating arbitrary functions on K. This allows, for example, the conversion of any pseudorandom function, signature scheme, or public-key encryption scheme into a leakage-resilient variant of itself. Our construction withstands a bounded amount of leakage per invocation (where an invocation occurs each time a function is evaluated on K), but the total amount of leakage is unbounded. Previously, only stream ciphers, signature schemes, and identification scheme have been made resilient to an unbounded total amount of leakage.

For our construction, we make use of fully homomorphic encryption [Gen09, vDGHV10], and an additional "leak-free" component. This component samples from a globally fixed distribution that does not depend on K.

Leakage-resilient cryptography. The problem of executing code in an adversarial environment has always been on the minds of cryptographers. Still, most cryptographic schemes are designed assuming that the hardware on which they will be implemented is a black-box device, and information is accessible to the adversary only through specified communication channels. Goldreich and Ostrovsky [GO96] consider the problem of protecting software from malicious users, and define the concept of an oblivious RAM – a CPU that is capable of evaluating encrypted programs using a constant amount of leak-free memory and an unbounded amount of memory that is fully visible to the adversary. The oblivious RAM is initialized with a secret key, which is used to decrypt encrypted instructions, execute them, and re-encrypt the output. The encrypted state of the program is stored in the clear. Oblivious RAMs provide the strong security guarantee that even if an adversary can keep track of the memory locations accessed by the computation, he is still unable to gain any additional information about the program over what would normally be revealed through black box access.

Since the work of Goldreich and Ostrovsky, the focus in leakage-resilient cryptography has been steadily shifting towards allowing the adversary ever-growing freedom in observing the *computation* of cryptographic primitives. Ishai, Sahai, and Wagner [ISW03] introduce "private circuits" – a generic compiler that transforms any circuit into one that is resilient to probing attacks. In a probing attack, the adversary selects a subset (of some fixed size) of the wires of the circuit and obtains the values of these wires. Goldwasser, Kalai, and Rothblum [GKR08] define one-time programs – programs that come with small secure hardware tokens, and can be executed a bounded number of times without revealing anything but the output, even if the adversary observes the entire computation. The secure tokens are the hardware equivalent of oblivious transfer – each token stores two keys and reveals one of them upon request, while the second key is erased.

Micali and Reyzin [MR04] outline a framework for defining and analyzing cryptographic security against adversaries that perform side channel attacks. They introduce an axiom: only computation leaks information. That is, at any point during the execution of an algorithm, only the part of memory that is actively computed on may leak information. This allows for convenient modeling of leakage: an algorithm is described as a sequence of procedures and the set of variables that is accessed by the procedure. The adversary may then obtain leakage from the contents of each set of variables as they are accessed during the execution of the algorithm. The only-computation-leaks model (OCL) has since been used to obtain stream ciphers [DP08, Pie09] and signature schemes [FKPR10] that remain secure even if the adversary obtains leakage from the active state each time the primitive is used, and the total amount of leakage is unbounded. We refer to such leakage as "continuous leakage" for the rest of the chapter.

Faust *et al* [FRR$^+$10] propose an alternative restriction on side-channel adversaries: restricting the computational power of the leakage function but allowing leakage on the entire state. Faust *et al* describe a circuit transformation that protects any circuit against leakage functions that can be described as AC^0 circuits[1]. The transformed circuit can leak information from the entire set of wires at each invocation, and makes use of a polynomial number of leak-free components that generate samples from a fixed distribution that does not depend on the computation of the circuit. We make use of a similar leak-free component, although the distribution generated by our component is significantly more complex than the one in [FRR$^+$10] due to the fact that we must defend against leakage functions that are not restricted to circuits of small depth.

Very recently (subsequently to our work), specific leakage-resilient cryptographic primitives have been constructed under even more general continuous leakage models. Dodis, Haralambiev, Lopez-Alt, and Wichs [DHLAW10] have constructed several primitives, including signature schemes and authenticated key agreement protocols, that remain secure even if the entire state (and not just the active part) leaks information continuously. The public key of the scheme remains fixed throughout the lifetime of the system. Brakerski, Kalai, Katz, and Vaikuntanathan [BKKV10] construct a public-key encryption scheme that allows continuous leakage on the entire state, and does not require a leak-free key update procedure. Brakerski *et al* also construct signature schemes and identity based encryption under slightly different leakage models. Malkin, Teranishi, Vahlis, and Yung [MTVY11] construct a signature scheme in the standard model that tolerates continuous leakage on the entire state as well as on all computation (that is, signing and key updates). As in our work, the constructions of Dodis *et al* , Brakerski *et al* , and Malkin *et al* provide protection against leakage that can be described by arbitrary polynomial-time computable functions with sufficiently short output.

In addition to the recent work on cryptographic constructions that are resilient to continuous leakage, there has been significant progress [AGV09, ADW09, NS09, KV09] on obtaining resilience to "memory attacks" – side channel attacks where the adversary obtains a bounded amount of information about the memory contents of the device throughout its lifetime. Per-

[1] AC^0 circuits have constant depth and unbounded fan-in.

haps due to the bounded nature of this type of leakage, constructions secure against memory attacks tend to be quite efficient and do not require the algorithm to maintain a state.

Concurrent work of Goldwasser and Rothblum. Concurrent to our work, Goldwasser and Rothblum [GR10] address the same problem that we do in this chapter. Their result can be viewed as a construction of a leakage-resilient key proxy in the "only computation leaks" model. Their construction relies on a linear number of leak-free components, while ours relies on a single component. Also, they use the "only computation leaks" assumption more strongly than we do, splitting the state of their construction into a number of separately-leaking pieces that is linear in the size of the circuit being evaluated (while we use only two such pieces). On the other hand, they rely on the standard Decisional Diffie Hellman assumption, whereas we rely on fully homomorphic encryption. They also tolerate more total leakage per round than we do.

On testable leak-free components. When constructing leakage-resilient cryptographic primitives, one has to take care in the nature and amount of components that are assumed not to leak any information. It is preferable, but may not always be possible, to avoid such components altogether. For example, one can protect any functionality against leakage given an arbitrary number of leak-free gates that can decrypt a ciphertext, perform a logical operation on the plaintext, and re-encrypt the result. Such a component can be used to evaluate the circuit F on K gate by gate, keeping all intermediate values encrypted, and thereby rendering leakage useless. However, building such leak-free components may be as difficult as constructing a leak-free computer and forgetting all about side-channels. Consequently, the focus of research in this area has always been to reduce the power and amount of computation that is assumed to be a-priori insulated from side-channel attacks.

Our construction uses a leak-free component that produces random encryptions of some fixed message (in our case – $\bar{0}$) under a given public key in the fully homomorphic encryption scheme. More specifically, the leak-free component we use is a randomized component that, given *pub*, produces two random encryptions of $\bar{0}$. Consequently, the computation performed by this component does not depend on any user or adversarially supplied inputs, and in particular does not depend on the key K or the function F that is evaluated on K. We call such a component *testable* because it can be accurately simulated in a controlled environment – all one has to do is feed the component random bits and randomly generated public keys and observe its behavior. More generally, we say that a component is testable if its inputs come from a globally fixed distribution that is independent from other inputs to the system.

We propose testability as a rule of thumb for secure hardware components in leakage resilient cryptography. All hardware components leak at least *some* information such as timing (every computation takes time) and power consumption. Therefore, the best we can hope for is that the information leaked by the components that we assume to be leak-free is useless to the adversary. Testability gives us the ability to observe the leakage from the secure component – as it will happen during actual usage – and estimate whether the component is safe to use. We note that the components used by [FRR$^+$10] and [GR10] are testable.

In contrast to [FRR$^+$10] and [GR10], where the number of leak-free components needed is linear in the size of the circuit that is evaluated on K, we use only one leak-free component.

Our contributions. We study the problem of computing on a cryptographic key in an environment that leaks information each time a computation is performed. We show that in the OCL model with a single leak-free randomized component, a cryptographic key can be protected in a manner that allows repeated computation on it while making sure that the adversary gains no information from side-channel information leakage.

More precisely, we propose a tool which we call a *key proxy* – a stateful cryptographic primitive that is initialized once with a key K, and then given any circuit F computes $F(K)$. Any leakage obtained by an adversary from the computation of the key proxy can be computed given just F and $F(K)$. Using any *fully homomorphic encryption* (FHE) scheme we construct a key proxy with the following properties:

Resilience to adaptive polynomial-time leakage. During each invocation of the key proxy, we allow the adversary to adaptively select leakage functions that are modeled as arbitrary circuits with a sufficiently short output. The exact amount of round leakage that our construction can withstand depends on the level of security of the underlying FHE scheme. Assuming the most basic security for the FHE scheme (i.e. against polynomial-time adversaries) permits security against $O(\log n)$ bits of leakage each time a function is evaluated on K. More generally, given a $2^{l(n)}$-secure FHE scheme, our construction can withstand roughly $l(n)$ bits of leakage per invocation.

Independent complexity. The starting point of leakage-resilient cryptography is that *computation leaks information*. It does not require a large leap of faith to suspect that *more* computation leaks more information. In fact, to the best of our knowledge, this is indeed the case for many side-channel attacks in practice. The amount of computation performed by our key proxy construction does not depend on the amount of leakage that the adversary obtains per invocation. Instead, to get resilience to larger amounts of leakage, a stronger assumption about the security of the underlying fully homomorphic encryption is used. This allows us to avoid a circular dependency where, in order to obtain resilience to larger amounts of leakage one must build a more complex device, which in turn leaks more information.

One-time programs with efficient refresh. The one-time programs of [GKR08] can be implemented without leak-free one-time memory tokens by storing the contents of the tokens in memory, and then accessing only the needed values during computation. The one-time programs can then be refreshed occasionally in a secure environment to allow continuous use. Currently, the refresh procedure performs as much computation as the evaluation of the program that it protects. If one is willing to trade resilience against complete exposure of the active memory (achieved by [GKR08]) for resilience against length-bounded leakage then by pre-computing the outputs of the leak-free tokens in our construction and storing them in memory, we obtain one-time programs with an update procedure of fixed complexity that does not depend on the protected program.

Our approach. The underlying building block for our construction is fully homomorphic encryption. An FHE scheme is a public-key encryption scheme that allows computation on encrypted data. That is, given a ciphertext with corresponding plaintext M, the public key, and a circuit F, there is an efficient algorithm that computes an encryption of $F(M)$.

For our construction, we partition the state of the key proxy into two parts, A and B (or, equivalently, two devices). Given a key K, the key proxy is initialized as follows. An FHE key pair (pri, pub) is generated and is stored in memory A. Then, a random encryption C of K under pub is computed and is stored in memory B. To evaluate a function F (described as a circuit) on K, the following actions are performed. First, a new pair of keys (pri', pub') is generated and stored in memory A, and an encryption $C_{pri} = \mathsf{Enc}_{pub'}(pri)$ of the old private key is written to a public channel. Then, computing on memory B and the public channel, the following two ciphertexts are generated homomorphically from C and C_{pri}: an encryption C_{res} of $F(K)$ and a fresh encryption C_{key} of K. Note that both C_{res} and C_{key} are encryptions under the new public key pub'. The ciphertext C_{res} is then sent back to memory A where it is decrypted, and $F(K)$ is returned as the output of the program. This basic approach is described in Figure 2.1.

It is clear that without leakage, the above construction is secure. Of course, the main difficulty is showing that leakage does not provide the adversary with any useful information. Below we provide an informal description of two main technical issues that arise.

Leakage on private keys and ciphertexts. It is easy to see that without refreshing the encryption C of K, a leakage adversary will eventually learn all of K by gradually leaking all of C and pri and then simply decrypting. Therefore, it is clear that an update procedure is necessary. The algorithm described in Figure 2.1 performs such an update: After each invocation, memory A contains a freshly generated private key and memory B contains an

Memory A		Memory B
Contents of memory: pri_i		Contents of memory: $C = \mathsf{Enc}_{pub_i}(K)$, Input: circuit F
$(pri_{i+1}, pub_{i+1}) = \mathsf{KeyGen}(1^n)$		
Encrypt $C_{pri} = \mathsf{Enc}_{pub_{i+1}}(pri_i)$		
Set memory to pri_{i+1}		
	$\xrightarrow{pub_{i+1}, C_{pri}}$	
		Homomorphically compute using C, C_{pri}:
		$C_{\mathsf{res}} = \mathsf{Enc}_{pub_{i+1}}(F(K))$
		and $C_{\mathsf{key}} = \mathsf{Enc}_{pub_{i+1}}(K)$
		Set memory to C_{key}
	$\xleftarrow{C_{\mathsf{res}}}$	
Compute $Y = \mathsf{Dec}_{pri_{i+1}}(C_{\mathsf{res}})$		
Return Y		

Figure 2.1: Informal description of the construction

encryption of K under the corresponding public key. However, we cannot directly claim that this refreshing procedure provides the necessary level of security. The main difficulty stems from the fact that the adversary obtains leakage on the private key in memory A both before and after he obtains leakage on the encryption C of K under the corresponding public key. In particular, if the adversary could obtain the entire ciphertext C, he would be able to hardcode it into the second leakage function that is applied to the private key. The leakage function would then decrypt C and leak bits of information about K.

This requires us to make use of the fact that the adversary obtains only a bounded amount of leakage on the ciphertext C, and never sees it completely. We argue that any leakage function that provides enough information about the ciphertext in order to later learn something about the plaintext given the private key, essentially acts as a distinguisher and can be used to break the semantic security of the FHE.

Randomizable FHE. Ciphertexts produced by fully homomorphic encryption schemes may carry information about the homomorphic computation that was performed to obtain them. For instance, it is possible that the ciphertext C_{res} is actually first decrypted to a string of the form $(F(K), K)$ and then the decryption algorithm ignores the second element in the pair. In this case, the adversarial leakage function is clearly not forced to follow the honest decryption algorithm and can make use of the intermediate values of the decryption process to leak information about K. Similarly, the ciphertext C_{key} may contain information about the function F that was evaluated on K. For some applications, such as encryption where F encodes in plain text the message to be encrypted, this is undesirable since the adversary may use future leakage functions to gain information about the message.

Fortunately, the homomorphic encryption schemes of Gentry [Gen09] and of van Dijk *et al* [vDGHV10] have the following additional property: given any encryption C of a message M and a random encryption C' of M', the ciphertext $C+C'$, where the addition is performed over the appropriate group of ciphertexts, is a random encryption of $M+M'$. Consequently, to address the issue described above, we randomize both C_{res} and C_{key} by adding random encryptions of zero to both ciphertexts. In order to make use of the property described above, the encryptions of zero need to be generated without leakage; otherwise, the leaked information maintains a correlation between the randomized ciphertext and the history of the computation that was used to produce the original ciphertext.

We note that in the FHE schemes of [Gen09] and [vDGHV10], C' has to be generated in a special way in order to have enough noise to annihilate any dependence between $C+C'$ and the computation history of C. For simplicity of exposition we ignore this distinction, and instead remark that the randomization procedures of both FHE schemes satisfy the properties needed for our construction.

Function privacy in key proxies. In the above description of key proxies, we require that the leakage obtained by the adversary can be simulated given just F and $F(K)$. However, in some applications, such as private-key encryption, the function F itself also needs to be hidden. In the case of encryption, F contains the message M, so an adversary can break semantic security simply by leaking information about F, ignoring K completely. This raises a subtle modeling issue: the message M must exist somewhere as plaintext, and if the adversary obtains leakage on that computation, he will trivially break semantic security. Therefore, irrespective of the definition of leakage-resilient key proxies, semantic security cannot be achieved when every invocation of every algorithm leaks information.

There are several ways in which this issue can be addressed. One solution is to weaken the definition of semantic security by requiring that the plaintexts have high pseudo-entropy[2] given the leakage obtained by the adversary. We avoid this approach both because it leads to complex definitions, and because it does not seem to have a clear advantage over the following

[2] A distribution has pseudo-entropy $\geq k$ if it is computationally indistinguishable from some distribution with min-entropy $\geq k$.

much cleaner solution. Instead, we allow the adversary to obtain leakage both before and after the challenge ciphertext is generated, but not on the computation of the challenge ciphertext itself. This essentially means that while leakage can compromise individual encryptions, the long-term key remains safe. Under this restriction, our definition of key proxies provides the needed level of security. This approach is consistent with previous definitions of leakage-resilient semantic security (see e.g. [DP08, NS09, DKL09, DGK$^+$10]), and allows us to avoid additional complexity in our definition. This is desirable especially given the fact that for some applications of key proxies, such as signature schemes, function privacy is not necessary.

We mention briefly that another option is to define a leakage model for private-key encryption which allows the encryption algorithm to perform some leak-free pre-processing that is independent of the key. Then, the encryptor can generate an encrypted version of the circuit F, which can be safely given to the adversary without compromising security.

Organization. In Section 2.2, we describe the computational and leakage models that we use, and define a leakage-resilient key proxy. In Section 2.3, we provide our main construction, and analyze its security. In Section 2.4, we describe several variants of our model and construction, and provide some applications of leakage-resilient key proxies.

2.1 Preliminaries

Notation. We write PPT to denote Probabilistic Polynomial Time. When we wish to fix the random bits of a PPT algorithm M to a particular value, we write $M(x;r)$ to denote running M on input x and randomness r. We write $time_n(M)$ to denote the running time of algorithm M on security parameter n. We use $x \in_{\mathrm{R}} S$ to denote the fact that x is sampled according to a distribution S. Similarly, when describing an algorithm we may write $x \leftarrow_{\mathrm{R}} S$ to denote the action of sampling an element from S and storing it in a variable x.

It is common in cryptography to describe probabilistic experiments that test the ability of an adversary to break a primitive. Given such an experiment Exp, and an adversary A, we write $A \leftrightarrows \mathsf{Exp}$ to denote the random variable representing outcome of Exp when run with the adversary A.

2.1.1 Fully Homomorphic Encryption

The main tool in our construction is a fully homomorphic public-key encryption (FHE) scheme. Intuitively, such a scheme has the usual semantic security properties of a public-key encryption (PKE) scheme, but in addition, can perform arbitrary computation on encrypted data. The outcome of this computation is, of course, also encrypted. The first construction of FHE was given by Gentry in [Gen09], and is based on ideal lattices. Recently another construction was proposed by van Dijk *et al* [vDGHV10].

We do not go into the details of the FHE constructions, but rather present the result with respect to an arbitrary FHE with an additional randomization property, which is satisfied by both constructions.

Definition 3 Let $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{EncEval}, \mathsf{Add}, \mathsf{Subtract})$ be a tuple of PPT algorithms, and let $l : \mathbb{N} \to \mathbb{N}$. We say that FHE is an *$l(n)$-secure fully homomorphic public key encryption scheme* if the following conditions hold:

1. The triple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a public-key encryption scheme. We assume without loss of generality that the private key is always the random bits of KeyGen.

2. The algorithm $\mathsf{EncEval}(pub, \boldsymbol{C}, F)$, where pub is a public key, $\boldsymbol{C} = (C_1, \ldots, C_n)$ is a vector of ciphertexts with plaintexts $(m_1, \ldots, m_n)$, and F is a circuit on n inputs, outputs a string C' which is a valid encryption of $F(m_1, \ldots, m_n)$.

3. The algorithms Add and $\mathsf{Subtract}$ have the following properties:

   (a) For all pri, for $pub = \mathsf{KeyGen}(pri)$, for all messages M_1 and M_2, for a random encryption C_1 of M_1 under pub and for every encryption C_2 of M_2 under pub, $\mathsf{Add}(pub, C_1, C_2)$ is distributed identically to $\mathsf{Enc}_{pub}(M_1 + M_2)$, and $\mathsf{Subtract}(pub, C_1, C_2)$ is distributed identically to $\mathsf{Enc}_{pub}(M_1 - M_2)$.

   (b) For all ciphertexts C_1 and C_2, $\mathsf{Add}(pub, \mathsf{Subtract}(pub, C_2, C_1), C_1) = C_2$. That is, subtracting a ciphertext is the inverse of adding it.

4. For every probabilistic adversary A running in time at most $l(n)$, the advantage of A in breaking the semantic security of FHE is at most $1/l(n)$.

Remark 2.1.1 The algorithms Add and $\mathsf{Subtract}$ may be implemented as addition and subtraction over the space of ciphertexts, though we do not require this. In some fully homomorphic encryption schemes, Add and $\mathsf{Subtract}$ may not achieve the exact requirement of step 3 above. Specifically, Add and $\mathsf{Subtract}$ may produce an encryption that cannot be computed on homomorphically using $\mathsf{EncEval}$. We note that this is not a problem for our construction since we only use $\mathsf{EncEval}$ on encryptions of pri, which are ephemeral and never the output of Add or $\mathsf{Subtract}$. We avoid formalizing this issue to improve exposition.

2.2 Models and Definitions

In this section, we present the definition of a leakage-resilient key proxy (LRKP). We start with a syntactic description of the primitive, and then describe the security experiment and the leakage model.

Stateful Algorithms. Due to the continuous nature of side-channel attacks, it is necessary for an LRKP to maintain a state in order to achieve security. We model stateful algorithms by considering algorithms with a special input and output structure. A stateful randomized algorithm takes as input a triple $(x; R, S)$ where x is the query to the algorithm, R is a random string, and S is a state (when R is clear from context we omit it, and denote the input by $(x; S)$). It then outputs (y, S_{new}) where y is the reply to the query, and S_{new} is the new state.

Definition 4 A *key proxy* is a pair $KP = (\mathsf{KPInit}, \mathsf{KPEval})$, where KPInit is an algorithm, and KPEval is a stateful algorithm. For fixed $c \in \mathbb{N}$ and for all $n \in \mathbb{N}$, $K \in \{0,1\}^{n^c}$, $\mathsf{KPInit}(1^n, K)$ outputs an initial state S. For every circuit $F : \{0,1\}^{|K|} \to \{0,1\}^n$, and random coins R, the stateful algorithm $\mathsf{KPEval}(1^n, F; R, S)$ outputs $F(K)$.

We now describe the security experiment of LRKPs. This experiment is parameterized by the leakage structure on a single invocation of the KPEval algorithm. However, for clarity we start with the description of the general experiment, and then provide details on the leakage that occurs at each invocation. We model the the leakage resilience of a key proxy by requiring the leaked information to be simulatable. That is, we require the existence of a simulator Sim that, given F and $F(K)$, can simulate the leakage and messages obtained by the adversary during the computation of $\mathsf{KPEval}(1^n, F; R, S)$. No efficient adversary should be able to tell whether he is getting actual leakage and messages, or interacting with a simulator. We now describe the real and ideal security experiments:

Let $KP = (\mathsf{KPInit}, \mathsf{KPEval})$ be a key proxy. Let A and Sim be PPT algorithms, $n \in \mathbb{N}$, and consider the following two experiments:

$\mathsf{ExpReal}$ **(Real Interaction).** The interaction of the adversary with the key proxy proceeds as follows:

1. A key K is chosen by the adversary, and $\mathsf{KPInit}(1^n, K)$ is used to generate an initial state S.
2. The adversary repeats the following steps an arbitrary number of times:
   (a) The adversary submits a circuit F, which is evaluated on K by KPEval. During the computation, the adversary acts as a single invocation leakage adversary (described below in Definition 7) for KPEval.

(b) At the end of the computation of KPEval, the adversary is given $F(K)$.

3. After the adversary is done making queries, it outputs a bit b.

$\mathsf{ExpIdeal}$ **(Ideal Interaction).** The interaction of the adversary with simulated leakage proceeds as follows:

1. The adversary submits a key K, which is not revealed to the simulator.
2. The adversary then repeats the following steps an arbitrary number of times:
   (a) The adversary submits a circuit F, and Sim is given F and $F(K)$. The adversary then acts as a single invocation leakage adversary according to Definition 7, except that the leakage functions are submitted to the simulator, which returns simulated leakage values and messages.
   (b) Eventually the adversary stops submitting leakage functions, and is given $F(K)$.
3. After the adversary is done making queries, it outputs a bit b.

Definition 5 We say that KP is a *Leakage-Resilient Key Proxy* if for every PPT A there exists a PPT S and a negligible function $neg(\cdot)$ such that

$$|\Pr[(A \leftrightarrows \mathsf{ExpReal}) = 1] - \Pr[(A \leftrightarrows \mathsf{ExpIdeal}) = 1]| \leq neg(n)$$

The above definition describes the security of an LRKP relative to some unspecified procedure which allows the adversary to obtain leakage during each invocation of KPEval. The exact procedure for a single-invocation leakage depends on the leakage model and on the structure of the implementation of KPEval. Below we formalize the structure of our solution, and describe the leakage obtained by the adversary during a single invocation of KPEval.

Our construction of KPEval is described as a protocol between two parties EvalA and EvalB that leak information separately, and where the messages between EvalA and EvalB are public. In this format, our construction requires two flows between the parties: one from EvalA to EvalB and one from EvalB to EvalA. The following definition formalizes this structure.

Definition 6 A 2-round *split state key proxy* $KP = (\mathsf{KPInit}, \mathsf{KPEval})$ is a key proxy such that the state S is represented as a pair $S = (\mathsf{MemA}, \mathsf{MemB}) \in (\{0,1\}^{n^d})^2$ for some fixed $d \in \mathbb{N}$, and the algorithm KPEval is described as four algorithms $(\mathsf{LeakFree}, \mathsf{EvalA}_1, \mathsf{EvalB}, \mathsf{EvalA}_2)$, each running in time polynomial in n, where

1. EvalA_1 takes as input MemA, OutLF_A, and randomness RandA, and outputs an updated state $\mathsf{MemA}' \in \{0,1\}^{n^d}$ and a message M_{AB} to EvalB.

2. $\mathsf{LeakFree}$ takes as input message M_{AB} and randomness RandLF, and outputs string OutLF.

3. EvalB takes as input MemB, randomness RandB, OutLF, the message M_{AB}, and a circuit $F : \{0,1\}^{|K|} \to \{0,1\}^n$ of arbitrary size. It then outputs an updated state $\mathsf{MemB}' \in \{0,1\}^{n^d}$ and a message M_{BA} to EvalA.

4. EvalA_2 takes as input MemA', the message M_{BA} and outputs an updated state MemA'' and the result $F(K)$.

The output of KPEval is $F(K)$, and the updated state is $(\mathsf{MemA}'', \mathsf{MemB}')$.

Recall that our construction requires a leak-free component. This leak-free component is modeled by algorithm $\mathsf{LeakFree}$ above. A crucial point here is that $\mathsf{LeakFree}$ receives only randomness and a public message as input, and, in particular, receives neither F nor the saved state $(\mathsf{MemA}, \mathsf{MemB})$ as inputs; therefore, regardless of the actual construction, the above definition prevents $\mathsf{LeakFree}$ from carrying out the evaluation of F on K, which would make the construction trivial.

We are now ready to describe the leakage structure on a single invocation of a 2-round split state key proxy. The leakage model we use, commonly known as "only computation leaks information" (OCL), lets the adversary obtain leakage only on the active part of memory during each computation.

Definition 7 Let $l : \mathbb{N} \to \mathbb{N}$ and let KP be a 2-round split state key proxy. A single invocation leakage adversary in the only-computation-leaks model chooses a circuit f_1, then sees $f_1(\mathsf{MemA}, \mathsf{RandA})$ and M_{AB}, chooses circuit f_2, then sees $f_2(\mathsf{MemB}, \mathsf{OutLF}, \mathsf{RandB})$ and M_{BA}, chooses a circuit f_3, and finally sees $f_3(\mathsf{MemA}')$. The adversary is l-bounded if for all n the range of f_1, f_2, f_3 is $\{0,1\}^{l(n)}$.

Note that in the above definition, the leakage functions can compute any internal values that appear during the computations of EvalA_1, EvalB, and EvalA_2. This means, for example, that it is unnecessary to explicitly provide M_{AB} to f_1 or M_{BA} to f_2.

History freeness. In Definition 5 we allow information about the functions F_i that are evaluated on K to leak to the adversary. In particular, it is possible that during some invocation j the adversary can obtain, through leakage, information about some previously queried function F_i. In the introduction we mentioned that leakage-resilient variants of some applications, such as private-key encryption, are defined to allow leakage both before and after the generation of the challenge ciphertext, but not on the challenge itself. However, if the state of LRKP keeps a history of some of the functions that were applied to K, then by leaking on it after the challenge was computed, the adversary may be able to break the semantic security of the encryption. We note that the above definition is sufficient to obtain security in the presence of what we call "lunch-time leakage" attacks – where the adversary obtains leakage only before the challenge ciphertext is generated, but not after.

To address the above issue, and allow full leakage in applications such as encryption, we introduce an additional information-theoretic property that requires that the state of the LRKP is distributed identically after all sequences of functions that are evaluated on K. This property is satisfied by our construction, and prevents the above mentioned "history attack".

Definition 8 An LRKP $(\mathsf{KPInit}, \mathsf{KPEval})$ is called *history free* if for all $n \in \mathbb{N}$ and all $K \in \{0,1\}^{poly(n)}$, there exists a distribution D over the states of the LRKP such that for all $j \in \mathbb{N}$, all sequences of functions $F_1, \ldots, F_j : \{0,1\}^{|K|} \to \{0,1\}^n$, and all sequences of random tapes $R_0, \ldots, R_{j-1}$, the random variable $\{S_{j+1} | S_1, \ldots, S_j\}$ over R_j is distributed according to D, where $S_1 = \mathsf{KPInit}(1^n, K; R_0)$ and S_i is the updated state after $\mathsf{KPEval}(1^n, F_{i-1}; R_i, S_{i-1})$.

2.3 Leakage-Resilient Key Proxies From Homomorphic Encryption

Given a fully homomorphic public-key encryption scheme $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{EncEval}, \mathsf{Add}, \mathsf{Subtract})$ we construct a leakage-resilient 2-round split state key proxy $\mathsf{LRKP} = (\mathsf{KPInit}, \mathsf{KPEval})$.

$\mathsf{KPInit}(1^n, K)$**:** The algorithm $\mathsf{KPInit}(1^n, K)$ first runs $\mathsf{KeyGen}(1^n)$ to obtain a public-private key pair (pub_1, pri_1) for the FHE. It then generates a ciphertext $C_{\mathsf{key}} = \mathsf{Enc}_{pub_1}(K)$ and assigns $\mathsf{MemA} \leftarrow pri_1$ and $\mathsf{MemB} \leftarrow C_{\mathsf{key}}$. The output is an initial state that consists of two parts $(\mathsf{MemA}, \mathsf{MemB})$.

$\mathsf{KPEval}(1^n, F; (\mathsf{MemA}, \mathsf{MemB}))$**:** The algorithm KPEval consists of four subroutines – $\mathsf{LeakFree}$, EvalA_1, EvalB, and EvalA_2 – that are used as follows: on input circuit F first generate $(\mathsf{OutLF}_A, \mathsf{OutLF}_B) \leftarrow_{\mathrm{R}} \mathsf{LeakFree}(1^n)$. Then, follow the protocol described in Figure 2.2 by computing

$$\begin{aligned}
&(M_{AB}, \mathsf{MemA}') \leftarrow_{\mathrm{R}} \mathsf{EvalA}_1(\mathsf{MemA}, \mathsf{OutLF}_A);\\
&(M_{BA}, \mathsf{MemB}') \leftarrow_{\mathrm{R}} \mathsf{EvalB}(\mathsf{MemB}, \mathsf{OutLF}_B, M_{AB});\\
&Y \leftarrow \mathsf{EvalA}_2(\mathsf{MemA}', M_{BA})
\end{aligned}$$

The final state after one evaluation of KPEval is $(\mathsf{MemA}', \mathsf{MemB}')$, and the output is Y.

We now describe the subroutines $\langle \mathsf{LeakFree}, \mathsf{EvalA}_1, \mathsf{EvalB}, \mathsf{EvalA}_2 \rangle$ of KPEval:

$\mathsf{LeakFree}(pub)$: Parse randomness as (r_{LF1}, r_{LF2}), and compute

$$C_{\mathsf{R0}} = \mathsf{Enc}_{pub}(\bar{0}; r_{LF1})$$
$$C_{\mathsf{R1}} = \mathsf{Enc}_{pub}(\bar{0}; r_{LF2})$$
$$\mathsf{OutLF} = (C_{\mathsf{R0}}, C_{\mathsf{R1}})$$

and output OutLF.

The subroutines EvalA_1, EvalB, and EvalA_2 are described in Figure 2.2 as a two round two party protocol where EvalA_1 and EvalA_2 specify the actions of party A and EvalB specifies the actions of party B. In the definition of EvalB we use subroutines $\mathsf{Evaluate}$ and $\mathsf{Refresh}$ that are defined as follows:

$\mathsf{Evaluate}(F, C, pri)$: Compute and output $F(\mathsf{Dec}_{pri}(C))$

$\mathsf{Refresh}(C, pri)$: Compute and output $\mathsf{Dec}_{pri}(C)$

The correctness of this construction follows in a straightforward manner from the correctness of the underlying FHE. We also note that our construction is *history free* according to Definition 8. This is due to the fact that the values assigned to MemA and MemB at the end of KPEval are independent from the function F. In particular, MemA is simply a random private key, and MemB contains an encryption of K which was obtained by a homomorphic evaluation of $\mathsf{Refresh}$ on the previous contents of MemB and an encryption of the previous private key, neither of which depends on F.

The bulk of the analysis is in showing that our construction is in fact leakage-resilient according to Definition 5, where during each invocation the leakage structure on the computation of KPEval is given in Definition 7. We now state our main theorem.

Theorem 2.3.1 *Let* LRKP *be the 2-round split state key proxy described in the above construction, and let* $l : \mathbb{N} \to \mathbb{N}$. *If* FHE *is a* $2^{O(l(n))}$*-secure fully homomorphic encryption then* LRKP *is leakage-resilient against all* $O(l(n))$*-bounded adversaries in the OCL model.*

The theorem follows as a corollary from the following lemma:

Lemma 2.3.2 *Consider the experiment ExpReal instantiated using scheme* $LRKP$*. Then, for every* $d > 0$*, every* $l : \mathbb{N} \to \mathbb{N}$*, and every* l*-bounded PPT adversary Adv that makes* n^d *queries and gets leakage according to the only-computation-leaks model, there exists a PPT simulator* S *such that for every function* $\varepsilon(n) > 0$*, if*

$$|\Pr[(Adv \leftrightarrows ExpReal) = 1] - \Pr[(Adv \leftrightarrows S) = 1]| \geq \varepsilon(n)$$

Party A

Contents of MemA: pri_i

Randomness: pri_{i+1}, r^i_{pri}

Party B

Contents of MemB: $C'_{\mathsf{key},i} = \mathsf{Enc}_{pub_i}(K)$

Randomness: r^i_{B1}, r^i_{B2}

Input: F_i

EvalA_1:

$pub_{i+1} = \mathsf{KeyGen}(pri_{i+1})$

$C^i_{pri} = \mathsf{Enc}_{pub_{i+1}}(pri_i; r^i_{pri})$

$\mathsf{MemA} \leftarrow pri_{i+1}$

$\xrightarrow{pub_{i+1}, C^i_{pri}}$

$(C_{\mathsf{R0},i}, C_{\mathsf{R1},i}) = \mathsf{LeakFree}(pub_{i+1})$

EvalB:

$C_{\mathsf{res},i} = \mathsf{EncEval}(pub_{i+1}, C^i_{pri}, \mathsf{Evaluate}(F_i, C'_{\mathsf{key},i}, \cdot); r^i_{B1})$

$C_{\mathsf{key},i+1} = \mathsf{EncEval}(pub_{i+1}, C^i_{pri}, \mathsf{Refresh}(C'_{\mathsf{key},i}, \cdot); r^i_{B2})$

$C'_{\mathsf{res},i} = \mathsf{Add}(pub_{i+1}, C_{\mathsf{R0},i}, C_{\mathsf{res},i})$

$C'_{\mathsf{key},i+1} = \mathsf{Add}(pub_{i+1}, C_{\mathsf{R1},i}, C_{\mathsf{key},i+1})$

$\mathsf{MemB} \leftarrow C'_{\mathsf{key},i+1}$

$\xleftarrow{C'_{\mathsf{res},i}}$

EvalA_2:

$Y_i = \mathsf{Dec}_{pri_{i+1}}(C'_{\mathsf{res},i})$

Output Y_i

Figure 2.2: The algorithm KPEval in its ith invocation.

for infinitely many n, then for every function $\varepsilon'(n) > 0$ there exists an adversary Adv' that runs in time

$$\frac{2^{3l(n)+7}}{\varepsilon'(n)^2}\left(3l(n) + 4 + \log\frac{1}{\varepsilon'(n)}\right) \cdot time_n\left(LRKP \leftrightarrow Adv\right)$$

and breaks the semantic security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with advantage*

$$\frac{\varepsilon(n)}{3 \cdot 2^{2l(n)}(n^d + 1)} - 2\varepsilon'(n)$$

for infinitely many n. Specifically, S runs in time $time_n(LRKP \leftrightarrow Adv)$.

We give an overview of the proof of Lemma 2.3.2 in Section 2.3.1 and we give the proof details in Section 2.3.2

2.3.1 Proof overview for Lemma 2.3.2

Let Adv be a PPT adversary according to Definition 5 that gets leakage according to the only-computation-leaks model described in Definition 7. We define a sequence of experiments where the initial experiment is the real security experiment $\mathsf{ExpReal}$, and the final experiment is such that the leakage obtained by the adversary for each KPEval query F can be simulated given only $(F, F(K))$. Specifically, the final experiment involves instantiating our construction with key $\bar{0}$ instead of K. We show that if Adv can distinguish the initial experiment and the final experiment, we can construct an adversary Adv' that, roughly speaking, distinguishes variants of these experiments that consist of only two rounds. We then show how pairs of the leakage queries of Adv' can be combined into a single query (of larger output length) using a guess-and-check approach: when the adversary would normally make the first of the pair of leakage queries, it instead guesses an output and verifies this guess when it makes the second leakage query; when the guess is wrong, the adversary outputs a randomly chosen bit. Repeatedly combining queries in this manner yields an adversary that just makes a single leakage query and (essentially) distinguishes encryptions of K and $\bar{0}$. To finish the proof, we use an observation of Akavia *et al* [AGV09] that every $2^{O(\ell(n))}$-semantically-secure public-key encryption scheme remains secure when the adversary gets $O(\ell(n))$ bits of leakage on KeyGen.

2.3.2 Proof of Lemma 2.3.2

We first introduce some notation. We denote by MemA_i and MemB_i the saved state of party A and party B before round i. We denote by F_i the ith function that the adversary submits to be evaluated on K, we denote by f_i^j the jth leakage query during the computation of KPEval on F_i, and we denote by λ_i^j the response to the leakage query f_i^j. For our construction, $j \in \{1, 2, 3\}$. Specifically, λ_i^1 is the initial leakage on EvalA_1 from $(pri_i, pri_{i+1}, r_{pri}^i)$ in round i, λ_i^2 is the leakage on EvalB from $(C'_{\mathsf{key},i}, C_{\mathsf{R0},i}, C_{\mathsf{R1},i}, r^i_{B1}, r^i_{B2})$, and λ^3_i is the final leakage on EvalA_2 from (pri_{i+1}). In addition to seeing leakage, the adversary also gets all communication between party A and party B. Specifically, after seeing λ^1_i but before submitting f^2_i, the adversary is given pub_{i+1} and C^i_{pri}, and after seeing λ^2_i but before submitting f^3_i, the adversary is given $C'_{\mathsf{res},i}$.

Hybrid Experiment Structure

We now describe the sequence of hybrid experiments:

Experiment Hyb_0. Hyb_0 is the real security experiment $\mathsf{ExpReal}$.

Experiment Hyb_1. Experiment Hyb_1 is the same as Hyb_0, except that a dummy round is added at the beginning and at the end of the experiment. More precisely, before the first evaluation query of the adversary, the initialization algorithm KPInit is run, and then a single round of KPEval is performed with a dummy function (e.g. one that always outputs $\bar{0}$). Similarly, after the adversary makes the last evaluation query, another dummy round of KPEval is performed.

Note that $(\mathsf{MemA}_1, \mathsf{MemB}_1)$ are distributed identically in Hyb_0 and Hyb_1, and the additional dummy round $n^d + 1$ has no effect on the view of the adversary. Thus, the above changes are purely conceptual.

We now describe a second hybrid experiment, where the changes are more substantial.

Experiment Hyb_2. In experiment Hyb_2 we remove the key K from all variables that are exposed to the adversary. In particular, MemB will now contain an encryption of $\bar{0}$ instead of K. This change, by itself, would corrupt the output of KPEval, which depends on the contents of MemB. We correct this error by changing the way we compute the ciphertext $C_{\mathsf{R0},i}$ so that when this ciphertext is added to $C_{\mathsf{res},i}$, the resulting ciphertext $C'_{\mathsf{res},i}$ contains the intended output $F_i(K)$. More formally: experiment Hyb_2 proceeds in the same way as Hyb_1 with the following changes.

1. During the initialization process, $C_{\mathsf{R1},0}$ is set to $\mathsf{Enc}_{pub_1}(-K)$.
2. In each round $0 < i \leq n^d$, $C_{\mathsf{R0},i}$ is computed as $\mathsf{Enc}_{pub_{i+1}}(F_i(K) - F_i(\bar{0}))$.

Observe that aside from in the initialization round, the only information about K that is needed to carry out Hyb_2 are the values $F_i(K)$ for each query F_i produced by the adversary. It is easy to see that, in fact, the initialization round can be modified so that K is not needed, without changing the distribution of the leakage values and communication seen by the adversary during the experiment. This modification is described by the following hybrid experiment:

Experiment Hyb_3**.** Experiment Hyb_3 proceeds in the same way as Hyb_2, except that dummy round $n^d + 1$ is omitted and the initialization process is done differently: dummy round 0 is omitted, and $C'_{\mathsf{key},1}$ is set directly to $\mathsf{Enc}_{pub_1}(\bar{0})$. The entire modified initialization is as follows:

1. Run KeyGen to obtain (pub_1, pri_1).

2. Compute $C'_{\mathsf{key},1} = \mathsf{Enc}_{pub_1}(\bar{0})$.

3. Set $\mathsf{MemA}_1 = pri_1$ and $\mathsf{MemB}_1 = C'_{\mathsf{key},1}$.

Note that $(\mathsf{MemA}_1, \mathsf{MemB}_1)$ are distributed identically in Hyb_2 and Hyb_3. Furthermore, omitting dummy round $n+1$ has no effect on the view of the adversary. Thus, the above change is purely conceptual.

Our simulator S interacts with the adversary as in Hyb_3. Note that S runs in time at most $time_n(LRKP \leftrightarrow Adv)$. To show that the adversary is unable to distinguish between leakage produced according to Hyb_3, and therefore between simulated leakage and real leakage, we show that each pair of consecutive hybrid experiments is indistinguishable.

To facilitate the analysis, we denote by X_i the random variable corresponding to the output of the adversary in experiment Hyb_i. We have already mentioned the following two facts:

Fact 2.3.3 $\Pr[X_0 = 1] = \Pr[X_1 = 1]$

Fact 2.3.4 $\Pr[X_2 = 1] = \Pr[X_3 = 1]$

The crux of the proof is comparing experiments Hyb_1 and Hyb_2. For this purpose, we first define a sequence of intermediate hybrids that are between Hyb_1 and Hyb_2. For $0 \le i \le n^d + 1$, Hyb^i_{12} behaves the same as Hyb_1 up to round $i - 1$, behaves the same as Hyb_2 from round $i + 1$ onward, and behaves specially in round i. For $0 \le i \le n^d$, Experiment Hyb^i_{12} is defined as follows.

Experiment Hyb^i_{12}**.**

1. For $0 \le j \le i - 1$, round j proceeds the same as in Hyb_1.

2. Round i proceeds the same as Hyb_1, except that $C_{\mathsf{R1},i}$ is set to $\mathsf{Enc}_{pub_{i+1}}(-K)$.

3. For $i + 1 \le j \le n^d + 1$, round j proceeds the same as in Hyb_2.

Note that the dummy round 0 that takes place during the initialization process proceeds identically to Hyb_2 if $i = 0$, and to Hyb_1 otherwise. Also, note that dummy round $n^d + 1$ always proceeds identically in both Hyb_1 and Hyb_2. Consequently, Hyb_1 is identical to $\mathsf{Hyb}^{n^d+1}_{12}$, and Hyb_2 is identical to Hyb^0_{12}.

We now show that if there exists an adversary A that distinguishes Hyb_{12}^{0} and $\mathsf{Hyb}_{12}^{n^d+1}$, then there exists an adversary A' that succeeds in the following experiment Exp_1.

Experiment Exp_1**.** Say that on inputs of length n, the output of Enc has length n'.

1. The adversary submits two messages m_0, m_1 such that $m_0 \neq m_1$.
2. A pair of public and private keys are generated $(pub, pri) = \mathsf{KeyGen}(1^n)$, and pub is given to the adversary.
3. The adversary submits a leakage function $\mathsf{leak}_1 : \{0,1\}^n \to \{0,1\}^{l(n)}$, and sees $\mathsf{leak}_1(pri)$.
4. A random bit b is chosen, and an encryption $C = \mathsf{Enc}_{pub}(m_b)$ is computed.
5. The adversary submits a leakage function $\mathsf{leak}_2 : \{0,1\}^{n'} \to \{0,1\}^{l(n)}$, and sees $\mathsf{leak}_2(C)$.
6. The adversary submits a leakage function $\mathsf{leak}_3 : \{0,1\}^n \to \{0,1\}^{l(n)}$, and sees $\mathsf{leak}_3(pri)$.
7. A new pair of public and private keys are generated $(pub', pri') = \mathsf{KeyGen}(1^n)$, and a random string $r_{pri'}$ is chosen. The public key pub' is given to the adversary.
8. The adversary submits a leakage function $\mathsf{leak}_4 : \{0,1\}^{3n} \to \{0,1\}^{l(n)}$, and then sees $\mathsf{leak}_4(pri, pri', r_{pri'})$.
9. The adversary sees $C' = \mathsf{Enc}_{pub'}(pri; r_{pri'})$.
10. The adversary submits a leakage function $\mathsf{leak}_5 : \{0,1\}^{n'} \to \{0,1\}^{l(n)}$, and sees $\mathsf{leak}_5(C)$.
11. The adversary sees pri, pri', and outputs a bit $\hat{b}$.

We say that an adversary A' succeeds with advantage $\varepsilon(n)$ in Exp_1 if $|\Pr[(A' \leftrightarrows Exp_1) = 1|b = 1] - \Pr[(A' \leftrightarrows Exp_1) = 1|b = 0]| \geq \varepsilon(n)$.

Claim 2.3.5 *Let A be an adversary and define, for all n, $\varepsilon(n) = |\Pr[(A \leftrightarrows \mathsf{Hyb}_{12}^{0}) = 1] - \Pr[(A \leftrightarrows \mathsf{Hyb}_{12}^{n^d+1}) = 1]|$. Then there exists an adversary A' that, for all n, runs in time $4 \cdot time_n(\mathsf{Enc}) + time_n(LRKP \leftrightarrow A)$ and succeeds with advantage $\varepsilon(n)/(n^d + 1)$ in Exp_1.*

Proof We first summarize the construction of A'. A' randomly selects an i, $0 \leq i \leq n^d$, and then simulates A according to Hyb_1 up to round $i-1$. Then, A' submits the two messages $m_0 = K$ and $m_1 = \bar{0}$, and uses the leakage queries permitted by Exp_1 to answer the queries of A during the ith and $i+1$st rounds. During the simulation, pub plays the role of pub_{i+1}, pub' the role of pub_{i+2}, C the role of $C'_{\mathsf{key},i+1}$ and C' the role of $C_{pri}^{i+1} = \mathsf{Enc}_{pub_{i+2}}(pri_{i+1})$. A' uses C and the properties of Add to "work backwards" and obtain correctly distributed values for $C_{\mathsf{R1},i}$, $C_{\mathsf{R0},i+1}$, and $C_{\mathsf{R1},i+1}$. Then, from round $i+2$ onward, A' simulates A according to Hyb_2, and outputs whatever A outputs. By construction, we have that if C is an encryption of $\bar{0}$ then A' simulates A perfectly in Hyb^{i}_{12}, and if C is an encryption of K, A is simulated perfectly in Hyb^{i+1}_{12}. The details follow.

A' begins by randomly selecting i such that $0 \le i \le n^d$. We first handle the case $1 \le i \le n^d - 1$. Our adversary A' simulates A according to Hyb_1 up to round $i-1$ (note that Hyb^{i}_{12} and Hyb^{i+1}_{12} proceed identically up to that round). Then, A' submits the two messages $m_0 = K$ and $m_1 = \bar{0}$, and obtains a public key pub. A' starts simulating A in round i by obtaining the first leakage function f^1_i. A' then generates uniformly r^i_{pri}, and creates the following leakage function:

- $\mathsf{leak}_1(pri)$: Compute and return $f^1_i(pri_i, pri, r^i_{pri})$.

A' submits the above leakage function in step 3, and obtains a string λ^1_i. A' also computes $C^i_{pri} = \mathsf{Enc}_{pub}(pri_i; r^i_{pri})$, and gives the tuple $(\lambda^1_i, pub, C^i_{pri})$ to A. A then outputs leakage functions f^2_i. A' generates an encryption $C'_{\mathsf{res},i} = \mathsf{Enc}_{pub}(F_i(K))$, and randomly selects r^i_{B1} and r^i_{B2}. Then A' constructs the following leakage function:

- $\mathsf{leak}_2(C)$:
  - Compute $C_{\mathsf{res},i} = \mathsf{EncEval}(pub, C^i_{pri}, \mathsf{Evaluate}(F_i, \mathsf{MemB}_i, \cdot); r^i_{B1})$.
  - Compute $C_{\mathsf{key},i+1} = EncEval(pub, C^i_{pri}, \mathsf{Refresh}(\mathsf{MemB}_i, \cdot); r^i_{B2})$.
  - Compute $C_{\mathsf{R0},i} = \mathsf{Subtract}(pub, C'_{\mathsf{res},i}, C_{\mathsf{res},i})$.
  - Compute $C_{\mathsf{R1},i} = \mathsf{Subtract}(pub, C, C_{\mathsf{key},i+1})$.
  - Compute and return $f^2_i(\mathsf{MemB}_i, C_{\mathsf{R0},i}, C_{\mathsf{R1},i}, r^i_{B1}, r^i_{B2})$.

A' submits the leakage function in step 5, and obtains λ^2_i. A' gives $(\lambda^2_i, C'_{\mathsf{res},i})$ to A. A then outputs f^3_i. A' sets:

- $\mathsf{leak}_3(pri)$: Compute and return $f^3_i(pri)$.

A' is now given λ^3_i and pub'. Using λ^3_i, A' obtains the first leakage function f^1_{i+1} for round $i+1$, and sets:

- $\mathsf{leak}_4(pri, pri', r^{i+1}_{pri})$: Compute and return $f^1_{i+1}(pri, pri', r^{i+1}_{pri})$.

A' is now given λ^1_{i+1} and a ciphertext C'. Using the tuple $(\lambda^1_{i+1}, pub', C')$ A' obtains from A a leakage function f^2_{i+1}. A' also computes encryptions $C'_{\mathsf{key},i+2} = \mathsf{Enc}_{pub'}(\bar{0})$ and $C'_{\mathsf{res},i+1} = \mathsf{Enc}_{pub'}(F_{i+1}(K))$, and randomly selects r^{i+1}_{B1} and r^{i+1}_{B2}. A' sets

- $\mathsf{leak}_5(C)$:

– Compute $C_{\mathsf{res},i+1} = \mathsf{EncEval}(pub', C', \mathsf{Evaluate}(F_{i+1}, C, \cdot); r_{B1}^{i+1})$.

– Compute $C_{\mathsf{key},i+2} = \mathsf{EncEval}(pub', C', \mathsf{Refresh}(C, \cdot); r_{B2}^{i+1})$.

– Compute $C_{\mathsf{R0},i+1} = \mathsf{Subtract}(pub', C'_{\mathsf{res},i+1}, C_{\mathsf{res},i+1})$.

– Compute $C_{\mathsf{R1},i+1} = \mathsf{Subtract}(pub', C'_{\mathsf{key},i+2}, C_{\mathsf{key},i+2})$.

– Compute and return $f_{i+1}^2(C, C_{\mathsf{R0},i+1}, C_{\mathsf{R1},i+1}, r_{B1}^{i+1}, r_{B2}^{i+1})$.

and obtains a value λ_{i+1}^2. A' uses $(\lambda_{i+1}, C'_{\mathsf{res},i+1})$ to obtain f_{i+1}^3 from A. A' is then given pri'. From this point onward, A' simulates A according to Hyb_2. Note that the only value which is not generated by A' that is needed to perform this simulation is pri'. At the end of the simulation A outputs a bit $\hat{b}$, which A' also outputs. By construction, we have that if C is an encryption of $\bar{0}$ then A' simulates A perfectly in Hyb_{12}^i, and if C is an encryption of K, A is simulated perfectly in Hyb_{12}^{i+1}.

Notice that since A' simulates A along with the experiment with which A is interacting, and does some additional work in rounds i and $i+1$, A' runs in time at most $4 \cdot time_n(\mathsf{Enc}) + time_n(LRKP \leftrightarrow A)$.

It remains to handle the cases $i = 0$ and $i = n^d$. These are handled similarly to the first case, except we have to take into account the fact that A sees no leakage or communication during "rounds" 0 and $n^d + 1$. More specifically, for the case $i = 0$, A' proceeds as in the first case except that it does not submit leak_1, leak_2, or leak_3 in round i (or, alternatively, it submits constant functions and ignores their output), nor does it produce $C'_{\mathsf{res},0}$ or give anything to A during round i; for round $i+1$, A' proceeds as in the first case, starting by obtaining leakage function f_{i+1}^1 from A. For the case $i = n^d$, A' proceeds as in the first case except that it does not submit leak_4 or leak_5 (or, alternatively, it submits constant functions and ignores their output), nor does it produce C'_{res,n^d+1} or give anything to A during round $i+1$.

Once again, for both these cases, we have by construction that if C is an encryption of $\bar{0}$ then A' simulates A perfectly in Hyb_{12}^i, and if C is an encryption of K, A is simulated perfectly in Hyb_{12}^{i+1}. It then follows by standard arguments that A' succeeds with advantage $\varepsilon(n)/(n^d+1)$ in Exp_1.

□

We shall now prove an upper bound on the advantage of adversaries in Exp_1. Let A_1 be an adversary in Exp_1 and let ε be its advantage. Then, we show that there exists an adversary A_2 that succeeds with advantage $\varepsilon(n)/2^{l(n)}$ in the following experiment Exp_2.

Experiment Exp_2**.** Exp_2 proceeds identically to Exp_1, except steps 8-11 are modified as follows

8. The adversary submits a leakage function $\mathsf{leak}_4 : \{0,1\}^{3n} \to \{0,1\}^{l(n)+1}$, and then sees $\mathsf{leak}_4(pri, pri', r_{pri'})$.

9. The adversary sees $C' = \mathsf{Enc}_{pub'}(pri; r_{pri'})$ and C.

10. The adversary outputs a bit $\hat{b}$.

Claim 2.3.6 *Let A_1 be an adversary and define, for all n, $\varepsilon(n)$ to be the advantage of A_1 in Exp_1. Then there exists an adversary A_2 that, for all n, runs in time at most $3 \cdot time_n(A_1) + time_n(\mathsf{Enc})$ and succeeds in Exp_2 with advantage $\varepsilon(n)/2^{l(n)}$.*

Proof A_2 randomly selects a string r_{sim} to use as the randomness of A_1. Then, using r_{sim}, A_2 simulates A_1 up to and including step 7 without any modifications. In step 8, A_2 obtains a leakage function leak_4 from A_1, and randomly selects guessed leakage value $\hat{\lambda}_5 \in \{0,1\}^{l(n)}$. A_2 then constructs a new leakage function:

- $\mathsf{leak}'_4(pri, pri', r_{pri'})$: First, compute $\lambda_4 = \mathsf{leak}_4(pri, pri', r_{pri'})$. Then, compute $C' = \mathsf{Enc}_{pub'}(pri; r_{pri'})$, and use $(\lambda_4, C', \hat{\lambda}_5)$ to complete the simulation of A_1 (using randomness r_{sim}). Let $\hat{b}$ be the bit output by A_1. The output of leak'_4 is then $(\lambda_4, \hat{b})$.

A_2 submits leak'_4 in step 8, and is given (λ'_4, C', C) where $\lambda'_4 = (\lambda_4, \hat{b})$. Using (λ_4, C'), A_2 obtains from A_1 the leakage function leak_5. Now, A_2 checks whether $\mathsf{leak}_5(C) = \hat{\lambda}_5$, and if so it outputs $\hat{b}$. Otherwise, A_2 flips an unbiased coin and outputs the outcome. Observe that A_2 runs in time at most $3 \cdot time_n(A_1) + time_n(\mathsf{Enc})$.

Notice that if A_2 guesses the leakage $\hat{\lambda}_5$ correctly then it simulates A_1 perfectly, and that the leakage is guessed correctly with probability $1/2^{l(n)}$. We therefore conclude:

$$\begin{aligned}&|\Pr[(A_2 \leftrightarrows Exp_2) = 1 | b = 0] - \Pr[(A_2 \leftrightarrows Exp_2) = 1 | b = 1]| \geq \\ &\frac{1}{2^{l(n)}} |\Pr[(A_1 \leftrightarrows Exp_1) = 1 | b = 0] - \Pr[(A_1 \leftrightarrows Exp_1) = 1 | b = 1]|\end{aligned}$$

□

We now simplify the experiment further. For clarity, we describe the modified experiment completely:

Experiment Exp_3**.** The new experiment proceeds as follows:

1. The adversary submits two messages m_0, m_1 such that $m_0 \neq m_1$.

2. Private keys pri, pri' are randomly chosen, and public keys $pub = \mathsf{KeyGen}(pri)$, $pub' = \mathsf{KeyGen}(pri')$ are computed, and given to the adversary.

3. The adversary submits a leakage function $\mathsf{leak}_1 : \{0,1\}^{3n} \to \{0,1\}^{3l(n)+1}$, and sees $\mathsf{leak}_1(pri, pri', r_{pri'})$.

4. A random bit b is chosen, and an encryption $C = \mathsf{Enc}_{pub}(m_b)$ is computed.

5. The adversary sees $C' = \mathsf{Enc}_{pub'}(pri; r_{pri'})$ and C.

6. The adversary outputs a bit $\hat{b}$.

Claim 2.3.7 *Let A_2 be an adversary and define, for all n, $\varepsilon(n)$ to be the advantage of A_2 in Exp_2. Then there exists an adversary A_3 that, for all n, runs in time at most $4 \cdot time_n(A_2)$ that succeeds in Exp_3 with advantage $\varepsilon(n)/2^{l(n)}$.*

Proof The basic idea is the same as in the proof of Claim 2.3.6. A_3 guesses a response $\hat{\lambda}_2 \in \{0,1\}^{l(n)}$ to A_2's leak_2 query, uses this guess to simulate A_2 within the leakage function that A_3 submits, and then verifies its guess. The details follow.

A_3 randomly selects a string r_{sim} to use as the randomness of A_2. Then, using r_{sim}, A_3 starts simulating A_2, obtaining a leakage function leak_1. A_3 randomly selects a guessed leakage value $\hat{\lambda}_2 \in \{0,1\}^{l(n)}$, and then constructs a new leakage function:

- $\mathsf{leak}'_1(pri, pri', r_{pri'})$: First, compute $\lambda_1 = \mathsf{leak}_1(pri)$. Simulate A_2, using randomness r_{sim} and using $(\lambda_1, \hat{\lambda}_2)$ as the responses to the first two leakage queries. A_2 then produces a leakage function leak_3. Compute $\lambda_3 = \mathsf{leak}_3(pri)$, and continue simulating A_2 using λ_3. A_2 produces a leakage function leak_4. Compute $\lambda_4 = \mathsf{leak}_4(pri, pri', r_{pri'})$. The output of leak'_1 is then $(\lambda_1, \lambda_3, \lambda_4)$.

A_3 submits leak'_1, and is given (λ'_1, C', C), where $\lambda'_1 = (\lambda_1, \lambda_3, \lambda_4)$. A_3 continues its simulation of A_2, using λ_1 as the response to the first leakage query. A_2 then produces a leakage function leak_2. Now, A_3 checks whether $\mathsf{leak}_2(C) = \hat{\lambda}_2$; if not, A_3 outputs a randomly selected bit. Otherwise, A_3 continues simulating A_2, using λ_3 and λ_4 as the responses to the next two leakage queries. Then, A_3 gives C' and C to A_2, and outputs the bit output by A_2. Observe that A_3 runs in time at most $4 \cdot time_n(A_2)$.

Notice that if A_3 guesses the leakage $\hat{\lambda}_2$ correctly then it simulates A_2 perfectly, and that the leakage is guessed correctly with probability $1/2^{l(n)}$. We therefore conclude:

$$\begin{aligned}&|\Pr[(A_3 \leftrightarrows Exp_3) = 1 | b = 0] - \Pr[(A_3 \leftrightarrows Exp_3) = 1 | b = 1]| \geq \\ &\frac{1}{2^{l(n)}} |\Pr[(A_2 \leftrightarrows Exp_2) = 1 | b = 0] - \Pr[(A_2 \leftrightarrows Exp_2) = 1 | b = 1]|\end{aligned}$$

$\square$

We again simplify the experiment, this time moving to a leakage-free setting.

Experiment Exp_4**.** Exp_4 proceeds identically to Exp_3, except step 3 is omitted.

Claim 2.3.8 *For all functions $\varepsilon'(n) > 0$ and $\varepsilon(n) > 0$, and for every adversary A_3 that succeeds in Exp_3 with advantage $\varepsilon'(n)$ for infinitely many n, there exists an adversary A_4 that runs in time at most $\frac{2^{3l(n)+1}}{\varepsilon(n)^2}(3l(n) + 4 + \log \frac{1}{\varepsilon(n)})(time_n(A_3) + time_n(\mathsf{Enc}))$ and succeeds in Exp_4 with advantage $\varepsilon'(n) - 6\varepsilon(n)$ for infinitely many n.*

Proof The key observation is that the response to A_3's leakage query is independent of bit b and the randomness used when producing encryption $C = \mathsf{Enc}_{pub}(m_b)$. This allows us to use the observation of Akavia *et al* [AGV09] that for every public-key encryption system, every adversary that breaks semantic security given leakage on KeyGen can be simulated by an adversary that is not given leakage but instead guesses the leakage and then tests whether the guessed leakage is good. Specifically, given pub, pub', and $C' = \mathsf{Enc}_{pub'}(pri)$, we can find a good response $\hat{\lambda}_1 \in \{0,1\}^{3l(n)+1}$ to A_3's leakage query that (almost) maximizes the distinguishing advantage of A_3 conditioned on pri, pri', and C'. To do so, we define an adversary A_4 that tests all strings $\hat{\lambda}_1 \in \{0,1\}^{l(n)}$ until it finds a leakage value that maximizes the gap between A_3's probability of outputting 1 on an encryption of m_0, and on an encryption of m_1. This is done by sampling, for each value $\hat{\lambda}_1$, many encryptions of m_0 and of m_1, and recording A_3's output. The details follow.

Without loss of generality, suppose

$$\Pr\left[(A_3 \leftrightarrows Exp_3) = 1 | b = 1\right] - \Pr\left[(A_3 \leftrightarrows Exp_3) = 1 | b = 0\right] \geq \varepsilon'(n)$$

for infinitely many n. A_4 behaves as follows. A_4 randomly selects a string r_{sim} to use as the randomness of A_3. Then, using r_{sim}, A_4 starts simulating A_3. A_3 submits messages m_0 and m_1, which are in turn submitted by A_4. Then, A_4 is given (pub, pub', C', C), where $C' = \mathsf{Enc}_{pub'}(pri, r_{pri'})$ and $C = \mathsf{Enc}_{pub}(m_b)$. A_4 continues simulating A_3, giving it pub and pub', and obtaining a leakage function leak_1. Recall from Exp_3 that leak_1 takes input $(pri, pri', r_{pri'})$. This means that the correct response to this leakage query is independent of bit b and the randomness used when producing C. Since A_4 cannot make any leakage queries, it runs experiments in order to determine the response that (almost) maximizes the distinguishing advantage of A_3 (conditioned on r_{sim}, pri, pri', and $r_{pri'}$).

Specifically, for each $\hat{\lambda}_1 \in \{0,1\}^{3l(n)+1}$ and for each $\hat{b} \in \{0,1\}$, A_4 does the following $m = \frac{1}{2\varepsilon(n)^2}(3l(n) + 3 + \log \frac{1}{\varepsilon(n)})$ times: A_4 produces a random encryption $C'' = \mathsf{Enc}_{pub}(m_{\hat{b}})$, runs A_3 with randomness r_{sim} on $(pub, pub', \hat{\lambda}_1, C', C'')$, and notes the output of A_3. This allows A_4 to obtain an estimate $p_{\hat{\lambda}_1, \hat{b}}$ of the probability that A_3 outputs 1 conditioned on r_{sim}, pri,

pri', $r_{pri'}$, $\hat{\lambda}_1$, and $\hat{b}$. Then, for each $\hat{\lambda}_1 \in \{0,1\}^{3l(n)+1}$, A_4 computes $\varepsilon_{\hat{\lambda}_1} = p_{\hat{\lambda}_1,1} - p_{\hat{\lambda}_1,0}$ to obtain an estimate of the distinguishing advantage of A_3 (conditioned on r_{sim}, pri, pri', and $r_{pri'}$) when $\hat{\lambda}_1$ is used as the response to the leakage query.

A_4 then continues its original simulation of A_3, letting the response to the leakage query be the $\hat{\lambda}_1$ which maximizes $\varepsilon_{\hat{\lambda}_1}$. A_4 then gives C' and C to A_3, and outputs the bit output by A_3. Observe that A_4 runs in time $time_n(A_3) + m(time_n(A_3) + time_n(\mathsf{Enc})) \leq \frac{2^{3l(n)+1}}{\varepsilon(n)^2}(3l(n) + 4 + \log \frac{1}{\varepsilon(n)})(time_n(A_3) + time_n(\mathsf{Enc}))$.

Now, fix an n such that $\Pr[(A_3 \leftrightarrows Exp_3) = 1 | b = 1] - \Pr[(A_3 \leftrightarrows Exp_3) = 1 | b = 0] \geq \varepsilon'(n)$ and consider the advantage A_4 in Exp_4 for such n. Note that A_4 produces $2(2^{3l(n)+1})$ estimates. Using Chernoff bounds, each estimate $p_{\hat{\lambda}_1,\hat{b}}$ is within additive error $\varepsilon(n)$ of its true value with probability at least $1 - 2e^{-2m\varepsilon(n)^2} = 1 - 2e^{-3l(n)-3-\log\frac{1}{\varepsilon}} \geq 1 - 2^{-3l(n)-2-\log\frac{1}{\varepsilon}} = 1 - \frac{\varepsilon(n)}{2^{3l(n)+2}}$. Then, by the union bound, *all* estimates $p_{\hat{\lambda}_1,\hat{b}}$ are within $\varepsilon(n)$ of their true values with probability at least $1 - \varepsilon(n)$. Observe that when this happens, all estimates $\varepsilon_{\hat{\lambda}_1}$ are within $2\varepsilon(n)$ of their true values. In this case, the $\hat{\lambda}_1$ chosen by A_4 results in A_3 having true distinguishing advantage within $4\varepsilon(n)$ of whichever response yields the best true distinguishing advantage conditioned on r_{sim}, pri, pri', and $r_{pri'}$

Putting this all together, conditioned on each r_{sim}, pri, pri', and $r_{pri'}$, we have that with probability at least $1 - \varepsilon(n)$, A_4 has distinguishing advantage within $4\varepsilon(n)$ of the distinguishing advantage of A_3 (subject to the same conditioning). It follows that overall (without conditioning), with probability at least $1 - \varepsilon(n)$, A_4 has distinguishing advantage within $4\varepsilon(n)$ of the distinguishing advantage of A_3. That is, A_4 has distinguishing advantage at least $(1 - \varepsilon(n))(\varepsilon'(n) - 4\varepsilon(n)) - \varepsilon(n) \geq \varepsilon'(n) - 6\varepsilon(n)$. $\square$

It is easy to see that an adversary that succeeds in experiment Exp_4 can be used to break the semantic security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The idea is that such an adversary must either distinguish $\mathsf{Enc}_{pub'}(pri)$ from $\mathsf{Enc}_{pub'}(\bar{0})$, or must succeed at guessing b even when given $\mathsf{Enc}_{pub'}(\bar{0})$ instead of $\mathsf{Enc}_{pub'}(pri)$.

Claim 2.3.9 *For every function $\varepsilon(n) > 0$ and for every adversary A_4 that succeeds in Exp_4 with advantage $\varepsilon(n)$ for infinitely many n, there exists an adversary A_5 that runs in time at most $time_n(A_4) + time_n(\mathsf{KeyGen}) + time_n(\mathsf{Enc})$ and breaks the semantic security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with advantage $\varepsilon(n)/3$ for infinitely many n.*

Proof Let experiment Exp_5 be identical to Exp_4 except C' is set to $\mathsf{Enc}_{pub'}(\bar{0})$ instead of $\mathsf{Enc}_{pub'}(pri)$. There are two cases to consider:

Case 1: For infinitely many n, A_4 has advantage at least $\varepsilon(n)$ in Exp_4 and has advantage at least $\varepsilon(n)/3$ in Exp_5.

Let A_5 behave as follows. A_5 starts simulating A_4. A_4 submits messages m_0 and m_1, which are in turn submitted by A_5. Then, A_5 is given (pub, C), where $C = \mathsf{Enc}_{pub}(m_b)$. A_5 randomly selects pri', and lets $pub' = \mathsf{KeyGen}(pri')$. Then A_5 produces $C' = \mathsf{Enc}_{pub'}(\bar{0})$, gives (pub, pub', C', C) to A_4, and outputs the bit output by A_4.

Notice that A_5 simulates $A_4 \leftrightarrows Exp_5$ perfectly, and hence has the same distinguishing advantage as A_4. That is, A_5 breaks the semantic security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with advantage $\varepsilon(n)/3$ for infinitely many n. Observe that A_5 runs in time $time_n(A_4) + time_n(\mathsf{KeyGen}) + time_n(\mathsf{Enc})$.

Case 2: For infinitely many n, A_4 has advantage at least $\varepsilon(n)$ is Exp_4 and has advantage less than $\varepsilon(n)/3$ in Exp_5. Without loss of generality, suppose

$$\Pr[(A_4 \leftrightarrows Exp_4) = 1|b = 1] - \Pr[(A_4 \leftrightarrows Exp_4) = 1|b = 0] \geq \varepsilon(n) \tag{2.1}$$

and

$$\Pr[(A_4 \leftrightarrows Exp_5) = 1|b = 1] - \Pr[(A_4 \leftrightarrows Exp_5) = 1|b = 0] < \frac{\varepsilon(n)}{3} \tag{2.2}$$

for infinitely many n. Let A_5 behave as follows. A_5 randomly selects pri, and lets $pub = \mathsf{KeyGen}(pri)$. A_5 submits $m'_0 = \bar{0}$ and $m'_1 = pri$. Then, A_5 is given (pub', C'), where $C' = \mathsf{Enc}_{pub'}(m'_{b'})$ for some $b' \in \{0, 1\}$. Now, A_5 starts A_5 starts simulating A_4. A_4 submits messages m_0 and m_1. A_5 randomly selects $b \in \{0, 1\}$, produces $C = \mathsf{Enc}_{pub}(m_b)$, and gives (pub, pub', C', C) to A_4. Then, A_4 outputs a bit. If this bit is b, A_5 outputs 1, and otherwise A_5 outputs 0.

Now, fix an n such that (2.1) and (2.2) both hold, and consider the advantage of A_5 in breaking the semantic security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The key observation is that when $b' = 0$, A_5 simulates $A_4 \leftrightarrows Exp_5$ perfectly, and when $b' = 1$, A_5 simulates $A_4 \leftrightarrows Exp_4$ perfectly. Then we have

$$\begin{aligned}
\Pr[A_5 \text{ outputs } 1|b' = 0] &= \frac{1}{2}\Pr[(A_4 \leftrightarrows Exp_5) = 1|b = 1] + \frac{1}{2}\Pr[(A_4 \leftrightarrows Exp_5) = 0|b = 0] \\
&= \frac{1}{2}\left(\Pr\left[(A_4 \leftrightarrows Exp_5) = 1|b = 1\right] + 1 - \Pr\left[(A_4 \leftrightarrows Exp_5) = 1|b = 0\right]\right) \\
&< \frac{1}{2}\left(1 + \frac{\varepsilon(n)}{3}\right)
\end{aligned}$$

where the inequality is by (2.2). We also have

$$\begin{aligned}
\Pr[A_5 \text{ outputs } 1|b' = 1] &= \frac{1}{2}\Pr[(A_4 \leftrightarrows Exp_4) = 1|b = 1] + \frac{1}{2}\Pr[(A_4 \leftrightarrows Exp_4) = 0|b = 0] \\
&= \frac{1}{2}\left(\Pr\left[(A_4 \leftrightarrows Exp_4) = 1|b = 1\right] + 1 - \Pr\left[(A_4 \leftrightarrows Exp_4) = 1|b = 0\right]\right) \\
&\geq \frac{1}{2}\left(1 + \varepsilon(n)\right)
\end{aligned}$$

where the inequality is by (2.1). But this means that

$$\Pr[A_5 \text{ outputs } 1|b' = 1] - \Pr[A_5 \text{ outputs } 1|b' = 0] > \frac{1}{2}(1 + \varepsilon(n)) - \frac{1}{2}\left(1 + \frac{\varepsilon(n)}{3}\right) = \frac{\varepsilon(n)}{3}$$

That is, A_5 breaks the semantic security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with advantage $\varepsilon(n)/3$ for infinitely many n. Observe that A_5 runs in time $time_n(A_4) + time_n(\mathsf{KeyGen}) + time_n(\mathsf{Enc})$. $\square$

Combining all the claims, we see that for all functions $\varepsilon(n) > 0$ if there exists an adversary A such that $|\Pr[(A \leftrightarrows \mathsf{Hyb}_{12}^{0}) = 1] - \Pr[(A \leftrightarrows \mathsf{Hyb}_{12}^{n^d+1}) = 1]| > \varepsilon(n)$ for infinitely many n, then for every function $\varepsilon'(n) > 0$ there exists an adversary A' that runs in time

$$\frac{2^{3l(n)+7}}{\varepsilon'(n)^2}\left(3l(n) + 4 + \log\frac{1}{\varepsilon'(n)}\right)(time_n(LRKP \leftrightarrow A))$$

and breaks the semantic security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with advantage $\frac{\varepsilon(n)}{3 \cdot 2^{2l(n)}(n^d+1)} - 2\varepsilon'(n)$ for infinitely many n.

2.4 Extensions and Applications

Below we describe some variants and applications of our scheme.

Resilience against simultaneous leakage. In Definition 7, the adversary is only allowed to see leakage from the part of memory where computation is occurring. Our construction is also secure under an alternative leakage model where the adversary is allowed to see *independent* leakage from *both* parts of memory each time it makes a leakage query. The basic idea is to first show that our construction is secure under a variant of Definition 7 where the adversary sees an additional leakage f_4 on memory B. Under this variant of Definition 7, the adversary's leakage queries strictly alternate between memory A and memory B. We then use an observation of Pietrzak [Pie09] that simultaneous but independent leakage on two pieces of memory can be perfectly simulated by strictly alternating leakage (of twice the output length) on these two pieces of memory.

Resilience against complete compromise. Our scheme can be viewed as a protocol between two devices that communicate over a public channel. The key remains hidden even if the memory contents of one of the devices are leaked completely (for example, in a cold boot attack), provided that the compromise is detected and no further computation is performed using the counterpart device.

One-time programs. Our construction can be modified to work without any leak-free components by pre-computing a large number of tuples of the form (pri, pub, C, C') where C and

C' are encryptions of 0 under *pub*, and storing the tuples in memory. Then, at each invocation, one such tuple is used (first *pri* and *pub* are used by EvalA_1, and then C, C' are used by EvalB). Assuming that only computation leaks information, the remaining tuples remain hidden until they are accessed. Therefore, security is obtained following essentially the same argument as the proof of Theorem 2.3.1. The number of invocations in this case is bounded by the number of pre-computed tuples. This approach provides a weaker security guarantee than the one time programs of [GKR08] (i.e. only security against leakage), but has the advantage that the pre-computing phase is independent from the functionality that is being protected.

Concurrent composition. We have shown that an adversary interacting with a single instance of our construction gains no information about the underlying key. However, for some applications, such as private-key encryption where several parties compute on the same agreed upon key, this may not suffice. It is quite possible that the adversary is performing side-channel attacks on several parties simultaneously, and is coordinating his leakage functions adaptively. In Section 2.4.1, we show that an adversary interacting concurrently with several instances of our construction still gains no information through leakage.

Leakage-resilient private-key encryption. Extending the traditional notions of semantically secure encryption to the leakage setting is non-trivial. In particular, suppose that every invocation of the encryption algorithm leaks information. Then, since the plaintext of the adversary's challenge message is an input to the encryption algorithm, the adversary can trivially break semantic security by leaking even a single bit about this message. To deal with this problem, several works [DP08, NS09, DKL09, DGK$^+$10] adopt the approach that the computation of the encryption of the challenge is *not* allowed to leak. In Sections 2.4.2 and 2.4.3, we follow this approach and show how to obtain semantically-secure private-key encryption in the leakage setting using LRKPs.

Leakage-resilient public-key encryption. Constructions of leakage-resilient public-key encryption schemes were recently given by [AGV09, NS09, DGK$^+$10, BKKV10]. However, no constructions are known of PKE schemes that are leakage-resilient under Chosen Ciphertext Attack (CCA), where the adversary can obtain leakage during each decryption query *even after* receiving the challenge. LRKPs provide a convenient way to achieve such a construction. Specifically, given a CCA-PKE scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, we construct a new PKE scheme $(\mathsf{KeyGen}', \mathsf{Enc}, \mathsf{Dec}')$ where the encryption algorithm stays the same; the key generation KeyGen' runs KeyGen to obtain (pub, pri) and then initializes an LRKP with *pri*. The public key is *pub*, and the private key is the initial state state_1 of the LRKP. The decryption algorithm is stateful, and to decrypt a ciphertext C, Dec' generates a circuit $H(x)$ that computes that function $\mathsf{Dec}_x(C)$, and then uses KPEval to evaluate it on the private key *pri*.

2.4.1 Concurrent Composition

In this section we show that an adversary interacting with several instances of LRKP concurrently still gains no information through leakage. This allows us to obtain some of the applications described in Section 2.4. We start with a definition.

Definition 9 Let A and S be PPT algorithms, let $n \in \mathbb{N}$, and consider the following two experiments:

$\mathsf{ExpConcurrentReal}$. The adversary chooses n keys $K_1, \ldots, K_n$ and interacts with n instances of $\mathsf{ExpReal}$ where in instance i, K_i is protected by LRKP_i. At the end, the adversary outputs a bit b. During the interaction the adversary controls the schedule of the queries completely, and in particular leakage queries on LRKP_i may depend on leakage obtained from LRKP_j for $i \neq j$.

$\mathsf{ExpConcurrentIdeal}$. The adversary chooses n keys $K_1, \ldots, K_n$, interacts with a single simulator S, and eventually outputs a bit b.

Then, we say that LRKP is a *Concurrent-Leakage-Resilient Key Proxy (C-LRKP)* if for every PPT A there exists a PPT S and a negligible function $neg(\cdot)$ such that

$$|\Pr[(A \leftrightarrows \mathsf{ExpConcurrentReal}) = 1] - \Pr[(A \leftrightarrows \mathsf{ExpConcurrentIdeal}) = 1]| \leq neg(n)$$

We now show that every LRKP is also concurrent-LRKP. This follows due to the strong security guarantee of LRKP: even when the adversary himself selects the key K, he still cannot distinguish between simulated and actual leakage.

Theorem 2.4.1 *Let* LRKP *be a leakage-resilient key proxy. Then,* LRKP *is also concurrent-leakage-resilient.*

Proof Suppose that LRKP is insecure according to Definition 9. Then, there exists an adversary A such that for every simulator, A distinguishes between $\mathsf{ExpConcurrentReal}$ and $\mathsf{ExpConcurrentIdeal}$ with non-negligible advantage $\varepsilon(n)$. Let S be the simulator of LRKP in the non-concurrent setting, and consider a simulator S' that runs n copies of S in parallel. Copy i is used to simulate leakage from LRKP_i.

Consider a sequence of hybrid experiments Hyb_i, $0 \leq i \leq n$ such that in Hyb_i the adversary obtains leakage from the actual state of LRKP_j for $1 \leq j \leq i$, and obtains simulated leakage for $i+1 \leq j \leq n$. Note that Hyb_0 is $\mathsf{ExpConcurrentIdeal}$ and Hyb_n is $\mathsf{ExpConcurrentReal}$.

We now construct an adversary A' that simulates A and breaks the non-concurrent security of LRKP. The simulation proceeds as follows: A' randomly selects $0 \leq i < n$. Then, A' starts simulating A, which chooses n keys $K_1, \ldots, K_n$. A' then initializes LRKP_j for $1 \leq j \leq i-1$ with K_j, submits K_i as its own key in the LRKP security experiment, and chooses the initial randomness independently for $n-i$ copies of S.

A' then continues simulating A, answering queries as follows: leakage queries about LRKP_j for $1 \leq j \leq i-1$ are answered by applying the leakage function to the actual state of LRKP_j. For $i+1 \leq j \leq n$ the leakage queries are forwarded to the $j-i$th copy of S. Leakage queries for LRKP_i are forwarded by A' as his own queries. At the end of the simulation A' outputs what A outputs. It is not hard to see that when A' is interacting with $\mathsf{ExpReal}$ and $\mathsf{ExpIdeal}$ it is simulating A perfectly in Hyb_i and Hyb_{i+1} respectively.

It follows by a standard calculation that A' distinguishes $\mathsf{ExpReal}$ from $\mathsf{ExpIdeal}$ with advantage $\varepsilon(n)/n$. □

2.4.2 Semantic Security Under Leakage

Encryption is one of the most important products of cryptography. In the classical setting, where side-channel attacks are not taken into account, there are widely accepted definitions of security for both the private- and the public-key setting. For a rigorous exposition, we direct the reader to [Gol04, KL07]. Informally, the accepted notion of privacy, which is commonly referred to as "semantic security", is to require that no efficient adversary can distinguish between the encryptions of two messages of his choice.

Extending the traditional notions of semantic security to the leakage setting is non-trivial. In particular, suppose that we assume that every invocation of the encryption algorithm leaks information. Then, since the message plaintext is an input to that algorithm, the adversary can trivially break semantic security by simply leaking a bit that differentiates the two messages in question. Consequently, in the setting where "everything leaks", traditional semantic security cannot be achieved. This leads us to consider several alternatives to the naive definition, which permit non-trivial results. Below, we outline some of the possible approaches for dealing with privacy under leakage, and describe the choice that we made.

Leak-free challenge. One approach to dealing with the trivial impossibility described above is to weaken the requirement that "everything leaks" to allowing everything to leak *except* the computation of the actual ciphertext that the adversary is trying to distinguish. This solution has been adopted by several works on leakage-resilient encryption (see e.g. [DP08, NS09, DKL09, DGK$^+$10]). These works all deal with what we call "bounded leakage", that is, the amount of information that the adversary obtains on the key during its entire lifetime is bounded. Still, the issue that we mentioned about semantic security applies, but for a different reason. In most constructions in the bounded leakage model, the key remains fixed after it is generated; such constructions are clearly insecure in the "everything leaks" model since the entire key eventually leaks. In the bounded leakage model, such constructions turn out to be insecure when there is leakage after the challenge ciphertext is obtained.

The problem is that if the adversary is allowed to obtain leakage on the key after he has seen the challenge ciphertext, he can simply use the key to decrypt the ciphertext within the leakage function, and leak the information that distinguishes the two messages in question, thereby breaking semantic security. Consequently, as in our setting, some restrictions on the leakage, or a weakening of the definition of security are necessary.

We adopt the leak-free challenge approach for our applications. We prefer this solution to the one listed below because it permits fairly clean definitions while allowing other notions to be achieved through simple transformations and reductions.

Leakage on random messages. Instead of weakening the requirement that everything leaks, we can relax the definition of semantic security so that it is still meaningful in the leakage setting. Instead of requiring that the adversary fails to distinguish between the encryption of two messages, we can require that he does not learn too much about a message, as long as it is sampled from a distribution with a sufficiently high min-entropy. That is, we can ask for essentially the best that can be hoped for: that the adversary obtains no more information through leakage on the encryption process than what he would be able to obtain through leakage only on the message that is being encrypted. This notion of security seems to capture accurately what is achievable in terms of privacy in a setting with leakage. However, we choose not to adopt this notion both because it is more cumbersome than assuming a leak free challenge, and, more importantly, because it does not seem to be easily usable in applications which require semantically secure private-key encryption as an underlying tool.

2.4.3 Leakage-Resilient Private-Key Encryption Using Key Proxies

We extend the standard definition of semantic security to the leakage setting with a leak-free challenge. One issue that arises in the private-key setting is that in a typical application, several parties will hold the same key K which is used both for encryption and decryption. In order to maintain generality, it is therefore important to allow a leakage adversary to obtain leakage on each of the parties according to his own schedule. With this in mind, we now define a leakage-resilient private-key encryption scheme.

A stateful private-key encryption scheme consists of three PPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm $\mathsf{KeyGen}(1^n)$ outputs n initial states $S_1^0, \ldots, S_n^0$ that are held by n

individual parties. These states correspond to the initial encodings of some key K. For $j \in \mathbb{N}$, the encryption algorithm $\mathsf{Enc}(M, S_i^j)$ outputs a ciphertext C, and an updated state S_i^{j+1}. The decryption algorithm $\mathsf{Dec}(C, S_i^j)$ outputs a message M, and an updated state S_i^{j+1}.

Definition 10 A triple of PPT algorithms ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$) is a correct *stateful private-key encryption scheme* if for all random tapes R, for all $1 \leq i, i' \leq n$, all $j, j' \in \mathbb{N}$, and all $M \in \{0,1\}^n$, $\mathsf{Dec}(\mathsf{Enc}(M, S_i^j; R), S_{i'}^{j'}) = M$.

We can now describe the experiment *ExpSemSec*(b), for $b \in \{0,1\}$, of semantic security under leakage.

1. *Initialization.* The key generation algorithm $\mathsf{KeyGen}(1^n)$ is run to obtain $S_1^0, \ldots, S_n^0$.

2. *Encryption Queries.* The adversary may initiate an arbitrary number of encryption processes by submitting a message M, and an index $1 \leq i \leq n$. An encryption $C = \mathsf{Enc}(M, S_i^j)$ is then computed, where j is the number of times party i encrypted until now, and the adversary concurrently obtains single invocation leakage on all the active encryption processes (the single invocation leakage model for private-key encryption is described later in this section). The adversary is then given C.

3. *Challenge.* At some point the adversary submits two messages M_0, M_1, and an index $1 \leq i \leq n$ for which there is no current active encryption process, and obtains $C^* = \mathsf{Enc}(M_b, S_i^j)$.

4. *Encryption Queries.* The adversary continues to initiate encryption processes, and concurrently obtain leakage on these processes.

5. *Guess.* The adversary outputs a bit b'.

Definition 11 A stateful private-key encryption scheme ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$) is *semantically secure under leakage*, if for all PPT adversaries A,

$$|\Pr[(A \leftrightarrows \mathit{ExpSemSec}(0)) = 1] - \Pr[(A \leftrightarrows \mathit{ExpSemSec}(1)) = 1]| \leq \mathit{neg}(n)$$

Construction

Let $F = \{F_n\}_{n \in \mathbb{N}}$ be a family of pseudo-random functions (when n is clear from context we write F instead of F_n) such that $F_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ for all $n \in \mathbb{N}$. Let LRKP be a leakage-resilient key proxy. Our stateful private-key encryption scheme $\mathsf{PRI} - \mathsf{ENC} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ works as follows:

Key Generation. The key generation algorithm, $\mathsf{KeyGen}(1^n)$, first chooses a key $K \in_R \{0,1\}^n$ at random, and runs $\mathsf{KPInit}(1^n, K)$ n times with independently chosen randomness to obtain n initial states $S_1, \ldots, S_n$. The states $S_1, \ldots, S_n$ are the output of KeyGen.

Encryption. For a message $M \in \{0,1\}^n$, the encryption algorithm $\mathsf{Enc}(M, S)$ chooses a random string $R \in_R \{0,1\}^n$, and generates a circuit $H(x)$ that computes the function $F_x(R) \oplus M$. It then runs $\mathsf{KPEval}(H, S)$ to obtain an output Y, and an updated state S'. The ciphertext is then $C = (Y, R)$, and output of Enc is (C, S').

Decryption. The decryption algorithm $\mathsf{Dec}(C, S)$ parses C as (Y, R), and generates a circuit $G(x)$ that computes the function $F_x(R) \oplus Y$. It then runs $\mathsf{KPEval}(G, S)$ to obtain an output M, and an updated state S'. The output of Dec is then (M, S').

Single Invocation Leakage Model

The single invocation leakage for our construction is quite simple: during encryption, the adversary is given R (and already knows M since he chose this himself), and then proceeds to interact in a single invocation leakage experiment with the computation of KPEval. During decryption, the adversary simply interacts in a single invocation leakage experiment with KPEval. In other words, for both encryption and decryption, the adversary obtains leakage on the computation of KPEval, and obtains all the other inputs completely.

We note that although it may seem more reasonable to allow the adversary to learn only part of the randomness and message of the encryption, it would give us a weaker theorem. The fact that our construction is secure in the above leakage model implies security in other more realistic but weaker models.

Security Analysis

We show that any adversary that breaks the semantic security of $\mathsf{PRI}-\mathsf{ENC}$ can be used to break either the concurrent leakage resilience of LRKP or the pseudo-randomness of F. We start by stating the theorem:

Theorem 2.4.2 *Let A be a PPT adversary for the semantic security under leakage of* $\mathsf{PRI}-\mathsf{ENC}$*, let* LRKP *be a concurrently leakage-resilient key proxy such that all adversaries running in time at most $time_n(A)$ can distinguish real and simulated leakage with advantage at most $\varepsilon_{\mathsf{c-lrkp}}(n)$, and let $F = \{F_n\}_{n\in\mathbb{N}}$ be a family of PRFs such that all adversaries running in time at most $time_n(A) \cdot (time_n(\mathsf{KPEval}) + time_n(\mathsf{Enc})) + time_n(\mathsf{KeyGen})$ can distinguish F_n from random with advantage at most $\varepsilon_{\mathsf{prf}}(n)$. Then, A breaks the semantic security of* $\mathsf{PRI}-\mathsf{ENC}$ *with advantage at most $\varepsilon_{\mathsf{c-lrkp}}(n) + \varepsilon_{\mathsf{prf}}(n) + \frac{time_n(A)}{2^n}$.*

To prove security we define several hybrid experiments where the first hybrid Hyb_0 is the original experiment of semantic security with leakage, and in the final hybrid the adversary obtains no information about the bit b.

Experiment Hyb_1. Experiment Hyb_1 proceeds as Hyb_0 except that the computation of the challenge is performed differently. Instead of using KPEval to compute it, the challenge ciphertext is computed directly by choosing a random string $R^* \in_{\mathrm{R}} \{0,1\}^n$ and outputting $(F_K(R^*) \oplus M_b, R^*)$. The LRKP is evaluated on some constant function (e.g. one that always outputs $\bar{0}$) in order to refresh its state.

Experiment Hyb_2. In this experiment, the leakage obtained by the adversary is replaced by simulated leakage. More precisely, let S be the simulator for concurrent leakage that is guaranteed by Theorem 2.4.1 to exist for LRKP. In experiment Hyb_2, whenever the adversary initiates an encryption process, the simulator S is given the corresponding circuit H, and the ciphertext C. Then, the adversary interacts with the simulator to obtain leakage on the underlying invocation of LRKP.

Experiment Hyb_3. In this experiment we replace the pseudo-random function F with a random one. Namely, for each new encryption process started by the adversary, the simulator S is given the circuit H, and a ciphertext of the form $C = (\hat{F}(R) \oplus M, R)$ where $\hat{F} : \{0,1\}^n \to \{0,1\}^n$ is a random function. The challenge ciphertext is also computed using the random function $\hat{F}$.

Let A be PPT adversary for the semantic security under leakage of $\mathsf{PRI} - \mathsf{ENC}$. We define X_i to be the random variable that is 1 if A guesses the bit b correctly in experiment Hyb_i, for $0 \leq i \leq 3$.

Claim 2.4.3 $\Pr[X_0 = 1] = \Pr[X_1 = 1]$

Proof sketch This follows directly from the fact that LRKP is history free according to Definition 8. In particular, it makes no difference whether we refresh the state of LRKP during the challenge by evaluating the actual circuit that is needed to compute the challenge, or a circuit that always outputs $\bar{0}$. □

Claim 2.4.4 $|\Pr[X_1 = 1] - \Pr[X_2 = 1]| \leq \varepsilon_{\mathsf{c-lrkp}}(n)$

Proof sketch Suppose that the claim is false. Then we can use the adversary A to distinguish between simulated and real leakage in the concurrent LRKP experiment: Our adversary A' initializes n copies of LRKPs with some random PRF key K, and then simply acts as a middleman between A and the security experiment of the concurrent LRKP. □

Claim 2.4.5 $|\Pr[X_2 = 1] - \Pr[X_3 = 1]| \leq \varepsilon_{\mathsf{prf}}(n)$

Proof sketch Suppose that the claim is false. Then we can use the adversary A to distinguish between an oracle for F_K and an oracle for a random function $\hat{F}$. To see this, note that in both Hyb_2 and Hyb_3, the leakage is simulated. Therefore, we can construct an adversary A' that, given an oracle O (that is either a pseudo-randomly chosen function or a randomly chosen function) simulates A using this oracle. If O is F_K for randomly chosen K, then A is simulated perfectly in Hyb_2. If O is a random function, then A is perfectly simulated in Hyb_3. □

Claim 2.4.6 $\Pr[X_3 = 1] \leq \frac{1}{2} + \frac{time_n(A)}{2^n}$

Proof sketch This follows from the fact that $\hat{F}$ is a random function, and that the simulated leakage is independent from the randomness of the challenge. □

2.5 Open problems

The necessity of the leak-free component Recall that our construction uses the "only computation leaks" assumption *and* a leak-free component. However, as noted earlier, specific leakage-resilient cryptographic primitives have been constructed under leakage models where the *entire* state (and not just the active part) leaks continuously [DHLAW10, BKKV10, MTVY11], without the use of any leak-free component. It is easy to see that no key proxy can be leakage-resilient under a leakage model where the entire state leaks, even if a (memory-less) leak-free component is used – with leakage on the entire state, the adversary can learn arbitrary bits of the protected key K via his leakage queries. On the other hand, it is not clear that a leak-free component is necessary in order to obtain a key proxy under the "only computation leaks" leakage model. We have worked on removing this component from our construction, but have found this to be a difficult problem. Can we provide evidence that it is necessary for constructions to use such a component? Faust *et al* [FRR$^+$10] have provided such evidence for their setting, showing that with respect to the proof technique they use, a proof of security for a construction without leak-free components yields a proof that $\mathsf{AC} = \mathsf{P/poly}$. To provide such evidence in our setting, we need to restrict our attention to some "reasonable" leakage model (such as the one we use) and some "reasonable" class of proof techniques, and show that for every construction of a key proxy that does *not* use leak-free components, a proof of leakage resilience has consequences that are believed to be false.

Securing the output of leakage-resilient key proxies Our definition of a leakage-resilient key proxy places no security requirement on the output of each query to the evaluation procedure. Suppose we want to protect such outputs from leakage and compute on them in a leakage-resilient way. Can we modify our definition and construction so that we can ask for outputs to be produced in the form of an initialized leakage-resilient key proxy? One issue to consider is *how* such a key proxy should be output. If the key proxy consists of two pieces of memory (as our construction does), outputting these pieces together may trivially break the key proxy's leakage resilience. Alternative approaches include producing the output piece-by-piece (which allows for the first piece of output to be moved elsewhere before the second piece is produced) or directly producing the pieces of output in separate pieces of memory.

Chapter 3

Leakage-resilient authentication

When two parties wish to send messages to each other securely, the two security properties that must be achieved are *privacy* – ensuring that no adversary learns anything at all about the contents of the messages – and *authentication* – ensuring that the purported sender of each message did indeed produce the message. Authentication is arguably the more important property, since in most settings, an adversary that succeeds in modifying messages can cause far more damage than one who simply learns about message contents.

In this chapter, we construct a shared-private-key authenticated session protocol that is resilient to leakage on both parties. Unlike our construction of leakage-resilient key proxies in the previous chapter, we do not use the "only computation leaks" assumption, nor do we use any leakage-free hardware. Instead, leakage occurs on the *entire* state, inputs, and randomness of the party performing the computation. The only assumption we make is the existence of pseudo-random generators; consequently, our construction is much more practical than the construction in the previous chapter, where we assumed the existence of fully-homomorphic public-key encryption. Finally, our construction has the feature that all randomness used by each party is made public; furthermore, this randomness can be chosen according a high min-entropy distribution instead of the uniform distribution.

Stream ciphers and authentication

A *stream cipher* is an object that takes a randomly-chosen seed as input, and outputs a pseudo-random sequence of strings (where the length of the sequence is unbounded). Intuitively, a stream cipher can be viewed as a pseudo-random function generator where the adversary must examine the bits of the generated function sequentially rather than having random access to these bits; that is, the adversary chooses only *how many* bits he wishes to see rather than *which* bits he wishes to see.

Dziembowski and Pietrzak [DP08] and Pietrzak [Pie09] construct leakage-resilient stream ciphers in the *only-computation-leaks* model. Their constructions use two pieces of memory connected by a public channel, and computation alternates between the two pieces. Specifically, for all $i \geq 1$, the $(2i-1)$-st string output by the stream cipher is produced by the first piece of memory, and the $2i$-th string output by the stream cipher is produced by the second piece of memory. Security for these constructions is defined with respect to an adversary who sees as many pieces of output as he likes along with leakage on each of the computations producing these outputs (where the adversary chooses leakage queries adaptively). The adversary is then given either the next output or a uniformly chosen string, and should not be able to tell which he is given. Note that the adversary is *not* given leakage for the computation producing this output, since this would make it trivial for him to distinguish; an alternative notion of security (also satisfied by these constructions) allows the adversary to see such leakage, and asserts the output is indistinguishable from a distribution of high min-entropy.

Consider using a leakage-resilient stream cipher to obtain a leakage-resilient version of shared-private-key authenticated sessions. We have two parties, A and B, where A is sending message pieces to B, and we wish to ensure that an adversary cannot reorder these pieces or insert message pieces of his own without this being detected by B. The adversary obtains leakage from both parties. An immediate problem we encounter is that the definition of security for leakage-resilient stream ciphers tells us nothing about what happens when two parties A and B share a randomly-chosen string K and both use K as the seed of the stream cipher. Indeed, we observe that for both constructions [DP08, Pie09] of leakage-resilient stream ciphers, an adversary that can cause parties A and B to "get out of sync" (meaning, for example, that party A uses the stream cipher to produce several outputs before party B gets started) can eventually learn the entire state of the stream cipher. This suggests we need a way to (almost) synchronize the stream cipher computation performed by the two parties.

Our construction

We begin by modifying Pietrzak's stream cipher construction so that it uses a *single* piece of memory along with a source of *public min-entropy.* By "public min-entropy", we mean strings that are chosen according to distribution of high min-entropy but not kept secret (that is, the adversary is given such strings in their entirety). We have in mind that in applications involving two parties, one or both parties will produce this high min-entropy string and communicate it to the other party over a public channel.

Our stream cipher uses a pseudo-random function generator $F_s : \{0,1\}^n \to \{0,1\}^{2n}$. The initial state is randomly chosen $K_0 \in \{0,1\}^n$. For each $i > 0$, the i-th output is produced and the state is updated as follows. A string $R_i \in \{0,1\}^n$ is chosen according to a distribution of min-entropy at least $\log^2(n)$. Then, $K_i, X_i \in \{0,1\}^n$ are computed as the left and right halves of $F_{K_{i-1}}(R_i)$. The new state is K_i and the output is X_i.

We use our stream cipher to construct a shared-private-key authenticated session protocol. The basic idea is that the sender A and receiver B run their own copies of the stream cipher in parallel, using their shared key k as the initial state K_0. The i-th stream cipher output X_i is used to sign the i-th message piece m_i. As discussed previously, it seems important to ensure that the parties A and B perform stream cipher computation in a synchronized manner. Our approach involves the receiver B generating the high min-entropy string R_i used each round. These bits are sent to A over a public channel. Intuitively, this means that party A must wait to receive a string from party B before moving on to its next computation, and party B can wait to receive a (properly signed) message from party A before generating the high min-entropy string for the next round. Of course, the adversary controls the public channel and may insert strings of his choice (purporting to be sent by the other party) to induce a party to continue its computation; we show that such tampering by the adversary will be detected by party B when he attempts to verify the signatures of the message pieces he receives.

The definition of security we use for our stream cipher is somewhat different than that of Pietrzak. First, we do not rely on the only-computation-leaks axiom – we allow leakage functions to be applied to the entire state. Second, we give a formal, precise definition of security for rounds on which the adversary is allowed to leak; Pietrzak's formal definition only asserts that for rounds i on which the adversary is not allowed to leak, the output X_i is indistinguishable from random. Pietrzak does note that the appropriate requirement for rounds i on which the adversary is allowed to leak is that the output X_i has high HILL pseudo-entropy; however, translating this idea into a precise definition is non-trivial since we must carefully handle subtleties in the definitions of the min-entropy and HILL pseudo-entropy of conditional distributions. Our proof of security uses similar ideas and similar structure to that of Pietrzak. However, there are significant differences in the details, due to the differences in our definitions of security.

Signature schemes and authentication

Given a leakage-resilient public-key signature scheme (that tolerates unbounded total leakage), a leakage-resilient shared-private-key authenticated session protocol can be constructed in a straightforward manner. The sender A and the receiver B use their shared key k as the randomness for the signature scheme's key generation algorithm, producing a key pair (pub, pri); the receiver B discards *pri*. Then, along with each message piece m_i, the sender A includes a signature (under *pri*) of the string $\langle m_i, \bar{i} \rangle$; the receiver B uses *pub* to verify each signature he receives.

How does this approach compare to our construction? Recall that our construction assumes only the existence of pseudo-random generators, and does not use the "only computation leaks" leakage model. Existing signatures schemes tolerating unbounded total leakage either make stronger assumptions than the existence of pseudo-random generators [BKKV10, DHLAW10, MTVY11] or rely on the "only computation leaks" leakage model [FKPR10]. In our construction, for each message piece, the sender must simply perform two evaluations of a pseudo-random function generator. The computational complexity of producing a signature in the existing signature schemes is higher. On the other hand, while our construction requires two flows per message piece, the signature-scheme-based authenticated session protocol requires only a single flow per message piece. Finally, while our construction tolerates only $O(\log n)$ bits of leakage per computation, the existing signature schemes (and hence the resulting authenticated session protocols) tolerate leakage whose length is a constant fraction of the length of the state; furthermore, it is easy to see that in the signature-scheme-based authenticated session protocol described above, the state of the receiver can be made public.

3.1 Preliminaries

3.1.1 Entropy

Definition 12 (Min-entropy) Let X be a distribution. The *min-entropy* of X, denoted $\mathrm{H}_\infty(X)$, is

$$\mathrm{H}_\infty(X) = -\log \max_x \Pr[X = x].$$

Definition 13 (HILL pseudo-entropy) Let X be a distribution over $\{0,1\}^n$. X has *HILL pseudo-entropy* at least k with respect to circuits of size n^e and distinguishing advantage $1/n^d$, denoted

$$\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d},n^e}(X) \geq k$$

if there exists a distribution Y over $\{0,1\}^n$ such that $\mathrm{H}_\infty(Y) \geq k$ and for every circuit D of size at most n^e, we have

$$\left|\Pr\left[D(X) = 1\right] - \Pr\left[D(Y) = 1\right]\right| \leq \frac{1}{n^d}$$

3.2 Authenticated session protocols

In this section, we give a definition for security for a leakage-resilient shared-private-key authenticated session protocol. We then describe our construction of such a protocol.

3.2.1 Security definition

The intuitive goal of an authenticated session protocol involving two parties A and B, where A is sending message pieces $m_1, m_2, \ldots,$ to B, is that B can verify that the message pieces he receives are indeed those sent by A, in the same order. This should hold even when all message pieces m_i sent by A are adversarially chosen. Of course, the adversary has complete control of the public channel over which A and B are communicating. This means that he controls the timing and contents of all communication.

To extend this informal definition to a *leakage-resilient* version, we strengthen the adversary by allowing him to obtain leakage on *both* parties. We are interested in the *continual* leakage setting, where the adversary obtains some bounded amount of leakage on each computation by each party but the total amount of leakage obtained by the adversary over the course of the execution of the protocol is unbounded. The leakage on each computation is computed by an adversarially-chosen function that is applied to the inputs and randomness involved in the computation along with the *entire* state of the party performing the computation. This means that we do *not* rely on the only-computation-leaks assumption.

We further strengthen the adversary by giving him all the entropy used by each party. Equivalently, we require that A and B are deterministic but each have access to a (separate) source of *public min-entropy*; whenever a party obtains a string its source of high min-entropy strings, this string is also given to the adversary. We will formalize the idea of "giving" such strings to the adversary by simply requiring that these strings are output on the public channel.

We begin by formally defining session protocols. Looking ahead, the protocol we have in mind involves two flows per message piece m_i. We will, for the sake of simplicity, restrict our attention to such protocols in our definition. This means that for each message piece m_i, the computation of party B will consist of two algorithms EvalB_1 and EvalB_2, where EvalB_1 is used to produce a flow sent from party B to party A, and EvalB_2 receives the flow sent from party A and outputs a message piece. On the other hand, for each message piece m_i, the computation of party A will consist of only a single algorithm EvalA, that receives the flow from party B, takes as input the message piece m_i, and produces the flow sent from party A to party B.

Definition 14 (Shared-private-key session protocol with public min-entropy) A *shared-private-key session protocol with public min-entropy* (which we will henceforth simply refer to as a *session protocol*) consists of deterministic polytime algorithms EvalB_1, EvalA, and EvalB_2, polynomials $s_B(n)$, $\ell_B(n)$, $s_A(n)$, and $\ell_A(n)$, and distribution ensembles $\{Z_n^A\}$ and $\{Z_n^B\}$ that satisfy the following properties for all $n \in \mathbb{N}$:

1. Z_n^A is a distribution over strings of length $s_A(n)$ such that $\mathrm{H}_\infty(Z_n^A) \geq \log^2(n)$. Similarly, Z_n^B is a distribution over strings of length $s_B(n)$ such that $\mathrm{H}_\infty(Z_n^B) \geq \log^2(n)$.

2. EvalB_1 takes as input $K_B \in \{0,1\}^n$ and $r_B \in \{0,1\}^{s_B(n)}$, and outputs $\beta \in \{0,1\}^{\ell_B(n)}$ and $K'_B \in \{0,1\}^n$ such that β has prefix r_B.

   Informally, the strings K_B and K'_B are the state of party B before and after it executes EvalB_1, r_B is the public min-entropy used by EvalB_1, and β is a flow from party B to party A.

3. EvalA takes as input $K_A \in \{0,1\}^n$, $m \in \{0,1\}^n$, $\beta \in \{0,1\}^{\ell_B(n)}$, and $r_A \in \{0,1\}^{s_A(n)}$, and outputs $e \in \{0,1\}^{\ell_A(n)}$ and $K'_A \in \{0,1\}^n$ such that e has prefix r_A.

   Informally, the strings K_A and K'_A are the state of party A before and after it executes EvalA, m is a message piece that party A would like to send to party B, β is a flow from party B to party A, r_A is the public min-entropy used by EvalA, and e is a flow from party A to party B.

4. EvalB_2 takes as input $K_B \in \{0,1\}^n$, $r_B \in \{0,1\}^{s_B(n)}$, and $e \in \{0,1\}^{\ell_A(n)}$, and outputs either $m \in \{0,1\}^n$ and $K'_B \in \{0,1\}^n$ or a special message `Fail`.

   Informally, the strings K_B and K'_B are the state of party B before and after it executes EvalB_2, r_B is the public min-entropy used by the immediately preceding run of EvalB_1, e is a flow from party A to party B, and m is a message piece received by party B.

5. For all $K \in \{0,1\}^n$, every polynomial $p(n)$, all $r_{A,1}, r_{A,2}, \ldots, r_{A,p(n)} \in \{0,1\}^{s_A(n)}$, all $r_{B,1}, r_{B,2}, \ldots, r_{B,p(n)} \in \{0,1\}^{s_B(n)}$, and all sequences of message pieces $m_1, m_2, \ldots, m_{p(n)} \in \{0,1\}^n$, if we define $K_{A,0} = K_{B,0} = K$ and, for $1 \le i \le p(n)$, we iteratively define $K_{A,i}, K'_{B,i}, K_{B,i}, e_i, \beta_i, m'_i$ in the following manner:

$$\begin{aligned}
(\beta_i, K'_{B,i}) &\leftarrow \mathsf{EvalB}_1(K_{B,i-1}, r_{B,i}) \\
(e_i, K_{A,i}) &\leftarrow \mathsf{EvalA}(K_{A,i-1}, m_i, \beta_i, r_{A,i}) \\
(m'_i, K_{B,i}) &\leftarrow \mathsf{EvalB}_2(K'_{B,i}, r_{B,i}, e_i)
\end{aligned}$$

   then $m'_i = m_i$ for all $1 \le i \le p(n)$.

   Informally, this means that in the absence of an adversary, the message pieces output by party B are exactly those sent by party A, in the same order.

We now define the security experiment for leakage-resilient authenticated session protocols. The adversary will be a family of polynomial-size circuits $C = \{C_n\}$. Letting $\lambda : \mathbb{N} \to \mathbb{N}$ be a function, we will say that an adversary C is $\lambda(n)$-bounded if the leakage functions produced by C_n over the course of the security experiment each have output length $\lambda(n)$. Fixing a session protocol $(\mathsf{EvalB}_1, \mathsf{EvalA}, \mathsf{EvalB}_2, s_B(n), \ell_B(n), s_A(n), \ell_A(n), \{Z^A_n\}, \{Z^B_n\})$, a function $\lambda : \mathbb{N} \to \mathbb{N}$, a $\lambda(n)$-bounded adversary $C = \{C_n\}$, and $n \in \mathbb{N}$, the security experiment proceeds as follows.

A string $K \in \{0,1\}^n$ is randomly chosen. We define $K_{A,0} = K_{B,0} = K$. Then, C_n is allowed to run EvalA, EvalB_1, and EvalB_2 in the following manner. C_n may run these algorithms as many times as he wishes and in any order of his choice as long as for every $i > 0$, the $(i+1)$-st invocation of EvalB_1 does not occur before the i-th invocation of EvalB_2, and the i-th invocation of EvalB_2 does not occur before the i-th invocation of EvalB_1. (This restriction captures the fact that even though the adversary controls the public channel, party B will still alternate between executing EvalB_1 and executing EvalB_2.) We now describe what happens when the adversary C_n runs each algorithm.

- For $i > 0$, the i-th invocation of EvalB_1 proceeds as follows. C_n produces the description of a circuit $f_{B1,i} : \{0,1\}^n \times \{0,1\}^{s_B(n)} \to \{0,1\}^{\lambda(n)}$. Then, $r_{B,i} \leftarrow Z_n^B$ is chosen. Next, $(\beta_i, K'_{B,i}) \leftarrow \mathsf{EvalB}_1(K_{B,i-1}, r_{B,i})$ and $leak_{B1,i} \leftarrow f_{B1,i}(K_{B,i-1}, r_{B,i})$ are computed. Finally, C_n is given β_i and $leak_{B1,i}$.

- For $i > 0$, the i-th invocation of EvalA proceeds as follows. C_n produces $m_i \in \{0,1\}^n$ and $\beta'_i \in \{0,1\}^{\ell_B(n)}$, and the description of a circuit $f_{A,i} : \{0,1\}^n \times \{0,1\}^{s_A(n)} \to \{0,1\}^{\lambda(n)}$. Then, $r_{A,i} \leftarrow Z_n^A$ is randomly chosen. Next, $(e_i, K_{A,i}) \leftarrow \mathsf{EvalA}(K_{A,i-1}, m_i, \beta'_i, r_{A,i})$ and $leak_{A,i} \leftarrow f_{A,i}(K_{A,i-1}, r_{A,i})$ are computed[1]. Finally, C_n is given e_i and $leak_{A,i}$.

- For $i > 0$, the i-th invocation of EvalB_2 proceeds as follows. C_n produces a string $e'_i \in \{0,1\}^{\ell_A(n)}$ and the description of a circuit $f_{B2,i} : \{0,1\}^n \to \{0,1\}^{\lambda(n)}$. Then, $(m'_i, K_{B,i}) \leftarrow \mathsf{EvalB}_2(K'_{B,i}, r_{B,i}, e'_i)$ and $leak_{B2,i} \leftarrow f_{B2,i}(K'_{B,i})$ are computed[2]; if EvalB_2 outputs `Fail`, the experiment ends immediately. If the i-th invocation of EvalA has previously occurred and $m'_i = m_i$, C_n is given $leak_{B2,i}$; otherwise, the experiment ends immediately.

Say that the final invocation of EvalB_2 is the j-th invocation. Define $q_C(n)$ to be the probability that the j-th invocation of EvalB_2 does not output `Fail` and either EvalA has been invoked fewer than j times or $m'_j \neq m_j$.

Definition 15 (Leakage-resilient authenticated session protocol) Let $\lambda : \mathbb{N} \to \mathbb{N}$ be a function. A session protocol is a $\lambda(n)$*-leakage-resilient authenticated session protocol* if for every $\lambda(n)$-bounded adversary C as described above, we have $q_C(n) \leq 1/n^d$ for all d and sufficiently large n.

[1] It is not necessary to provide m_i or β'_i as inputs to $f_{A,i}$ since C_n chose these values himself and hence he can simply hardcode them into $f_{A,i}$ if he wishes.

[2] It is not necessary to provide $r_{B,i}$ to $f_{B2,i}$ since this was previously provided to C_n as the prefix of β_i, and it is not necessary to provide e'_i to $f_{B2,i}$ since C_n chose this value himself.

3.2.2 Our construction

In our construction, only party B requires a source of public min-entropy. Accordingly, to simplify notation, we use Z_n rather that Z_n^B to denote the high min-entropy distribution used by B.

Given pseudo-random function generators $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ and $F' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, and given a distribution ensemble $\{Z_n\}$ such that for all n, Z_n is a distribution over $\{0,1\}^n$ and $\mathrm{H}_\infty(Z_n) \geq \log^2(n)$, we construct a leakage-resilient authenticated session protocol SP as follows.

EvalB_1: On input (K_B, r_B), where $K_B \in \{0,1\}^n$ and $r_B \in \{0,1\}^n$, EvalB_1 lets $K'_B = K_B$ and $\beta = r_B$, and outputs (β, K'_B).

EvalA: On input (K_A, m, β), where $K_A, m \in \{0,1\}^n$ and $\beta \in \{0,1\}^n$, EvalA computes $K'_A || X_A \leftarrow F_{K_A}(\beta)$ (where $|K'_A| = |X_A| = n$) and $\alpha = F'_{X_A}(m)$, lets $e = \langle m, \alpha \rangle$, and outputs (e, K'_A).

EvalB_2: On input (K_B, r_B, e'), where $K_B \in \{0,1\}^n$, $r_B \in \{0,1\}^n$, and $e' \in \{0,1\}^{2n}$, EvalB_2 parses $\langle m', \alpha' \rangle \leftarrow e'$, computes $K'_B || X_B \leftarrow F_{K_B}(r_B)$ (where $|K'_B| = |X_B| = n$), and $\alpha = F'_{X_B}(m')$. If $\alpha' = \alpha$, EvalB_2 outputs (m', K'_B); otherwise, EvalB_2 outputs `Fail`.

It is not hard to see that SP satisfies the definition of a session protocol. The idea is that parties A and B both run a stream cipher (see Section 3.4) starting from the same key and using the same inputs, and use the i-th output X_i to compute a signature $F'_{X_i}(m_i)$ of the i-th message piece m_i.

Theorem 3.2.1 *For all $c > 0$, SP is a $c \log n$-leakage-resilient authenticated session protcol.*

We prove Theorem 3.2.1 in Section 3.6.

3.3 Running multiple instances of a stream cipher

In this section, we show what goes wrong if two parties run Pietrzak's stream cipher [Pie09] with the same initial state, when the adversary controls the scheduling of each party's computation and the adversary obtains leakage from both parties. Note that the same general approach can be used to attack the stream cipher of Dziembowski and Pietrzak [DP08] (and, in fact, any stream cipher whose construction consists of two pieces of memory connected by a public channel, where all computation is deterministic) when two parties run this stream cipher with the same initial state.

Pietrzak's construction uses two pieces of memory, A and B. Computation alternates between these two pieces, and leakage for each computation occurs only on the single piece of memory accessed by the computation. Initially, A stores a randomly chosen string $K_0 \in \{0,1\}^n$, B stores a randomly chosen string $K_1 \in \{0,1\}^n$, and there is a randomly chosen publicly-known string $X_0 \in \{0,1\}^n$. Computation then proceeds in a sequence of rounds. Computation for odd-numbered rounds occurs on memory A, and computation for even-numbered rounds occurs on memory B. In each round i, $K_{i+1}||X_i \leftarrow F_{K_{i-1}}(X_{i-1})$ is computed, where $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ is a pseudo-random function generator. X_i is output, and K_{i+1} replaces K_{i-1} in memory.

Leakage in each round i is computed by an adversarially-chosen function whose output length is bounded by some value λ (such as $\lambda = \log n$). Specifically, before computation takes places in round i, the adversary outputs the description of a function $f_i : \{0,1\}^n \to \{0,1\}^\lambda$. Then, at the end of the round, the adversary is given X_i and $f_i(K_{i-1})$.

Now, suppose two parties P_1 and P_2 compute Pietrzak's stream cipher, and both parties start with the same initial state (K_0, K_1, X_0). Suppose further that the adversary controls the timing of each party's computation; that is, he controls the manner in which the computation of P_1 and P_2 is interleaved.

We now show that the adversary can eventually obtain the entire contents of both pieces memory, even if his leakage is restricted to 1 bit per round. The adversary begins by scheduling party P_1 for $2n$ rounds. He does not bother to obtain any leakage during these rounds, but he stores the outputs $X_1, X_2, \ldots X_{2n}$. Now the adversary schedules party P_2 for $2n$ rounds. We claim that by the end of these $2n$ rounds, the adversary will have learned K_{2n} and K_{2n+1}. He accomplishes this as follows. For each odd i, the adversary in round i will choose a leakage function f_i that, on input K_{i-1}, uses $X_{i-1}, X_{i+1}, X_{i+3}, \ldots, X_{2n-2}$ (which are known by the adversary) to compute $K_{i+1}, K_{i+3}, K_{i+5}, \ldots, K_{2n}$, and then outputs bit $(i+1)/2$ of K_{2n}. Similarly, for each even i, the adversary in round i will choose a leakage function f_i that, on input K_{i-1}, uses $X_{i-1}, X_{i+1}, X_{i+3}, \ldots, X_{2n-1}$ (which are known by the adversary) to compute $K_{i+1}, K_{i+3}, K_{i+5}, \ldots, K_{2n+1}$, and then outputs bit $i/2$ of K_{2n+1}. In this manner, after $2n$ rounds, the adversary learns K_{2n} and K_{2n+1} in their entirety; then, since he also knows X_{2n}, he knows the entire current state of party P_2.

Note that if the parties are not outputting the X_i, but instead use the X_i to perform some task (like authentication), it is not clear how to extend the attack described above to such a setting. On the other hand, in such a setting, it is also not clear how prove that the adversary *cannot* mount successful attacks where he takes advantage of his ability to schedule the parties; at the very least, by scheduling in the manner described above, he can accumulate some information about the X_i through leakage on party P_1, and he can obtain additional information from P_1's computation as a result of the usage of X_i (e.g. if the X_i are used to sign message pieces, the adversary obtains some information about the X_i by seeing the signatures output by P_1).

3.4 Stream cipher construction

In this section, we present our modified version of Pietrzak's stream cipher. Our construction uses only a single piece of memory but requires a public source of min-entropy.

3.4.1 The construction

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudo-random function generator.

Let $\{Z_n\}$ be such that for all n, Z_n is a distribution over strings of length n and $\mathrm{H}_\infty(Z_n) \geq \log^2(n)$.

The initial state is K_0, where $K_0 \in \{0,1\}^n$ is randomly chosen.

For each $i > 0$, the i-th round consists of:

1. $R_i \leftarrow Z_n$ is chosen.
2. $K_i || X_i \leftarrow F_{K_{i-1}}(R_i)$.
3. The new state is K_i.

3.4.2 The adversary's interaction

Fix $c > 0$. A $(c \log n)$-bounded adversary interacts as follows.

For each $i > 0$:

1. Before round i, the adversary outputs the description of a function $f_i : \{0,1\}^{2n} \to \{0,1\}^{c \log n}$.
2. After round i, the adversary sees $R_i, X_i, f_i(K_{i-1}, R_i)$.

3.4.3 Security

We begin by defining some notation.

For an adversary A, we will use $\texttt{real}_i$ to denote the adversary's view after the first i rounds along with the corresponding X_j. That is,

$$\texttt{real}_i = \langle R_1, f_1(K_0, R_1), X_1, R_2, f_2(K_1, R_2), X_2, \ldots, R_i, f_i(K_{i-1}, R_i), X_i \rangle$$

Note that the f_j are not fixed functions, but rather are chosen adaptively by the adversary A as described in Section 3.4.2.

We next consider the distribution produced by having an adversary A interact with a "simulated" version of the construction. For distribution $K'_0X'_1K'_1X'_2K'_2 \dots K'_{i-1}X'_{i-1}X'_i|R_1R_2 \dots R_i$, define

$$\mathtt{sim}_i = \left\langle R_1, f_1(K'_0, R_1), X'_1, R_2, f_2(K'_1, R_2), X'_2, \dots, R_i, f_i(K'_{i-1}, R_i), X'_i \right\rangle$$

That is, $\mathtt{sim}_i$ is the distribution produced by having A interact with a modified version of the construction where in step 2 in each round j, K_jX_j take on values from the distribution $K'_jX'_j$ rather than a value computed using F. Note that each f_j is the function that is chosen in round j by A based on its view (interacting with the "simulated" version of the construction) up to that point.

We will sometimes need to use a version of $\mathtt{sim}_i$ which omits X'_i. We will denote this using $\mathtt{sim}_i^-$.

We will also define versions of $\mathtt{real}_i$ and $\mathtt{sim}_i$ that include an additional round where there is no leakage. Specifically, we define

$$\mathtt{real}_i^+ = \langle \mathtt{real}_i, R_{i+1}, K_{i+1}, X_{i+1} \rangle$$

That is, $\mathtt{real}_i^+$ includes the inputs and outputs of an additional leak-free round along with the entire state at the end of that round. We also define

$$\mathtt{sim}_i^+ = \left\langle \mathtt{sim}_i, R_{i+1}, K''_{i+1}, X''_{i+1} \right\rangle$$

where X''_{i+1} and K''_{i+1} are independent random variables that are each uniformly distributed over $\{0,1\}^n$.

We need to define hybrid distributions that are in-between $\mathtt{real}_i^+$ and $\mathtt{sim}_i^+$. For $0 \le j \le i+1$, define $\mathtt{hybrid}_i^j$ to be the distribution produced by having A interact as in $\mathtt{sim}_i$ for the first j rounds, and then continuing the interaction with the real construction (starting from state K'_j) for the remaining $i+1-j$ rounds. Observe that $\mathtt{hybrid}_i^{i+1} = \mathtt{sim}_i^+$, and when K'_0 is distributed uniformly we have $\mathtt{hybrid}_i^0 = \mathtt{real}_i^+$.

Theorem 3.4.1 *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudo-random function generator. Let $\{Z_n\}$ be such that for all n, Z_n is a distribution over strings of length n and $\mathrm{H}_\infty(Z_n) \ge \log^2(n)$. For all $c > 0, d > 0, e > 0$, every function $p : \mathbb{N} \to \mathbb{N}$, and sufficiently large n, and for all $(c \log n)$-bounded adversaries A interacting as described in section 3.4.2 and obtaining leakage for $p(n)$ rounds, there exists a distribution $K'_0K'_1X'_1K'_2X'_2 \dots K'_{p(n)}X'_{p(n)}|R_1R_2 \dots R_{p(n)}$, with each K'_i and each X'_i over $\{0,1\}^n$, such that the following properties hold.*

1. *For all $1 \le i \le p(n)$, and all $\alpha \leftarrow \mathtt{sim}_i^-$, $\mathrm{H}_\infty(K'_iX'_i|\mathtt{sim}_i^- = \alpha) \ge 2n - (c + 2d + 1)\log n$.*

2. *For all* $0 \le i \le p(n)$*, and all* $\beta \leftarrow \texttt{sim}_i$*,* $\mathrm{H}_\infty(K'_i|\texttt{sim}_i = \beta) \ge n - (c + 3d + 1)\log n$*.*

3. *For all* $1 \le i \le p(n)$*, and all* $\alpha \leftarrow \texttt{sim}_i^-$*,*

$$\Pr_{k \leftarrow K'_i}\left[\mathrm{H}_\infty(X'_i|\texttt{sim}_i^- = \alpha \wedge K'_i = k) \ge n - (c + 3d + 1)\log n\right] \ge 1 - \frac{1}{n^d}$$

4. *For all adversaries* D *such that* $2 \cdot \texttt{size}(A) + \texttt{size}(D) + p(n)\texttt{size}(F) \le n^e$*, and all* $1 \le i \le p(n) + 1$*,*

$$\left|\Pr\left[D(\texttt{real}^+_{p(n)}) = 1\right] - \Pr\left[D(\texttt{hybrid}^i_{p(n)}) = 1\right]\right| \le \frac{3i}{n^d}$$

5. *For all adversaries* D *such that* $2 \cdot \texttt{size}(A) + \texttt{size}(D) + p(n)\texttt{size}(F) \le n^e$*,*

$$\left|\Pr\left[D(\texttt{real}^+_{p(n)}) = 1\right] - \Pr\left[D(\texttt{real}_{p(n)}, R_{p(n)+1}, K''_{p(n)+1}, X''_{p(n)+1}) = 1\right]\right| \le \frac{6p(n) + 6}{n^d}$$

Specifically, for sufficiently large n *(depending only on* c*,* d*, and* e*):*

- *If, for every distribution* $K'_0K'_1X'_1K'_2X'_2 \ldots K'_{p(n)}X'_{p(n)}|R_1R_2 \ldots R_{p(n)}$ *satisfying (1), (2), and (3) there exists an adversary* D *breaking (4), then there exists an adversary of size* $n^{e+8d+2c+8}$ *breaking* F *with advantage* $1/n^{5d+2c+3}$*.*

- *If there exists an adversary* D *breaking (5), then there exists an adversary of size* $n^{e+8d+2c+8}$ *breaking* F *with advantage* $1/n^{5d+2c+3}$*.*

Since it is not obvious that Theorem 3.4.1 is the "right" theorem to prove, let us consider how this theorem can be applied. Suppose we would like to use the X_i produced each round to perform some "cryptographic task". In general, we would like a robust notion of security where the use of each X_i remains secure even when the adversary happens to learn all the other X_j (in addition to getting leakage from every round, including round i). The theorem tells us that it is sufficient to show that the use of each X'_i is secure in the experiment implied by $\texttt{hybrid}^i_{n^b}$. Furthermore, the theorem tells us that in the experiment implied by $\texttt{hybrid}^i_{n^b}$, X'_i has high entropy even given the adversary's view, all the previous X'_j, and the entire state at the end of round i. Finally, in certain applications (like authentication) we might follow a proof approach that allows us to eliminate leakage in a particular round of interest; property 5 in the theorem tells us that for such a round i, the output X_i is pseudo-random even given the entire state at the end of round i.

We prove Theorem 3.4.1 in Section 3.5.

3.5 Proof of Theorem 3.4.1

We begin by giving an informal overview of the proof.

3.5.1 Proof overview

The high-level approach is similar to that of Pietrzak [Pie09]. Intuitively, to show that our stream cipher is leakage-resilient, we need to show that as a result of the refreshing that takes place each round, the state K_i maintains almost maximal[3] pseudo-entropy conditioned on the view of the adversary, and as a result, the outputs X_i have almost maximal pseudo-entropy conditioned on the adversary's view.

In a first attempt at formalizing this intuition, one might try to show inductively that for every $1 \leq i \leq p(n)$, we have with very high probability over the adversary's view $\texttt{real}_i^-$ that $K_i X_i$ has almost maximal HILL pseudo-entropy conditioned on $\texttt{real}_i^-$. In the induction step, we would use the assumption that $K_i X_i$ has almost maximal pseudo-entropy conditioned on $\texttt{real}_i^-$ to argue that K_i has almost maximal pseudo-entropy conditioned on $\langle \texttt{real}_i^-, X_i \rangle$ (that is, conditioned on $\texttt{real}_i$), as a precursor to arguing about the pseudo-entropy of $K_{i+1}X_{i+1} = F_{K_i}(R_{i+1})$ conditioned on $\texttt{real}_{i+1}^-$. However, as we will see in Lemma 3.5.3, such an argument will not go through, due to subtleties in the definition of the HILL pseudo-entropy of conditional distributions; specifically, even if $K_i X_i$ has *maximal* HILL pseudo-entropy conditioned on $\texttt{real}_i^-$, it may be the case that K_i has nearly *zero* HILL pseudo-entropy conditioned on $\langle \texttt{real}_i^-, X_i \rangle$.

To overcome this problem, we instead (as detailed in the statement of Theorem 3.4.1) argue about the actual (not computational) min-entropy of distributions that are indistinguishable from the "real" distributions that yield the adversary's view. As we will see in Lemma 3.5.2, the min-entropy of conditional distributions behaves in exactly the manner we need in order to follow an inductive approach similar to the failed approach described above. Specifically, if distribution $K_i' X_i'$ has almost maximal min-entropy conditioned on a distribution $\texttt{sim}_i^-$, then with high probability over X_i' we have that K_i' has almost maximal min-entropy conditioned on $\langle \texttt{sim}_i^-, X_i' \rangle$.

Note that we do not show how to sample the high min-entropy distributions in question. Instead, we non-constructively argue about the existence of such distributions. This means we cannot use a traditional hybrid argument where all the hybrid experiments are *first* defined and *then* shown to be indistinguishable from each other. Instead, we use an inductive hybrid-style argument where we simultaneously argue about the existence of distributions with desired properties and about the indistinguishability of these distributions from the "real" distribution. Specifically, our initial hybrid is the real experiment, and in each step i of the argument, we use the approach described above to argue that round i of the adversary's view can be replaced with distributions that have the desired entropy properties and such that the resulting i-th hybrid is indistinguishable from the initial hybrid.

[3] By "almost maximal", we mean "within logarithmically-many bits of maximal".

One of the main ingredients we need for our induction argument is Lemma 3.5.7, where we show that for every pseudo-random function generator F and every distribution K of almost maximal min-entropy, we have for almost every x that the distribution $F_K(x)$ is indistinguishable from random. (Note that if K has maximal min-entropy, that is, if K is randomly chosen, then we have for *every* x that $F_K(x)$ is indistinguishable from random.) The approach we use for proving this is similar to the approach used by Pietrzak [Pie09] in his proof that weak-pseudo-random function generators (that is, function generators that are pseudo-random when their inputs are chosen randomly rather than adversarially) remain weak-pseudo-random when their seed is chosen according to a distribution of almost-maximal entropy instead of the uniform distribution.

The other ingredient we need is the lemma of Dziembowski and Piertrzak [DP08] (which we state as Lemma 3.5.5) about leakage on the seed of pseudo-random number generators. They show that the output of a pseudo-random number generator has almost maximal HILL pseudo-entropy even when there is logarithmic-length leakage on the seed. Combining this lemma with Lemma 3.5.7 (the lemma discussed above), we conclude in Lemma 3.5.8 that pseudo-random function generators whose seed is chosen from a distribution of almost maximal min-entropy and whose input is chosen from a distribution of super-logarithmic min-entropy have output that has almost maximal HILL pseudo-entropy, even when there is logarithmic-length leakage on the seed[4]. This allows us to proceed with our induction argument.

3.5.2 Entropy-related lemmas

In this section, we first give a lemma about the behaviour of min-entropy and conditional distributions. We will need such a lemma when proving Theorem 3.4.1. Then, we show that HILL pseudo-entropy does not exhibit the same behaviour, illustrating the need to be careful when working with the entropy of conditional distributions in the statement and proof of Theorem 3.4.1.

Min-entropy and conditional distributions

Lemma 3.5.1 *Let XY be a distribution, with X and Y each over $\{0,1\}^n$. For all $c \geq 0$ and $d > 0$, if $\mathrm{H}_\infty(XY) \geq 2n - c\log n$, then*

$$\Pr_{y \leftarrow Y}[\mathrm{H}_\infty(X|Y=y) \geq n-(c+d)\log n] \geq 1 - \frac{1}{n^d} \tag{3.1}$$

Proof Fix c, d, X, Y, and assume $\mathrm{H}_\infty(XY) \geq 2n - c\log n$.

[4] Pietrzak [Pie09] obtains a similar result for the case where the input is chosen from a distribution of almost maximal min-entropy rather than super-logarithmic min-entropy.

Suppose that (3.1) is false. Let $S \subseteq \{0,1\}^n$ be the set of all y such that $\mathrm{H}_\infty(X|Y = y) < n - (c+d)\log n$. Then, since (3.1) is false, we have $\Pr[Y \in S] > 1/n^d$.

Now, consider an arbitrary $y \in S$. Then, by definition of S, there exists an x such that $\Pr[X = x|Y = y] > 2^{-(n-(c+d)\log n)} = n^{c+d}/2^n$. On the other hand, since $\mathrm{H}_\infty(XY) \geq 2n - c\log n$, we have that $\Pr[XY = xy] \leq n^c/2^{2n}$. This means we must have $\Pr[Y = y] < \frac{1}{n^d 2^n}$.

So for each $y \in S$, we have $\Pr[Y = y] < \frac{1}{n^d 2^n}$. Since $\Pr[Y \in S] > 1/n^d$, it follows that

$$|S| > \frac{\frac{1}{n^d}}{\frac{1}{n^d 2^n}} = 2^n$$

which is a contradiction (since $S \subseteq \{0,1\}^n$). □

Corollary 3.5.2 *Let XY be a distribution, with X and Y each over $\{0,1\}^n$. For all $c \geq 0$ and $d > 0$ if $\mathrm{H}_\infty(XY) \geq 2n - c\log n$, then there exists a distribution $X'|Y$ over $\{0,1\}^n$ such that $X'Y$ is within statistical distance $1/n^d$ of XY, $\mathrm{H}_\infty(X'Y) \geq 2n - c\log n$, and*

$$\Pr_{y \leftarrow Y}\left[\mathrm{H}_\infty(X'|Y = y) \geq n - (c+d)\log n\right] = 1 \tag{3.2}$$

Proof Fix c, d, X, Y, and assume $\mathrm{H}_\infty(XY) \geq 2n - c\log n$. By Lemma 3.5.1, we have

$$\Pr_{y \leftarrow Y}\left[\mathrm{H}_\infty(X|Y = y) \geq n - (c+d)\log n\right] \geq 1 - \frac{1}{n^d} \tag{3.3}$$

We define distribution $X'|Y$ as follows. For each $y \in Y$ such that $\mathrm{H}_\infty(X|Y = y) \geq n - (c + d)\log n$, distribution $X'|Y = y$ is identical to $X|Y = y$. For all other $y \in Y$, distribution $X'|Y = y$ is the uniform distribution. By construction, we have

$$\Pr_{y \leftarrow Y}\left[\mathrm{H}_\infty(X'|Y = y) \geq n - (c+d)\log n\right] = 1 \tag{3.4}$$

Also, by construction, we have that with probability at least $1-1/n^d$ over $y \leftarrow Y$, the conditional distributions $X|Y = y$ and $X'|Y = y$ are identical. It follows that $X'Y$ is within statistical distance $1/n^d$ of XY.

It remains to show that $\mathrm{H}_\infty(X'Y) \geq 2n - c\log n$. For all $y \in Y$ such that $\mathrm{H}_\infty(X|Y = y) \geq n-(c+d)\log n$, we have (by construction of X' and using the fact that $\mathrm{H}_\infty(XY) \geq 2n-c\log n$) that for all $x \in \{0,1\}^n$, $\Pr[X'Y = xy] \leq 2^{-2n+c\log n}$. For all other $y \in \{0,1\}^n$, we have that for all $x \in \{0,1\}^n$, $\Pr[X'Y = xy] = 2^{-n}\Pr[Y = y]$. But this is at most $2^{-2n+c\log n}$, since the assumption $\mathrm{H}_\infty(XY) \geq 2n - c\log n$ implies that for all $y \in \{0,1\}^n$, $\Pr[Y = y] \leq 2^{-n+c\log n}$. □

HILL pseudo-entropy and conditional distributions

We now show that a HILL pseudo-entropy analog of Lemma 3.5.1 does *not* hold.

Lemma 3.5.3 *Suppose one-way permutations exist. Then for all d, e, and sufficiently large n, there exists a distribution XY, with X and Y each over $\{0,1\}^n$, such that $\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d},n^e}(XY) = 2n$ yet with probability 1 over $y \in Y$, we have $\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d},n^e}(X|Y=y) \approx 0$.*

To prove this lemma, we need the following claim that uses a standard construction of a pseudo-random number generator from a one-way permutation.

Claim 3.5.4 *Suppose one-way permutations exist. Then there exists a pseudo-random number generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ with the property that the function $p : \{0,1\}^n \to \{0,1\}^n$ that on input x outputs the right-half of $G(x)$ (that is, $p(x) = G(x)|_{last\ n\ bits}$) is a permutation.*

Proof Suppose one-way permutations exist. Then there exists a one-way permutation $f : \{0,1\}^n \to \{0,1\}^n$ that has a hard-core predicate $b : \{0,1\}^n \to \{0,1\}$. We define number generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ as follows:

$$G(x) = b(x)\, b(f(x))\, b(f^{(2)}(x)) \ldots b(f^{(n-1)}(x))\, f^{(n)}(x) \tag{3.5}$$

where $f^{(i)}$ denotes the composition of f with itself i times. A standard proof shows that G is pseudo-random (for example, see section 3.4.1.2 in [Gol01]).

Note that the right-half of $G(x)$ is $f^{(n)}(x)$, and $f^{(n)}$ is a permutation (since f is a permutation).
□

We now prove the lemma.

Proof (Lemma 3.5.3) Suppose one-way permutations exist. Then, by Claim 3.5.4, there exists a pseudo-random number generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ with the property that the function $p : \{0,1\}^n \to \{0,1\}^n$ that on input x outputs the right-half of $G(x)$ (that is, $p(x) = G(x)|_{last\ n\ bits}$) is a permutation.

Fix constants d and e. Then, for sufficiently large n, the distributions X and Y, each over $\{0,1\}^n$, defined as

$$XY = G(U_n) \tag{3.6}$$

have the property that

$$\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d},n^e}(XY) = 2n \tag{3.7}$$

Now, note that $X = G(p^{-1}(Y))|_{first\ n\ bits}$. Then, for all y, given that $Y = y$, the value of X is completely determined. That is, for all y, there exists an x such that $\Pr[X = x | Y = y] = 1$. It follows that with probability 1 over $y \in Y$, we have

$$\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d},n^e}(X|Y=y) \approx 0 \tag{3.8}$$

□

3.5.3 Pseudo-random generators with bounded leakage on the seed

By definition, the output of a pseudo-random generator $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ has $\ell(n)$ bits of HILL pseudo-entropy with respect to an adversary that is not given the randomly chosen seed of G. Clearly, when the adversary is allowed to see leakage on the seed, we can no longer expected the output of G to have $\ell(n)$ bits of pseudo-entropy. However, Dziembowski and Pietrzak [DP08] show that the output of a pseudo-random generator has very high HILL pseudo-entropy even when there is leakage on the seed. In fact, when the leakage has logarithmic length, there is only a logarithmic loss in pseudo-entropy.

Lemma 3.5.5 ([DP08]) *Let $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ be a pseudo-random generator and let $f : \{0,1\}^n \to \{0,1\}^{c \log n}$ be a function. Then for all d, e and sufficiently large n, when independent $X, Y \sim U_n$ we have:*

$$\Pr_{y \leftarrow f(Y)} \left[\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d}, n^e} (G(X) | f(X) = y) \geq \ell(n) - (c + 2d) \log n - 3 \right] \geq 1 - \frac{1}{2n^d} \tag{3.9}$$

Specifically, for sufficiently large n (depending only on d, e and $\ell(\cdot)$), given adversaries breaking (3.9), there exists an adversary of size $(\ell(n))^2 n^{e+4d+1}$ that breaks G with advantage at least $1/(8n^{2d+c})$.

By examining Dziembowski and Pietrzak's proof of Lemma 3.5.5, it is easy to observe that their proof does not actually require $\ell(n) > n$ or $X, Y \sim U_n$. Instead, it only requires that X and Y are independent identical distributions over strings of length n. That is, Lemma 3.5.5 generalizes to the case where $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is an arbitrary function (not necessarily length-increasing) whose output is indistinguishable from random when its input is chosen according to some distribution D_n.

Corollary 3.5.6 *Let $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ and $f : \{0,1\}^n \to \{0,1\}^{c \log n}$ be functions and let D_n be a distribution on strings of length n. Suppose the distribution $G(D_n)$ is computationally indistinguishable from the uniform distribution $U_{\ell(n)}$. Then for all d, e and sufficiently large n, when independent $X, Y \sim D_n$ we have:*

$$\Pr_{y \leftarrow f(Y)} \left[\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d}, n^e} (G(X) | f(X) = y) \geq \ell(n) - (c + 2d) \log n - 3 \right] \geq 1 - \frac{1}{2n^d} \tag{3.10}$$

Specifically, for sufficiently large n (depending only on d, e and $\ell(\cdot)$), given adversaries breaking (3.10), there exists an adversary of size $(\ell(n))^2 n^{e+4d+1}$ that distinguishes $G(D_n)$ with advantage at least $1/(8n^{2d+c})$.

3.5.4 Pseudo-random function generators with high-entropy seeds

It is easy to see that if $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ is a pseudo-random function generator, then for all $x \in \{0,1\}^n$, the function $G^x(y) = F_y(x)$ is a pseudo-random *number* generator[5]. Now, suppose K is a high-entropy (but not necessarily uniform) distribution on $\{0,1\}^n$. Then, there may exist x such that $G^x(K)$ is easy to distinguish from U_m. However, we will show that for *almost all* $x \in \{0,1\}^n$, the distribution $G^x(K)$ is computationally indistinguishable from U_m. Our proof uses ideas similar to those used by Pietrzak [Pie09] in his lemma about weak-pseudo-random function generators with high-entropy seeds.

Lemma 3.5.7 *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{\ell(n)}$ be a pseudo-random function generator. Then, for all c, d, e and sufficiently large n: If K is a distribution on $\{0,1\}^n$ such that $\mathrm{H}_\infty(K) \geq n - c\log n$, then for all but fewer than n^{2d+1} strings $x \in \{0,1\}^n$, for all circuits D of size n^e,*

$$\left|\Pr\left[D(F_K(x)) = 1\right] - \Pr\left[D(U_{\ell(n)}) = 1\right]\right| \leq \frac{1}{n^d} \tag{3.11}$$

Specifically, for sufficiently large n (depending only on c and d), if for at least n^{2d+1} strings x there exists a D of size n^e breaking (3.11), then there exists an adversary of size n^{e+2d+2} that distinguishes F (as a pseudo-random function generator) with advantage at least $1/n^{c+d+1}$.

Proof Fix c, d, e. Let n be large enough so that $3/(4n^{d+c}) - 1/\exp(\frac{n}{8}) \geq 1/n^{c+d+1}$. Let K be a distribution on $\{0,1\}^n$ such that $\mathrm{H}_\infty(K) \geq n - c\log n$.

Suppose there exists a set $S \subseteq \{0,1\}^n$ of size n^{2d+1} and a collection of distinguishers $\{D_x\}_{x\in S}$ each of size n^e such that for all $x \in S$,

$$\Pr\left[D_x(F_K(x)) = 1\right] - \Pr\left[D_x(U_{\ell(n)}) = 1\right] > \frac{1}{n^d} \tag{3.12}$$

For each $x \in S$, define $r_x = \Pr[D_x(U_{\ell(n)}) = 1]$ and define $p_x = \Pr[D_x(F_K(x)) = 1]$. Also, define $\alpha = \sum_{x\in S} r_x$ and $\beta = \sum_{x\in S} p_x$. Note that we have

$$\beta - \alpha = \sum_{x\in S}(p_x - r_x) > |S|\frac{1}{n^d} = \frac{n^{2d+1}}{n^d} = n^{d+1} \tag{3.13}$$

We now construct an adversary D' of size at most n^{e+2d+2} for breaking F. Given an oracle f, D' runs each D_x on input $f(x)$, and accepts iff the number of D_x that accept is at least $t = \alpha + \frac{n^{d+1}}{4}$.

Consider the probability that D' accepts a randomly chosen function f. For each $x \in S$, let R_x be a random variable whose value is the output of $D_x(f(x))$. Note that since f is

[5] More precisely, we need to require that $m > n$ in order to call G^x a number *generator*. But even without this condition, we have that $G^x(U_n)$ is computationally indistinguishable from U_m.

randomly chosen, $\{R_x\}_{x\in S}$ is a set of independent random variables. Define random variable $R = \sum_{x\in S} R_x$. That is, R is the number of D_x that accept. Note that $E[R_x] = r_x$ for each $x \in S$, and hence $E[R] = \alpha$. Then, by Hoeffding's inequality, we have

$$\Pr\left[R \geq t\right] = \Pr\left[R \geq \alpha + \frac{n^{d+1}}{4}\right] = \Pr\left[R - \alpha \geq |S|\frac{1}{4n^d}\right] \leq \exp\left(-\frac{|S|}{8n^{2d}}\right) = \frac{1}{\exp(\frac{n}{8})} \quad (3.14)$$

It follows that the probability D' accepts a randomly chosen function f is at most $1/\exp(\frac{n}{8})$.

Now consider the probability that D' accepts F_k for k chosen according to K. For each $x \in S$, let P_x be a random variable whose output is the value of $D_x(F_k(x))$. Define random variable $P = \sum_{x\in S} P_x$. That is, P is the number of D_x that accept. Note that $E[P_x] = p_x$ for each $x \in S$, and hence $E[P] = \beta$. By reasoning similar to Markov's inequality, and using the fact that P does not take on any value larger than $|S|$, we have

$$\Pr\left[P \leq \beta - \frac{3n^{d+1}}{4}\right] = \Pr\left[P \leq \beta - |S|\frac{3}{4n^d}\right] \leq 1 - \frac{3}{4n^d} \quad (3.15)$$

Now, since $\beta - \alpha > n^{d+1}$, we have $\beta - \frac{3n^{d+1}}{4} > \alpha + \frac{n^{d+1}}{4} = t$. This means

$$\Pr\left[P \geq t\right] \geq \Pr\left[P \geq \beta - \frac{3n^{d+1}}{4}\right] \geq \frac{3}{4n^d} \quad (3.16)$$

It follows that the probability D' accepts F_k for k chosen according to K is at least $\frac{3}{4n^d}$.

Finally, consider the probability that D' accepts F_k for k chosen according to U_n. Define *Good* to be the set of strings $k \in \{0,1\}^n$ such that D' accepts F_k. For each $k \in$ *Good*, define $q_k = \Pr[K = k]$. We know that

$$\sum_{k\in Good} q_k = \Pr[K \in Good] \geq \frac{3}{4n^d} \quad (3.17)$$

Now, how does q_k compare to $\Pr[U_n = k]$? Since $\mathrm{H}_\infty(K) \geq n - c\log n$, we know that $q_k \leq n^c/2^n$ for each k. On the other hand, we know that $\Pr[U_n = k] = 1/2^n$ for each n. This means that for each $k \in$ *Good*,

$$\Pr[U_n = k] \geq \frac{1}{n^c} q_k \quad (3.18)$$

Then, we have

$$\Pr[U_n \in Good] = \sum_{k\in Good} Pr[U_n = k] \geq \sum_{k\in Good} \frac{1}{n^c} q_k = \frac{1}{n^c}\sum_{k\in Good} q_k \geq \frac{3}{4n^{d+c}} \quad (3.19)$$

It follows that the probability D' accepts F_k for k chosen according to U_n is at least $\frac{3}{4n^{d+c}}$.

Recall that the probability D' accepts a randomly chosen function f is at most $1/\exp(\frac{n}{8})$. This means that D' distinguishes F with advantage at least $3/(4n^{d+c}) - 1/\exp(\frac{n}{8})$. By our choice of n, this is at least $1/n^{c+d+1}$. □

3.5.5 Pseudo-random function generators with high-entropy seeds and leakage

We re-work Pietrzak's Lemma 6 [Pie09] using our Lemma 3.5.7, focusing on the case where the leakage length is $O(\log n)$ and the input to F has super-logarithmic min-entropy (rather than almost maximal min-entropy).

Lemma 3.5.8 *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudo-random function generator, and let $f : \{0,1\}^{2n} \to \{0,1\}^{c \log n}$ be a function. For all d, e, ℓ, and sufficiently large n, if K and R are independent distributions over $\{0,1\}^n$ such that $\mathrm{H}_\infty(K) \geq n - \ell \log n$ and $\mathrm{H}_\infty(R) \geq \log^2(n)$, then we have*

$$\Pr_{\alpha \leftarrow \langle R, f(K,R)\rangle}\left[\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d}, n^e}\left(F_K(R) | \langle R, f(K,R)\rangle = \alpha\right) \geq 2n - (c + 2d + 1)\log n\right] \geq 1 - \frac{1}{n^d} \quad (3.20)$$

Specifically, for sufficiently large n (depending only on c, d, e, and ℓ), given adversaries breaking (3.20), there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{\ell+2d+c+2}$.

Proof Fix d, e, ℓ, and n. Suppose (3.20) is false.

So we have

$$\Pr_{\alpha \leftarrow \langle R, f(K,R)\rangle}\left[\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d}, n^e}\left(F_K(R) | \langle R, f(K,R)\rangle = \alpha\right) < 2n - (c + 2d + 1)\log n\right] > \frac{1}{n^d} \quad (3.21)$$

Then we have with probability at least $1/(2n^d)$ over $r \leftarrow R$ that

$$\Pr_{\alpha \leftarrow \langle r, f(K,r)\rangle}\left[\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d}, n^e}\left(F_K(r) | \langle r, f(K,r)\rangle = \alpha\right) < 2n - (c + 2d + 1)\log n\right] > \frac{1}{2n^d} \quad (3.22)$$

For sufficiently large n, we have $1/(2n^d) > n^{2d+2c+3}/2^{\log^2(n)}$. Then, since $\mathrm{H}_\infty(R) \geq \log^2(n)$, there exists a set $S \subseteq \{0,1\}^n$ of size at least $n^{2d+2c+3}$ such that for each $r \in S$, (3.22) holds. Now, for sufficiently large n, we also have $2n - (c+2d+1)\log n < 2n - (c+2d)\log n - 3$. But then by Corollary 3.5.6, we have (for sufficiently large n depending only on d and e) that for each $r \in S$, there exists an adversary D_r of size $4n^{e+4d+3} < n^{e+4d+4}$ that distinguishes $F_K(r)$ from U_m with advantage at least $1/(8n^{2d+c}) > 1/n^{2d+c+1}$.

Applying Lemma 3.5.7 (with constants $c' = \ell$, $d' = 2d + c + 1$, $e' = e + 4d + 4$), we get (for sufficiently large n depending only on c, d, and ℓ) an adversary of size $n^{e'+2d'+2} = n^{e+8d+2c+8}$ that breaks F with advantage at least $1/n^{c'+d'+1} = 1/n^{\ell+2d+c+2}$. □

Corollary 3.5.9 *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudo-random function generator, and let $f : \{0,1\}^{2n} \to \{0,1\}^{c \log n}$ be a function. For all d, e, and sufficiently large n, if K and R are independent distributions over $\{0,1\}^n$ such that $\mathrm{H}_\infty(K) \geq n - (c + 3d + 1)\log n$ and $\mathrm{H}_\infty(R) \geq \log^2(n)$, then there exists a collection of distributions $\{Y_\alpha Z_\alpha\}_{\alpha \in \langle R, f(K,R)\rangle}$, where Y_α and Z_α are each over $\{0,1\}^n$, such that*

1. *For all* $\alpha \in \langle R, f(K,R)\rangle$*, we have* $\mathrm{H}_\infty(Y_\alpha Z_\alpha) \geq 2n - (c+2d+1)\log n$.

2. *For all* $\alpha \in \langle R, f(K,R)\rangle$ *and* $z \in Z_\alpha$*, we have* $\mathrm{H}_\infty(Y_\alpha | Z_\alpha = z) \geq n - (c+3d+1)\log n$.

3. *For all adversaries* D *of size* n^e*,*

$$\left| \Pr_{K,R}[D(F_K(R), R, f(K,R)) = 1] - \Pr_{\alpha \leftarrow \langle R, f(K,R)\rangle,\, Y_\alpha,\, Z_\alpha}[D(Y_\alpha Z_\alpha, \alpha) = 1] \right| \leq \frac{3}{n^d}$$

Specifically, for sufficiently large n *(depending only on* c*,* d*, and* e*), given adversaries breaking the above, there exists an adversary of size* $n^{e+8d+2c+8}$ *breaking* F *with advantage* $1/n^{5d+2c+3}$*.*

Proof Fix d, e, n. Suppose that there exist independent distributions K and R such that $\mathrm{H}_\infty(K) \geq n-(c+3d+1)\log n$ and $\mathrm{H}_\infty(R) \geq \log^2(n)$, but that for all collections of distributions $\{Y_\alpha Z_\alpha\}_{\alpha \in \langle R, f(K,R)\rangle}$ satisfying (1) and (2), there exists an adversary of size n^e breaking (3). Fix such distributions K and R.

Then, by Corollary 3.5.2, we have that for all collections of distributions $\{Y_\alpha Z_\alpha\}_{\alpha \in \langle R, f(K,R)\rangle}$ satisfying (1), there exists an adversary D of size n^e such that

$$\left| \Pr_{K,R}[D(F_K(R), R, f(K,R)) = 1] - \Pr_{\alpha \leftarrow \langle R, f(K,R)\rangle,\, Y_\alpha,\, Z_\alpha}[D(Y_\alpha Z_\alpha, \alpha) = 1] \right| > \frac{2}{n^d} \tag{3.23}$$

This means that for all collections of distributions $\{Y_\alpha Z_\alpha\}_{\alpha \in \langle R, f(K,R)\rangle}$ satisfying (1), there exists an adversary D of size n^e such that with probability greater than $1/n^d$ over $\alpha \leftarrow \langle R, f(K,R)\rangle$,

$$|\Pr[D(F_K(R), \alpha) = 1 | \langle R, f(K,R)\rangle = \alpha] - \Pr[D(Y_\alpha Z_\alpha, \alpha) = 1]| > \frac{1}{n^d} \tag{3.24}$$

It follows that

$$\Pr_{\alpha \leftarrow \langle R, f(K,R)\rangle}\left[\mathrm{H}^{\mathsf{HILL}}_{\frac{1}{n^d}, n^e}(F_K(R) | \langle R, f(K,R)\rangle = \alpha) < 2n - (c+2d+1)\log n\right] > \frac{1}{n^d} \tag{3.25}$$

But then by Lemma 3.5.8, there exists (for sufficiently large n depending only on c, d, and e) an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$. □

3.5.6 Main lemmas

We first prove a lemma that allows us to use an inductive approach to proving the theorem.

Lemma 3.5.10 *Let* $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ *be a function generator. Let* $\{Z_n\}$ *be such that for all* n*,* Z_n *is a distribution over strings of length* n *and* $\mathrm{H}_\infty(Z_n) \geq \log^2(n)$*. For all* $c > 0, d > 0, e > 0$*, every function* $p : \mathbb{N} \to \mathbb{N}$*, and sufficiently large* n *(depending only on* c*,* d*, and* e*), and for all* $(c\log n)$*-bounded adversaries* A *interacting as described in Section 3.4.2 and obtaining leakage for* $p(n)$ *rounds, and for some* $0 \leq i < p(n)$*, suppose there exists a distribution* $K'_0 K'_1 X'_1 K'_2 X'_2 \ldots K'_i X'_i | R_1 R_2 \ldots R_i$ *with each* X'_j *and each* K'_j *over* $\{0,1\}^n$*, such that the following properties hold:*

1. *For all* $1 \leq j \leq i$, *and all* $\alpha \leftarrow \mathtt{sim}_j^-$, $\mathrm{H}_\infty(K'_j X'_j | \mathtt{sim}_j^- = \alpha) \geq 2n - (c + 2d + 1) \log n$.
2. *For all* $0 \leq j \leq i$, *and all* $\beta \leftarrow \mathtt{sim}_j$, $\mathrm{H}_\infty(K'_j | \mathtt{sim}_j = \beta) \geq n - (c + 3d + 1) \log n$.
3. *For all adversaries* D *such that* $2 \cdot \mathtt{size}(A) + \mathtt{size}(D) + p(n)\mathtt{size}(F) \leq n^e$, *if*
$$\left| \Pr\left[D(\mathtt{real}^+_{p(n)}) = 1 \right] - \Pr\left[D(\mathtt{hybrid}^i_{p(n)}) = 1 \right] \right| > \frac{3i}{n^d}$$
then there exists an adversary of size $n^{e+8d+2c+8}$ *breaking* F *with advantage* $1/n^{5d+2c+3}$.

Then there exist distributions $K'_{i+1} X'_{i+1} | \mathtt{sim}^-_{i+1}$, *with* K'_{i+1} *and* X'_{i+1} *each over* $\{0,1\}^n$, *such that the following properties hold:*

1. *For all* $\alpha \leftarrow \mathtt{sim}^-_{i+1}$, $\mathrm{H}_\infty(K'_{i+1} X'_{i+1} | \mathtt{sim}^-_{i+1} = \alpha) \geq 2n - (c + 2d + 1) \log n$.
2. *For all* $\beta \leftarrow \mathtt{sim}_{i+1}$, $\mathrm{H}_\infty(K'_{i+1} | \mathtt{sim}_{i+1} = \beta) \geq n - (c + 3d + 1) \log n$.
3. *For all adversaries* D *such that* $2 \cdot \mathtt{size}(A) + \mathtt{size}(D) + p(n)\mathtt{size}(F) \leq n^e$, *if*
$$\left| \Pr\left[D(\mathtt{real}^+_{p(n)}) = 1 \right] - \Pr\left[D(\mathtt{hybrid}^{i+1}_{p(n)}) = 1 \right] \right| > \frac{3(i+1)}{n^d}$$
then there exists an adversary of size $n^{e+8d+2c+8}$ *breaking* F *with advantage* $1/n^{5d+2c+3}$.

Proof Fix $c > 0, d > 0, e > 0$, function $p : \mathbb{N} \to \mathbb{N}$, n, $(c \log n)$-bounded adversary A, and $0 \leq i < p(n)$. Suppose there exists a distribution $K'_0 K'_1 X'_1 K'_2 X'_2 \ldots K'_i X'_i | R_1 R_2 \ldots R_i$ satisfying the properties specified in the statement of the lemma.

Fix $\beta \leftarrow \mathtt{sim}_i$. Let K_i^β denote the distribution $K'_i | \mathtt{sim}_i = \beta$. By assumption, we have $\mathrm{H}_\infty(K_i^\beta) \geq 2n - (c + 3d + 1) \log n$. Now, applying Corollary 3.5.9, there exists (for sufficiently large n depending only on c, d, and e) a collection of distributions $\{K'_\alpha X'_\alpha\}_{\alpha \in \left\langle R_{i+1}, f_{i+1}(K_i^\beta, R_{i+1}) \right\rangle}$ such that: for all α, we have $\mathrm{H}_\infty(K'_\alpha X'_\alpha) \geq 2n - (c + 2d + 1) \log n$; for all α and $x \in X'_\alpha$, we have $\mathrm{H}_\infty(K'_\alpha | X'_\alpha = x) \geq n - (c + 3d + 1) \log n$; and for all adversaries D' of size n^e, if
$$\left| \Pr_{K_i^\beta, R_{i+1}} \left[D'(F_{K_i^\beta}(R_{i+1}), R_{i+1}, f_{i+1}(K_i^\beta, R_{i+1})) = 1 \right] - \Pr_{\alpha \leftarrow \left\langle R_{i+1}, f_{i+1}(K_i^\beta, R_{i+1}) \right\rangle, K'_\alpha, X'_\alpha} \left[D'(K'_\alpha X'_\alpha, \alpha) = 1 \right] \right| > \frac{3}{n^d}$$
then there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$.

Now let $K'_{i+1} X'_{i+1} | \mathtt{sim}^-_{i+1}$ be the distribution obtained by repeating the above argument over all values $\beta \leftarrow \mathtt{sim}_i$ and $\alpha \leftarrow \left\langle R_{i+1}, f_{i+1}(K_i^\beta, R_{i+1}) \right\rangle$. By this, we mean $K'_{i+1} X'_{i+1} | \mathtt{sim}^-_{i+1}$ are such that for each $\beta \leftarrow \mathtt{sim}_i$ and $\alpha \leftarrow \left\langle R_{i+1}, f_{i+1}(K_i^\beta, R_{i+1}) \right\rangle$, the distribution
$$K'_{i+1} X'_{i+1} | \mathtt{sim}^-_{i+1} = \langle \beta, \alpha \rangle$$

is the distribution given by the above argument instantiated with β and α.

Then, for all $\delta \leftarrow \mathtt{sim}^-_{i+1}$, we have $\mathrm{H}_\infty(K'_{i+1}X'_{i+1}|\mathtt{sim}^-_{i+1} = \delta) \geq 2n - (c + 2d + 1)\log n$. Also, for all $\beta \leftarrow \mathtt{sim}_{i+1}$, we have $\mathrm{H}_\infty(K'_{i+1}|\mathtt{sim}_{i+1} = \beta) \geq n - (c + 3d + 1)\log n$. In addition, for all adversaries D' of size n^e and all $\gamma \leftarrow \mathtt{sim}_i$, if

$$\Big|\Pr\Big[D'\Big(F_{K'_i}(R_{i+1}), R_{i+1}, f_{i+1}(K'_i, R_{i+1})\Big) = 1|\mathtt{sim}_i = \gamma\Big] \\ - \Pr\big[D'\big(K'_{i+1}X'_{i+1}, R_{i+1}, f_{i+1}(K'_i, R_{i+1})\big) = 1|\mathtt{sim}_i = \gamma\big]\Big| \quad > \frac{3}{n^d}$$

then there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$. It follows that for all adversaries D'' of size n^e, if

$$\Big|\Pr\Big[D''(\mathtt{sim}^-_{i+1}, F_{K'_i}(R_{i+1})) = 1\Big] - \Pr\big[D''(\mathtt{sim}^-_{i+1}, K'_{i+1}X'_{i+1}) = 1\big]\Big| > \frac{3}{n^d} \tag{3.26}$$

then there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$.

Now suppose there exists an adversary D such that $2\cdot\mathtt{size}(A)+\mathtt{size}(D)+p(n)\mathtt{size}(F) \leq n^e$ and

$$\Big|\Pr\Big[D(\mathtt{real}^+_{p(n)}) = 1\Big] - \Pr\Big[D(\mathtt{hybrid}^{i+1}_{p(n)}) = 1\Big]\Big| > \frac{3(i+1)}{n^d}$$

There are two cases to consider:

- **Case 1:** $\Big|\Pr\Big[D(\mathtt{real}^+_{p(n)}) = 1\Big] - \Pr\Big[D(\mathtt{hybrid}^{i}_{p(n)}) = 1\Big]\Big| > \frac{3i}{n^d}$

  But then by our assumptions, there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$.

- **Case 2:** $\Big|\Pr\Big[D(\mathtt{hybrid}^{i}_{p(n)}) = 1\Big] - \Pr\Big[D(\mathtt{hybrid}^{i+1}_{p(n)}) = 1\Big]\Big| > \frac{3}{n^d}$

  We will construct an adversary D'' satisfying (3.26). On input skx, where $x, k \in \{0,1\}^n$, D'' treats s as a sample of $\mathtt{sim}^-_{i+1}$, and treats kx as the string produced in step 2 of round $i+1$ of the stream cipher construction. Then, using A and F, it continues the interaction between A and the construction (starting at round $i+2$) for another $p(n)+1-(i+1) = p(n)-i$ rounds, and produces a string t encoding a transcript of this interaction. Finally, D'' runs D on input sxt , and outputs whatever D outputs.

  Note that since D'' runs D and A, and must carry out the stream cipher construction for $(p(n)-i)$ rounds as well as evaluate the leakage functions produced by A in these rounds, we have $\mathtt{size}(D'') \leq \mathtt{size}(D)+2\cdot\mathtt{size}(A)+(p(n)-i)\mathtt{size}(F)$, and hence $\mathtt{size}(D'') \leq n^e$.

  Also, observe that when the input to D'' has distribution $(\mathtt{sim}^-_{i+1}, F_{K'_i}(R_{i+1}))$, then the input D'' gives to D has distribution $\mathtt{hybrid}^{i}_{p(n)}$. On the other hand, when the input to D'' has distribution $(\mathtt{sim}^-_{i+1}, K'_{i+1}X'_{i+1})$, then the input D'' gives to D has distribution $\mathtt{hybrid}^{i+1}_{p(n)}$. This means that D'' satisfies (3.26). Also, D'' can be made deterministic by a standard argument. It follows that there exists an adversary of size $n^{e+8d+2c+8}$ that breaks F with advantage $1/n^{5d+2c+3}$.

So in both cases, there exists an adversary of size $n^{e+8d+2c+8}$ that breaks F with advantage $1/n^{5d+2c+3}$. $\square$

We now show that the final two hybrids, $\mathtt{hybrid}_{p(n)}^{p(n)}$ and $\mathtt{hybrid}_{p(n)}^{p(n)+1}$, are also "close".

Lemma 3.5.11 *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ be a function generator. Let $\{Z_n\}$ be such that for all n, Z_n is a distribution over strings of length n and $\mathrm{H}_\infty(Z_n) \geq \log^2(n)$. For all $c > 0, d > 0, e > 0$, all functions $p : \mathbb{N} \to \mathbb{N}$, and sufficiently large n (depending only on c and d), and for all $(c \log n)$-bounded adversaries A interacting as described in Section 3.4.2, obtaining leakage for $p(n)$ rounds, suppose there exists a distribution $K'_0 K'_1 X'_1 K'_2 X'_2 \dots K'_{p(n)} X'_{p(n)} | R_1 R_2 \dots R_{p(n)}$ with each X'_i and each K'_i over $\{0,1\}^n$, such that for all $\beta \leftarrow \mathtt{sim}_{p(n)}$, we have $\mathrm{H}_\infty(K'_{p(n)} | \mathtt{sim}_{p(n)} = \beta) \geq n - (c + 3d + 1) \log n$. Then, for all adversaries D of size n^e, if*

$$\left| \Pr\left[D(\mathtt{hybrid}_{p(n)}^{p(n)}) = 1 \right] - \Pr\left[D(\mathtt{hybrid}_{p(n)}^{p(n)+1}) = 1 \right] \right| > \frac{2}{n^d}$$

then there exists an adversary of size n^{e+2d+2} breaking F with advantage $1/n^{4d+c+2}$.

Proof Fix $c > 0, d > 0, e > 0$, function $p : \mathbb{N} \to \mathbb{N}$, n, $(c \log n)$-bounded adversary A, and $1 \leq i < p(n)$. Suppose there exists a distribution $K'_0 K'_1 X'_1 K'_2 X'_2 \dots K'_{p(n)} X'_{p(n)} | R_1 R_2 \dots R_{p(n)}$ satisfying the properties specified in the statement of the lemma.

Suppose there exists an adversary D of size n^e such that

$$\left| \Pr\left[D(\mathtt{hybrid}_{p(n)}^{p(n)}) = 1 \right] - \Pr\left[D(\mathtt{hybrid}_{p(n)}^{p(n)+1}) = 1 \right] \right| > \frac{2}{n^d}$$

By the definitions of $\mathtt{hybrid}_{p(n)}^{p(n)}$ and $\mathtt{hybrid}_{p(n)}^{p(n)+1}$, we have that

$$\begin{aligned} &\Big| \Pr\left[D(\mathtt{sim}_{p(n)}, R_{p(n)+1}, F_{K'_{p(n)}}(R_{p(n)+1}) = 1 \right] - \\ &\Pr\left[D(\mathtt{sim}_{p(n)}, R_{p(n)+1}, K''_{p(n)+1}, X''_{p(n)+1}) = 1 \right] \Big| \qquad > \frac{2}{n^d} \end{aligned}$$

where $K''_{p(n)+1}$ and $X''_{p(n)+1}$ are independent random variables, each uniformly distributed over $\{0,1\}^n$. Then, there must exist $\beta \leftarrow \mathtt{sim}_{p(n)}$ such that, letting $K^\beta_{p(n)}$ denote the distribution $K'_{p(n)} | \mathtt{sim}_{p(n)} = \beta$, we have

$$\left| \Pr\left[D(\beta, R_{p(n)+1}, F_{K^\beta_{p(n)}}(R_{p(n)+1}) = 1 \right] - \Pr\left[D(\beta, R_{p(n)+1}, K''_{p(n)+1}, X''_{p(n)+1}) = 1 \right] \right| > \frac{2}{n^d}$$

Fix such β. Then we have with probability at least $1/n^d$ over $r \leftarrow R_{p(n)+1}$ that

$$\left| \Pr\left[D(\beta, r, F_{K^\beta_{p(n)}}(r) = 1 \right] - \Pr\left[D(\beta, r, K''_{p(n)+1}, X''_{p(n)+1}) = 1 \right] \right| > \frac{1}{n^d} \tag{3.27}$$

For sufficiently large n, we have $1/n^d > n^{2d+1}/2^{\log^2(n)}$. Then, recalling that $R_{p(n)+1} \sim Z_n$ and $\mathrm{H}_\infty(Z_n) \geq \log^2(n)$, there exists a set $S \subseteq \{0,1\}^n$ of size at least n^{2d+1} such that for all $r \in S$, (3.27) holds. Also, we have by assumption that $\mathrm{H}_\infty(K^\beta_{p(n)}) \geq n - (c + 3d + 1) \log n$.

Applying Lemma 3.5.7 (with constants $c' = c + 3d + 1$, $d' = d$, $e' = e$), there exists (for sufficiently large n depending only on c and d) an adversary D' of size n^{e+2d+2} that breaks F with advantage at least $1/n^{4d+c+2}$. □

3.5.7 Finishing up

We now have everything we need to prove Theorem 3.4.1.

Proof (Theorem 3.4.1) Fix $c > 0, d > 0, e > 0$, and function $p : \mathbb{N} \to \mathbb{N}$. Fix n large enough (depending on c, d, and e) as required by Lemma 3.5.10 and Lemma 3.5.11. Fix $(c \log n)$-bounded adversary A. Let K'_0 be distributed uniformly over $\{0,1\}^n$.

We would like to apply Lemma 3.5.10 for the case $i = 0$. To do so, we need to ensure that the three properties required by this lemma are satisfied. Observe that property 1 is vacuously true for $i = 0$. Property 2 is satisfied by our choice of K'_0. Property 3 is trivially satisfied, since $\mathtt{real}^+_{n^b} = \mathtt{hybrid}^0_{n^b}$ when K'_0 is uniformly distributed. So we can apply Lemma 3.5.10 for the case $i = 0$. But then this gives us everything we need to apply Lemma 3.5.10 for the case $i = 1$. We continue in this fashion, repeatedly applying Lemma 3.5.10 for $i = 2, 3, \ldots, p(n) - 1$, using the previous applications of the Lemma to satisfy the properties needed each time. (We can view this as complete induction.) This gives us distributions $K'_1, X'_1, K'_2, X'_2, \ldots, K'_{p(n)}, X'_{p(n)}$ satisfying properties 1 and 2, that also satisfy property 4 for $1 \leq i \leq p(n)$. Then, by applying Lemma 3.5.11, we see that these distributions also satisfy property 4 for $i = p(n) + 1$.

We next show that these distributions satisfy property 3. Fix $1 \leq i \leq p(n)$, and consider X'_i. We have by property 1 that for all $\alpha \leftarrow \mathtt{sim}^-_i$, $\mathrm{H}_\infty(K'_i X'_i | \mathtt{sim}^-_i = \alpha) \geq 2n - (c + 2d + 1) \log n$. Then, by Lemma 3.5.1, we have that for all $\alpha \leftarrow \mathtt{sim}^-_i$,

$$\Pr_{k \leftarrow K'_i}\left[\mathrm{H}_\infty(X'_i | \mathtt{sim}^-_i = \alpha \wedge K'_i = k) \geq n - (c + 3d + 1) \log n\right] \geq 1 - \frac{1}{n^d}$$

That is, property 3 is satisfied.

It remains to consider property 5. Suppose there exists an adversary D such that we have $2 \cdot \mathtt{size}(A) + \mathtt{size}(D) + p(n)\mathtt{size}(F) \leq n^e$ and

$$\left|\Pr\left[D(\mathtt{real}^+_{p(n)}) = 1\right] - \Pr\left[D(\mathtt{real}_{p(n)}, R_{p(n)+1}, K''_{p(n)+1}, X''_{p(n)+1}) = 1\right]\right| > \frac{6p(n) + 6}{n^d}$$

There are two cases to consider:

- **Case 1:** $\left|\Pr\left[D(\mathtt{real}^+_{p(n)}) = 1\right] - \Pr\left[D(\mathtt{hybrid}^{p(n)+1}_{p(n)}) = 1\right]\right| > \frac{3p(n)+3}{n^d}$
  But then by property 4, there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$.

- **Case 2:** $\left|\Pr\left[D(\texttt{real}^{+}_{p(n)}) = 1\right] - \Pr\left[D(\texttt{hybrid}^{p(n)+1}_{p(n)}) = 1\right]\right| \leq \frac{3p(n)+3}{n^d}$
Then we must have that

$$\left|\Pr\left[D(\texttt{real}_{p(n)}, R_{p(n)+1}, K''_{p(n)+1}, X''_{p(n)+1}) = 1\right] - \Pr\left[D(\texttt{hybrid}^{p(n)+1}_{p(n)}) = 1\right]\right| > \frac{3p(n)+3}{n^d}$$

By definition of $\texttt{hybrid}^{p(n)+1}_{p(n)}$, we then have

$$\begin{aligned}&\Big|\Pr\left[D(\texttt{real}_{p(n)}, R_{p(n)+1}, K''_{p(n)+1}, X''_{p(n)+1}) = 1\right]\\ &- \Pr\left[D(\texttt{sim}_{p(n)}, R_{p(n)+1}, K''_{p(n)+1}, X''_{p(n)+1}) = 1\right]\Big| \qquad > \frac{3p(n)+3}{n^d}\end{aligned}$$

But then, since $K''_{p(n)+1}$ and $X''_{p(n)+1}$ are independent distributions, there exist particular strings $k, x \in \{0,1\}^n$ such that

$$\left|\Pr\left[D(\texttt{real}_{p(n)}, R_{p(n)+1}, k, x) = 1\right] - \Pr\left[D(\texttt{sim}_{p(n)}, R_{p(n)+1}, k, x) = 1\right]\right| > \frac{3p(n)+3}{n^d}$$

This allows us to construct an adversary D' that distinguishes $\texttt{real}^{+}_{p(n)}$ and $\texttt{hybrid}^{p(n)+1}_{p(n)}$ with advantage $(3p(n)+3)/n^d$. We construct D' as follows. On input α (of the same length as samples of $\texttt{real}^{+}_{p(n)}$), D' lets α_1 be the first $|\alpha| - 2n$ bits of α, runs D' on input (α_1, k, x) and outputs whatever D outputs. By construction, we have

$$\left|\Pr\left[D'(\texttt{real}^{+}_{p(n)}) = 1\right] - \Pr\left[D'(\texttt{hybrid}^{p(n)+1}_{p(n)}) = 1\right]\right| > \frac{3p(n)+3}{n^d}$$

We also have $\texttt{size}(D') = \texttt{size}(D)$, and hence $2 \cdot \texttt{size}(A) + \texttt{size}(D') + p(n)\texttt{size}(F) \leq n^e$. Then, by property 4, there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$.

So in both cases, there exists an adversary of size $n^{e+8d+2c+8}$ breaking F with advantage $1/n^{5d+2c+3}$.

□

3.6 Proof of Theorem 3.2.1

We begin by giving an overview of the proof.

3.6.1 Proof overview

First, consider a setting where there is no leakage. Observe that in the absence of leakage, our authenticated session protocol is secure even when the high min-entropy distribution Z_n is replaced with a distribution of zero entropy (that is, a distribution that assigns all weight to a single string). Here we simply use the fact that, in the absence of leakage, the output of pseudo-random function generator F with a randomly chosen seed and an arbitrary input is, by definition, indistinguishable from a randomly chosen string. It follows by induction that for every i, the strings X_i, K_i computed by the i-th invocation of EvalB_2 are indistinguishable from random strings. But an adversary that breaks our protocol is able to predict an output of some F'_{X_i} and hence breaks the pseudo-randomness of F'.

The above idea breaks down when there is leakage – when Z_n has zero entropy, the computation of X_i, K_i is deterministic, and hence the adversary can use leakage on rounds prior to some round i in order to output bits of X_i. This means that the adversary can eventually learn an entire X_i and trivially break the authentication protocol. Now, by requiring Z_n to be a distribution of high min-entropy, we can attempt to recover some of the above intuition even in a setting with leakage. This is more subtle than it might first seem. To see this, suppose we modify our protocol so that strings from high min-entropy distribution Z_n are sampled by A and sent to B over the public channel (instead of vice-versa). In this case, the adversary can simply run A for several rounds (without running B yet), accumulating its output. At this point, the adversary knows all the strings sampled from distribution Z_n for those rounds, and hence the computation performed by B for those rounds will be deterministic from the adversary's perspective. This means that the adversary can once again mount an attack where he uses leakage on B on rounds prior to some round i in order to learn X_i. We overcome this problem by having B sample strings from Z_n and send these strings to A; the idea is that this introduces probabilism *and* essentially forces the adversary to properly interleave calls to EvalB_1, EvalA and EvalB_2.

Observe that if the adversary chooses to properly interleave the scheduling of EvalB_1, EvalA and EvalB_2, and also properly passes on the output of EvalB_1 as input to EvalA, then the leakage he obtains on consecutive calls to EvalB_1, EvalA and EvalB_2 can easily be combined into a single leakage of three times the output length. The idea is that in this case, the state of A and the state of B remain equal to each other, and hence a single leakage (of sufficiently long output) on this common state suffices. But now we can essentially view the adversary as interacting with a single instance of the stream cipher we described in Section 3.4, for some number j of rounds. Furthermore, we can disallow leakage in the final two rounds (or indeed, in any constant number of rounds) with only a polynomial loss in the adversary's success probability, since the adversary can simply attempt to guess the leakage for these rounds. We can then use Theorem 3.4.1 to argue that the adversary also does well when the string K_{j-1} produced by the parties in the second-last round is replaced with a randomly chosen string. But then, in the final round, we are in a situation very similar to the leakage-free setting we discussed at the beginning of this section: since K_{j-1} is randomly chosen, and since there is no leakage, the string X_j produced by EvalB_2 will be indistinguishable from random, and hence an adversary that succeeds in round j breaks the pseudo-randomness of F'.

What if the adversary does not properly interleave the scheduling of EvalB_1, EvalA and EvalB_2, or what if he does not properly pass on the output of EvalB_1 as an input to EvalA? We argue that an adversary that does not properly interleave scheduling must correctly guess an output of EvalB_1 in order to prevent the following invocation of EvalB_2 from outputting `Fail`; however, the output of EvalB_1 is chosen according to distribution Z_n which has high min-entropy and hence is unpredictable by definition. To handle the case of an adversary that does not properly pass on the output of EvalB_1 to EvalA, we first relax the conditions needed for the adversary to succeed so that if EvalB_2 does not output `Fail` in a round where the adversary does not properly pass on the output of EvalB_1 to EvalA, the adversary is considered to have succeeded. With this change, a round where the adversary does not properly pass on the output of EvalB_1 to EvalA *is* the final round. Finally, to handle the case of an adversary not properly passing on the output of EvalB_1 to EvalA in the final round j, we observe that this does not prevent us from applying the reasoning of the previous paragraph to argue that we can make the final two rounds leakage-free and replace the string K_{j-1} produced by the parties in round $j-1$ with a randomly chosen string. Then, it is not difficult to use the pseudo-randomness of F and F' to argue that the failure of the adversary to pass on the output of EvalB_1 to EvalA in the final round does not provide the adversary with any benefit.

3.6.2 Proof details

Let $c > 0$ and let $C = \{C_n\}$ be a $(c \log n)$-bounded adversary that breaks SP (in the sense described in Definition 15). Then we have $q_C(n) > 1/n^d$ for some d and infinitely many n.

We will describe a sequence of experiments where the first experiment is the security experiment for leakage-resilient authenticated session protocols (that is, the experiment that gives rise to $q_C(n)$) and the final security experiment is one in which an adversary that "does well" yields an adversary that breaks F'. We show that for every pair of consecutive experiments in the sequence, an adversary that "does well" in the first experiment either yields an adversary "doing well" in the second experiment or yields an adversary that breaks F.

We first introduce some notation. We write $\mathtt{size}_n(F)$ and $\mathtt{size}_n(F')$ to denote the size of circuits computing F and F' on seeds of length n. For $i \geq 0$, we denote by $K_{A,i}$ the state of party A before invocation $i+1$ of EvalA, and we denote by $K_{B,i}$ the state of party B before invocation $i+1$ of EvalB_1 (note that this is also the state of B before the invocation $i+1$ of EvalB_2 since EvalB_1 does not change the state of B in our construction SP). For $i \geq 1$, we denote by r_i the string output to the adversary by invocation i of EvalB_1, we denote by m_i and r_i' the strings given by the adversary as input to invocation i of EvalA, we denote by $e = \langle m_i, \alpha_i \rangle$ the string output to the adversary by invocation i of EvalA, we denote by

$e' = \langle m'_i, \alpha'_i \rangle$ the string given by the adversary as input to invocation i of EvalB_2, we denote by $X_{A,i}$ the string computed by invocation i of EvalA as the rightmost n bits of $F_{K_{A,i-1}}(r'_i)$, and we denote by $X_{B,i}$ the string computed by invocation i of EvalB_2 as the rightmost n bits of $F_{K_{B,i-1}}(r_i)$. Turning our attention to leakage functions and their outputs, for $i \geq 1$, we denote by $f_{B1,i}$ the leakage function submitted by the adversary for invocation i of EvalB_1, we denote by $leak_{B1,i}$ the value $f_{B1,i}(K_{B,i-1}, r_i)$, we denote by $f_{A,i}$ the leakage function submitted by the adversary for invocation i of EvalA, we denote by $leak_{A,i}$ the value $f_{A,i}(K_{A,i-1})$, we denote by $f_{B2,i}$ the leakage function submitted by the adversary for invocation i of EvalB_2, and we denote by $leak_{B2,i}$ the value $f_{B2,i}(K_{B,i-1})$.

Consider the following modified version of the security experiment for SP, where we now make it easier for the adversary to win. Informally, the main difference is that now the adversary also wins if he chooses $r'_i \neq r_i$ but invocation i of EvalB_2 does not `Fail`.

Experiment $Exp1$**.** Experiment $Exp1$ proceeds exactly as the security experiment for SP, except we modify the conditions under which the experiment terminates immediately after an execution of EvalB_2. Previously, the experiment terminates after invocation i of EvalB_2 if at least one of the following conditions holds: 1) invocation i of EvalB_2 outputs `Fail`; 2) EvalA has been invoked fewer than i times; or 3) $m_i \neq m'_i$. In $Exp1$, the experiment terminates after invocation i of EvalB_2 if one of the previous conditions (1) or (2) holds, or if one of the following conditions holds: 3$'$) $e_i \neq e'_i$; or 4) $r_i \neq r'_i$. Observe that (3$'$) holds whenever (3) holds. It follows that whenever at least one of the termination conditions for the previous experiment is satisfied, one of the stopping conditions for $Exp1$ is satisfied.

Let $D = \{D_n\}$ be an adversary interacting as in $Exp1$. Define $q^1_D(n)$ to be the probability that, letting j denote the number of times that EvalB_2 is invoked by D_n during the experiment, we have that the j-th invocation of EvalB_2 does not output `Fail` and either EvalA has been invoked fewer than j times, or $r'_j \neq r_j$, or $e'_j \neq e_j$.

Lemma 3.6.1 *Let $D = \{D_n\}$ be an adversary. Then, for all n, we have $q^1_D(n) \geq q_D(n)$.*

Proof First, observe that whenever experiment $Exp1$ is terminated due only to one of the new stopping conditions (that is, condition (3$'$) or (4) holds, but (1), (2), and (3) do not hold, and, in particular, EvalB_2 does not output `Fail`), the definition of q^1_D considers such a run to be a success for the adversary. To conclude, it suffices to observe that whenever $Exp1$ is *not* terminated due only to one of the new stopping conditions (that is, it is terminated due either to conditions (1), (2), or (3), or because the adversary simply chooses to stop), it proceeds exactly as the security experiment for SP, and furthermore, since $m'_j \neq m_j$ implies $e'_j \neq e_j$, the winning condition implicit in the definition of q_D^1 is achieved if the winning condition implicit in the definition of q_D is achieved. It follows that for all n, we have $q_D^1(n) \geq q_D(n)$. $\square$

We now define an experiment where the adversary is required to be passive until after his next-to-last invocation of EvalB_2; by "passive", we mean that he chooses leakage functions and message pieces m_i, sees the communication over the public channel, but he does not control scheduling and cannot change the contents of the public channel.

Experiment $Exp2$**.** The adversary D consists of a circuit family $\{D_n\}$ and a sequence of integers $\{j_n\}$. For each n, the experiment proceeds as follows. A string $K_0 \in \{0,1\}^n$ is randomly chosen. Then the experiment proceeds in a sequence of j_n rounds.

For $1 \leq i \leq j_n - 2$, round i proceeds as follows. D_n produces the description of a circuit $f_{B1,i} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{c \log n}$. Then $r_i \leftarrow Z_n$ is chosen, $leak_{B1,i} \leftarrow f_{B1,i}(K_{i-1}, r_i)$ is computed, and D_n is given r_i and $leak_{B1,i}$. D_n then produces the description of a circuit $f_{A,i} : \{0,1\}^n \to \{0,1\}^{c \log n}$ and a string $m_i \in \{0,1\}^n$. Then, $(e_i, K_i) \leftarrow \mathsf{EvalA}(K_{i-1}, m_i, r_i)$ and $leak_{A,i} \leftarrow f_{A,i}(K_{i-1})$ are computed, and D_n is given e_i and $leak_{A,i}$. D_n then produces the description of a circuit $f_{B2,i} : \{0,1\}^n \to \{0,1\}^{c \log n}$. Finally, $leak_{B2,i} \leftarrow f_{B1,i}(K_{i-1})$ is computed and given to D_n.

Then, round $j_n - 1$ proceeds as the previous rounds, except that after being given e_{j_n-1} and $leak_{j_n-1}$ and before having to produce f_{B2,j_n-1}, D_n is given the opportunity to invoke EvalA an additional time if he wishes. If he chooses to do so, he produces the description of a circuit $f_{A,j_n} : \{0,1\}^n \to \{0,1\}^{c \log n}$, and strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$; then $(e_{j_n}, K_{j_n}) \leftarrow \mathsf{EvalA}(K_{j_n-1}, m_{j_n}, r'_{j_n})$ and $leak_{A,j_n} \leftarrow f_{A,j_n}(K_{j_n-1})$ are computed; then D_n is given e_{j_n}, $leak_{A,j_n}$, *and he is also given* K_{j_n}.

Next, in round j_n, the experiment proceeds as follows. If D_n elected to invoke EvalA an additional time in round $j_n - 1$, then in round j_n he is not allowed to invoke EvalA; otherwise, he may invoke EvalA at most once. He must also invoke EvalB_1 exactly once. If he is invoking both EvalA and EvalB_1, the order in which he does this is his choice. When D_n invokes EvalB_1, the experiment proceeds as follows: D_n produces the description of a circuit $f_{B1,j_n} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{c \log n}$; then $r_{j_n} \leftarrow Z_n$ is chosen, $leak_{B1,j_n} \leftarrow f_{B1,j_n}(K_{i-1}, r_{j_n})$ is computed, and D_n is given r_{j_n} and $leak_{B1,j_n}$. If and when D_n invokes EvalA, the experiment proceeds as follows: D_n produces the description of a circuit $f_{A,j_n} : \{0,1\}^n \to \{0,1\}^{c \log n}$, and strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$; then $(e_{j_n}, K_{j_n}) \leftarrow \mathsf{EvalA}(K_{j_n-1}, m_{j_n}, r'_{j_n})$ and $leak_{A,j_n} \leftarrow f_{A,j_n}(K_{j_n-1})$ are computed; then D_n is given e_{j_n}, $leak_{A,j_n}$, *and he is also given* K_{j_n}.

Finally, D_n produces a string $e'_{j_n} = \langle m'_{j_n}, \alpha'_{j_n} \rangle \in \{0,1\}^{2n}$.

Define $q_D^2(n)$ to be the probability that $\mathsf{EvalB}_2(K_{j_n-1}, r_{j_n}, e'_{j_n})$ does not output `Fail` and either D_n elected not to invoke EvalA in round j_n, or $r'_{j_n} \neq r_j$, or $e'_{j_n} \neq e_{j_n}$.

Lemma 3.6.2 *For every $e > 0$ and every adversary $D = \{D_n\}$ for $Exp1$ such that D_n has size at most n^e, there exists an adversary D' for $Exp2$ consisting of a circuit family $\{D'_n\}$ of size at most $n^{2e} + n^e(\mathtt{size}_n(F) + \mathtt{size}_n(F'))$ and a sequence $\{j_n\}$ such that $q^2_{D'}(n) \geq \frac{q^1_D(n)}{n^e} - \frac{1}{2^{\log^2(n)}}$ for all n.*

Proof Let $e > 0$ and let $D = \{D_n\}$ be an adversary for $Exp1$ such that D_n has size at most n^e. Fix $n > 0$. We will define a circuit D'_n and an integer j_n for $Exp2$.

Note that since D_n is of size at most n^e, the number of times it invokes EvalB_2 must be at most n^e. Now, for $1 \leq i \leq n^e$, define $q^1_{D,i}(n)$ to be the probability (when D_n is run according to $Exp1$) that the number of times EvalB_2 is invoked is exactly i, the i-th invocation of EvalB_2 does not output `Fail`, and either EvalA has been invoked fewer than i times, or $r'_i \neq r_i$, or $e'_i \neq e_i$. Observe that we must $q^1_D(n) = \sum_{1 \leq i \leq n^e} q^1_{D,i}(n)$. It follows that there must be an i, $1 \leq i \leq n^e$, such that $q^1_{D,i}(n) \geq q^1_D(n)/n^e$. Define j_n to be such an i.

For $i > 0$, we say that D_n *behaves non-passively in round i* if he does one of the following: invokes EvalA for the i-th time before invoking EvalB_1 for the i-th time; invokes EvalA for the $(i+1)$-st time before invoking EvalB_2 for the i-th time; invokes EvalB_2 for the i-th time before invoking EvalA for the i-th time; provides a string $r'_i \neq r_i$ as an input to the i-th invocation of EvalA; or provides a string $e'_i \neq e_i$ as an input to the i-th invocation of EvalB_2.

We now define D'_n. D'_n begins by running D_n for $j_n - 2$ rounds or until D_n behaves non-passively, whichever comes first. Specifically, D'_n uses the leakage functions and message pieces produced by D_n as the leakage functions and message pieces it is expected to produce, and D'_n provides D_n with all the output it is given. If D_n behaves non-passively in any of these rounds, D'_n stops running D_n and instead continues on its own in some arbitrary manner (for example, using 0^n as each remaining message piece and using a constant function as each remaining leakage function).

In round $j_n - 1$, D'_n slightly relaxes the passiveness requirement on D_n. Specifically, after D_n invokes EvalA for the $(j_n - 1)$-st time, D'_n allows D_n to invoke EvalA additional times, on inputs of its choice, before invoking EvalB_2 for the $(j_n - 1)$-st time. To handle the first such additional invocation, D'_n simply uses the additional call to EvalA that it is allowed to make in round $j_n - 1$ of $Exp2$. Recall that in $Exp2$, after this additional call, one of the strings given to D'_n is K_{j_n}; D'_n uses this string to simulate any futher invocations of EvalA made by D_n. If D_n exhibits any other kind of non-passive behavior during round $j_n - 1$, D'_n stops running D_n and instead continues on its own in some arbitrary manner.

Then, in round j_n D'_n continues running D_n until it invokes EvalB_2 one more time; during this phase D'_n does not care if D_n is non-passive, since D'_n itself is allowed to behave non-passively at this point in $Exp2$. As before, D'_n uses the leakage functions output by D_n as its own leakage functions, and passes on the outputs it receives to D_n. Invocations of EvalA by D_n

are handled as follows: if D_n made additional invocations of EvalA during the previous round, the D'_n is able to simulate invocations of EvalA since it was given the string K_{j_n}; otherwise, the first time (if any) that D_n invokes EvalA during this phase, D'_n uses the message piece m_{j_n} and string r'_{j_n} produced by D_n as the strings it is expected to produce for this invocation of EvalA, and uses the string K_{j_n} it is given after this invocation to simulate future invocations of EvalA. When D_n invokes EvalB_2, D'_n uses the string e'_{j_n} produced by D_n as the final string it is expected to produce.

Observe that since D'_n runs D_n and also needs to simulate any invocations of EvalA made by D_n after the j_n-th invocation of EvalA, we have that D'_n is of size at most $n^e(n^e + \mathtt{size}_n(F) + \mathtt{size}_n(F')) = n^{2e} + n^e(\mathtt{size}_n(F) + \mathtt{size}_n(F'))$.

Note that as long as D_n behaves passively, it is properly simulated according to $Exp1$ when D'_n is run according to $Exp2$. Furthermore, observe that if D_n behaves non-passively in some round i, then in $Exp1$ we have that the experiment ends after the i-th invocation of EvalB_2 *unless* one of the following happens: 1) the non-passive behavior consists of D_n invoking EvalA for the i-th time before invoking EvalB_1 for the i-th time *and* nevertheless $r'_i = r_i$, that is, D_n predicts r_i before seeing it; or 2) the non-passive behavior consists of D_n invoking EvalA for the $(i+1)$-st time before invoking EvalB_2 for the i-th time. First consider (1): since each r_i is chosen according to Z_n and since $\mathrm{H}_\infty(Z_n) \geq \log^2(n)$, the probability that a prediction of r_i is correct is at most $1/2^{\log^2(n)}$. Now, note that if (2) occurs, then we have that D_n invokes EvalA for the $(i+1)$-st time before invoking EvalB_1 for the $(i+1)$-st time; hence, the experiment will end after the $(i+1)$-st invocation of EvalB_2 unless D_n successfully predicts r_{i+1}, and this occurs with probability at most $1/2^{\log^2(n)}$.

That is, whenever D'_n stops running D_n in some round $i \leq j_n - 2$ because of non-passive behavior, the probability that, if we instead continued running D_n according to $Exp1$ then the experiment would continue past the $(i+1)$-st invocation of EvalB_2 is at most $1/2^{\log^2(n)}$; in particular, the probability that a j_n-th invocation of EvalB_2 would occur is at most $1/2^{\log^2(n)}$. Furthermore, note that whenever D'_n stops running D_n in round $j_n - 1$ because of non-passive behavior, it will not be in situation (2), since this form of non-passiveness is allowed in round $j_n - 1$. This means that whenever D'_n stops running D_n in round $j_n - 1$ because of non-passive behavior, the probability that, if we instead continued running D_n according to $Exp1$ then the experiment would continue past the $(j_n - 1)$-st invocation of EvalB_2 is at most $1/2^{\log^2(n)}$; in particular, the probability that a j_n-th invocation of EvalB_2 would occur is at most $1/2^{\log^2(n)}$. We conclude that the probability that the stopping of D_n by D'_n prevents D_n from invoking EvalB_2 a j_n-th time when it would have done so had it been allowed to continue running according to $Exp1$ is at most $1/2^{\log^2(n)}$.

We conclude that $q^2_{D'}(n) \geq q^1_{D,j_n}(n) - \frac{1}{2^{\log^2(n)}} \geq \frac{q^1_D(n)}{n^e} - \frac{1}{2^{\log^2(n)}}$.

□

We now define an experiment where there is no leakage in the final two rounds.

Experiment $Exp3$**.** The adversary D consists of a circuit family $\{D_n\}$ and a sequence of integers $\{j_n\}$. For each n, the experiment proceeds exactly as experiment $Exp2$, except that in rounds $j_n - 1$ and j_n, D_n does not produce circuits for leakage functions (that is, he does not produce f_{B1,j_n-1}, f_{A,j_n-1}, f_{B2,j_n-1}, f_{B1,j_n}, or f_{A,j_n}) and hence is not given output for such functions. We define $q_D^3(n)$ in $Exp3$ identically to $q_D^2(n)$ in $Exp2$; that is, $q_D^3(n)$ is the probability that $\mathsf{EvalB}_2(K_{j_n-1}, r_{j_n}, e'_{j_n})$ does not output `Fail` and either D_n elected not to invoke EvalA in round j_n, or $r'_{j_n} \neq r_j$, or $e'_{j_n} \neq e'_{j_n}$.

Lemma 3.6.3 *Let D be an adversary for $Exp2$. Then there exists an adversary D' for $Exp3$ such that D' has the same size as D and $q_{D'}^3(n) \geq q_D^2(n)/n^{5c}$ for all n.*

Proof Let circuit family $\{D_n\}$ and sequence $\{j_n\}$ be an adversary for $Exp2$. We will define a circuit family $\{D'_n\}$ such that $\{D'_n\}$ and $\{j_n\}$ is an adversary for $Exp3$. Fix $n > 0$. We first describe a probabilistic circuit D'_n and then observe that it can be made deterministic.

Circuit D'_n works as follows. It randomly selects $z_{B1,j_n-1}, z_{A,j_n-1}, z_{B2,j_n-1}, z_{B1,j_n}, z_{A,j_n} \in \{0,1\}^{c \log n}$. Then D'_n simulates D_n, using the leakage functions and message pieces produced by D_n as the leakage functions and message pieces it is expected to produce, and D'_n provides D_n with all the output it is given. However, in rounds $j_n - 1$ and j_n, D'_n is not allowed to produce leakage functions, and hence it uses $z_{B1,j_n-1}, z_{A,j_n-1}, z_{B2,j_n-1}, z_{B1,j_n}$, and z_{A,j_n} as responses to the functions $f_{B1,j_n-1}, f_{A,j_n-1}, f_{B2,j_n-1}, f_{B1,j_n}$, and f_{A,j_n} produced by D_n. At the end, D'_n outputs the string e'_{j_n} that is output by D_n.

Observe that whenever D'_n guesses the correct answers to the leakage queries made by D_n in rounds $j_n - 1$ and j_n, it perfectly simulates D_n according to $Exp2$. Furthermore, the probability that D'_n correctly guesses all of these answers is exactly $1/n^{5c}$. It follows that $q_{D'}^3(n) \geq q_D^2(n)/n^{5c}$. Then, there must exist fixed strings $z_{B1,j_n-1}, z_{A,j_n-1}, z_{B2,j_n-1}, z_{B1,j_n}, z_{A,j_n} \in \{0,1\}^{c \log n}$ such that if we hardcode these values into D'_n (instead of having D'_n choose these values randomly), we still have $q_{D'}^3(n) \geq q_D^2(n)/n^{5c}$.

Finally, note that since D'_n simply simulates D_n, we have that D'_n is of the same size as D_n.
□

We now describe an experiment where the adversary does not have control over the ordering of events in rounds $j_n - 1$ and j_n.

Experiment *Exp*4**.** The adversary D consists of a circuit family $\{D_n\}$ and a sequence of integers $\{j_n\}$. For each n, the experiment proceeds exactly as experiment *Exp*3 for the first $j_n - 2$ rounds.

Round $j_n - 1$ proceeds as follows. $r_{j_n-1} \leftarrow Z_n$ is chosen and given to D_n. D_n then produces a string $m_{j_n-1} \in \{0,1\}^n$. Then, $(e_{j_n-1}, K_{j_n-1}) \leftarrow \mathsf{EvalA}(K_{j_n-2}, m_{j_n-1}, r_{j_n-1})$ is computed, and D_n is given e_{j_n-1}.

Round j_n proceeds as follows. $r_{j_n} \leftarrow Z_n$ is chosen and given to D_n. D_n produces strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$. Then, $(e_{j_n}, K_{j_n}) \leftarrow \mathsf{EvalA}(K_{j_n}, m_{j_n}, r'_{j_n})$ is computed, and D_n is given e_{j_n-1}.

Finally, D_n produces a string $e'_{j_n} = \langle m'_{j_n}, \alpha'_{j_n} \rangle \in \{0,1\}^{2n}$.

Define $q^4_D(n)$ to be the probability that $\mathsf{EvalB}_2(K_{j_n-1}, r_{j_n}, e'_{j_n})$ does not output `Fail` and either $r'_{j_n} \neq r_{j_n}$ or $e'_{j_n} \neq e_{j_n}$.

Lemma 3.6.4 *Let D be an adversary for Exp*3*. Then there exists an adversary D' for Exp*4 *such that D' has the same size as D and $q^4_{D'}(n) = q^3_D(n)$ for all n.*

Proof Let circuit family $\{D_n\}$ and sequence $\{j_n\}$ be an adversary for *Exp*3. We will define a circuit family $\{D'_n\}$ such that $\{D'_n\}$ and $\{j_n\}$ is an adversary for *Exp*4. Fix $n > 0$.

Circuit D'_n works as follows. D'_n simulates D_n using the leakage functions and message pieces produced by D_n as the leakage functions and pieces it is expected to produce, and D'_n provides D_n with all of the output it is given.

However, if in round $j_n - 1$, D_n wishes to invoke EvalA an additional time (as he is allowed to do in *Exp*3), then D'_n first proceeds to round j_n and is given r_{j_n}, then D'_n invokes EvalA using the inputs that D_n had produced for the additional invocation it wished to perform in round $j_n - 1$, then D'_n passes on the output it receives from EvalA to D_n, then D_n will ask to invoke EvalB_1 (as it is required to do in *Exp*3) and D'_n gives D_n the string r_{j_n}.

If D_n does not invoke EvalA an additional time in round $j_n - 1$ and also does not invoke EvalA in round j_n (that is, after invoking EvalB_1 in round j_n, he immediately produces his output e'_{j_n}) then D'_n chooses an arbitrary $r'_{j_n} \neq r_{j_n}$ and an arbitrary m_{j_n} as the strings it is required to provide as input to EvalA in round j_n.

At the end, D'_n outputs the string e'_{j_n} that is output by D_n.

Observe that D'_n perfectly simulates D_n according to *Exp*3. Further, note that D'_n achieves one of the winning conditions implicit in the definition of $q^4_{D'}(n)$ exactly when D_n achieves one of the winning conditions implicit in the definition of $q^3_D(n)$. In particular, note that when D_n wins by not invoking EvalA in round j_n, D'_n wins since he chooses $r'_{j_n} \neq r_j$ in this situation. It follows that $q^4_{D'}(n) = q^3_D(n)$.

Finally, note that since D'_n simply simulates D_n, we have that D'_n is of the same size as D_n.
□

We now describe an experiment where in rounds in which leakage occurs, it occurs only once rather than three times, but is allowed to be three times as long.

Experiment $Exp5$**.** The adversary consists of a circuit family $\{D_n\}$ and a sequence of integers $\{j_n\}$. For each n, the experiment proceeds as follows. A string $K_0 \in \{0,1\}^n$ is randomly chosen. Then the experiment proceeds in a sequence of j_n rounds.

For $1 \leq i \leq j_n - 2$, round i proceeds as follows. D_n produces the description of a circuit $f_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{3c \log n}$. Then $r_i \leftarrow Z_n$ is chosen, $leak_i \leftarrow f_i(K_{i-1}, r_i)$ and $K_i||X_i \leftarrow F_{K_{i-1}}(r_i)$ (where $|K_i| = |X_i| = n$) are computed, and D_n is given r_i, X_i, and $leak_i$.

Round $j_n - 1$ proceeds as follows. $r_{j_n-1} \leftarrow Z_n$ is chosen, $K_{j_n-1}||X_{j_n-1} \leftarrow F_{K_{j_n-2}}(r_{j_n-1})$ is computed, and D_n is given r_{j_n-1} and X_{j_n-1}.

Then, round j_n proceeds as follows. $r_{j_n} \leftarrow Z_n$ is chosen and given to D_n. D_n produces strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$. Then, $(e_{j_n}, K_{j_n}) \leftarrow \mathsf{EvalA}(K_{j_n-1}, m_{j_n}, r'_{j_n})$ is computed, and D_n is given e_{j_n}.

Finally, D_n produces a string $e'_{j_n} = \langle m'_{j_n}, \alpha'_{j_n} \rangle \in \{0,1\}^{2n}$.

Define $q^5_D(n)$ to be the probability that $\mathsf{EvalB}_2(K_{j_n-1}, r_{j_n}, e'_{j_n})$ does not output $\texttt{Fail}$ and either $r'_{j_n} \neq r_{j_n}$ or $e'_{j_n} \neq e_{j_n}$.

Lemma 3.6.5 *For every* $e > 0$ *and every adversary* D *of size at most* n^e *for* $Exp4$*, there exists an adversary* D' *for* $Exp5$ *such that* D' *is of size at most* $n^{2e} + n^e(\texttt{size}_n(F) + 2\texttt{size}_n(F'))$ *and* $q^5_{D'}(n) = q^4_D(n)$ *for all* n*.*

Proof Let $e > 0$, and let circuit family $\{D_n\}$ of size n^e and sequence $\{j_n\}$ be an adversary for $Exp4$. We will define a circuit family $\{D'_n\}$ such that $\{D'_n\}$ and $\{j_n\}$ is an adversary for $Exp5$. Fix $n > 0$.

Circuit D'_n works in the following way. D'_n simulates D_n.

For $1 \leq i \leq j_n - 2$, each round i proceeds as follows. D_n produces a circuit $f_{B1,i} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{c \log n}$. Then, D'_n produces a circuit $f_i : \{0,1\}^n \times \{0,1\}^n$ that has a copy of the current state of D_n embedded within it (which f_i uses to compute the two remaining leakage functions that D_n will produce for round i, and then f_i computes the outputs of these functions), and works as follows:

- $f_i(K_{i-1}, r_i)$: First, compute $z_1 \leftarrow f_{B1,i}(K_{i-1}, r_i)$. Then, using the copy of D_n, continue the simulation of D_n by giving it r_i and z_1. The simulation of D_n produces the description of a circuit $f_{A,i} : \{0,1\}^n \to \{0,1\}^{c \log n}$ and a string $m_i \in \{0,1\}^n$. Compute $K_i||X_i \leftarrow F_{K_{i-1}}(r_i)$, $\alpha_i \leftarrow F'_{X_i}(m_i)$, and $z_2 \leftarrow f_{A,i}(K_{i-1})$. Let $e_i = \langle m_i, \alpha_i \rangle$. Give e_i and z_2 to the simulation of D_n. Then the simulation of D_n produces a circuit $f_{B2,i} : \{0,1\}^n \to \{0,1\}^{c \log n}$. Compute $z_3 \leftarrow f_{B2,i}(K_{i-1})$. Output $z_1||z_2||z_3$.

Then, D'_n is given strings r_i, X_i, and $leak_i = z_1||z_2||z_3$. D'_n gives r_i and z_1 to D_n. D_n then produces a function $f_{A,i} : \{0,1\}^n \to \{0,1\}^{c \log n}$ and a string $m_i \in \{0,1\}^n$. D'_n computes $\alpha_i \leftarrow F'_{X_i}(m_i)$, lets $e_i = \langle m_i, \alpha_i \rangle$, and gives e_i and z_2 to D_n. Then D_n produces a circuit $f_{B2,i} : \{0,1\}^n \to \{0,1\}^{c \log n}$, and D'_n gives D_n the string z_3.

Round $j_n - 1$ proceeds as follows. D'_n is given strings r_{j_n-1} and X_{j_n-1}. D'_n gives r_{j_n-1} to D_n. Then, D_n produces a string $m_{j_n-1} \in \{0,1\}^n$. D'_n computes $\alpha_{j_n-1} \leftarrow F'_{K_{j_n-1}}(m_{j_n-1})$, lets $e_{j_n-1} = \langle m_{j_n-1}, \alpha_{j_n-1} \rangle$, and gives e_{j_n-1} to D_n.

In round j_n, D'_n is given a string r_{j_n}, provides this string to D_n, and uses D_n to produce the strings m_{j_n} and r_{j_n} it is expected to produce. Then D'_n is given e_{j_n}, which it passes on to D_n.

At the end, D'_n outputs the string e'_{j_n} that is output by D_n.

Observe that D'_n perfectly simulates D_n according to *Exp*4. Further, the winning conditions implicit in the definition of $q^4_D(n)$ are exactly the same as those in the definition of $q^5_{D'}(n)$. It follows that $q^5_{D'}(n) = q^4_D(n)$.

Finally, consider the size of D'_n. D'_n simulates D_n. Also, in each round other than the final round, D'_n evaluates F' exactly once. Additionally, in each round other than the final two rounds, D'_n has to produce a leakage function that itself simulates D_n, evaluates F exactly once, and evaluates F' exactly once. It follows that D'_n is of size at most $n^e(\mathtt{size}_n(F') + n^e + \mathtt{size}_n(F) + \mathtt{size}_n(F')) = n^{2e} + n^e(\mathtt{size}_n(F) + 2\mathtt{size}_n(F'))$.

□

We now define an experiment where in the second-last round, we use a randomly chosen string in place of the output of F.

Experiment *Exp*6**.** The adversary D consists of a circuit family $\{D_n\}$ and a sequence of integers $\{j_n\}$. For each n, the experiment proceeds exactly as experiment *Exp*5 for the first $j_n - 2$ rounds.

Round $j_n - 1$ proceeds as follows. Strings $K_{j_n-1}, X_{j_n-1} \in \{0,1\}^n$ are randomly chosen, string $r_{j_n-1} \leftarrow Z_n$ is chosen, and D_n is given r_{j_n-1} and X_{j_n-1}.

Round j_n proceeds as follows. $r_{j_n} \leftarrow Z_n$ is chosen and given to D_n. D_n produces strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$. Then, $(e_{j_n}, K_{j_n}) \leftarrow \mathsf{EvalA}(K_{j_n-1}, m_{j_n}, r'_{j_n})$ is computed, and D_n is given e_{j_n}.

Finally, D_n produces a string $e'_{j_n} = \langle m'_{j_n}, \alpha'_{j_n} \rangle \in \{0,1\}^{2n}$.

Define $q^6_D(n)$ to be the probability that $\mathsf{EvalB}_2(K_{j_n-1}, r_{j_n}, e'_{j_n})$ does not output $\mathtt{Fail}$ and either $r'_{j_n} \neq r_{j_n}$ or $e'_{j_n} \neq e_{j_n}$.

Lemma 3.6.6 *For every $d, e > 0$ and every adversary D of size at most n^e for $Exp5$ such that $q_D^5(n) > 1/n^d$ for infinitely many n, at least one of the following holds:*

1. *$q_D^6(n) > 1/(2n^d)$ for infinitely many n.*
2. *For every $e' > 0$ such that $3n^e + n^e \cdot \mathtt{size}_n(F) + 2 \cdot \mathtt{size}_n(F') < n^{e'}$ for sufficiently large n, there exists an adversary of size $n^{e'+8d+8e+6c+16}$ that breaks F with advantage $1/n^{5d+5e+6c+8}$ for infinitely many n.*

Proof Let $d, e > 0$. Let circuit family $\{D_n\}$ of size n^e and sequence $\{j_n\}$ be an adversary for $Exp5$ such that $q_D^5(n) > 1/n^d$ for infinitely many n. Let $e' > 0$ be such that $3n^e + n^e \cdot \mathtt{size}_n(F) + 2 \cdot \mathtt{size}_n(F') < n^{e'}$ for sufficiently large n.

If $q_D^6(n) > 1/(2n^d)$ for infinitely many n then we are done, so suppose $q_D^6(n) \leq 1/(2n^d)$ for sufficiently large n. This means that $q_D^5(n) - q_D^6(n) > 1/(2n^d)$ for infinitely many n. Fix such an n.

We will construct a $(3c \log n)$-bounded adversary A_n that interacts with our stream cipher (as described in section 3.4.2), obtaining leakage for $j_n - 2$ rounds, and we will construct adversary D'_n such that in the terminology of Section 3.4.3 (and Theorem 3.4.1), we have

$$\left|\Pr\left[D'_n(\mathtt{real}^+_{j_n-2}) = 1\right] - \Pr\left[D'_n(\mathtt{real}_{j_n-2}, R_{j_n-1}, K''_{j_n-1}, X''_{j_n-1}) = 1\right]\right| > \frac{6j_n - 6}{n^{d'}}$$

for some d' whose value will be specified later.

Our adversary A_n will work as follows. A_n will simulate D_n for $j_n - 2$ rounds. In each round i, D_n will produce the description of a circuit $f_i : \{0,1\}^{2n} \rightarrow \{0,1\}^{3c \log n}$. A_n will use this f_i as the circuit it is expected to produce in round i. A_n will then be given strings $R_i, X_i, f_i(K_{i-1}, R_i)$, which it will pass on to D_n (as the strings $r_i, X_i, leak_i$).

Our adversary D'_n will work as follows. The input to D'_n will be of the form

$$\langle r_1, leak_1, x_1, r_2, leak_2, x_2, \ldots, r_{j_n-2}, leak_{j_n-2}, x_{j_n-2}, r_{j_n-1}, k_{j_n-1}, x_{j_n-1}\rangle$$

where each $r_i \in \{0,1\}^n$, each $x_i \in \{0,1\}^n$, each $leak_i \in \{0,1\}^{3c \log n}$, and $k_{j_n-2} \in \{0,1\}^n$. D'_n will begin by simulating D_n for $j_n - 2$ rounds. In each such round i, D'_n will ignore the function f_i produced by D_n, and will give the strings $r_i, x_i, leak_i$ to D_n. Then, in round $j_n - 1$, D'_n will give D_n the strings r_{j_n-1} and x_{j_n-1}. In round j_n, D'_n will select $r_{j_n} \leftarrow Z_n$ and give this string to D_n. D_n produces strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$. Then, D'_n computes $(e_{j_n}, k_{j_n}) \leftarrow \mathsf{EvalA}(k_{j_n-1}, m_{j_n}, r'_{j_n})$ and gives D_n the string e_{j_n}. Then D_n produces a string $e'_{j_n} = \langle m'_{j_n}, \alpha'_{j_n}\rangle \in \{0,1\}^{2n}$. Then, D'_n computes $\mathsf{EvalB}_2(k_{j_n-1}, r_{j_n}, e'_{j_n})$. If EvalB_2 does not output $\mathtt{Fail}$, and either $r'_{j_n} \neq r_{j_n}$ or $e'_{j_n} \neq e_{j_n}$, D'_n outputs 1; otherwise, D'_n outputs 0.

We claim that $\Pr\left[D'_n(\mathtt{real}^+_{j_n-2}) = 1\right] = q_D^5(n)$. To see this, observe that when the input to D'_n is chosen according to $\mathtt{real}^+_{j_n-2}$, it is a transcript of an interaction of A_n with $j_n - 2$ rounds of the stream cipher. But by the construction of A_n, this is also a transcript of an interaction of D_n with the first $j_n - 2$ rounds of *Exp*5. Since D_n is deterministic, when D'_n runs D_n using the strings from the given transcript, D_n proceeds exactly as it did when this transcript was produced by A_n (that is, D_n produces the same sequence of leakage functions). This means that when D'_n begins by simulating D_n for $j_n - 2$ rounds, it simply brings D_n to the point where it was at the end of the run of A_n. Furthermore, since the input to D'_n is chosen according to $\mathtt{real}^+_{j_n-2}$, the final two input strings k_{j_n-1} and x_{j_n-1} to D'_n are computed using the stream cipher, and hence they are computed the same way that they are computed in *Exp*5. This means that when D'_n continues simulating D_n in rounds $j_n - 1$ and j_n, it does so according to *Exp*5. It follows that $\Pr\left[D'_n(\mathtt{real}^+_{j_n-2}) = 1\right] = q^5_D(n)$.

We also claim that $\Pr\left[D'_n(\mathtt{real}_{j_n-2}, R_{j_n-1}, K''_{j_n-1}, X''_{j_n-1}) = 1\right] = q^6_D(n)$. To see this, observe that the only difference between an input chosen according to $\mathtt{real}^+_{j_n-2}$ and an input chosen according to $(\mathtt{real}_{j_n-2}, R_{j_n-1}, K''_{j_n-1}, X''_{j_n-1})$ is that in the latter case, the final two input strings k_{j_n-1} and x_{j_n-1} are chosen randomly rather than using the stream cipher. That is, in the latter case, D'_n will use randomly chosen strings k_{j_n-1} and x_{j_n-1} when carrying out the final two rounds of the simulation of D_n. But this is exactly the manner in which the final two rounds of *Exp*6 proceed, and, furthermore, this is the only difference between *Exp*6 and *Exp*5. It follows that $\Pr\left[D'_n(\mathtt{real}_{j_n-2}, R_{j_n-1}, K''_{j_n-1}, X''_{j_n-1}) = 1\right] = q^6_D(n)$.

We then have that

$$\Pr\left[D'_n(\mathtt{real}^+_{j_n-2}) = 1\right] - \Pr\left[D'_n(\mathtt{real}_{j_n-2}, R_{j_n-1}, K''_{j_n-1}, X''_{j_n-1}) = 1\right] = q^5_D(n) - q^6_D(n) > \frac{1}{2n^d}$$

Now, note that we must have $n^e \geq j_n$, since the size of D_n must upper-bound the number of rounds of interaction that it performs. Then, observe that for sufficiently large n, we have

$$\frac{1}{2n^d} = \frac{6n^e}{12n^{d+e}} \geq \frac{6n^e}{n^{d+e+1}} \geq \frac{6j_n}{n^{d+e+1}}$$

Then, letting $d' = d + e + 1$, we have that

$$\Pr\left[D'_n(\mathtt{real}^+_{j_n-2}) = 1\right] - \Pr\left[D'_n(\mathtt{real}_{j_n-2}, R_{j_n-1}, K''_{j_n-1}, X''_{j_n-1}) = 1\right] > \frac{6j_n}{n^{d'}} \tag{3.28}$$

Note that D'_n can be made deterministic by a standard argument (that involves fixing a particular choice of r_{j_n} such that when D'_n uses this fixed value, (3.28) still holds).

Observe that A_n has size n^e (since it simply simulates D_n). Also note that D'_n has size at most $n^e + 2\mathtt{size}_n(F) + 2\mathtt{size}(F')$, since it simulates D and evaluates EvalA and EvalB_2 once each. It follows that for sufficiently large n,

$$2 \cdot \mathtt{size}(A_n) + \mathtt{size}(D'_n) + (j_n - 2) \cdot \mathtt{size}_n(F) \leq 3n^e + n^e \cdot \mathtt{size}_n(F) + 2 \cdot \mathtt{size}_n(F') \leq n^{e'}$$

where the first inequality uses the fact that $j_n \leq n^e$ and the second inequality is by our choice of e'.

Now, defining function $p : \mathbb{N} \to \mathbb{N}$ to be such that for all n, $p(n) = j_n - 2$, we have that for infinitely many n, there exists an adversary D'_n such that

$$2 \cdot \texttt{size}(A_n) + \texttt{size}(D'_n) + p(n) \cdot \texttt{size}_n(F) \leq n^{e'}$$

and

$$\Pr\left[D'_n(\texttt{real}^{+}_{j_n-2}) = 1\right] - \Pr\left[D'_n(\texttt{real}_{j_n-2}, R_{j_n-1}, K''_{j_n-1}, X''_{j_n-1}) = 1\right] > \frac{6p(n)+6}{n^{d'}}$$

Applying Theorem 3.4.1, there exists an adversary of size $n^{e'+8d'+6c+8} = n^{e'+8d+8e+6c+16}$ that breaks F with advantage $1/n^{5d'+6c+3} = 1/n^{5d+5e+6c+8}$ for infinitely many n.

□

We now describe an experiment where in the final round, randomly chosen strings are used in place of the output of F.

Experiment *Exp*7 The adversary consists of a circuit family $\{D_n\}$ and a sequence of integers $\{j_n\}$. For each n, the experiment proceeds exactly as *Exp*6 for the first $j_n - 1$ rounds.

Round j_n proceeds as follows. Strings $X_{j_n}, X'_{j_n} \in \{0,1\}^n$ are randomly chosen and string $r_{j_n} \leftarrow Z_n$ is chosen. r_{j_n} is given to D_n. D_n produces strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$. If $r'_{j_n} = r_{j_n}$, then $\alpha_{j_n} \leftarrow F'_{X_{j_n}}(m_{j_n})$ is computed; otherwise, $\alpha_{j_n} \leftarrow F'_{X'_{j_n}}(m_{j_n})$ is computed. D_n is given $e_{j_n} = \langle m_{j_n}, \alpha_{j_n} \rangle$.

Finally, D_n produces a string $e'_{j_n} = \langle m'_{j_n}, \alpha'_{j_n} \rangle \in \{0,1\}^{2n}$.

Define $q^7_D(n)$ to be the probability that $\alpha'_{j_n} = F'_{X_{j_n}}(m'_{j_n})$ and either $r'_{j_n} \neq r_{j_n}$ or $e'_{j_n} \neq e_{j_n}$.

Lemma 3.6.7 *For every $d, e > 0$ and every adversary D of size at most n^e for Exp6 such that $q^6_D(n) > 1/n^d$ for infinitely many n, at least one of the following holds:*

1. *$q^7_D(n) > 1/(2n^d)$ for infinitely many n.*
2. *There exists an adversary of size $2n^e + n^e \cdot \texttt{size}_n(F) + 2 \cdot \texttt{size}_n(F')$ that breaks F with advantage $1/(2n^d)$ for infinitely many n.*

Proof Let $d, e > 0$. Let circuit family $\{D_n\}$ of size n^e and sequence $\{j_n\}$ be an adversary for *Exp*6 such that $q^6_D(n) > 1/n^d$ for infinitely many n.

If $q^7_D(n) > 1/(2n^d)$ for infinitely many n then we are done, so suppose $q^7_D(n) \leq 1/(2n^d)$ for sufficiently large n. This means that $q^6_D(n) - q^7_D(n) > 1/(2n^d)$ for infinitely many n. Fix such an n.

We will define an adversary D'_n for breaking F. D'_n is given an oracle $g : \{0,1\}^n \to \{0,1\}^{2n}$. Then D'_n proceeds as follows. D'_n first runs D_n according to *Exp*6 for $j_n - 1$ rounds (note that this is the same as running D_n according to $Exp7$ for $j_n - 1$ rounds). Then, D'_n selects $r_{j_n} \leftarrow Z_n$ and gives this string to D_n. Next, D_n produces strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$. Then, D'_n computes x' as the rightmost n bits of $g(r'_{j_n})$, and computes $\alpha_{j_n} \leftarrow F'_{x'}(m_{j_n})$. D'_n then gives $e_{j_n} = \langle m_{j_n}, \alpha_{j_n} \rangle$ to D_n. Then, D_n produces a string $e'_{j_n} = \langle m'_{j_n}, \alpha'_{j_n} \rangle$. D'_n computes x as the rightmost n bits of $g(r_{j_n})$. D'_n outputs 1 if $\alpha'_{j_n} = F'_x(m'_{j_n})$ and either $r'_{j_n} \neq r_{j_n}$ or $m'_{j_n} \neq m_{j_n}$; otherwise, D'_n outputs 0.

Observe that when the oracle $g : \{0,1\}^n \to \{0,1\}^{2n}$ given to D'_n is randomly chosen, D'_n simulates D_n according to $Exp7$. This means that the probability that D'_n accepts a randomly chosen function is exactly $q^7_D(n)$. On the other hand, when the oracle g is F_z for randomly chosen $z \in \{0,1\}^n$, D'_n simulates D_n according to $Exp6$. Then, the probability that D'_n accepts a pseudo-randomly generated function is exactly $q^6_D(n)$. It follows that D'_n breaks F with advantage $q^6_D(n) - q^7_D(n) > 1/(2n^d)$.

Finally, consider the size of D'_n. D'_n simulates D_n. In addition, D'_n simulates $j_n - 1$ rounds of $Exp6$; this involves evaluating F and evaluating the leakage functions produced by D_n (for the first $j_n - 2$ rounds). D'_n also evaluates F' twice. Then, using the fact that $j_n \leq n^e$ (since the size of D_n must upper bound the number of rounds of interaction it performs), we have that the size of D'_n is at most $2n^e + n^e \cdot \mathtt{size}_n(F) + 2 \cdot \mathtt{size}_n(F')$. □

We now show that an adversary that does well in Experiment $Exp7$ yields an adversary that breaks the pseudo-randomness of F'.

Lemma 3.6.8 *For every $d, e > 0$ and every adversary D of size at most n^e for $Exp7$ such that $q^7_D(n) > 1/n^d$ for infinitely many n, there exists an adversary of size $2n^e + n^e\mathtt{size}_n(F) + \mathtt{size}_n(F')$ that breaks F' with advantage $1/(2n^d)$ for infinitely many n.*

Proof Let $d, e > 0$. Let circuit family $\{D_n\}$ of size n^e and sequence $\{j_n\}$ be an adversary for $Exp7$ such that $q^7_D(n) > 1/n^d$ for infinitely many n. Fix such an n.

We will define an adversary D'_n for breaking F'. D'_n is given an oracle $g : \{0,1\}^n \to \{0,1\}^n$. Then, D'_n proceeds as follows. D'_n first runs D_n according to $Exp7$ for $j_n - 1$ rounds. Then, D'_n selects $r_{j_n} \leftarrow Z_n$ and gives this string to D_n. Next, D_n produces strings $m_{j_n}, r'_{j_n} \in \{0,1\}^n$. If $r'_{j_n} = r_{j_n}$, then D'_n computes $\alpha_{j_n} \leftarrow g(m_{j_n})$; otherwise, D'_n randomly selects $X'_{j_n} \in \{0,1\}^n$ and computes $\alpha_{j_n} \leftarrow F'_{X'_{j_n}}(m_{j_n})$. Then, D'_n gives $e_{j_n} = \langle m_{j_n}, \alpha_{j_n} \rangle$ to D_n. Next, D_n produces a string $e'_{j_n} = \langle m'_{j_n}, \alpha'_{j_n} \rangle$. D'_n outputs 1 if $\alpha'_{j_n} = g(m'_{j_n})$ and either $r'_{j_n} \neq r_{j_n}$ or $e'_{j_n} \neq e_{j_n}$; otherwise, D'_n outputs 0.

Observe that when the oracle $g : \{0,1\}^n \to \{0,1\}^n$ that is given to D'_n is F'_z for randomly chosen $z \in \{0,1\}^n$, D'_n simulates D_n according to $Exp7$. This means that the probability that D'_n accepts a pseudo-randomly generated function is exactly $q^7_D(n)$. Now consider the probability that D'_n accepts randomly chosen g. Note that in this case, D'_n will only accept if D_n is able to predict g on an input on which it has not yet been queried. This will happen with probability at most $1/2^n$. It follows that D'_n breaks F' with advantage $q^7_D(n) - 1/2^n > 1/n^d - 1/2^n$.

Finally, consider the size of D'_n. D'_n simulates D_n. In addition, D'_n simulates $j_n - 1$ rounds of $Exp7$; this involves evaluating F and evaluating the leakage functions produced by D_n (for the first $j_n - 2$ rounds). D'_n also evaluates F' at most once. Then, using the fact that $j_n \leq n^e$ (since the size of D_n must upper bound the number of rounds of interaction it performs), we have that the size of D'_n is at most $2n^e + n^e \cdot \mathtt{size}_n(F) + \mathtt{size}_n(F')$. □

We can now complete the proof of Theorem 3.2.1. Recall that $C = \{C_n\}$ is a $(c \log n)$-bounded adversary for SP such that $q_C(n) > 1/n^d$ for infinitely many n.

Let $a, b, e > 0$ be such that C_n is of size at most n^e, $\mathtt{size}_n(F)$ is at most n^a, and $\mathtt{size}_n(F')$ is at most n^b.

By Lemma 3.6.1, we have $q^1_C(n) \geq 1/n^d$ for infinitely many n. Then, by Lemma 3.6.2, there exists an adversary D^2 of size at most $n^{2e} + n^e(n^a + n^b)$ such that $q^2_{D^2}(n) \geq 1/n^{d+e+1}$ for infinitely many n. For sufficiently large n, the size of D^2 is at most n^{2e+a+b}. Now, by Lemma 3.6.3, there exists an adversary D^3 of size at most n^{2e+a+b} such that $q^3_{D^3}(n) \geq 1/n^{d+e+5c+1}$ for infinitely many n. Then, by Lemma 3.6.4, there exists an adversary D^4 of size at most n^{2e+a+b} such that $q^4_{D^4}(n) \geq 1/n^{d+e+5c+1}$ for infinitely many n. It follows by Lemma 3.6.5 that there exists an adversary D^5 of size at most $n^{4e+2a+2b} + n^{2e+a+b}(n^a + 2n^b)$ such that $q^5_{D^5}(n) \geq 1/n^{d+e+5c+1}$ for infinitely many n. For sufficiently large n, the size of D^5 is at most $n^{4e+2a+2b+1}$.

We next apply Lemma 3.6.6. Observe that if we choose $e' = 4e + 3a + 2b + 2$, then we have $3n^{4e+2a+2b+1} + n^{4e+2a+2b+1}\mathtt{size}_n(F) + 2\mathtt{size}_n(F') \leq n^{e'}$ for sufficiently large n. We then have by Lemma 3.6.6 that either $q^6_{D^5}(n) \geq 1/(2n^{4e+2a+2b+1})$ for infinitely many n, or there exists an adversary of size $n^{44e+19a+18b+8d+46c+27}$ that breaks F with advantage $1/n^{5d+25e+10a+10b+31c+18}$. In the latter case we are done (since we have an adversary breaking the pseudo-randomness of F), so suppose $q^6_{D^5}(n) \geq 1/(2n^{4e+2a+2b+1})$ for infinitely many n.

Now, applying Lemma 3.6.7, we have that either $q^7_{D^5}(n) \geq 1/(4n^{4e+2a+2b+1})$ for infinitely many n, or there exists an adversary of size $n^{4e+3a+2b+2}$ that breaks F with advantage $1/(4n^{4e+2a+2b+1})$ for infinitely many n. In the latter case we are done (since we have an adversary breaking the pseudo-randomness of F), so suppose $q^7_{D^5}(n) \geq 1/(4n^{4e+2a+2b+1})$ for infinitely many n. But then by Lemma 3.6.8, there exists an adversary of size $n^{4e+3a+2b+2}$ that breaks the pseudo-randomness of F' with advantage $1/(8n^{4e+2a+2b+1})$ for infinitely many n.

3.7 Open problems

One-flow leakage-resilient authentication Our leakage-resilient authenticated session protocol uses two flows for each message piece. As we noted previously, a protocol that uses only one flow per message can be constructed using leakage-resilient signature schemes. However, existing leakage-resilient signature schemes either require stronger assumptions than the existence of pseudo-random generators, or use the "only computation leaks" assumption. Further, these schemes are more complex (computationally and conceptually) than simply evaluating a pseudo-random function generator. Can we get a one-flow authenticated session protocol construction that is simpler and more efficient than protocols based on existing signature schemes? Can we construct a one-flow protocol using only the minimal assumption that pseudo-random generators exist (without requiring the "only computation leaks" assumption)? Recall that in the authenticated session protocol setting, we seem to have an important advantage over the signature scheme setting: the parties have a shared private key. This suggests that constructing a better one-flow leakage-resilient authenticated session protocol might be an easier problem than simplifying existing leakage-resilient signature schemes.

Leakage-resilient privacy In our session protocol, each message piece is made public. Can we construct a leakage-resilient session protocol that achieves authentication *and* privacy? In Section 2.4.2, we discussed the issues involved when defining privacy in a setting with leakage. Recall that there are two approaches one might follow – a leak-free challenge, or weakening security so that we only require that the adversary doesn't learn "too much" about the challenge (as long as the challenge message is sampled from a distribution of high min-entropy). We believe it should be possible to modify our construction to achieve privacy according to either approach.

At first glance, it might seem sufficient to modify our construction so that F outputs $3n$ bits instead of $2n$ bits, with the additional n bits of output used as a key Y_i for encrypting a single message piece m_i; that is, instead of sending m_i in the clear over the public channel, EvalA would send $Y_i \oplus m_i$. However, it turns out that this protocol fails to achieve privacy in the worst possible way – the adversary can learn every bit of some message piece m_j. The adversary can simply ignore party B (by never invoking EvalB_1 or EvalB_2), providing EvalA with $\bar{0}$ as input in place of the output of EvalB_1. In this way, the computation of EvalA becomes deterministic, and hence the adversary can use leakage to eventually learn every bit of the state of party A at some future round j. Then, designating this round j as the challenge round, the adversary succeeds trivially according to both notions of privacy.

What if we further modify our construction so that party A also samples a string from a distribution of high min-entropy, and uses the concatenation of this string and the purported output of EvalB_1 as the input to F? (Of course, A must then also include this sampled string in the output of EvalA, so that EvalB_2 can evaluate F on the same input.) This prevents the computation of either party from becoming deterministic, no matter what the adversary does, and hence seems to prevent the kind of attack described in the previous paragraph. Does this protocol indeed achieve authentication and privacy? We believe this to be the case, but we have not proved it.

Chapter 4

Black-box impossibility results

It is well known that if there exist pseudo-random generators obtaining even one bit of stretch, then for every polynomial $p(n)$, there exist pseudo-random generators obtaining $p(n)$ bits of stretch. The usual approach for constructing a pseudo-random generator of large stretch from a pseudo-random generator of smaller stretch involves composing the smaller-stretch generator with itself repeatedly. Similarly, the usual approach for constructing a pseudo-random generators of large stretch from a one-way permutation involves composing the one-way permutation with itself repeatedly.

In this chapter, we consider whether there exist such constructions that do *not* involve composition. To formalize this requirement about composition, we consider constructions that only have oracle access to the given object (a smaller-stretch pseudo-random generator or a one-way permutation) and query this oracle *non-adaptively.* We refer to such constructions as *non-adaptive (oracle) constructions.*

We give a number of black-box impossibility results for non-adaptive oracle constructions of pseudo-random generators. Some of these arguments are rather technically involved. Roughly speaking, we answer in the negative whether we can obtain, with only a constant number of queries to a pseudo-random generator, a pseudo-random generator of much larger stretch, where *answers to these non-adaptive queries are combined arbitrarily.* The challenge is to deal with this arbitrary computation phase.

Non-adaptive constructions are conceptually related to *streaming cryptography*; that is, computing private-key primitives with a device that uses small space and accesses the seed a small number of times. One of the three non-adaptive settings we consider in this chapter is motivated by questions in streaming cryptography.

Our results. Observe that if pseudo-random generators exist, then there exist trivial non-adaptive oracle constructions of large-stretch pseudo-random generators: such constructions can simply ignore their oracle and directly compute a large-stretch pseudo-random generator. Since we are interested in constructions that use their oracle in a non-trivial way, we focus on constructions whose pseudo-randomness is proven using a *black-box reduction* [IR89] to the the security (pseudo-randomness or one-wayness) of their oracle.

We consider three classes of such constructions, and give bounds on the stretch that can be obtained by each class. For each class, our results demonstrate a contrast between the stretch that can be achieved by adaptive and non-adaptive constructions. We show that, in some sense, whatever was already known regarding algorithms for non-adaptive constructions is the best we can hope for. While we are primarily interested in constructions that are polynomial-time computable, our bounds hold even for computationally-unbounded constructions (where the number of oracle queries is still bounded).

- *Class 1: Constructions with short seeds*

  We begin by considering constructions whose seed length is not too much longer than the length of each oracle query. Suppose we have a pseudo-random generator $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$ and we wish to obtain a pseudo-random generator with larger stretch, say stretch $2 \cdot s(n)$. We can easily define such a generator $G^f : \{0,1\}^n \to \{0,1\}^{n+2 \cdot s(n)}$ as follows: on input $x \in \{0,1\}^n$, G^f computes $y_0 || y_1 = f(x)$ (where $|y_0| = s(n)$ and $|y_1| = n$), and outputs $y_0 || f(y_1)$. G^f can be formalized as a fully black-box construction making two *adaptive* oracle queries, each of the same length as G's seed x, to an oracle mapping n bits to $n + s(n)$ bits. This idea can easily be extended to obtain, for every $k \in \mathbb{N}$, a fully black-box construction making k adaptive oracle queries and achieving stretch $k \cdot s(n)$.

  We show that fully black-box constructions making constantly-many queries, each of the same length as their seed length n, *must* make *adaptive* queries even to achieve stretch $s(n) + 1$, that is, even to achieve a one-bit increase in stretch. We show that this also holds for constructions whose seed length is at most $O(\log n)$ bits longer than the length n of each oracle query.

- *Class 2: Constructions with long seeds*

  What about constructions whose seed length is significantly longer than the length of each oracle query? Can we also show that such constructions must make adaptive oracle queries in order to achieve greater stretch than their oracle? In fact, a very simple way for such a construction to make non-adaptive oracle queries, yet achieve greater stretch than its oracle, involves splitting up its seed into two or more portions, and using each portion as an oracle query. For example, if $f : \{0,1\}^n \to \{0,1\}^{n+1}$ is pseudo-random, then the generator $G^f : \{0,1\}^{2n} \to \{0,1\}^{2n+2}$ defined for all $x_1, x_2 \in \{0,1\}^n$ as

$G^f(x_1||x_2) = f(x_1)||f(x_2)$ is also pseudo-random. Observe that when this construction is given an input chosen uniformly at random, the oracle queries x_1 and x_2 are chosen independently (and uniformly at random); this property is crucial for the construction's security.

What about constructions where oracle queries cannot be chosen independently and uniformly at random? Specifically, what if we consider constructions where we place no restriction on the seed length, but insist that oracle queries are collectively chosen in a manner that depends only on a portion of the seed that is not too much longer than the length of each oracle query (making it impossible to simply split up the seed into multiple queries)? While this setting may seem unnatural at first, it *is* possible in this setting to obtain a construction that makes constantly-many non-adaptive oracle queries to a pseudo-random generator and achieves more stretch than its oracle; indeed, even a single query suffices. For example, if $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$ is pseudo-random, then by the Goldreich-Levin theorem [GL89] we have that for all functions $m(n) \in O(\log n)$, the number generator $G^f : \{0,1\}^{n \cdot m(n)+n} \to \{0,1\}^{n \cdot m(n)+n+s(n)+m(n)}$ defined for all $r_1, r_2, \ldots, r_{m(n)}, x \in \{0,1\}^n$ as

$$G^f\left(r_1||r_2||\ldots||r_{m(n)}||x\right) = r_1||r_2||\ldots||r_{m(n)}||f(x)||\langle r_1, x\rangle||\langle r_2, x\rangle||\ldots||\langle r_{m(n)}, x\rangle$$

is pseudo-random; the stretch of G^f is $m(n)$ bits greater than the stretch of f. Also observe that the query made by $G^{(\cdot)}$ depends only on a portion of the seed of $G^{(\cdot)}$ whose length is the same as the length of the query (indeed, the query is identical to this portion of the seed). Using this Goldreich-Levin-based approach, it is easy to see that *adaptive* black-box constructions whose input length is much longer than the length n of each oracle query can obtain stretch $k \cdot s(n) + O(\log n)$ by making k queries to an oracle of stretch $s(n)$, even when the portion of the seed that is used to choose oracle queries has length n.

We show that fully black-box constructions $G^{(\cdot)}$ making constantly-many queries of length n to a pseudo-random generator $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$, such that only the rightmost $n + O(\log n)$ bits of the seed of $G^{(\cdot)}$ are used to choose oracle queries, *must* make *adaptive* queries in order to achieve stretch $s(n) + \omega(\log n)$. That is, such constructions making *constantly*-many non-adaptive queries cannot achieve greater stretch than the stretch provided by Goldreich-Levin with just a *single* query. This holds no matter how long a seed is used by the construction $G^{(\cdot)}$.

- *Class 3: Goldreich-Levin-like constructions*

The final class of constructions we consider is motivated by the streaming computation of pseudo-random generators. What is the relationship between non-adaptivity and streaming? Using a one-way permutation π, we wish to compute a pseudo-random generator G of linear stretch in a streaming manner, that is, using small space and a small number of passes over the seed. Even if we are able to compute π under such restrictions, adaptive use of π seems to require either storing intermediate results (but in streaming we lack sufficient space) or recomputing them (but we also lack sufficient access to the seed). In this sense, non-adaptivity serves as a clean setting for studying black-box streaming constructions.

We consider a class of constructions where the seed has a public portion that is always included in the output, the choice of each oracle query does not depend on the public portion of the seed, and the computation of each individual output bit depends only on the seed and on the response to a *single* oracle query. We refer to such constructions making non-adaptive oracle queries as *bitwise-nonadaptive* constructions. It is not hard to see that such constructions making polynomially-many *adaptive* queries to a one-way permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ can achieve arbitrary polynomial stretch; the idea is to repeatedly compose π with itself, outputting a hardcore bit of π on each composition. For example, using the Goldreich-Levin hardcore bit [GL89], a standard way of constructing a pseudo-random generator G of polynomial stretch $p(n)$ is the following: On input $r, x \in \{0,1\}^n$,

$$G^{\pi}(r||x) = r||\langle r, x\rangle||\langle r, \pi(x)\rangle||\langle r, \pi^2(x)\rangle|| \ldots ||\langle r, \pi^{p(n)+n}(x)\rangle$$

where $\pi^i := \underbrace{\pi \circ \pi \circ \ldots \circ \pi}_{i \text{ times}}$, and $\langle \alpha, \beta\rangle$ denotes the standard inner product of α and β. Observe that the leftmost n bits of the seed of G are public in the sense that they are included in the output. Also observe that each of the remaining output bits of G is computed using only a single output of π along with the input bits of G. Finally, observe that the queries made to π do not depend on the public input bits of G, and the number of non-public input bits is no greater than the length n of each oracle query. It is natural to ask whether the *adaptive* use of π in a construction of this form is necessary. This is particularly interesting if we wish to compute G in a streaming setting where we have small workspace, we are allowed to produce the output bit-by-bit, and we are allowed to re-read the input once per output bit.

We show that fully black-box bitwise-nonadaptive constructions $G^{(\cdot)}$ making queries of length n to a one-way permutation, such that the non-public portion of the seed of $G^{(\cdot)}$ is of length at most $n + O(\log n)$, cannot achieve linear stretch. This holds no matter the length of the public portion of the seed of $G^{(\cdot)}$.

Related work. Black-box reductions were formalized by Impagliazzo and Rudich [IR89], who observed that most proofs of security in cryptography are of this form. Impagliazzo and Rudich also gave the first black-box impossibility results. In their most general form, such results show that for particular security properties P_1 and P_2, it is impossible to give a black-box construction of P_1 from P_2. The same approach can also be applied to particular classes of black-box constructions, such as those making some restricted number of oracle queries or those that query their oracle non-adaptively. A large number of impossibility results have been given using this framework. The results most closely related to the problem we are considering are those of Gennaro *et al* [GGKT05], Viola [Vio05], Lu [Lu06], and Miles and Viola [MV11].

Gennnaro *et al* [GGKT05] consider black-box constructions of pseudo-random generators from one-way permutations. They show that such constructions cannot achieve $\omega(\log n)$ bits of stretch per oracle query of length n, even when queries are chosen adaptively. Their result can be extended in a straightforward way to show that for the second class of constructions we consider (and also for a more general class where queries are allowed to depend on the entire seed), for every $k \in \mathbb{N}$, constructions making k oracle queries to a pseudo-random generator of stretch $s(n)$ cannot achieve stretch $k \cdot s(n) + \omega(\log n)$, even when these queries are chosen adaptively. By contrast, recall that we show that for this class of constructions, for every $k \in \mathbb{N}$, constructions making k *non-adaptive* oracle queries to a pseudo-random generator of stretch $s(n)$ cannot achieve stretch $s(n) + \omega(\log n)$.

Viola [Vio05] considers black-box constructions of pseudo-random generators from one-way *functions* where oracle queries are non-adaptive but chosen in a computationally unbounded way, while the output of the construction is computed from the query responses by an AC^0 (polynomial-size and constant-depth) circuit. He shows that such constructions cannot achieve linear stretch. The class of constructions considered by Viola is, in general, incomparable to the classes we consider. His class is more general in terms of the numbers of queries allowed and the way that queries are chosen: he places no bounds on the number of queries, allows the queries to be chosen arbitrarily based on the seed (while we require queries to be chosen in a computable manner), and places no restrictions on the length of the queries relative to the length of the seed. On the other hand, his class is more restrictive in terms of the computational power allowed after the query responses are received: he only allows AC^0 computation, while we allow unbounded computation.

Lu [Lu06] considers the same class of constructions as Viola, except that Lu allows the output to be computed from the query responses by a subexponential-size constant-depth circuit (rather than an AC^0 circuit). He shows that such constructions cannot achieve linear stretch.

Miles and Viola [MV11] consider black-box constructions of pseudo-random generators from pseudo-random generators of 1-bit stretch, where the oracle queries are non-adaptive but chosen in a computationally unbounded way, while the output of the construction consists simply of query response bits; that is, these constructions are not allowed to perform any computation on query responses. They show that such constructions cannot achieve linear stretch. Like the constructions considered by Viola [Vio05] and Lu [Lu06], the class of constructions considered by Miles and Viola is, in general, incomparable to the classes we consider: the constructions they consider are more general in the manner in which queries are chosen (they place no restrictions on the length of queries relative to the length of the seed), but much more restrictive in terms of the computational power allowed after query responses are received.

In the positive direction, Haitner *et al* [HRV10] give the first *non-adaptive* black-box construction of a pseudo-random generator from a one-way *function*. Their construction achieves sublinear stretch. They also give a non-adaptive black-box construction achieving linear stretch, but this requires an *exponentially-hard* one-way function. In both of these constructions, the oracle queries are collectively chosen based on a portion of the seed that is significantly longer than the length of each oracle query. By contrast, recall that all of our impossibility results are for constructions where the oracle queries are collectively chosen based on a portion of the seed that is no more than logarithmically-many bits longer than the length of each oracle query.

Organization. Section 4.1 contains definitions and preliminaries. In Section 4.2, we show that if a distribution yields pseudo-random generators "on the average", then it does so with probability 1; we need this result for our impossibility results about the first two classes of constructions that we consider. The impossibility results for constructions with short seeds and long seeds are discussed in Sections 4.3 and 4.4 respectively. In Section 4.5, we state a restriction on the way that constructions choose oracle queries, and under this restriction we extend the results of Sections 4.3 and 4.4 to constructions making polynomially-many queries. The impossibility result for Goldreich-Levin-like constructions is found in Section 4.6.

4.1 Preliminaries

Notation. We use "PPT" to denote "probabilistic polynomial time". We denote by $\langle a\rangle_n$ the n-bit binary string representation of $a \in \mathbb{N}$, padded with leading zeros when necessary. If the desired representation length is clear from the context, we write $\langle a\rangle$ instead of $\langle a\rangle_n$. If $a \geq 2^n$, then $\langle a\rangle_n$ denotes the n least significant bits of the binary representation of a. We denote by $x||y$ the concatenation of strings x and y.

4.1.1 Pseudo-random generators and one-way functions

A length-increasing function $G : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ is a *pseudo-random generator* if for every PPT adversary M, we have $\left| \Pr_{x \leftarrow \{0,1\}^{\ell_1(n)}} [M(G(x)) = 1] - \Pr_{z \leftarrow \{0,1\}^{\ell_2(n)}} [M(z) = 1] \right| \leq 1/n^c$ for all c and sufficiently large n.

A function $f : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ is *one-way* if for every PPT adversary M, we have $\Pr_{x \leftarrow \ell_1(n)} [f(M(f(x))) = f(x)] \leq 1/n^c$ for all c and sufficiently large n.

4.1.2 Non-adaptive constructions

Our impossibility results are for constructions that use their oracle in a non-adaptive manner.

Definition 16 (Non-adaptive oracle machine) Let $M^{(\cdot)}$ be a deterministic oracle Turing machine. We say that $M^{(\cdot)}$ is a *non-adaptive oracle machine* if the oracle queries made by $M^{(\cdot)}$ are determined by only the input to $M^{(\cdot)}$, and, in particular, do not depend on the responses to previous queries.

We will sometimes need to refer to the *querying function* of a non-adaptive oracle machine.

Definition 17 (Querying function) Let $\ell_1(n)$, $\ell_2(n)$, and $p(n)$ be polynomials, and let $M^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ be a non-adaptive oracle machine that makes $p(n)$ oracle queries, each of length n. The *querying function* of $M^{(\cdot)}$, denoted Q_M, is the function $Q_M : \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log p(n)} \to \{0,1\}^n$ such that for all $x \in \{0,1\}^{\ell_1(n)}$ and $0 \leq i < p(n)$, the i-th oracle query made by $M^{(\cdot)}(x)$ is $Q_M(x, \langle i \rangle)$. When $p(n) \equiv 1$, the second argument to Q_M is omitted.

If there exists a polynomial $r(n)$ such that the queries made by $M^{(\cdot)}$ depend only on the rightmost $r(n)$ bits of the input of $M^{(\cdot)}$, then the *$r(n)$-restricted querying function* of $M^{(\cdot)}$, denoted $Q_M^{r(n)}$, is the function $Q_M^{r(n)} : \{0,1\}^{r(n)} \times \{0,1\}^{\log p(n)} \to \{0,1\}^n$ such that for all $v \in \{0,1\}^{\ell_1(n) - r(n)}$, $w \in \{0,1\}^{r(n)}$, and $0 \leq i < p(n)$, the i-th oracle query made by $M^{(\cdot)}(v||w)$ is $Q_M^{r(n)}(w, \langle i \rangle)$.

4.1.3 Black-box reductions

Reingold, Trevisan, and Vadhan [RTV04] give a classification of black-box security reductions. Our impossibility results apply to what Reingold *et al* call *fully-black box* reductions. We avoid defining such reductions in their full generality and instead focus on security reductions for constructions of pseudo-random number generators from pseudo-random generators of smaller stretch.

Definition 18 (Fully black-box reduction [IR89]) Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ be a number generator whose construction has access to an oracle for a length-increasing function mapping $\ell'_1(n)$ bits to $\ell'_2(n)$ bits. There is a *fully black-box* reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle if there exists a PPT oracle machine $M^{(\cdot,\cdot)}$ such that for every function $f : \{0,1\}^{\ell'_1(n)} \to \{0,1\}^{\ell'_2(n)}$ and every function $A : \{0,1\}^{\ell_2(n)} \to \{0,1\}$, if A breaks the pseudo-randomness of G^f then $M^{(f,A)}$ breaks the pseudo-randomness of f.

Definition 18 can be modified in a straightforward way for constructions of pseudo-random number generators from other primitives, such as from one-way permutations.

An oracle construction whose security is proven using a black-box reduction is called a *black-box construction*.

4.2 Pseudo-random "on the average" $\Longrightarrow$ pseudo-random with probability 1

When giving a black-box impossibility result showing that an object with some security property P_2 cannot be obtained from security property P_1, the usual approach is to define a joint distribution $(\mathcal{F}, \mathcal{A})$ over pairs of functions and then show that with probability one over $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, A can be used to break P_2-ness but f satisfies property P_1 even with respect to adversaries that are given oracle access to f and A. How might we go about proving that with probability one over $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, f satisfies property P_1 with respect to adversaries that are given oracle access to f and A? Impagliazzo and Rudich [IR89] consider the case when property P_1 is *one-wayness*, and show that if $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ is one-way "on the average" (where "on the average" means that the choice $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ is part of the security experiment) with respect to adversaries that are given oracle access to f and A, then $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ is one-way with probability one with respect to such adversaries. Specifically, Impagliazzo and Rudich show that if for every PPT oracle machine $D^{(\cdot,\cdot)}$ we have that

$$\Pr_{\substack{(f,A)\leftarrow(\mathcal{F},\mathcal{A}) \\ x\leftarrow_r\{0,1\}^n}}\left[f\left(D^{(f,A)}\left(f(x)\right)\right) = f(x)\right] < \frac{1}{n^c}$$

for all c and sufficiently large n, then with probability one over the choice $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ we have that for every PPT oracle machine $D^{(\cdot,\cdot)}$,

$$\Pr_{x\leftarrow_r\{0,1\}^n}\left[f\left(D^{(f,A)}\left(f(x)\right)\right) = f(x)\right] < \frac{1}{n^c}$$

for all c and sufficiently large n.

For some of our black-box impossibility results, we need a similar result for *pseudo-randomness* instead of one-wayness. We are not aware of any previous such result, so we give a proof here.

The main differences between our proof and that of Impagliazzo and Rudich arise from the fact that the definition of pseudo-randomness involves comparing two probabilities, while the definition of one-wayness involves only a single probability.

Theorem 4.2.1 *Let $\ell_1(n)$ and $\ell_2(n)$ be polynomials. Let $(\mathcal{F},\mathcal{A}) = \{(\mathcal{F}_n,\mathcal{A}_n)\}$ be a joint distribution such that for all $n > 0$, $\mathcal{F}_n$ is a distribution over functions mapping n bits to $\ell_1(n)$ bits and $\mathcal{A}_n$ is a distribution over functions mapping $\ell_2(n)$ bits to 1 bit. Suppose that for every PPT oracle machine $D^{(\cdot,\cdot)}$, we have*

$$\left| \Pr_{\substack{(f,A)\leftarrow(\mathcal{F},\mathcal{A})\\ s\leftarrow_r\{0,1\}^n}}\left[D^{(f,A)}\left(f(s)\right)=1\right] - \Pr_{\substack{(f,A)\leftarrow(\mathcal{F},\mathcal{A})\\ z\leftarrow_r\{0,1\}^{\ell_1(n)}}}\left[D^{(f,A)}\left(z\right)=1\right]\right| < \frac{1}{n^c} \tag{4.1}$$

for all c and sufficiently large n. Then, with probability 1 over the choice $(f,A)\leftarrow(\mathcal{F},\mathcal{A})$, we have that for every PPT oracle machine $D^{(\cdot,\cdot)}$,

$$\left| \Pr_{s\leftarrow_r\{0,1\}^n}\left[D^{(f,A)}\left(f(s)\right)=1\right] - \Pr_{z\leftarrow_r\{0,1\}^{\ell_1(n)}}\left[D^{(f,A)}\left(z\right)=1\right]\right| < \frac{1}{n^c}$$

for all c and sufficiently large n.

Proof The main part of the proof is the following claim, which essentially says that for a fixed adversary $D^{(\cdot,\cdot)}$ and large enough n, "most" $(f,A)\in(\mathcal{F},\mathcal{A})$ are such that $D^{(f,A)}$ does not break the pseudo-randomness of f for security parameter n. Once we have the claim, we proceed similarly to the proof of Impagliazzo and Rudich, using the Borel-Cantelli lemma.

Claim 4.2.2 *Let $D^{(\cdot,\cdot)}$ be a PPT oracle machine. For all $c > 0$ and sufficiently large n, we have*

$$\Pr_{(f,A)\leftarrow(\mathcal{F},\mathcal{A})}\left[\left(\Pr_{s\leftarrow_r\{0,1\}^n}\left[D^{(f,A)}\left(f\left(s\right)\right)=1\right] - \Pr_{z\leftarrow_r\{0,1\}^{\ell_1(n)}}\left[D^{(f,A)}\left(z\right)=1\right]\right)\geq\frac{1}{n^c}\right] < \frac{1}{n^2}$$

Proof *(Claim 4.2.2)* Suppose for the sake of contradiction that there exists a PPT oracle machine $D^{(\cdot,\cdot)}$ and a $c\in\mathbb{N}$ such that

$$\Pr_{(f,A)\leftarrow(\mathcal{F},\mathcal{A})}\left[\left(\Pr_{s\leftarrow_r\{0,1\}^n}\left[D^{(f,A)}\left(f\left(s\right)\right)=1\right] - \Pr_{z\leftarrow_r\{0,1\}^{\ell_1(n)}}\left[D^{(f,A)}\left(z\right)=1\right]\right)\geq\frac{1}{n^c}\right] \geq \frac{1}{n^2} \tag{4.2}$$

for infinitely many n. We will obtain a contradiction to our assumption (4.1).

We define a PPT oracle machine $\hat{D}^{(\cdot,\cdot)}$. On input $t\in\ell_1(n)$ and given access to oracles f and A, $\hat{D}$ behaves as follows. First, $\hat{D}$ runs experiments to approximate the probabilities $p_{D,f,A}(n)$ and $r_{D,f,A}(n)$, where we define

$$p_{D,f,A}(n) = \Pr_{s\leftarrow_r\{0,1\}^n}\left[D^{(f,A)}\left(f\left(s\right)\right)=1\right]$$

and

$$r_{D,f,A}(n) = \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[D^{(f,A)}(z) = 1 \right]$$

In particular, for each of the two probabilities above, $\hat{D}$ runs n^{2c+3} experiments, obtaining estimates $\hat{p}_{D,f,A}(n)$ and $\hat{r}_{D,f,A}(n)$. Then, if $\hat{p}_{D,f,A}(n) - \hat{r}_{D,f,A}(n) \geq 1/n^c - 2/n^{c+1}$, $\hat{D}$ simulates $D^{(f,A)}(t)$, outputting whatever the simulation outputs. Otherwise, $\hat{D}$ outputs a randomly chosen bit.

Now, fix $n \geq 4$ such that (4.2) holds. That is, we have

$$\Pr_{(f,A) \leftarrow (\mathcal{F},\mathcal{A})} \left[p_{D,f,A}(n) - r_{D,f,A}(n) \geq \frac{1}{n^c} \right] \geq \frac{1}{n^2} \tag{4.3}$$

Using Chernoff bounds, we have that for all (f, A), the probability that $|\hat{p}_{D,f,A}(n) - p_{D,f,A}(n)| \leq 1/n^{c+1}$ is at least $1 - 1/2^n$. Similarly, the probability that $|\hat{r}_{D,f,A}(n) - r_{D,f,A}(n)| \leq 1/n^{c+1}$ is at least $1 - 1/2^n$. Then, by the union bound, the probability that both estimates $\hat{p}_{D,f,A}(n)$ and $\hat{r}_{D,f,A}(n)$ each have additive error at most $1/n^{c+1}$ is at least $1 - 2/2^n$. It follows that if

$$p_{D,f,A}(n) - r_{D,f,A}(n) \geq \frac{1}{n^c}$$

then with probability at least $1 - 2/2^n$ we have

$$\hat{p}_{D,f,A}(n) - \hat{r}_{D,f,A}(n) \geq \frac{1}{n^c} - \frac{2}{n^{c+1}}, \tag{4.4}$$

and hence for such (f, A) we have

$$\Pr_{s \leftarrow_r \{0,1\}^n} \left[\hat{D}^{(f,A)}(f(s)) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[\hat{D}^{(f,A)}(z) = 1 \right] \geq \left(1 - \frac{2}{2^n} \right) \frac{1}{n^c}. \tag{4.5}$$

Also, if

$$p_{D,f,A}(n) - r_{D,f,A}(n) < \frac{1}{n^c} - \frac{4}{n^{c+1}}$$

then with probability at least $1 - 2/2^n$ we have

$$\hat{p}_{D,f,A}(n) - \hat{r}_{D,f,A}(n) < \frac{1}{n^c} - \frac{2}{n^{c+1}}, \tag{4.6}$$

and hence for such (f, A) we have

$$\Pr_{s \leftarrow_r \{0,1\}^n} \left[\hat{D}^{(f,A)}(f(s)) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[\hat{D}^{(f,A)}(z) = 1 \right] \geq -\frac{2}{2^n}. \tag{4.7}$$

Finally, if

$$\frac{1}{n^c} - \frac{4}{n^{c+1}} \leq p_{D,f,A}(n) - r_{D,f,A}(n) < \frac{1}{n^c}$$

then our choice of $n \geq 4$ ensures $p_{D,f,A}(n) - r_{D,f,A}(n) \geq 0$, and hence for such (f, A) we have

$$\Pr_{s \leftarrow_r \{0,1\}^n} \left[\hat{D}^{(f,A)}(f(s)) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[\hat{D}^{(f,A)}(z) = 1 \right] \geq 0. \tag{4.8}$$

Putting everything together, we have

$$
\begin{aligned}
& \Pr_{\substack{(f,A)\leftarrow(\mathcal{F},\mathcal{A})\\ s\leftarrow_r\{0,1\}^n}}\left[\hat{D}^{(f,A)}\left(f(s)\right)=1\right]-\Pr_{\substack{(f,A)\leftarrow(\mathcal{F},\mathcal{A})\\ z\leftarrow_r\{0,1\}^{\ell_1(n)}}}\left[\hat{D}^{(f,A)}\left(z\right)=1\right] \\
= \; & \mathop{\mathrm{E}}_{(f,A)\leftarrow(\mathcal{F},\mathcal{A})}\left[\Pr_{s\leftarrow_r\{0,1\}^n}\left[\hat{D}^{(f,A)}\left(f\left(s\right)\right)=1\right]-\Pr_{z\leftarrow_r\{0,1\}^{\ell_1(n)}}\left[\hat{D}^{(f,A)}\left(z\right)=1\right]\right] \qquad (4.9)\\
\geq \; & \frac{1}{n^2}\left(1-\frac{2}{2^n}\right)\left(\frac{1}{n^c}\right)+\left(1-\frac{1}{n^2}\right)\left(-\frac{2}{2^n}\right) \qquad (4.10)\\
\geq \; & \frac{1}{n^{c+2}}-\frac{4}{2^n} \qquad (4.11)
\end{aligned}
$$

where equality (4.9) is by linearity of expectation, and inequality (4.10) follows from (4.3), (4.5), (4.7), (4.8), and the definition of expected value.

It follows that

$$\Pr_{\substack{(f,A)\leftarrow(\mathcal{F},\mathcal{A})\\ s\leftarrow_r\{0,1\}^n}}\left[\hat{D}^{(f,A)}\left(f(s)\right)=1\right]-\Pr_{\substack{(f,A)\leftarrow(\mathcal{F},\mathcal{A})\\ z\leftarrow_r\{0,1\}^{\ell_1(n)}}}\left[\hat{D}^{(f,A)}\left(z\right)=1\right]>1/n^{c+3}$$

for infinitely many n, contradicting our assumption (4.1). □

By Claim 4.2.2, we have that for all PPT oracle machines $D^{(\cdot,\cdot)}$, all c, and sufficiently large n, the measure of $(f,A)\in(\mathcal{F},\mathcal{A})$ such that

$$\Pr_{s\leftarrow_r\{0,1\}^n}\left[D^{(f,A)}\left(f\left(s\right)\right)=1\right]-\Pr_{z\leftarrow_r\{0,1\}^{\ell_1(n)}}\left[D^{(f,A)}\left(z\right)=1\right]\geq\frac{1}{n^c}$$

is less that $1/n^2$. Then, by the Borel-Cantelli lemma, we have that for all PPT oracle machines $D^{(\cdot,\cdot)}$ and all c, the the measure of $(f,A)\in(\mathcal{F},\mathcal{A})$ such that

$$\Pr_{s\leftarrow_r\{0,1\}^n}\left[D^{(f,A)}\left(f\left(s\right)\right)=1\right]-\Pr_{z\leftarrow_r\{0,1\}^{\ell_1(n)}}\left[D^{(f,A)}\left(z\right)=1\right]\geq\frac{1}{n^c}$$

for infinitely many n is 0.

Then, since there are only countably-many PPT oracle machines and (of course) only countably many $c\in\mathbb{N}$, the measure of $(f,A)\in(\mathcal{F},\mathcal{A})$ such that there exists a PPT oracle machine $D^{(\cdot,\cdot)}$ and a natural number c such that

$$\Pr_{s\leftarrow_r\{0,1\}^n}\left[D^{(f,A)}\left(f\left(s\right)\right)=1\right]-\Pr_{z\leftarrow_r\{0,1\}^{\ell_1(n)}}\left[D^{(f,A)}\left(z\right)=1\right]\geq\frac{1}{n^c}$$

for infinitely many n is 0.

Now, observe that for every $(f,A)\in(\mathcal{F},\mathcal{A})$ and every $c\in\mathbb{N}$, there exists a PPT oracle machine $D^{(\cdot,\cdot)}$ such that

$$\Pr_{s\leftarrow_r\{0,1\}^n}\left[D^{(f,A)}\left(f\left(s\right)\right)=1\right]-\Pr_{z\leftarrow_r\{0,1\}^{\ell_1(n)}}\left[D^{(f,A)}\left(z\right)=1\right]\geq\frac{1}{n^c}$$

for infinitely many n if and only if there exists a PPT oracle machine $D^{(\cdot,\cdot)}$ such that

$$\left| \Pr_{s \leftarrow_r \{0,1\}^n} \left[D^{(f,A)} \left(f \left(s \right) \right) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[D^{(f,A)} \left(z \right) = 1 \right] \right| \geq \frac{1}{n^c}$$

for infinitely many n. The "only if" direction is obvious. For the "if" direction, we note that if $\left| \Pr_{s \leftarrow_r \{0,1\}^n} \left[D^{(f,A)} \left(f \left(s \right) \right) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[D^{(f,A)} \left(z \right) = 1 \right] \right| \geq \frac{1}{n^c}$ for infinitely many n, then either $\Pr_{s \leftarrow_r \{0,1\}^n} \left[D^{(f,A)} \left(f \left(s \right) \right) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[D^{(f,A)} \left(z \right) = 1 \right] \geq \frac{1}{n^c}$ for infinitely many n, or $\Pr_{s \leftarrow_r \{0,1\}^n} \left[D^{(f,A)} \left(f \left(s \right) \right) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[D^{(f,A)} \left(z \right) = 1 \right] \leq -\frac{1}{n^c}$ for infinitely many n; the former case is again obvious, and for the latter case it is sufficient to complement the output of D.

It follows that the measure of $(f, A) \in (\mathcal{F}, \mathcal{A})$ such that there exists a PPT oracle machine $D^{(\cdot,\cdot)}$ and a natural number c such that

$$\left| \Pr_{s \leftarrow_r \{0,1\}^n} \left[D^{(f,A)} \left(f \left(s \right) \right) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_1(n)}} \left[D^{(f,A)} \left(z \right) = 1 \right] \right| \geq \frac{1}{n^c}$$

for infinitely many n is 0, completing the proof of Theorem 4.2.1. □

4.3 Constructions with short seeds

In this section, we consider constructions whose seed length is not more than $O(\log n)$ bits longer than the length n of each oracle query. Recall that such constructions making k adaptive queries to a given pseudo-random generator can achieve stretch that is k times the stretch of the given generator. We show that such constructions making constantly-many *non-adaptive* queries cannot achieve stretch that is *even a single bit* longer than the stretch of the given generator.

Theorem 4.3.1 *Let $k \in \mathbb{N}$, and let $\ell_1(n)$ and $\ell_2(n)$ be polynomials such that $\ell_1(n) \leq n + O(\log n)$ and $\ell_2(n) > n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ be a non-adaptive oracle construction of a number generator, making k queries of length n to an oracle mapping n bits to $\ell_2(n)$ bits. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

The approach we use to prove Theorem 4.3.1 does not seem to extend to the case of polynomially-many (or even $\omega(1)$-many) queries. However, a similar approach does work for polynomially-many queries when we place a restriction on the many-oneness of the number generator's querying function. We state this restriction in Section 4.5.

We give an overview of the proof of Theorem 4.3.1 in Section 4.3.1, and we give the proof details in Section 4.3.2 and Section 4.3.3.

4.3.1 Proof overview for Theorem 4.3.1

A simpler case

We first consider the simpler case of constructions making just a single query, where the query made is required to be the same as the construction's input. That is, we consider constructions $G^{(\cdot)} : \{0,1\}^n \to \{0,1\}^{\ell_2(n)+1}$ such that on every input $x \in \{0,1\}^n$, G makes query x to an oracle mapping n bits to $\ell_2(n)$ bits. Fix such a construction $G^{(\cdot)}$. We need to show the existence of functions $f : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$ and $A : \{0,1\}^{\ell_2(n)} \to \{0,1\}$ such that A breaks the pseudo-randomness of G^f but f is pseudo-random even with respect to adversaries that have oracle access to f and A. Following the approach for proving black-box impossibility results initiated by Impagliazzo and Rudich [IR89], we actually define a *joint distribution* $(\mathcal{F}, \mathcal{A})$ over pairs of functions, such that with probability one over $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, A breaks the pseudo-randomness of G^f but f is pseudo-random even with respect to adversaries that have oracle access to f and A.

Consider how we might define such a joint distribution $(\mathcal{F}, \mathcal{A})$. The most obvious approach is to let $(\mathcal{F}, \mathcal{A})$ be the distribution defined by the following procedure for sampling a tuple $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$: randomly select f from the (infinite) set of all functions that, for each $n \in \mathbb{N}$, map n bits to $\ell_2(n)$ bits; let A be the function such that for every $z \in \{0,1\}^{\ell_2(n)+1}$, $A(z) = 1$ if and only if there exists an $s \in \{0,1\}^n$ such that $G^f(s) = z$. Following this approach, we have that with probability one over $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, A breaks the pseudo-randomness of G^f but f is pseudo-random with respect to adversaries that have oracle access to f alone. However, it is *not* necessarily the case that f is pseudo-random with respect to adversaries that have oracle access to f *and* A. For example, suppose construction G is such that for every $x \in \{0,1\}^{n-1}$ and every $b \in \{0,1\}$, $G^f(x||b) = f(x||b)||b$. In this case, it is easy to use A to break f: on input $y \in \{0,1\}^{\ell_2(n)}$, output 1 if and only if either $A(y||0) = 1$ or $A(y||1) = 1$.

To overcome this problem, we add some "noise" to A. We need to be careful that we add enough noise to A so that it is no longer useful for breaking f, but we do not add so much noise that A no longer breaks G^f. Our basic aproach is to modify A so that instead of only accepting $G^f(s)$ for all $s \in \{0,1\}^n$, A accepts $G^{f_i(s)}$ for all s, all i, and some appropriate collection of functions $\{f_0, f_1, f_2, \dots\}$ where $f_0 = f$. How should this collection of functions be defined? Since we want to make sure that A still breaks G^f, and since we have that A accepts $G^f(s)$ with probability 1 over $s \leftarrow \{0,1\}^n$, we need to ensure that A accepts randomly chosen strings with probability non-negligibly less than 1. For this, it suffices to ensure that (# of n-bit strings s)*(# of functions f_i) is at most, say, half the number of strings of length $\ell_2(n)+1$. At the same time, to prevent A from helping to break f, we would like it to be the case that, intuitively, A treats strings that are *not* in the image of f on an equal footing with strings that *are* in the image of f. One way to accomplish these objectives, which we follow, is to randomly select a permutation π on $\{0,1\}^{\ell_2(n)}$, define $f(x) = \pi(0^{\ell_2(n)-n}||x)$ for all $x \in \{0,1\}^n$, and define A to accept $G^{\pi(y||\cdot)}(s)$ for every $y \in \{0,1\}^{\ell_2(n)-n}$ and every $s \in \{0,1\}^n$. We formalize this as a joint distribution $(\mathcal{F}, \mathcal{A}, \Pi)$ over tuples (f, A, π) that are sampled in the manner just described.

It is easy to show that with probability one over $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, A does indeed break G^f. It is much more difficult to show that with probability one over $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, f is pseudo-random ever with respect to PPT adversaries that have oracle access to f and A. We argue that it suffices to show that for every PPT oracle machine $D^{(\cdot,\cdot)}$, the probability over $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$ and $s \leftarrow \{0,1\}^n$ that $D^{(f,A)}(f(s))$ makes oracle query s to f is negligible. Now, instead of only showing this for every PPT oracle machine $D^{(\cdot,\cdot)}$, we find it more convenient to show this for every computationally unbounded probabilistic oracle machine $D^{(\cdot,\cdot)}$ that makes at most polynomially-many oracle queries. How might we do so? We would like to argue that A does not help D to find s since a computationally unbounded D can try to compute A by itself. More formally, we would like to show that given D, we can build a D' that, given input $f(s)$ and given oracle access only to f, simulates D on input $f(s)$, answers f-queries of D using the given oracle, and "makes up" answers to the A-queries of D in a manner that ensures that the probability that the simulation of D makes query s is very close to the probability that $D^{(f,A)}(f(s))$ makes oracle query s. Of course, D' does not "know" π, so it is not immediately clear how it should answer the A-queries of the simulation of D. If D' simply randomly chooses its own permutation π' and answers A-queries using π' in place of the unknown π, the simulation of D may "notice" this sleight of hand. For example, since D is given $f(s)$ as input, it might (depending on the definition of G) be able to compute the value of $G^f(s)$, and hence make query $G^f(s)$ to A; if this query does not produce response 1, D will "know" that queries are not being responded to properly.

We address this by showing that D' can still compute "most" of A on its own, and that the "rest" of A is not helpful for finding s. Specifically, we split A into two functions, A_1 and A_2, that together can be used to compute A. Function A_1 outputs 1 only on input $G^f(s)$. For every $(\ell_2(n)+1)$-bit string $z \neq G^f(s)$, $A_2(z) = 1$ if and only if $A(z) = 1$. We then argue that querying A_1 provides very little help for finding s. Let X be the set of all strings $x \in \{0,1\}^n$ such that $G^f(x) = G^f(s)$. Roughly speaking, if X is large, then A_1 gives no information about s beyond the fact that $s \in X$. On the other hand, if X is small, then we argue it is unlikely that an adversary making polynomially-many queries to A_1 will receive a non-zero response to any of its queries (in other words, it is unlikely that query $G^f(s)$ will be made). It remains to argue that D' can compute A_2 on its own. We show that if D' randomly selects a permutation π', computes an A_2' based on π' (rather than π), uses this A_2' along with the given A_1 to answer the A-queries of the simulation of D, and answers the f-queries of the simulation of D based on $\pi'(0^{\ell_2(n)-n}||\cdot)$ (rather than using the given oracle f), then it is unlikely that the simulation of D will make a query that "exposes" the fact that its oracle queries are not being answered by f and A.

The general case

We extend the above argument to constructions $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ making constantly-many non-adaptive queries, where the length $\ell_1(n)$ of the construction's input is allowed to be $O(\log n)$ bits longer than the length n of each oracle query. The high-level idea is the same: we define a joint distribution $(\mathcal{F}, \mathcal{A}, \Pi)$ by specifying a procedure for sampling a tuple $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, and the way we sample π and f is (almost) the same as before. But now we change the way A behaves. Our goal is to follow the same style of argument as before. To accomplish this, we would still like it to be the case that when we "split up" A into functions A_1 and A_2, there is still at most one string accepted by A_1 (this helps us ensure that A_1 does not provide too much information about s). Recall that before, when D' was run on an input $f(s)$, the unique string accepted by A_1 was $G^f(s)$. This made sense because in the previous setting, the only input on which $G^{(\cdot)}$ made oracle query s was s itself. But in the current setting, for each $s \in \{0,1\}^n$, there may be many inputs $x \in \{0,1\}^{\ell_1(n)}$ on which $G^{(\cdot)}$ makes oracle query s. We would like to modify the definition of A so that rather than accepting $G^{\pi(y||\cdot)}(x)$ for every $y \in \{0,1\}^{\ell_2(n)-n}$ and *every* $x \in \{0,1\}^{\ell_1(n)}$, A accepts $G^{\pi(y||\cdot)}(x)$ for every $y \in \{0,1\}^{\ell_2(n)-n}$ and x in some subset $\mathit{Good}(n) \subseteq \{0,1\}^{\ell_1(n)}$ such that for every $s \in \{0,1\}^n$, there is at most one $x \in \mathit{Good}(n)$ such that $G^{(\cdot)}$ on input x makes query s. But we cannot do exactly this (and still have that A breaks G^f), since, for example, there might be some string t that $G^{(\cdot)}$ queries no matter what its input is.

Instead, we need to proceed very carefully, partitioning the set of strings t of length n into those that are queried by $G^{(\cdot)}$ for "many" of its inputs $x \in \{0,1\}^{\ell_1(n)}$, and those queried by $G^{(\cdot)}$ for "at most a few" of its inputs $x \in \{0,1\}^{\ell_1(n)}$. We call the former set $\mathit{Fixed}(n)$ and the latter set $\mathit{NotFixed}(n)$. We then define a set $\mathit{Good}(n) \subseteq \{0,1\}^{\ell_1(n)}$ of inputs to $G^{(\cdot)}$ such that for no pair of distinct inputs from $\mathit{Good}(n)$ does $G^{(\cdot)}$ make the same query $t \in \mathit{NotFixed}(n)$. That is, each $t \in \mathit{NotFixed}(n)$ is queried by $G^{(\cdot)}$ for at most one of its inputs $x \in \mathit{Good}(n)$. The challenge, of course, is ensuring that that the set $\mathit{Good}(n)$ defined this way is "large enough".

We define A to accept $G^{\pi(y||\cdot)}(x)$ for every $y \in \{0,1\}^{\ell_2(n)-n}$ and every $x \in \mathit{Good}(n)$. Now we can "split up" A into A_1 and A_2 in a manner similar to what we did before: on input $f(s)$ to D', where $s \in \mathit{NotFixed}(n)$, if there exists a string $x \in \mathit{Good}(n)$ such that $G^{(\cdot)}(x)$ makes query s (note that there can be at most one such string x by definition of $\mathit{Good}(n)$), then A_1 only accepts $G^f(x)$, and if there is no such string x then A_1 does not accept any strings; as before, we define A_2 to accept the remaining strings accepted by A. We then argue as before about the (lack of) usefulness of A_1 and A_2 for helping to find s. Finally, we argue that our definition of $Fixed(n)$ ensures that this set will be of negligible size, and hence it does not hurt to ignore the case $s \in Fixed(n)$ (since this case will occur with negligible probability).

4.3.2 Proof of Theorem 4.3.1: The case $k = 1$

To develop the intuition needed to prove Theorem 4.3.1, we first consider the special case of fully black-box constructions $G^{(\cdot)} : \{0,1\}^n \to \{0,1\}^{\ell_2(n)+1}$ making a *single* query to an oracle mapping n bits to $\ell_2(n)$ bits. Towards this goal, we begin by considering the special case where the only query made by the construction is required to be the same as the construction's input.

Theorem 4.3.2 *Let $\ell(n)$ be a length function. Let $G^{(\cdot)} : \{0,1\}^n \to \{0,1\}^{\ell(n)+1}$ be an oracle construction of a number generator, using its own input as the only query to an oracle mapping n bits to $\ell(n)$ bits. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

Proof To prove this theorem, it suffices to show the existence of functions $f : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ and $A : \{0,1\}^{\ell(n)+1} \to \{0,1\}$ such that A breaks the pseudo-randomness of G^f but every PPT oracle machine $M^{(\cdot,\cdot)}$ is such that $M^{(f,A)}$ does not break the pseudo-randomness of f.

Let $g : \{0,1\}^n \times \{0,1\}^{\ell(n)} \to \{0,1\}^{\ell(n)+1}$ be such that for all $x \in \{0,1\}^n$ and all $\alpha \in \{0,1\}^{\ell(n)}$, $g(x,\alpha)$ is the output of $G^{(\cdot)}$ on input x when given α as the response to its oracle query. That is, for all $x \in \{0,1\}^n$ and for every function $O : \{0,1\}^n \to \{0,1\}^{\ell(n)}$, we have $g(x, O(x)) = G^O(x)$.

We begin by defining a joint distribution $(\mathcal{F}, \mathcal{A}, \Pi) = \{(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)\}$. Specifically, for each $n > 0$, $(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ is the distribution defined by the following procedure for sampling a triple (f_n, A_n, π_n).

- Randomly select a permutation $\pi_n : \{0,1\}^{\ell(n)} \to \{0,1\}^{\ell(n)}$.
- Define function $f_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ as follows: for all $x \in \{0,1\}^n$, $f_n(x) = \pi_n(0^{\ell(n)-n}||x)$.
- Define function $A_n : \{0,1\}^{\ell(n)+1} \to \{0,1\}$ as follows: for all $z \in \{0,1\}^{\ell(n)+1}$, $A_n(z) = 1$ if and only if there exists $x \in \{0,1\}^n$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $g(x, \pi_n(y||x)) = z$.

We now consider the pseudo-randomness of $G^{(\cdot)}$ when its oracle f and the adversary A are chosen as $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$. Also, for $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, we consider the pseudo-randomness of f with respect to adversaries that have oracle access to f and A.

Lemma 4.3.3 *With probability 1 over the choice $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, adversary A breaks the pseudo-randomness of G^f.*

Lemma 4.3.4 *Let $D^{(\cdot,\cdot)}$ be a PPT oracle machine. For all $n \in \mathbb{N}$, define $p_D(n)$ to be the probability that when $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ and $s \leftarrow_r \{0,1\}^n$, $D^{(f,A)}$ accepts $f(s)$. For all $n \in \mathbb{N}$, define $r_D(n)$ to be the probability that when $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ and $z \leftarrow_r \{0,1\}^{\ell(n)}$, $D^{(f,A)}$ accepts z. Then, $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large n.*

Observe that Theorem 4.3.2 follows from Lemma 4.3.3, Lemma 4.3.4, and Theorem 4.2.1. We first prove Lemma 4.3.3.

Proof *(Lemma 4.3.3)* Fix a sample $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$. We will show that A breaks G^f. The main idea is that A accepts *every* pseudo-randomly generated string, but we have by a counting argument that A accepts at most half of all strings in $\{0,1\}^{\ell(n)+1}$.

Fix $n > 0$.

Define $p_A(n)$ to be the probability that A accepts $G^f(s)$ for randomly chosen $s \in \{0,1\}^n$. Fix $s \in \{0,1\}^n$. Recall that by definition of g, we have that $G^f(s) = g(s, f(s))$. Also, by definition of $(\mathcal{F}, \mathcal{A}, \Pi)$, we have that that $f(s) = \pi(0^{\ell(n)-n}||s)$ and A accepts $g(s, \pi(0^{\ell(n)-n}||s))$. That is, A accepts $G^f(s)$. So we have $p_A(n) = 1$.

Define $r_A(n)$ to be the probability that A accepts randomly chosen $z \in \{0,1\}^{\ell(n)+1}$. By definition of $(\mathcal{F}, \mathcal{A}, \Pi)$, we have that the set of $(\ell(n) + 1)$-bit strings accepted by A is exactly

$$\left\{ g\left(x, \pi_n\left(y||x\right)\right) \;:\; x \in \{0,1\}^n \text{ and } y \in \{0,1\}^{\ell(n)-n} \right\}.$$

But this set clearly has size at most $2^{\ell(n)}$. It follows that $r_A(n) \leq 2^{\ell(n)}/2^{\ell(n)+1} = 1/2$. □

Proof *(Lemma 4.3.4)* Intuitively, this lemma says that when we choose $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, even adversaries that have oracle access to A cannot break the pseudo-randomness of f. We first observe that the probabilities defined in the statement of Lemma 4.3.4 can be expressed in a different (but equivalent) way that is more convenient.

For each probabilistic oracle machine $D^{(\cdot,\cdot)}$ and each $n \in \mathbb{N}$, consider the following experiments and associated probabilities.

<u>Experiment 1</u>

(a) Choose $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$ and $s \leftarrow_r \{0,1\}^n$.

(b) Run $D^{(f,A)}$ on input $\pi(0^{\ell(n)-n}||s)$.

Define $p'_D(n)$ to be the probability that D accepts. Define $q^p_D(n)$ to be the probability that D makes oracle query s to f.

<u>Experiment 2</u>

(a) Choose $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, $y \leftarrow_r \{0,1\}^{\ell(n)-n}$, and $s \leftarrow_r \{0,1\}^n$.

(b) Run $D^{(f,A)}$ on input $\pi(y||s)$.

Define $r'_D(n)$ to be the probability that D accepts. Define $q^r_D(n)$ to be the probability that D makes oracle query s to f.

Finally, define $q_D(n) = \max\{q^p_D(n), q^r_D(n)\}$.

Observe that we have $p'_D(n) = p_D(n)$, since $\pi(0^{\ell(n)-n}||s) = f(s)$. Also, observe that we have $r'_D(n) = r_D(n)$, since π is a permutation.

Now, note that in the two experiments defined above, unless D queries f on s, its view will be distributed identically. This holds even if we do not place any computational bound on D.

Claim 4.3.5 *For all probabilistic oracle machines $D^{(\cdot,\cdot)}$ and all $n \in \mathbb{N}$, we have $q_D(n) = q^p_D(n) = q^r_D(n)$ and $|p'_D(n) - r'_D(n)| \leq q_D(n)$.*

Proof *(Claim 4.3.5)* Consider the following experiments, parametrized by probabilistic oracle machine $D^{(\cdot,\cdot)}$ and $n \in \mathbb{N}$.

Experiment 3

(a) Choose $s \leftarrow_r \{0,1\}^n$.

(b) Randomly choose $2^{\ell(n)-n}$ *distinct* strings $z_1, z_2, \ldots, z_{2^{\ell(n)-n}} \in \{0,1\}^{\ell(n)}$.

(c) Let $V = \left\{y||s \ : \ y \in \{0,1\}^{\ell(n)-n}\right\}$. Let $W = \{z_1, z_2, \ldots, z_{2^{\ell(n)-n}}\}$.

(d) Randomly choose bijection $\pi_n : \left(\{0,1\}^{\ell(n)} - V\right) \to \left(\{0,1\}^{\ell(n)} - W\right)$.

(e) Define function $A_n : \{0,1\}^{\ell(n)+1} \to \{0,1\}$ as follows: for all $\gamma \in \{0,1\}^{\ell(n)+1}$, $A_n(\gamma) = 1$ if and only if either there exists $x \in (\{0,1\}^n - \{s\})$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $g(x, \pi_n(y||x)) = \gamma$ OR there exists $z \in W$ such that $g(s,z) = \gamma$.

(f) Define function $f_n : (\{0,1\}^n - \{s\}) \to \{0,1\}^{\ell(n)}$ as follows: for all $x \in (\{0,1\}^n - \{s\})$, $f_n(x) = \pi_n(0^{\ell(n)-n}||x)$.

(g) Define $f_n(s) = z_1$.

(h) For all $i \neq n$, choose $(f_i, A_i) \leftarrow (\mathcal{F}_i, \mathcal{A}_i)$. Define $f = \{f_m\}$ and $A = \{A_m\}$.

(i) Run $D^{(f,A)}$ on input z_1.

Experiment 4

This experiment is identical to Experiment 3, except we modify step (g) as follows:

(g) Choose $z \leftarrow_r W$. Define $f_n(s) = z$.

It is not difficult to verify that s, f, A, and the input to D in Experiment 1 are jointly distributed identically to s, f, A, and the input to D in Experiment 3. Similarly, s, f, A, and the input to D in Experiment 2 are jointly distributed identically to s, f, A, and the input to D in Experiment 4.

The intuition is that even though we do not define π_n on V, we have in mind that $\pi_n(V) = W$ without fixing a particular bijection between V and W; this information is sufficient for defining A_n. Of course, in order to define $f_n(s)$, we need to have a value in mind for $\pi_n(0^{\ell(n)-n}||s)$. In Experiment 3, this value is z_1, the input to D. In Experiment 4, we view the input z_1 to D as the value of $\pi_n(y||s)$ for randomly chosen $y \in \{0,1\}^{\ell(n)-n}$, that is, as the value of $\pi_n(v)$ for randomly chosen $v \in V$; in this case, it suffices to view π_n on V as a randomly chosen bijection between V and W, and hence we view the value of $\pi_n(0^{\ell(n)-n}||s)$ as simply a random element of W.

Observe that Experiments 3 and 4 differ only in step (g), and this difference only affects the view[1] of D when query s is made to f. Indeed, step (g) of these experiments can even be deferred until D makes query s to f. This means that so long as D has *not* made query s to f, the joint distribution of s and the view of D in Experiment 3 is identical to the joint distribution of s and the view of D in Experiment 4. Equivalently, so long as D has *not* made query s to f, the joint distribution of s and the view of D in Experiment 1 is identical to the joint distribution of s and the view of D in Experiment 2. It follows that $q_D(n) = q^p_D(n) = q^r_D(n)$. It also follows that whenever D fails to make query s, it has no information whatsoever to distinguish Experiment 1 from Experiment 2. We conclude that $|p'_D(n) - r'_D(n)| \leq q_D(n)|$. $\square$

To complete the proof of Lemma 4.3.4, it suffices to show that for every PPT oracle machine $D^{(\cdot,\cdot)}$, $q^p_D(n)$ is negligible. It turns out to be more convenient to show something stronger: for every *computationally unbounded* probabilistic oracle machine $D^{(\cdot,\cdot)}$ that makes *at most polynomially-many oracle queries*, we have that $q^p_D(n)$ is negligible.

Recall that $q^p_D(n)$ is the probability, in Experiment 1, that $D^{(f,A)}$ queries oracle f on s. Also recall that in Experiment 1, D's oracles (f, A) are chosen according to distribution $(\mathcal{F}, \mathcal{A})$. By the definition of $(\mathcal{F}, \mathcal{A})$, choosing $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ means choosing (f_i, A_i) according to $(\mathcal{F}_i, \mathcal{A}_i)$ independently for each $i > 0$. It follows that in Experiment 1, for all $i \neq n$, (f_i, A_i) is independent of s even given D's input $f_n(s)$. This means that a computationally unbounded probabilistic oracle machine $D^{(\cdot,\cdot)}$ can simulate (f_i, A_i) for $i \neq n$ on its own, without reducing the probability that it queries f on s. That is, it is sufficient to give such $D^{(\cdot,\cdot)}$ only oracles for (f_n, A_n). For the sake of simplifying our analysis, we in fact replace A_n with a pair of oracles that together are at least as strong as A_n. Consider the following modified version of

[1]The *view* of D consists of its input as well as the responses to its oracle queries.

Experiment 1, parametrized by probabilistic oracle machine $D^{(\cdot,\cdot,\cdot)}$ and $n \in \mathbb{N}$.

Experiment 1′

(a) Choose $(f_n, A_n, \pi_n) \leftarrow (\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ and $s \leftarrow_r \{0,1\}^n$.

(b) Let $\alpha = \pi_n(0^{\ell(n)-n}||s)$.

(c) Define function $A_n^1 : \{0,1\}^{\ell(n)+1} \to \{0,1\}$ as follows: for all $\gamma \in \{0,1\}^{\ell(n)+1}$, $A_n^1(\gamma) =$ 1 if and only if $g(s,\alpha) = \gamma$.

(d) Define function $A_n^2 : \{0,1\}^{\ell(n)+1} \to \{0,1\}$ as follows: for all $\gamma \in \{0,1\}^{\ell(n)+1}$, $A_n^2(\gamma) =$ 1 if and only if there exists $x \in \{0,1\}^n$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $y||x \neq 0^{\ell(n)-n}||s$ and $g(x, \pi_n(y||x)) = \gamma$.

(e) Run $D^{(f_n, A_n^1, A_n^2)}$ on input α.

Observe that for all $\gamma \in \{0,1\}^{\ell(n)+1}$, we have $A_n(\gamma) = \max(A_n^1(\gamma), A_n^2(\gamma))$ and hence D can compute A_n using the oracles it is given for A_n^1 and A_n^2.

Define $q'_D(n)$ to be the probability that D makes oracle query s to f_n.

We need to show that for every computationally unbounded probabilistic oracle machine $D^{(\cdot,\cdot,\cdot)}$ that makes at most polynomially-many oracle queries, $q'_D(n)$ is negligible.

We begin by considering probabilistic oracle machines $E^{(\cdot,\cdot,\cdot)}$ that make no queries to A_n^2 and that query A_n^1 and f_n in a particular structured manner.

Claim 4.3.6 *Let $E^{(\cdot,\cdot,\cdot)}$ be a probabilistic oracle machine. Let function $m(n)$ be a bound on the number of oracle queries made by $E^{(\cdot,\cdot,\cdot)}$ when run on inputs of length $\ell(n)$. Suppose E makes no queries to its third oracle, and uses its first and second oracles in the following restricted "two-phase" manner: initially, E makes queries only to its second oracle; if, at some point, E receives response 1 to an oracle query, then E makes no further queries to its second oracle, and makes queries only to its first oracle. Then, for all n, $q'_E(n) \leq [(m(n))^2 - m(n)]/2^{n+1}$.*

Proof *(Claim 4.3.6)* Fix n. Consider running $E^{(\cdot,\cdot,\cdot)}$ as in Experiment 1′. Let $\mathcal{X}$ denote the set of $x \in \{0,1\}^n$ such that $g(x,\alpha) = g(s,\alpha)$. We will abuse notation by using m to denote $m(n)$; that is, E makes at most m oracle queries.

By assumption, E's behaviour can be viewed as consisting of two phases. In the first phase, E makes queries only to A_n^1. If some query to A_n^1 has response 1, then E immediately enters a second phase where it makes queries only to f_n.

Note that before E begins making queries, we have that given the view of E, every $x \in \{0,1\}^n$ is equally likely to be the value of s. Each query γ to A_n^1 whose response is 0 rules out (as potential values of s) all x such that $g(x,\alpha) = \gamma$. But note that after such a query, the "un-ruled-out" values $x \in \{0,1\}^n$ – that is, all $x \in \{0,1\}^n$ such that query $g(x,\alpha)$ has *not* yet been made to A_n^1 – are all equally likely to be the value of s given the view of E. Similarly, note that a query γ to A_n^1 whose response is 1 rules out (as potential values of s) all x such that $g(x,\alpha) \neq \gamma$ (that is, all $x \notin \mathcal{X}$ are ruled out); immediately following such a query γ, all x' such that $g(x',\alpha) = \gamma$ (that is, all $x' \in \mathcal{X}$) are equally likely, given the view of E, to be the value of s. Once E begins querying f_n (and has so far not queried f_n on s), each query $x \in \mathcal{X}$ whose response is not α rules out x as a potential value of s; after such a query, all the $x' \in \mathcal{X}$ that have not yet been queried to f_n are equally likely, given the view of E, to be the value of s. Also note that once E begins querying f_n, each query $x \notin \mathcal{X}$ provides no information about s, since such x has already been ruled out as a potential value of s.

For $1 \leq N \leq 2^n$ and $w \geq 0$, define $q_E^{(N,w)}$ to be the probability that if E has not yet made a query to A_n^1 whose response is 1, there are N "un-ruled-out" values $x \in \{0,1\}^n$, and E is allowed to make at most w additional oracle queries, then E queries f_n on s. Note that $q'_E(n) \leq q_E^{(2^n,m)}$, and hence $q_E^{(2^n,m)}$ is the value we are ultimately interested in upper bounding.

We will prove by strong induction on w that for all $1 \leq N \leq 2^n$ and $w \geq 0$, we have $q_E^{(N,w)} \leq w(w-1)/(2N)$.

It is clear that $q_E^{(N,0)} = 0$ for all $1 \leq N \leq 2^n$. We also have that $q_E^{(N,1)} = 0$ for all $1 \leq N \leq 2^n$, since E must make a query to A_n^1 whose response is 1 before making queries to f_n, and hence if E is allowed only a single oracle query then it cannot query f_n.

Now consider $q_E^{(N,w)}$ for $w \geq 2$ and $1 \leq N \leq 2^n$. Let γ denote the next query to A_n^1 that will be made by E. Let V be the set of strings x such that $g(x,\alpha) = \gamma$. If $|V| = 0$ or if E has previously made query γ, then $A_n^1(\gamma) = 0$ but no additional $x \in \{0,1\}^n$ will be ruled out by this query; in this case, E has simply "wasted a query", and the probability E queries f_n on s is $q_E^{(N,w-1)}$, which by induction is at most $(w-1)(w-2)/(2N) < w(w-1)2N$. So suppose $|V| > 0$ and E has not previously made query γ. Observe that given the view of E before making query γ, the probability that $s \in V$ is $|V|/N$. Now, if $s \in V$, then $A_n^1(\gamma) = 1$, and hence E will stop querying A_n^1 and henceforth only query f_n. In this case, the probability that E queries f_n on s is at most $(w-1)/|V|$. On the other hand, if $s \notin V$, then $A_n^1(\gamma) = 0$, and hence E will continue making queries to A_n^1. However, since $A_n^1(\gamma) = 0$, all $x \in V$ are ruled out as potential values of s. It follows that, in this case, the probability that E queries f_n on s is at most $q_E^{(N-|V|,w-1)}$; by induction, we have $q_E^{(N-|V|,w-1)} \leq (w-1)(w-2)/(2N-2|V|)$. We then have that

$$\begin{aligned} q_E^{(N,w)} &\leq \frac{|V|}{N} \cdot \frac{w-1}{|V|} + \frac{N-|V|}{N} \cdot \frac{(w-1)(w-2)}{2(N-|V|)} \\ &= \frac{w-1}{N} + \frac{(w-1)(w-2)}{2N} \\ &= \frac{2(w-1)+(w-1)(w-2)}{2N} \\ &= \frac{w(w-1)}{2N}, \end{aligned}$$

as required.

We then have that $q_E^{(2^n,m)} \leq m(m-1)/2^{n+1} = (m^2-m)/2^{n+1}$. □

To finish the proof of Lemma 4.3.4, we consider probabilistic oracle machines $D^{(\cdot,\cdot,\cdot)}$ that have no restrictions in the manner in which they query their oracles.

Claim 4.3.7 *Let $D^{(\cdot,\cdot,\cdot)}$ be a probabilistic oracle machine. Let function $m(n)$ be a bound on the number of oracle queries made by $D^{(\cdot,\cdot,\cdot)}$ when run on inputs of length $\ell(n)$. Then, for all n, $q'_D(n) \leq [(m(n))^2 + 5m(n) + 1]/2^n$.*

Proof *(Claim 4.3.7)* We define a probabilistic oracle machine $E^{(\cdot,\cdot,\cdot)}$ that, when run according to Experiment $1'$, simulates D "almost" according to Experiment $1'$, but uses its own oracles in the restricted two-phase manner described in Claim 4.3.6. $E^{(f_n,A_n^1,A_n^2)}$ will simulate $D^{(\hat{f}_n,\hat{A}_n^1,\hat{A}_n^2)}$. E will answer $\hat{A}_n^1$ queries using its own oracle A_n^1. To answer $\hat{A}_n^2$ and $\hat{f}_n$ queries, E will randomly select a permutation $\hat{\pi}_n : \{0,1\}^{\ell(n)} \to \{0,1\}^{\ell(n)}$, and define $\hat{A}_n^2$ and $\hat{f}_n$ using $\hat{\pi}$ in the same way that A_n^2 and f_n are defined from permutation π_n in Experiment $1'$. For some of the queries x made by D to $\hat{f}_n$, E will query f_n on x (but will not use the response when responding to D). We will bound $q'_E(n)$ using Claim 4.3.6, and argue that E queries f_n on s in Experiment $1'$ whenever its simulation of D queries $\hat{f}_n$ on s. Then, to bound $q'_D(n)$, we will argue that it is unlikely that D will make a query that "exposes" the fact that E is making up the answers it is providing to $\hat{f}_n$ and $\hat{A}_n^2$ queries, and hence the bound on the probability that the simulation of D queries s must be "very close" to a bound on the probability that D queries s in Experiment $1'$.

On input $\alpha \in \{0,1\}^{\ell(n)}$ and with access to oracles $f_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}$, $A_n^1 : \{0,1\}^{\ell(n)+1} \to \{0,1\}$, and $A_n^2 : \{0,1\}^{\ell(n)+1} \to \{0,1\}$, E behaves as follows. E randomly selects $s' \in \{0,1\}^n$, and randomly selects a permutation $\hat{\pi}_n : \{0,1\}^{\ell(n)} \to \{0,1\}^{\ell(n)}$ such that $\hat{\pi}_n(0^{\ell(n)-n}||s') = \alpha$. Then, E simulates $D^{(\cdot,\cdot,\cdot)}$ on input α and with access to oracles $\hat{f}_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}$, $\hat{A}_n^1 : \{0,1\}^{\ell(n)+1} \to \{0,1\}$, and $\hat{A}_n^2 : \{0,1\}^{\ell(n)+1} \to \{0,1\}$ whose behaviour we define now.

Whenever D makes a query x to its oracle $\hat{f}_n$, E provides response $\hat{\pi}_n(0^{\ell(n)-n}||x)$ to D. Then, E checks if $A_n^1(g(x,\alpha)) = 1$: if E has previously made query $\gamma = g(x,\alpha)$ to A_n^1, it checks the response it received to that query; if E has previously received response 1 to a query to A_n^1 different from γ, then it "knows" that $A_n^1(\gamma) \neq 1$ without needing to make any query; otherwise, E makes query γ to A_n^1. If $A_n^1(g(x,\alpha)) = 1$, E makes query x to its oracle f_n.

Whenever D makes a query γ to its oracle $\hat{A}_n^1$, E first checks if it has previously made query γ to (its own oracle) A_n^1. If so, E gives the (previously obtained) value $A_n^1(\gamma)$ to D as the response to query γ. If not, E checks if any previous query (of its own) to A_n^1 had response 1; if so, E gives 0 to D as the response to query γ. Otherwise, E makes query γ to A_n^1, and then gives $A_n^1(\gamma)$ to D as the response to query γ.

Whenever D makes a query γ to its oracle $\hat{A}_n^2$, E checks if there exists $x \in \{0,1\}^n$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $\hat{\pi}_n(y||x) \neq \alpha$ and $g(x, \hat{\pi}_n(y||x)) = \gamma$; if so, E gives 1 to D as the response to query γ, and otherwise E gives 0 to D as the response to query γ. If there is a *unique* string $x \in \{0,1\}^n$ such that $\hat{\pi}_n(0^{\ell(n)-n}||x) \neq \alpha$ and $g(x, \hat{\pi}_n(0^{\ell(n)-n}||x)) = \gamma$, E checks if $A_n^1(g(x,\alpha)) = 1$ (using the same approach it uses to check if $A_n^1(g(x,\alpha)) = 1$ when answering query x to $\hat{f}_n$), and if so, E makes query x to its oracle f_n.

Note that the number of oracle queries made by E is at most one more than the number of oracle queries made by the simulation of D. For each $\hat{A}_n^1$ query made by D, E makes at most one query to A_n^1 and no queries to $\hat{f}_n$. For each $\hat{f}_n$ query x made by D such that $A_n^1(g(x,\alpha)) = 0$, E makes at most one query to A_n^1 and no queries to $\hat{f}_n$. Next, consider $\hat{f}_n$ queries x such that $A_n^1(g(x,\alpha)) = 1$. For the first such query, E makes one query to A_n^1 one query to f_n; for the rest of these queries E makes one query to f_n and no queries to A_n^1. It remains to consider the $\hat{A}_n^2$ queries γ made by D. When there does *not* exist a unique string $x \in \{0,1\}^n$ such that $\hat{\pi}_n(0^{\ell(n)-n}||x) \neq \alpha$ and $g(x, \hat{\pi}_n(0^{\ell(n)-n}||x)) = \gamma$, E does not make any oracle queries. When such a unique string x does exist, E makes the same queries that it would make if D made query x to $\hat{f}_n$. It follows that the number of oracle queries made by E when run on inputs of length $\ell(n)$ is at most $m(n) + 1$.

Consider running $E^{(\cdot,\cdot,\cdot)}$ according to Experiment $1'$, and let s and α be as chosen in this experiment. We claim that E simulates $D^{(\cdot,\cdot,\cdot)}$ "almost according" to Experiment $1'$ with the same choice of α and s. Specifically, if E defined permutation π'_n to be the same as $\hat{\pi}_n$ except that the values of $\hat{\pi}_n$ at points $0^{\ell(n)-n}||s$ and $0^{\ell(n)-n}||s'$ are interchanged (so $\pi'_n(0^{\ell(n)-n}||s) = \hat{\pi}_n(0^{\ell(n)-n}||s') = \alpha$ and $\pi'_n(0^{\ell(n)-n}||s') = \hat{\pi}_n(0^{\ell(n)-n}||s)$) and then used π'_n in place of $\hat{\pi}_n$ to answer the oracle queries of D, then when E is run according to Experiment $1'$ it would simulate D according to Experiment $1'$ with the same choice of α and s. Of course, E is not given s, so it cannot actually construct π'_n. However, since π'_n and $\hat{\pi}_n$ are identical except at points $0^{\ell(n)-n}||s$ and $0^{\ell(n)-n}||s'$, the only queries made by D that may receive a different response from E when

E answers using $\hat{\pi}_n$ instead of π'_n are the following (where we assume, for now, that $s \neq s'$):

1. **Query** s **to** $\hat{f}_n$**.** When E uses $\hat{\pi}_n$, it gives answer $\hat{\pi}_n(0^{\ell(n)-n}||s)$ instead of α.

2. **Query** s' **to** $\hat{f}_n$**.** When E uses $\hat{\pi}_n$, it gives answer α instead of $\hat{\pi}_n(0^{\ell(n)-n}||s)$.

3. **Query** $\gamma_s = g(s, \hat{\pi}_n(0^{\ell(n)-n}||s)) = g(s, \pi'_n(0^{\ell(n)-n}||s'))$ **to** $\hat{A}^2_n$**.** When E uses $\hat{\pi}_n$, it gives answer 1. When E uses π'_n, it gives answer 1 if and only if there exists $x \in \{0,1\}^n$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $\pi'_n(y||x) \neq \alpha$ (that is, $y||x \neq 0^{\ell(n)-n}||s$) and $g(x, \pi'_n(y||x)) = \gamma_s$.

4. **Query** $\delta_{s'} = g(s', \hat{\pi}_n(0^{\ell(n)-n}||s)) = g(s', \pi'_n(0^{\ell(n)-n}||s'))$ **to** $\hat{A}^2_n$**.** When E uses $\hat{\pi}_n$, it gives answer 1 if and only if there exists $x \in \{0,1\}^n$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $\hat{\pi}_n(y||x) \neq \alpha$ (that is, $y||x \neq 0^{\ell(n)-n}||s'$) and $g(x, \hat{\pi}_n(y||x)) = \delta_{s'}$. When E uses π'_n, it gives answer 1 since $\delta_{s'} = g(s', \pi'_n(0^{\ell(n)-n}||s'))$.

We will say that $\gamma_s = g(s, \hat{\pi}_n(0^{\ell(n)-n}||s))$ is *bad* if there does *not* exist $x \in \{0,1\}^n$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $y||x \neq 0^{\ell(n)-n}||s$ and $g(x, \pi'_n(y||x)) = \gamma_s$. We will say that $\delta_{s'} = g(s', \pi'_n(0^{\ell(n)-n}||s'))$ is *bad* if there does *not* exist $x \in \{0,1\}^n$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $y||x \neq 0^{\ell(n)-n}||s'$ and $g(x, \hat{\pi}_n(y||x)) = \delta_{s'}$. We will say that D makes a *bad* query if it makes one of the following queries: s to $\hat{f}_n$, s' to $\hat{f}_n$, bad γ_s to $\hat{A}^2_n$, or bad $\delta_{s'}$ to $\hat{A}^2_n$.

Observe that E simulates D according to Experiment $1'$ until D makes a bad query. This means that the probability that D makes query s in Experiment $1'$ must be at most the probability that the simulation of D makes query s or a bad query, which is simply the probability that the simulation of D makes a bad query. That is, $q'_D(n)$ is at most the probability that the simulation of D makes a bad query[2]. We now bound the probability that the simulation of D makes a bad query.

We first claim that whenever the simulation of D queries s to $\hat{f}_n$ or queries bad γ_s to $\hat{A}^2_n$, E makes query s to f_n. To see this, first suppose the simulation of D makes query s to $\hat{f}_n$. Then, since $A^1_n(g(s,\alpha)) = 1$, E will make query s to f_n. Now suppose the simulation of D queries bad γ_s to $\hat{A}^2_n$. Since γ_s is bad, there is no $x \in \{0,1\}^n$ such that $x \neq s$ and $g(x, \pi'(0^{\ell(n)-n}||x)) = \gamma_s$. Then, since π'_n and $\hat{\pi}_n$ agree everywhere except at $0^{\ell(n)-n}||s$ and $0^{\ell(n)-n}||s'$, there is no $x \in \{0,1\}^n$ such that $x \neq s$, $x \neq s'$, and $g(x, \hat{\pi}(0^{\ell(n)-n}||x)) = \gamma_s$. Also, recall that $\hat{\pi}_n(0^{\ell(n)-n}||s') = \alpha$. This means that there is a unique string $x \in \{0,1\}^n$ – in particular, $x = s$ – such that $\hat{\pi}_n(0^{\ell(n)-n}||x) \neq \alpha$ and $g(x, \hat{\pi}_n(0^{\ell(n)-n}||x)) = \gamma_s$. Then, since $A^1_n(g(s,\alpha)) = 1$, E will make query s to f_n. It follows that the probability that the simulation of D queries s to $\hat{f}_n$ or queries bad γ_s to $\hat{A}^2_n$ is at most $q'_E(n)$.

[2] More precisely, since we have assumed $s' \neq s$, $q'_D(n)$ is at most the probability that either $s' = s$ or the simulation of D makes a bad query.

We next upper-bound the probability that the simulation of D makes at least one bad query *and* that the first such bad query is either s' to $\hat{f}_n$ or bad $\delta_{s'}$ to $\hat{A}^2_n$. When upper-bounding this probability of this event, we will assume that E answers oracle queries using π'_n rather than $\hat{\pi}_n$, since this only changes answers to bad queries, and hence will not change the probability that at at least one bad query is made by the simulation of D, nor will it change the *first* bad query made by D. Under this assumption, observe that the response to each oracle query made by the simulation of D is completely determined by π'_n and α (recall that s itself is completely determined by α and π'_n since $\pi'_n(0^{\ell(n)-n}||s) = \alpha$).

Note that given π'_n, α, and s, we have that s' is simply a random n-bit string different from s. That is, given s along with the entire view of the simulation of D, we have that s' is a random n-bit string different from s. Now, fix π'_n and α (and hence s), and, conditioned on these values, consider the probability that the simulation of D makes at least one bad query and that the first such bad query is either s' to $\hat{f}_n$ or bad $\delta_{s'}$ to $\hat{A}^2_n$. Define $Bad \subseteq \{0,1\}^n - \{s\}$ to be the set of strings z such that if $s' = z$, then $\delta_{s'} = g(s', \pi'_n(0^{\ell(n)-n}||s'))$ is bad (note that fixing s' also fixes $\hat{\pi}_n$, allowing us to determine if $\delta_{s'}$ is bad).

We claim that for all distinct $z_1, z_2 \in Bad$, we have $\delta_{z_1} \neq \delta_{z_2}$. Suppose not, that is, suppose $z_1, z_2 \in Bad$ are distinct strings such that $\delta_{z_1} = \delta_{z_2}$. Now, if $s' = z_1$, we have that $\hat{\pi}_n(0^{\ell(n)-n}||z_2) = \pi'_n(0^{\ell(n)-n}||z_2)$ since z_2 is neither s nor s', and hence $g(z_2, \hat{\pi}_n(0^{\ell(n)-n}||z_2)) = g(z_2, \pi'_n(0^{\ell(n)-n}||z_2)) = \delta_{z_2} = \delta_{z_1} = \delta_{s'}$. But this means that $\delta_{s'}$ is not bad, contradicting $z_1 \in Bad$. Now, definining $Bad^\delta = \{\delta_z \ : \ z \in Bad\}$, we have $|Bad^\delta| = |Bad|$.

First condition on the case that $s' \notin Bad$. Then, s' is uniformly distributed over $(\{0,1\}^n - Bad) - \{s\}$. It follows that the probability that the simulation of D queries s' to $\hat{f}_n$ is at most $m(n)/(2^n - |Bad| - 1)$.

Now condition on the case that $s' \in Bad$. Then, s' is uniformly distributed over Bad, and $\delta_{s'}$ is uniformly distributed over Bad^δ. We will say that $z \in Bad$ is *covered* by the simulation of D if D queries z to $\hat{f}_n$ or D queries δ_z to $\hat{A}^2_n$. Observe that each query made by D can cover at most one $z \in Bad$. It follows that the probability that s' is covered is at most $m(n)/|Bad|$.

Removing conditioning on s', π'_n and α, we have that the probability that the simulation of D makes at least one bad query and that the first such bad query is either s' to $\hat{f}_n$ or bad $\delta_{s'}$ to $\hat{A}^2_n$ is at most

$$\frac{2^n - |Bad| - 1}{2^n - 1} \cdot \frac{m(n)}{2^n - |Bad| - 1} + \frac{|Bad|}{2^n - 1} \cdot \frac{m(n)}{|Bad|} = \frac{2m(n)}{2^n - 1}.$$

Putting everything together, we have that the probability that the simulation of D makes a bad query is at most $q'_E(n) + \frac{2m(n)}{2^n-1}$. Then, recalling that we did our analysis under the assumption that $s' \neq s$, we have that $q'_D(n) \leq q'_E(n) + \frac{2m(n)}{2^n-1} + \frac{1}{2^n}$. It remains to bound $q'_E(n)$.

Observe that E makes no queries to its oracle A^2_n. Furthermore, E's querying behaviour consists of two phases, where in the first phase E makes queries only to A_n^1, the second phase commences as soon as some query receives response 1, and in the second phase E makes queries only to f_n. Then, since E makes at most $m(n)+1$ queries, we have by Claim 4.3.6 that $q'_E(n) \leq \frac{(m(n))^2+m(n)}{2^{n+1}}$.

We now have

$$\begin{aligned} q'_D(n) &\leq \frac{(m(n))^2+m(n)}{2^{n+1}} + \frac{2m(n)}{2^n-1} + \frac{1}{2^n} \\ &< \frac{(m(n))^2+m(n)}{2^n} + \frac{4m(n)}{2^n} + \frac{1}{2^n} \end{aligned}$$

We conclude that $q'_D(n) < [(m(n))^2+5m(n)+1]/2^n$. □

This completes the proof of Lemma 4.3.4. □

This also completes the proof of Theorem 4.3.2.

□

It is easy to extend Theorem 4.3.2 to the case of constructions whose querying function is a permutation.

Corollary 4.3.8 *Let $\ell(n)$ be a length function. Let $G^{(\cdot)}:\{0,1\}^n \rightarrow \{0,1\}^{\ell(n)+1}$ be an oracle construction of a number generator, making a single query of length n to an oracle mapping n bits to $\ell(n)$ bits, such that the querying function of $G^{(\cdot)}$ is a permutation. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

Proof sketch We briefly describe how to modify the proof of Theorem 4.3.2 to obtain Corollary 4.3.8.

Let $Q_G:\{0,1\}^n \rightarrow \{0,1\}^n$ be the querying function of $G^{(\cdot)}$. As before, we define $g:\{0,1\}^n \times \{0,1\}^{\ell(n)} \rightarrow \{0,1\}^{\ell(n)+1}$ to be such that for all $x \in \{0,1\}^n$ and all $\alpha \in \{0,1\}^{\ell(n)}$, $g(x,\alpha)$ is the output of $G^{(\cdot)}$ on input x when given α as the response to its oracle query. That is, for all $x \in \{0,1\}^n$ and every function $O:\{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$, we have $g(x, O(Q_G(x)) = G^O(x)$.

We modify the definition of distribution $(\mathcal{F},\mathcal{A},\Pi)$ in the following way to take into account the behaviour of Q_G. When sampling a triple $(f_n, A_n, \pi_n) \in (\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$, we sample π_n and f_n as before, but we now define function $A_n:\{0,1\}^{\ell(n)+1} \rightarrow \{0,1\}$ as follows: for all $z \in \{0,1\}^{\ell(n)+1}$, $A_n(z)=1$ if and only if there exists $x \in \{0,1\}^n$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $g(Q_G^{-1}(x), \pi_n(y||x)) = z$.

Recall that the proof of Theorem 4.3.2 consists of Lemmas 4.3.3 and 4.3.4. It is straightforward to modify the proof of Lemma 4.3.3 to take into account the new definition of distribution

$(\mathcal{F}, \mathcal{A}, \Pi)$. The changes to the proof of Lemma 4.3.4 are more numerous but none involve any new or difficult ideas – they simply consist of small changes to experiments and simulations to make them consistent with the modified definition of $(\mathcal{F}, \mathcal{A}, \Pi)$. For example, when defining Experiment 3, we modify step (e) so that Q_G^{-1} is applied to the first argument of each occurrence of $g(\cdot, \cdot)$; the other changes to the proof of Lemma 4.3.4 are similar. $\square$

Observe that the changes we make in order to obtain the proof of Corollary 4.3.8 rely on the fact that Q_G^{-1} is well-defined on $\{0,1\}^n$. Once we can no longer rely on this fact (that is, once we consider constructions whose querying function Q_G is not required to be a permutation), the changes we need to make to the proof are more significant.

Theorem 4.3.9 *Let $k \in \mathbb{N}$, and let $\ell_1(n)$ and $\ell_2(n)$ be length functions such that $\ell_1(n) \leq n + O(\log n)$ and $\ell_2(n) > n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ be a non-adaptive oracle construction of a number generator, making a single query of length n to an oracle mapping n bits to $\ell_2(n)$ bits. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

Proof Let $Q_G : \{0,1\}^{\ell_1(n)} \to \{0,1\}^n$ be the querying function of $G^{(\cdot)}$. Define $g : \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\ell_2(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ to be such that for all $x \in \{0,1\}^{\ell_1(n)}$ and all $\alpha \in \{0,1\}^{\ell_2(n)}$, $g(x, \alpha)$ is the output of $G^{(\cdot)}$ on input x when given α as the response to its oracle query. That is, for all $x \in \{0,1\}^{\ell_1(n)}$ and every function $O : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$, we have $g(x, O(Q_G(x))) = G^O(x)$.

There are two cases to consider, one where the image of Q_G is "small" and the other where the image of Q_G is "large".

Define $Q_{G,n}$ to be Q_G restricted to inputs of length $\ell_1(n)$; that is, $Q_{G,n}$ is the querying function of $G^{(\cdot,\cdot)}$ for security parameter n.

For every function f, let $Im(f)$ denote the image of f.

Case 1: $|Im(Q_{G,n})| < \frac{2^n}{n^d}$ for all d and sufficiently large n

This is the easy case. The basic idea is to define a distribution $\mathcal{F} = \{\mathcal{F}_n\}$ over functions mapping n bits to $\ell_2(n)$ bits such that functions chosen according to this distribution are "very non-random" on $Im(Q_{G,n})$ but random everywhere else. Since $Im(Q_G)$ is small, the "non-random" behaviour on $Im(Q_G)$ cannot be used to break the pseudo-randomness of $f \in \mathcal{F}$. On the other hand, the "non-random" behaviour on $Im(Q_G)$ makes it easy to break the pseudo-randomness of G^f for all $f \in \mathcal{F}$.

We begin by defining a distribution $\mathcal{F} = \{\mathcal{F}_n\}$. For each $n > 0$, $\mathcal{F}_n$ is the distribution over functions mapping n bits to $\ell_2(n)$ bits defined by following procedure for sampling a function

f_n. For all $x \in \{0,1\}^n$ such that $x \notin Im(Q_{G,n})$, $f_n(x)$ is a randomly chosen $\ell_2(n)$-bit string. For all $x \in Im(Q_{G,n})$, $f_n(x)$ is $0^{\ell_2(n)}$. In other words, $\mathcal{F}_n$ is the uniform distribution on the set of all functions $f_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$ such that $f_n(x) = 0^{\ell_2(n)}$ for all $x \in Im(Q_{G,n})$.

We now define a function $A = A_n$ (which we will show breaks G^f for all $f \in \mathcal{F}$). For each $n > 0$, $A_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1\}$ is defined as follows: for all $z \in \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$, $A_n(z) = 1$ if and only if there exists $x \in \{0,1\}^{\ell_1(n)}$ such that $g(x, 0^{\ell_2(n)}) = z$.

Claim 4.3.10 *With probability 1 over the choice $f \leftarrow \mathcal{F}$, adversary A breaks the pseudo-randomness of G^f.*

Proof *(Claim 4.3.10)* Fix $f \in \mathcal{F}$. We will show that A breaks G^f. Specifically, we show that A accepts *every* pseudo-randomly generated string, but accepts randomly chosen strings with probability at most $1/4$.

Fix $n > 0$.

First consider the probability that A accepts pseudo-randomly generated strings. Observe that by definition of distribution $\mathcal{F}$, we have that for all $s \in \{0,1\}^{\ell_1(n)}$, $G^f(s) = g(s, f(Q_G(s))) = g(s, 0^{\ell_2(n)})$. Then we have by definition of A that A accepts $G^f(s)$. That is, A accepts pseudo-randomly generated strings with probability 1.

Now consider the probability that A accepts a randomly chosen $(\ell_1(n) + (\ell_2(n) - n) + 1)$-bit string. It is easy to see from the definition of A that A accepts at most $2^{\ell_1(n)}$ strings of length $\ell_1(n) + (\ell_2(n) - n) + 1$. This means that A accepts randomly chosen strings with probability at most $2^{\ell_1(n)}/2^{\ell_1(n)+(\ell_2(n)-n)+1} = 1/2^{\ell_2(n)-n+1}$. Then, since $\ell_2(n) > n$, we have that A accepts randomly chosen strings with probability at most $1/4$. $\square$

To complete Case 1, it suffices to prove the following claim and then apply Theorem 4.2.1.

Claim 4.3.11 *Let $D^{(\cdot,\cdot)}$ be a PPT oracle machine. For all $n \in \mathbb{N}$, define $p_D(n)$ to be the probability that when $f \leftarrow \mathcal{F}$ and $s \leftarrow_r \{0,1\}^n$, $D^{(f,A)}$ accepts $f(s)$. For all $n \in \mathbb{N}$, define $r_D(n)$ to be the probability that when $f \leftarrow \mathcal{F}$ and $z \leftarrow_r \{0,1\}^{\ell_2(n)}$, $D^{(f,A)}$ accepts z. Then, $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large n.*

Proof *(Claim 4.3.11)* It turns out to be more convenient to consider *computationally unbounded* probabilistic oracle machines $D^{(\cdot,\cdot)}$ that make at most polynomially-many queries, rather than only considering PPT oracle machines. Observe that since A is a computable function, a computationally unbounded machine can compute A for itself (that is, without being given an oracle for A). We will therefore show that for every computationally unbounded probabilistic oracle machine $D^{(\cdot)}$ that makes at most polynomially-many oracle queries, if we define

$$\hat{p}_D(n) = \Pr_{\substack{f \leftarrow \mathcal{F} \\ s \leftarrow_r \{0,1\}^n}} \left[D^f\left(f(s)\right) = 1 \right]$$

and

$$\hat{r}_D(n) = \Pr_{\substack{f \leftarrow \mathcal{F} \\ z \leftarrow_r \{0,1\}^{\ell_2(n)}}} \left[D^f(z) = 1 \right],$$

then $|\hat{p}_D(n) - \hat{r}_D(n)| < 1/n^c$ for all c and sufficiently large n.

Let $D^{(\cdot)}$ be a probabilistic oracle machine that makes at most polynomially-many oracle queries. Let $m(n)$ be a polynomial that bounds the number of oracle queries made by $D^{(\cdot)}$ when $D^{(\cdot)}$ is run on inputs of length $\ell_2(n)$. Consider the following experiments.

Experiment 1

(a) Choose $f \leftarrow \mathcal{F}$

(b) Choose $s \leftarrow_r \{0,1\}^n$.

(c) Define $z = f(s)$

(d) Run D^f on input z.

Experiment 2

(a) Choose $f \leftarrow \mathcal{F}$

(b) Choose $s \leftarrow_r \{0,1\}^n$.

(c) Choose $z \leftarrow_r \{0,1\}^{\ell_2(n)}$

(d) Run D^f on input z.

Observe that $\hat{p}_D(n)$ is the probability that D outputs 1 in Experiment 1, and $\hat{r}_D(n)$ is the probability that D outputs 1 in Experiment 2.

Now note that conditioned on $s \notin Im(Q_G)$, D's view is identical in these two experiments until it queries s; specifically, D sees a randomly chosen input and (until it queries s) sees random and independently chosen answers to each distinct oracle query. It follows that $|\hat{p}_D(n) - \hat{r}_D(n)|$ is at most the probability that either $s \in Im(Q_G)$, or both $s \notin Im(Q_G)$ and D queries s. The probability that $s \in Im(Q_G)$ is exactly $|Im(Q_{G,n})|/2^n$. Also, it is easy to see that conditioned on $s \notin Im(Q_G)$, the probability that D queries s is at most $m(n)/(2^n - |Im(Q_{G,n})|)$. We then have

$$\begin{aligned} |\hat{p}_D(n) - \hat{r}_D(n)| &\leq \frac{|Im(Q_{G,n})|}{2^n} + \frac{2^n - |Im(Q_{G,n})|}{2^n} \cdot \frac{m(n)}{2^n - |Im(Q_{G,n})|} \\ &= \frac{|Im(Q_{G,n})|}{2^n} + \frac{m(n)}{2^n}. \end{aligned}$$

But then since (by assumption) $|Im(Q_{G,n})| < \frac{2^n}{n^d}$ for all d and sufficiently large n, and since $m(n)$ is a polynomial, we have that $|\hat{p}_D(n) - \hat{r}_D(n)| < 1/n^c$ for all c and sufficiently large n. $\square$

Case 2: $|Im(Q_G)| \geq \frac{2^n}{n^d}$ for some d and infinitely many n The main idea for this case is to restrict our attention to a portion of the domain of Q_G such that Q_G is 1-1 on this portion. This allows us to proceed largely as in the proof of Theorem 4.3.2. We need this portion of the domain of Q_G to be large, and for this we rely on the assumption about $|Im(Q_G)|$.

Fix d such that $|Im(Q_G)| \geq \frac{2^n}{n^d}$ for infinitely many n. Let $\mathcal{N}$ be the set of all $n \in \mathbb{N}$ such that $|Im(Q_G)| \geq \frac{2^n}{n^d}$.

For each $n \in \mathcal{N}$, we will abuse notation by letting $Q_{G,n}^{-1} : Im(Q_{G,n}) \to \{0,1\}^{\ell_1(n)}$ be the function defined as follows: for all $x \in Im(Q_{G,n})$, $Q_{G,n}^{-1}(x)$ is the lexicographically-first string y such that $Q_{G,n}(y) = x$.

Now, as in the proof of Theorem 4.3.2, we define a joint distribution $(\mathcal{F}, \mathcal{A}, \Pi) = \{(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)\}$.

For each $n \notin \mathcal{N}$, $(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ is the distribution defined by the following procedure for sampling a triple (f_n, A_n, π_n).

- Define $\pi_n : \{0,1\}^{\ell_2(n)} \to \{0,1\}^{\ell_2(n)}$ to be the identity function.
- Randomly select a function $f_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$.
- Define function $A_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1\}$ to be the constant 0 function.

For each $n \in \mathcal{N}$, $(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ is the distribution defined by the following procedure for sampling a triple (f_n, A_n, π_n).

- Randomly select a permutation $\pi_n : \{0,1\}^{\ell_2(n)} \to \{0,1\}^{\ell_2(n)}$.
- Define function $f_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$ as follows: for all $x \in \{0,1\}^n$, $f_n(x) = \pi_n(0^{\ell_2(n)-n}||x)$.
- Define function $A_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1\}$ as follows: for every z, $A_n(z) = 1$ if and only if there exists $x \in Im(Q_{G,n})$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $g(Q_{G,n}^{-1}(x), \pi_n(y||x)) = z$.

Observe that for $n \in \mathcal{N}$, the definition of distribution $(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ is very similar to the definition in the proof of Theorem 4.3.2, except that we now need to take into account $Q_{G,n}^{-1}$ when sampling function A_n.

Now, for $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, we consider the pseudo-randomness of G^f with respect to adversary A, and we consider the pseudo-randomness of f with respect to adversaries that have oracle access to f and A.

Lemma 4.3.12 *With probability 1 over the choice $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, adversary A breaks the pseudo-randomness of G^f.*

Lemma 4.3.13 *Let $D^{(\cdot,\cdot)}$ be a PPT oracle machine. For all $n \in \mathbb{N}$, define $p_D(n)$ to be the probability that when $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ and $s \leftarrow_r \{0,1\}^n$, $D^{(f,A)}$ accepts $f(s)$. For all $n \in \mathbb{N}$, define $r_D(n)$ to be the probability that when $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ and $z \leftarrow_r \{0,1\}^{\ell_2(n)}$, $D^{(f,A)}$ accepts z. Then, $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large n.*

Observe that Lemma 4.3.12, Lemma 4.3.13, and Theorem 4.2.1 complete the proof of Case 2.

We first prove Lemma 4.3.12.

Proof *(Lemma 4.3.12)* Fix a sample $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$. We will show that A breaks G^f. The main idea is that for all $n \in \mathcal{N}$, A accepts at least a $|Im(Q_{G,n})|/2^{\ell_1(n)}$ fraction of pseudo-randomly generated strings, but by a counting argument, A accepts at most a $|Im(Q_{G,n})|/2^{\ell_1(n)+1}$ fraction of randomly chosen strings.

Fix $n \in \mathcal{N}$.

Define $p_A(n)$ to be the probability that A accepts $G^f(s)$ for randomly chosen $s \in \{0,1\}^{\ell_1(n)}$. We claim that A accepts $G^f(s)$ for every $s \in Im(Q_{G,n}^{-1})$. Fix $s \in Im(Q_{G,n}^{-1})$. Let $t = Q_{G,n}(s)$ (and hence we have $s = Q_{G,n}^{-1}(t)$). Recall that by definition of g, we have that $G^f(s) = g(s, f(t))$. Also, by definition of $(\mathcal{F}, \mathcal{A}, \Pi)$, we have that that $f(t) = \pi(0^{\ell_2(n)-n}||t)$ and A accepts $g(s, \pi(0^{\ell_2(n)-n}||t))$. That is, A accepts $G^f(s)$. So we have $p_A(n) \geq |Im(Q_{G,n})|/2^{\ell_1(n)}$.

Define $r_A(n)$ to be the probability that A accepts randomly chosen $z \in \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$. It is easy to see by the definition of $(\mathcal{F}, \mathcal{A}, \Pi)$ that A accepts at most $|Im(Q_{G,n})| \cdot 2^{\ell_2(n)-n}$ strings of length $\ell_1(n) + (\ell_2(n) - n) + 1$. This means that $r_A(n)$ is at most $|Im(Q_{G,n})| \cdot 2^{\ell_2(n)-n}/2^{\ell_1(n)+(\ell_2(n)-n)+1} = |Im(Q_{G,n})|/2^{\ell_1(n)+1}$.

So we have $p_A(n) - r_A(n) \geq |Im(Q_{G,n})|/2^{\ell_1(n)+1}$ for all $n \in \mathcal{N}$. Then, since (by assumption) $|Im(Q_G)| \geq 2^n/n^d$ for all $n \in \mathcal{N}$, we have $p_A(n) - r_A(n) \geq 2^n/2^{\ell_1(n)+1+d\log n}$ for all $n \in \mathcal{N}$. Finally, since $\ell_1(n) \leq n + O(\log n)$, we conclude that $p_A(n) - r_A(n) > 1/n^c$ for some c and sufficiently large $n \in \mathcal{N}$ (and hence infinitely many $n \in \mathbb{N}$). □

It remains to prove Lemma 4.3.13.

Proof *(Lemma 4.3.13)* The proof is very similar to the proof of Lemma 4.3.4. For the sake of conciseness, we focus only on the differences.

First, note that by the definition of distribution $(\mathcal{F}, \mathcal{A}, \Pi)$, it is easy to see that for every PPT oracle machine $D^{(\cdot,\cdot)}$, we have $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large $n \notin \mathcal{N}$. The point is that for such n, when we choose $(f_n, A_n) \leftarrow (\mathcal{F}_n, \mathcal{A}_n)$, f_n is a randomly-chosen function and A_n is a constant function (which is of no help for breaking the pseudo-randomness of f_n). It therefore suffices to show that $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large $n \in \mathcal{N}$.

Now, for each probabilistic oracle machine $D^{(\cdot,\cdot)}$ and each $n \in \mathcal{N}$, consider the following experiment.

Experiment 1

(a) Choose $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$ and $s \leftarrow_r \{0,1\}^n$.

(b) Run $D^{(f,A)}$ on input $\pi(0^{\ell_2(n)-n}||s)$.

Define $q_D^p(n)$ to be the probability that D makes oracle query s to f.

By the same argument as in the proof of Lemma 4.3.4, we have $|p_D(n) - r_D(n)| \leq q_D^p(n)$, and hence it suffices to bound $q_D^p(n)$.

Now, as in the proof of Lemma 4.3.4, instead of considering only PPT oracle machine $D^{(\cdot,\cdot)}$, it is more convenient to consider computationally unbounded probabilistic oracle machines $D^{(\cdot,\cdot)}$ that make at most polynomially-many queries. Also as before, it suffices to give such machines $D^{(\cdot,\cdot)}$ oracles for (f_n, A_n) in Experiment 1, since for all $i \neq n$, we have that (f_i, A_i) is independent of s even given D's input $f(s)$, and hence such machines $D^{(\cdot,\cdot)}$ can sample (f_i, A_i) for $i \neq n$ by themselves without reducing the probability that they make oracle query s. Finally, as before, we modify Experiment 1 by replacing A_n with a pair of oracle that together are at least as strong as A_n. The details are different, since the definition of A_n itself is different, and hence we now describe the entire modified experiment, parametrized by probabilistic oracle machine $D^{(\cdot,\cdot,\cdot)}$ and $n \in \mathcal{N}$.

Experiment 1′

(a) Choose $(f_n, A_n, \pi_n) \leftarrow (\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ and $s \leftarrow_r \{0,1\}^n$.

(b) Let $\alpha = \pi_n(0^{\ell_2(n)-n}||s)$.

(c) Define function $A_n^1 : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1,\perp\}$ as follows:
  - If $s \notin Im(Q_{G,n})$: for all $\gamma \in \{0,1\}^{\ell_2(n)+(\ell_1(n)-n)+1}$, $A_n^1(\gamma) = \perp$.
  - If $s \in Im(Q_{G,n})$: for all $\gamma \in \{0,1\}^{\ell_2(n)+(\ell_1(n)-n)+1}$, if $g(Q_{G,n}^{-1}(s), \alpha) = \gamma$ then $A_n^1(\gamma) = 1$, and otherwise $A_n^1(\gamma) = 0$.

(d) Define function $A_n^2 : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1\}$ as follows:
For all $\gamma \in \{0,1\}^{\ell_2(n)+(\ell_1(n)-n)+1}$, $A_n^2(\gamma) = 1$ if and only if there exists $x \in Im(Q_{G,n})$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $y||x \neq 0^{\ell_2(n)-n}||s$ and $g(Q_{G,n}^{-1}(x), \pi_n(y||x)) = \gamma$.

(e) Run $D^{(f_n, A_n^1, A_n^2)}$ on input α.

Observe that for all $\gamma \in \{0,1\}^{\ell(n)+1}$, we have $A_n(\gamma) = 1$ if either $A_n^1(\gamma) = 1$ or $A_n^2(\gamma) = 1$, and otherwise we have $A_n(\gamma) = 0$. This means that D can compute A_n using the oracles it is given for A_n^1 and A_n^2.

Define $q'_D(n)$ to be the probability that D makes oracle query s to f_n.

To complete the proof of Lemma 4.3.13, it suffices to prove the following claim.

Claim 4.3.14 *Let $D^{(\cdot,\cdot,\cdot)}$ be a probabilistic oracle machine. Let function $m(n)$ be a bound on the number of oracle queries made by $D^{(\cdot,\cdot,\cdot)}$ when run on inputs of length $\ell_2(n)$. Then, for all $n \in \mathcal{N}$, $q'_D(n) \leq [(m(n))^2 + 7m(n) + 1]/2^n$.*

Proof *(Claim 4.3.14)* We proceed in a manner very similar to the proof of Claim 4.3.7. As before, we define a probabilistic machine $E^{(\cdot,\cdot,\cdot)}$ that, when run according to Experiment $1'$, simulates D "almost" according to Experiment $1'$, and we analyze this simulation in order to bound $q'_D(n)$. The key difference is that in our definition of E, we need to do cases based on whether $s \in Im(Q_{G,n})$. In particular, E will use its oracle A^1_n to determine if $s \in Im(Q_{G,n})$. When $s \in Im(Q_{G,n})$, E will randomly choose $s' \in Im(Q_{G,n})$ and then proceed as before, and otherwise E will randomly choose $s' \in (\{0,1\}^n - Im(Q_{G,n}))$ and then proceed almost as before.

We now provide more details, and rely on the arguments in the proof of Claim 4.3.7 where possible.

On input $\alpha \in \{0,1\}^{\ell_2(n)}$ and with access to oracles $f_n : \{0,1\}^n \rightarrow \{0,1\}^{\ell_2(n)}$, $A^1_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$, and $A^2_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$, E behaves as follows. E first makes query $0^{\ell_1(n)+(\ell_2(n)-n)+1}$ to A^1_n. If the response to this query is $\perp$, then E randomly selects $s' \in (\{0,1\}^n - Im(Q_{G,n}))$; otherwise, E randomly selects $s' \in Im(Q_{G,n})$. Then, E randomly selects a permutation $\hat{\pi}_n : \{0,1\}^{\ell_2(n)} \rightarrow \{0,1\}^{\ell_2(n)}$ such that $\hat{\pi}_n(0^{\ell_2(n)-n}||s') = \alpha$. Then, E simulates $D^{(\cdot,\cdot,\cdot)}$ on input α and with access to oracles $\hat{f}_n : \{0,1\}^n \rightarrow \{0,1\}^{\ell_2(n)}$, $\hat{A}^1_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$, and $\hat{A}^2_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ defined almost as in the proof of Claim 4.3.7, except that we need to use $Q^{-1}_{G,n}$ when answering certain queries. For the sake of completeness, we now fully describe the behaviour of these oracles.

If the response to E's initial oracle query was $\perp$, then E responds to D's oracle queries as follows. Whenever D makes query x to its oracle $\hat{f}_n$, E provides response $\hat{\pi}_n(0^{\ell_2(n)-n}||x)$ to D, and makes query x to f_n. Whenever D makes a query γ its oracle $\hat{A}^1_n$, E provides response $\perp$. Whenever D makes a query γ to its oracle $\hat{A}^2_n$, E checks if there exists $x \in Im(Q_{G,n})$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $g(Q^{-1}_{G,n}(x), \hat{\pi}_n(y||x)) = \gamma$; if so, E gives 1 to D as the response to query γ, and otherwise E gives 0 to D as the response to query γ.

If the response to E's initial oracle query was not $\perp$, then E responds to D's oracle queries as follows.

Whenever D makes a query x to its oracle $\hat{f}_n$, E provides response $\hat{\pi}_n(0^{\ell_2(n)-n}||x)$ to D. If $x \notin Im(Q_{G,n})$, then E takes no further action for query x. Otherwise, E checks if $A^1_n(g(Q^{-1}_{G,n}(x), \alpha)) = 1$: if E has previously made query $\gamma = g(Q^{-1}_{G,n}(x), \alpha)$ to A^1_n, it checks the response it received to that query; if E has previously received response 1 to a query to A_n^1 different from γ, then it "knows" that $A_n^1(\gamma) \neq 1$ without needing to make any query; otherwise, E makes query γ to A_n^1. If $A_n^1(g(Q_{G,n}^{-1}(x), \alpha)) = 1$, E makes query x to its oracle f_n.

Whenever D makes a query γ to its oracle $\hat{A}_n^1$, E first checks if it has previously made query γ to (its own oracle) A_n^1. If so, E gives the (previously obtained) value $A_n^1(\gamma)$ to D as the response to query γ. If not, E checks if any previous query (of its own) to A_n^1 had response 1; if so, E gives 0 to D as the response to query γ. Otherwise, E makes query γ to A_n^1, and then gives $A_n^1(\gamma)$ to D as the response to query γ.

Whenever D makes a query γ to its oracle $\hat{A}_n^2$, E checks if there exists $x \in Im(Q_{G,n})$ and $y \in \{0,1\}^{\ell(n)-n}$ such that $\hat{\pi}_n(y||x) \neq \alpha$ and $g(Q_{G,n}^{-1}(x), \hat{\pi}_n(y||x)) = \gamma$; if so, E gives 1 to D as the response to query γ, and otherwise E gives 0 to D as the response to query γ. If there is a *unique* string $x \in Im(Q_{G,n})$ such that $\hat{\pi}_n(0^{\ell_2(n)-n}||x) \neq \alpha$ and $g(Q_{G,n}^{-1}(x), \hat{\pi}_n(0^{\ell_2(n)-n}||x)) = \gamma$, E checks if $A_n^1(g(Q_{G,n}^{-1}(x), \alpha)) = 1$ (using the same approach it uses to check if $A_n^1(g(Q_{G,n}^{-1}(x), \alpha)) = 1$ when answering query x to $\hat{f}_n$), and if so, E makes query x to its oracle f_n.

Observe that when the response to E's initial oracle query is $\perp$, the total number of oracle queries made by E is at most one more than the number of oracle queries made by simulation of D, that is, at most $m(n) + 1$. Also, when the response to E's initial oracle query is not $\perp$, then by exactly the reasoning given in the proof of Claim 4.3.7, the number of oracle queries made by E (not counting the initial query) is at most one more than the number of oracle queries made by the simulation of D; it follows that in this case, the total number of oracle queries made by E is at most $m(n) + 2$.

Consider running $E^{(\cdot,\cdot,\cdot)}$ according to Experiment $1'$, and let s and α be as chosen in this experiment. We claim that, as in the proof of Claim 4.3.7, E simulates $D^{(\cdot,\cdot,\cdot)}$ "almost according" to Experiment $1'$ with the same choice of α and s. Specifically, if E defined permutation π'_n to be the same as $\hat{\pi}_n$ except that the values of $\hat{\pi}_n$ at points $0^{\ell_2(n)-n}||s$ and $0^{\ell_2(n)-n}||s'$ are interchanged (so $\pi'_n(0^{\ell_2(n)-n}||s) = \hat{\pi}_n(0^{\ell_2(n)-n}||s') = \alpha$ and $\pi'_n(0^{\ell_2(n)-n}||s') = \hat{\pi}_n(0^{\ell_2(n)-n}||s)$) and then used π'_n in place of $\hat{\pi}_n$ to answer the oracle queries of D, then when E is run according to Experiment $1'$ it would simulate D according to Experiment $1'$ with the same choice of α and s. Of course, E is not given s, so it cannot actually construct π'_n. However, since π'_n and $\hat{\pi}_n$ are identical except at points $0^{\ell_2(n)-n}||s$ and $0^{\ell_2(n)-n}||s'$, there are only a small number of possible queries made by D that receive a different response from E when E answers using $\hat{\pi}_n$ instead of π'_n.

First consider the case that the first query made by E receives response $\perp$, that is, the case $s \notin Im(Q_{G,n})$.

This case occurs with probability exactly $(2^n - |Im(Q_{G,n})|)/2^n$. In this case, by definition of E, we also have $s' \notin Im(Q_{G,n})$. But then it is easy to see by the way that E answers oracle queries in this case that all queries made by D to $\hat{A}_n^1$ and $\hat{A}_n^2$ receive the same response whether E answers using π_n' or $\hat{\pi}_n$. This means that the only queries made by D that receive a different response from E when E answers using $\hat{\pi}_n$ instead of π_n' are queries s and s' to $\hat{f}_n$ (of course, if it happens to be the case that $s = s'$, then even these queries receive the same response whether E answers using π_n' and $\hat{\pi}_n$).

Now, it is easy to see that the probability that D makes query s' when $s \neq s'$ (and hence s' is uniformly distributed over $(\{0,1\}^n - Im(Q_{G,n})) - \{s\})$ is at most $m(n)/(2^n - |Im(Q_{G,n})| - 1)$. The probability that $s' \neq s$ is exactly $(2^n - |Im(Q_{G,n})| - 1)/(2^n - |Im(Q_{G,n})|)$. Then, the probability that both $s' \neq s$ *and* D makes query s is at most $m(n)/(2^n - |Im(Q_{G,n})|)$.

Also, note that whenever D makes query s, so does E. But it is easy to see that the probability that E makes query s is at most $m(n)/(2^n - |Im(Q_{G,n})|)$, since the only information that E gets from oracle A_n^1 is that $s \notin Im(Q_{G,n})$, E makes no queries to A_n^2, and E makes at most $m(n)$ queries to f_n. This means that the probability that D makes query s is at most $m(n)/(2^n - |Im(Q_{G,n})|)$.

Note that E simulates D according to Experiment $1'$ until D makes query s or query $s' \neq s$. This means the probability that D makes query s in Experiment $1'$ is at most the probability that the simulation of D makes query s or query $s' \neq s$. That is, conditioned on $s \notin Im(Q_{G,n})$, the probability that D makes query s in Experiment $1'$ is at most $2m(n)/(2^n - |Im(Q_{G,n})|)$.

Now consider the case that the first query made by E receives a response different from $\perp$, that is, the case $s \in Im(Q_{G,n})$.

This case occurs with probability exactly $|Im(Q_{G,n})|/2^n$. In this case, as in the proof of Claim 4.3.7, there are at most four possible queries made by D that receive a different response from E (where we assume, for now, that $s \neq s'$):

1. **Query s to $\hat{f}_n$.** When E uses $\hat{\pi}_n$, it gives answer $\hat{\pi}_n(0^{\ell_2(n)-n}||s)$ instead of α.

2. **Query s' to $\hat{f}_n$.** When E uses $\hat{\pi}_n$, it gives answer α instead of $\hat{\pi}_n(0^{\ell_2(n)-n}||s)$.

3. **Query $\gamma_s = g(Q_{G,n}^{-1}(s), \hat{\pi}_n(0^{\ell_2(n)-n}||s)) = g(Q_{G,n}^{-1}(s), \pi_n'(0^{\ell_2(n)-n}||s'))$ to $\hat{A}_n^2$.** When E uses $\hat{\pi}_n$, it gives answer 1. When E uses π_n', it gives answer 1 if and only if there exists $x \in Im(Q_{G,n})$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $\pi_n'(y||x) \neq \alpha$ (that is, $y||x \neq 0^{\ell_2(n)-n}||s$) and $g(Q_{G,n}^{-1}(x), \pi_n'(y||x)) = \gamma_s$.

4. **Query $\delta_{s'} = g(Q_{G,n}^{-1}(s'), \hat{\pi}_n(0^{\ell_2(n)-n}||s)) = g(Q_{G,n}^{-1}(s'), \pi_n'(0^{\ell_2(n)-n}||s'))$ to $\hat{A}_n^2$.** When E uses $\hat{\pi}_n$, it gives answer 1 if and only if there exists $x \in Im(Q_{G,n})$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $\hat{\pi}_n(y||x) \neq \alpha$ (that is, $y||x \neq 0^{\ell_2(n)-n}||s'$) and $g(Q_{G,n}^{-1}(x), \hat{\pi}_n(y||x)) = \delta_{s'}$. When E uses π_n', it gives answer 1 since $\delta_{s'} = g(Q_{G,n}^{-1}(s'), \pi_n'(0^{\ell_2(n)-n}||s'))$.

Staying consistent with the terminology from the proof of Claim 4.3.7, we will say that $\gamma_s = g(Q^{-1}_{G,n}(s), \hat{\pi}_n(0^{\ell_2(n)-n}||s))$ is *bad* if there does *not* exist $x \in Im(Q_{G,n})$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $y||x \neq 0^{\ell_2(n)-n}s$ and $g(Q^{-1}_{G,n}(x), \pi'_n(y||x)) = \gamma_s$. We will say $\delta_{s'} = g(Q^{-1}_{G,n}(s'), \pi'_n(0^{\ell_2(n)-n}||s'))$ is *bad* if there does *not* exist $x \in Im(Q_{G,n})$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $y||x \neq 0^{\ell_2(n)-n}||s'$ and $g(Q^{-1}_{G,n}(x), \hat{\pi}_n(y||x)) = \delta_{s'}$. We will say that D makes a *bad* query if it makes one of the following queries: s to $\hat{f}_n$, s' to $\hat{f}_n$, bad γ_s to $\hat{A}^2_n$, or bad $\delta_{s'}$ to $\hat{A}^2_n$.

Observe that E simulates D according to Experiment $1'$ until D makes a bad query. This means that the probability that D makes query s in Experiment $1'$ must be at most the probability that the simulation of D makes query s or a bad query, which is simply the probability that the simulation of D makes a bad query. Now, to bound the probability that the simulation of D makes a bad query, it suffices to apply the reasoning used in the proof of Claim 4.3.7, modified to account for the fact that in the current proof, s and s' are randomly chosen from $Im(Q_{G,n})$, not from $\{0,1\}^n$. We then have that the probability that the simulation of D makes a bad query is at most $((m(n))^2 + 5m(n) + 1)/|Im(Q_{G,n})|$ (this also accounts for the possibility that $s' = s$). That is, conditioned on $s \in Im(Q_{G,n})$, the probability that D makes query s in Experiment $1'$ is at most $((m(n))^2 + 5m(n) + 1)/|Im(Q_{G,n})|$.

Then, putting together our probability bounds for the two cases $s \in Im(Q_{G,n})$ and $s \notin Im(Q_{G,n})$, we have

$$\begin{aligned} q'_D(n) &\leq \frac{2^n - |Im(Q_{G,n})|}{2^n} \cdot \frac{2m(n)}{2^n - |Im(Q_{G,n})|} + \frac{|Im(Q_{G,n})|}{2^n} \cdot \frac{(m(n))^2 + 5m(n) + 1}{|Im(Q_{G,n})|} \\ &= \frac{(m(n))^2 + 7m(n) + 1}{2^n}, \end{aligned}$$

and hence we have finished proving the claim. □

This completes the proof of Lemma 4.3.13. □

This also completes the proof of Theorem 4.3.9.

□

4.3.3 Proof of Theorem 4.3.1: The general case

We now consider constructions that make constantly-many queries. Recall that when considering constructions that make a single query (that is, when proving Theorem 4.3.9), we did cases based on whether the image of the querying function was "large" or "small". When the image was "large", we restricted our attention (when defining our distribution over oracles) to a sufficiently large portion of the domain of the querying function such that the querying function was 1-1 on this portion. This had the effect of uniquely associating particular inputs to the construction with particular queries. We would like to follow a similar approach here.

However, the querying function is more complicated now, since in addition to an input x of the construction, it also takes a query number (between 0 and $k-1$) as input. When the image of the querying function is "large", we would now like to define a sufficiently large set $\mathit{Good}(n)$ of inputs to the construction where each input in $\mathit{Good}(n)$ is uniquely associated with a particular *set* of k queries, such that no pair of these inputs shares a common query. But this might not be possible – for example, the construction might make query $\langle 0 \rangle$ on *every* input, and yet still have a querying function with a large image as a result of the other queries made on each input. This means we need to proceed more carefully when defining $\mathit{Good}(n)$, by considering, for each i, whether the size of the image of the *querying function restricted to the i-th query* is "large" or "small". We then combine ideas from Case 1 and Case 2 of the proof of Theorem 4.3.9.

Proof *(Theorem 4.3.1)* Let $Q_G : \{0,1\}^{\ell_1(n)+\log k} \to \{0,1\}^n$ be the querying function of $G^{(\cdot)}$. We assume without loss of generality that $G^{(\cdot)}$ always makes k *distinct* queries. We also assume without loss of generality that Q_G encodes the queries of $G^{(\cdot)}$ in lexicographical order: specifically, we assume that for every $x \in \{0,1\}^{\ell_1(n)}$ and every $0 \le i < j < k$, we have $Q_G(x, \langle i \rangle) < Q_G(x, \langle j \rangle)$. For all $n > 0$, define $Q_{G,n}$ to be Q_G restricted to inputs of length $\ell_1(n) + \log k$; that is, $Q_{G,n}$ is the querying function of $G^{(\cdot,\cdot)}$ for security parameter n.

Define $g : \{0,1\}^{\ell_1(n)} \times \{0,1\}^{k \cdot \ell_2(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ to be such that for all $x \in \{0,1\}^{\ell_1(n)}$ and all $\alpha_0, \ldots, \alpha_{k-1} \in \{0,1\}^{\ell_2(n)}$, $g(x, \alpha_0, \ldots, \alpha_{k-1})$ is the output of $G^{(\cdot)}$ on input x when given $\alpha_0, \ldots, \alpha_{k-1}$ as the responses to its oracle queries. That is, for every $x \in \{0,1\}^{\ell_1(n)}$ and for every function $O : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$, we have $g(x, O(Q_G(x, \langle 0 \rangle)), \ldots, O(Q_G(x, \langle k-1 \rangle))) = G^O(x)$.

We now give a procedure that iteratively defines a sequence of sets $\mathcal{N}_k \subseteq \mathcal{N}_{k-1} \cdots \subseteq \mathcal{N}_0 \subseteq \mathbb{N}$ and, at the same time, for each $0 \le i \le k$ and for each $n \in \mathcal{N}_i$, defines a set $\mathit{Good}_i(n) \subseteq \{0,1\}^{\ell_1(n)}$. The procedure also defines a set $\mathit{Small} \subseteq \{0, \ldots k-1\}$, and a sequence of polynomials $p_{i+1}(n)$ for $i \notin \mathit{Small}$. We need the following properties:

(i) For all $0 \le i \le k$, $\mathcal{N}_i$ is of infinite size.

(ii) For all $0 \le i \le k$, there exists a polynomial $p(n)$ such that for sufficiently large $n \in \mathcal{N}_i$, $|\mathit{Good}_i(n)| \ge 2^{\ell_1(n)}/p(n)$.

(iii) For all $0 \le i \le k$, all $n \in \mathcal{N}_i$, all $x, x' \in \mathit{Good}_i(n)$, all $0 \le j < i$ and all $0 \le j' < k$, if $x \ne x'$ and $j \notin \mathit{Small}$, then $Q_G(x, \langle j \rangle) \ne Q_G(x', \langle j' \rangle)$.

Initially, $\mathit{Small} = \emptyset$. We define $\mathcal{N}_0 = \mathbb{N}$ and $\mathit{Good}_0(n) = \{0,1\}^{\ell_1(n)}$ for all $n \in \mathbb{N}$. We then proceed as follows:

For every $1 \le i \le k$ do:

1. For each $n \in \mathcal{N}_{i-1}$, define $Image_{i-1}(n) = \{Q_G(x, \langle i-1 \rangle) : x \in Good_{i-1}(n)\}$.

2. If there exists a polynomial $p(n)$ such that $|Image_{i-1}(n)| \geq 2^n/p(n)$ for infinitely many $n \in \mathcal{N}_{i-1}$ then:

2.1 Define $p_i(n)$ to be such a polynomial.

2.2 Define $\mathcal{N}_i$ to be the maximal subset of $\mathcal{N}_{i-1}$ such that $|Image_{i-1}(n)| \geq 2^n/p_i(n)$ for all $n \in \mathcal{N}_i$.

2.3 For each $n \in \mathcal{N}_i$ do:

2.3.1 Let $Image'_{i-1}(n) = Image_{i-1}(n) - \bigcup_{j \in Small} Image_j(n)$.

2.3.2 While $Image'_{i-1}(n) \neq \emptyset$ do:

2.3.2.1 Let $y \in Image'_{i-1}(n)$ be lexicographically first.

2.3.2.2 Let $x \in Good_{i-1}(n)$ be the lexicographically first string such that $Q_G(x, \langle i-1 \rangle) = y$.

2.3.2.3 Add x to $Good_i(n)$.

2.3.2.4 For every $y \in \{Q_G(x, \langle j \rangle) : i-1 \leq j < k\} \cap Image'_{i-1}(n)$, remove y from $Image'_{i-1}(n)$.

3. Else:

3.1 Define $\mathcal{N}_i = \mathcal{N}_{i-1}$.

3.2 Define $Good_i(n) = Good_{i-1}(n)$.

3.3 Add $i-1$ to $Small$.

Now, define $\mathcal{N} = \mathcal{N}_k$, and for all $n \in \mathcal{N}$, define $Good(n) = Good_k(n)$.

We note that the set $Good(n)$ is computable given input $n \in \mathcal{N}$. It is not hard to see that a Turing machine that has set $Small$ and polynomials p_{i+1} for $i \in (\{0, \ldots k-1\} - Small)$ hardcoded can use the ideas from the above procedure to compute $Good(n)$.

It is easy to see that property (i) is satisfied. The fact that property (iii) is satisfied can be shown by induction on i, $0 \leq i \leq k$, where we use the fact that Q_G encodes queries in lexicographical order and the fact that step 2.3.2.1 in the procedure selects elements of $Image'_{i-1}$ in lexicographical order, and hence steps 2.3.1 and 2.3.2.4 prevent any potential violations of property (iii). We now show that property (ii) is satisfied.

Claim 4.3.15 *For all $0 \leq i \leq k$, there exists a polynomial $p(n)$ such that for sufficiently large $n \in \mathcal{N}_i$, $|Good_i(n)| \geq 2^{\ell_1(n)}/p(n)$.*

Proof *(Claim 4.3.15)* We will use induction on i. The base case is trivial since $Good_0(n) = \{0,1\}^{\ell_1(n)}$. So fix $0 \leq i < k$, and suppose there exists a polynomial $p(n)$ such that for all $n \in \mathcal{N}_i$, $|Good_i(n)| \geq 2^{\ell_1(n)}/p(n)$. If $i \in Small$, then $Good_{i+1}(n) = Good_i(n)$ and $\mathcal{N}_{i+1} = \mathcal{N}_i$ so we are done. So suppose $i \notin Small$; then, by definition of $\mathcal{N}_{i+1}$ and p_{i+1}, we have

$|Image_i(n)| \geq 2^n/p_{i+1}(n)$ for all $n \in \mathcal{N}_{i+1}$. Now, note that by definition of $Image'_i(n)$, we have $|Image'_i(n)| \geq |Image_i(n)| - \sum_{j \in Small, j<i} |Image_j(n)|$ for all $n \in \mathcal{N}_{i+1}$. But for every $j \in Small$, we must have $|Image_j(n)| < 2^n/(k \cdot p_{i+1}(n))$ for sufficiently large $n \in \mathcal{N}_{j+1}$. Then, for $j \in Small$ such that $j < i$, we must have $|Image_j(n)| < 2^n/(k \cdot p_{i+1}(n))$ for sufficiently large $n \in \mathcal{N}_{i+1}$, since $\mathcal{N}_{i+1} \subseteq \mathcal{N}_{j+1}$ for all $j < i$. It follows that $|Image'_i(n)| \geq |Image_i(n)| - (k-1)2^n(k \cdot p_{i+1}(n)) \geq 2^n/(k \cdot p_{i+1}(n))$ for sufficiently large $n \in \mathcal{N}_{i+1}$. Now, observe that by the procedure used to construct $Good_{i+1}(n)$, we have $|Good_{i+1}(n)| \geq |Image'_i(n)|/k$ for all $n \in \mathcal{N}_{i+1}$. It follows that for sufficiently $n \in \mathcal{N}_{i+1}$, we have $|Good_{i+1}(n)| \geq 2^n/(k^2 \cdot p_i(n))$. To conclude, recall that by assumption, we have $\ell_1(n) \leq n + O(\log n)$. Let c be such that $\ell_1(n) \leq n + c \log n$ for sufficiently large n. Then, for sufficiently large $n \in \mathcal{N}_{i+1}$, we have $|Good_{i+1}(n)| \geq 2^{n + c\log n}/(k^2 n^c p_i(n)) \geq 2^{\ell_1(n)}/(k^2 n^c p_i(n))$. □

Claim 4.3.16 *There exists a polynomial $p(n)$ such that for sufficiently large $n \in \mathcal{N}$, $|Good(n)| \geq 2^{\ell_1(n)}/p(n)$.*

Proof *(Claim 4.3.16) Follows immediately from Claim 4.3.15 and from the definitions of $Good(n)$ and $\mathcal{N}$.* □

Claim 4.3.17 *For all $n \in \mathcal{N}$, all $x, x' \in Good(n)$, and all $0 \leq j, j' < k$, if $x \neq x'$ and $j \notin Small$, then $Q_G(x, \langle j \rangle) \neq Q_G(x', \langle j' \rangle)$.*

Proof *(Claim 4.3.17)* Follows immediately from property (iii) of the above procedure and from the definitions of $Good(n)$ and $\mathcal{N}$. □

For all $n \in \mathcal{N}$, define $Fixed(n) = \{Q_G(x, \langle i \rangle) : i \in Small \text{ and } x \in Good(n)\}$, $NotFixed(n) = \{0,1\}^n - Fixed(n)$. and $GoodQueries(n) = \{Q_G(x, \langle i \rangle) : i \notin Small \text{ and } x \in Good(n)\}$.

For each $n \in \mathcal{N}$, we will abuse notation by letting $Q^{-1}_{G,n} : GoodQueries(n) \to Good(n)$ be the function defined as follows: for all $x \in GoodQueries(n)$, $Q^{-1}_{G,n}(x)$ is the unique string $y \in Good(n)$ for which there exists an $i \notin Small$ such that $Q_{G,n}(y, \langle i \rangle) = x$. $Q^{-1}_{G,n}$ is well-defined by Claim 4.3.17 and the definition of $GoodQueries(n)$.

Finally, for each $n \in \mathcal{N}$ and each $x \in \{0,1\}^n$, if $x \in GoodQueries(n)$ then define $Siblings_n(x)$ to be the set $\left\{Q_G(Q^{-1}_{G,n}(x), \langle i \rangle) : i \notin Small\right\}$, and if $x \notin GoodQueries(n)$ then define $Siblings_n(x)$ to be the set $\{x\}$.

We note that the sets $Fixed(n)$ and $GoodQueries(n)$ are computable given input $n \in \mathcal{N}$, since the set $Good(n)$ and the function Q_G are computable.

Claim 4.3.18 $|Fixed(n)| < \frac{2^n}{n^d}$ *for all d and sufficiently large $n \in \mathcal{N}$.*

Proof *(Claim 4.3.18)* By the above procedure, we have that $|\{Q_G(x, \langle i \rangle) : x \in \mathit{Good}_i(n)\}| < \frac{2^n}{n^d}$ for all $i \in \mathit{Small}$, all d, and sufficiently large $n \in \mathcal{N}$. Then, since $\mathit{Good}_i(n) \subseteq \mathit{Good}(n)$ for all $i \leq k$ and all $n \in \mathcal{N}$, and since $|\mathit{Small}| \leq k$, we have

$$|\mathit{Fixed}(n)| = \left| \bigcup_{i \in \mathit{Small}} \{Q_G(x, \langle i \rangle) : x \in \mathit{Good}(n)\} \right| < \frac{2^n}{n^d}$$

for all d and sufficiently large $n \in \mathcal{N}$. □

Now, as in the proof of Theorem 4.3.9, we define a joint distribution $(\mathcal{F}, \mathcal{A}, \Pi) = \{(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)\}$.

For each $n \notin \mathcal{N}$, $(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ is the distribution defined by the following procedure for sampling a triple (f_n, A_n, π_n).

- Define $\pi_n : \{0,1\}^{\ell_2(n)} \to \{0,1\}^{\ell_2(n)}$ to be the identity function.
- Randomly select a function $f_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$.
- Define function $A_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1\}$ to be the constant 0 function.

For each $n \in \mathcal{N}$, $(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ is the distribution defined by the following procedure for sampling a triple (f_n, A_n, π_n).

- Randomly select a permutation $\rho_n : \{0,1\}^{\ell_2(n)-n} \times \mathit{NotFixed}(n) \to \{0,1\}^{\ell_2(n)-n} \times \mathit{NotFixed}(n)$. Define function $\pi_n : \{0,1\}^{\ell_2(n)} \to \{0,1\}^{\ell_2(n)}$ as follows: for all $y \in \{0,1\}^{\ell_2(n)-n}$ and all $x \in \{0,1\}^n$, if $x \in \mathit{NotFixed}(n)$ then $\pi_n(y||x) = \rho_n(y, x)$, and otherwise $\pi_n(y||x) = 0^{\ell_2(n)-n}||x$.
- Define function $A_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1\}$ as follows: for every z, $A_n(z) = 1$ if and only if there exists an $x \in \mathit{Good}(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that

$$g(x, \pi_n(y||Q_G(x, \langle 0 \rangle)), \ldots, \pi_n(y||Q_G(x, \langle k-1 \rangle))) = z.$$

In light of the way we have defined distribution $(\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ for $n \in \mathcal{N}$, we will also find it convenient to define a modified version of function g that "automatically" uses response $0^{\ell_2(n)}q$ for each query $q \in \mathit{Fixed}(n)$. For each $n \in \mathcal{N}$, define $\hat{g}_n : \mathit{Good}(n) \times \{0,1\}^{(k-|\mathit{Small}|)\ell_2(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ to be such that for all $x \in \mathit{Good}(n)$ and all $\alpha_0, \ldots \alpha_{k-|\mathit{Small}|-1} \in \{0,1\}^{\ell_2(n)}$, $\hat{g}_n(x, \alpha_0, \ldots \alpha_{k-|\mathit{Small}|-1})$ is the output of $G^{(\cdot)}$ on input x when given $\alpha_0, \ldots \alpha_{k-|\mathit{Small}|-1}$ as the responses its queries that are not in $\mathit{Fixed}(n)$ and when given $0^{\ell_2(n)}q$ for each query $q \in \mathit{Fixed}(n)$. That is, letting $I = \{0, \ldots, k-1\} - \mathit{Small}$ and letting $i_0, \ldots, i_{k-|\mathit{Small}|-1}$ denote the members of I in lexicographical order, we have that for all $x \in \mathit{Good}(n)$ and all $\alpha_0, \ldots \alpha_{k-|\mathit{Small}|-1} \in \{0,1\}^{\ell_2(n)}$, $\hat{g}_n(x, \alpha_0, \ldots \alpha_{k-|\mathit{Small}|-1}) = g(x, \alpha'_0, \ldots, \alpha'_k)$ where for $0 \leq j \leq k - |\mathit{Small}| - 1$ we have $\alpha'_{i_j} = \alpha_j$ and for $i \notin I$ we have $\alpha'_i = 0^{\ell_2(n)-n}Q_G(x, \langle i \rangle)$.

Now, for $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, we consider the pseudo-randomness of G^f with respect to adversary A, and we consider the pseudo-randomness of f with respect to adversaries that have oracle access to f and A.

Lemma 4.3.19 *With probability 1 over the choice $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, adversary A breaks the pseudo-randomness of G^f.*

Lemma 4.3.20 *Let $D^{(\cdot,\cdot)}$ be a PPT oracle machine. For all $n \in \mathbb{N}$, define $p_D(n)$ to be the probability that when $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ and $s \leftarrow_r \{0,1\}^n$, $D^{(f,A)}$ accepts $f(s)$. For all $n \in \mathbb{N}$, define $r_D(n)$ to be the probability that when $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$ and $z \leftarrow_r \{0,1\}^{\ell_2(n)}$, $D^{(f,A)}$ accepts z. Then, $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large n.*

Observe that Lemma 4.3.19, Lemma 4.3.20, and Theorem 4.2.1 complete the proof of Theorem 4.3.1.

We first prove Lemma 4.3.19.

Proof *(Lemma 4.3.19)* Fix a sample $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$. We will show that A breaks G^f.

Fix $n \in \mathcal{N}$.

Define $p_A(n)$ to be the probability that A accepts $G^f(s)$ for randomly chosen $s \in \{0,1\}^{\ell_1(n)}$. We have by definition of $(\mathcal{F}, \mathcal{A}, \Pi)$ that A accepts $G^f(s)$ for every $s \in \mathit{Good}(n)$. This means that $p_A(n) \geq |\mathit{Good}(n)|/2^{\ell_1(n)}$.

Define $r_A(n)$ to be the probability that A accepts randomly chosen $z \in \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$. It is easy to see by the definition of $(\mathcal{F}, \mathcal{A}, \Pi)$ that A accepts at most $|\mathit{Good}(n)| \cdot 2^{\ell_2(n)-n}$ strings of length $\ell_1(n) + (\ell_2(n) - n) + 1$. This means that $r_A(n)$ is at most $|\mathit{Good}(n)| \cdot 2^{\ell_2(n)-n}/2^{\ell_1(n)+(\ell_2(n)-n)+1} = |\mathit{Good}(n)|/2^{\ell_1(n)+1}$.

So we have $p_A(n) - r_A(n) \geq |\mathit{Good}(n)|/2^{\ell_1(n)+1}$ for all $n \in \mathcal{N}$. Now, let $q(n)$ be a polynomial such that for sufficiently large $n \in \mathcal{N}$, $|\mathit{Good}(n)| \geq 2^{\ell_1(n)}/q(n)$; such a polynomial exists by Claim 4.3.16. We then have that $p_A(n) - r_A(n) \geq 1/(2q(n))$ for sufficiently large $n \in \mathcal{N}$ (and hence for infinitely many $n \in \mathbb{N}$). $\square$

It remains to prove Lemma 4.3.20.

Proof *(Lemma 4.3.20)* The approach we use is similar to the proof of Lemma 4.3.13.

First, note that by the definition of distribution $(\mathcal{F}, \mathcal{A}, \Pi)$, it is easy to see that for every PPT oracle machine $D^{(\cdot,\cdot)}$, we have $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large $n \notin \mathcal{N}$. The point is that for such n, when we choose $(f_n, A_n) \leftarrow (\mathcal{F}_n, \mathcal{A}_n)$, f_n is a randomly-chosen function and A_n is a constant function (which is of no help for breaking the pseudo-randomness of f_n). It therefore suffices to show that for every PPT oracle machine $D^{(\cdot,\cdot)}$, we have $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large $n \in \mathcal{N}$.

Now, for each probabilistic oracle machine $D^{(\cdot,\cdot)}$ and each $n \in \mathcal{N}$, consider the following experiments.

Experiment 1

(a) Choose $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$ and $s \leftarrow_r \{0,1\}^n$.

(b) Run $D^{(f,A)}$ on input $\pi(0^{\ell_2(n)-n}||s)$.

Define $p'_D(n)$ to be the probability that D accepts. Define $q^p_D(n)$ to be the probability that D queries a string from $Siblings_n(s)$ to f.

Experiment 2

(a) Choose $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, $y \leftarrow_r \{0,1\}^{\ell_2(n)-n}$, and $s \leftarrow_r \{0,1\}^n$.

(b) Run $D^{(f,A)}$ on input $\pi(y||s)$.

Define $r'_D(n)$ to be the probability that D accepts. Define $q^r_D(n)$ to be the probability that D queries a string from $Siblings_n(s)$ to f.

Finally, define $q_D(n) = \max\{q^p_D(n), q^r_D(n)\}$.

Observe that we have $p'_D(n) = p_D(n)$, since $\pi(0^{\ell(n)-n}||s) = f(s)$. Also, observe that in Experiment 2, we have by definition of $(\mathcal{F}, \mathcal{A}, \Pi)$ that each string $z \in \{0,1\}^{\ell_2(n)-n} \times NotFixed(n)$ has probability exactly $1/2^{\ell_2(n)}$ of being the input to D – that is, such strings are chosen as the input to D with the same probability in Experiment 2 as in an alternative experiment where the input to D is chosen uniformly at random. This means that we have $|r'_D(n) - r_D(n)| \le |Fixed(n)|/2^n$.

Then, for every PPT oracle machine $D^{(\cdot,\cdot)}$ and every $n \in \mathcal{N}$, we have $|p_D(n) - r_D(n)| = |p'_D(n) - r_D(n)| \le |p'_D(n) - r'_D(n)| + |Fixed(n)|/2^n$. However, by Claim 4.3.18, we have that $|Fixed(n)|/2^n < 1/n^d$ for all d and sufficiently large n. This means that in order to show that for every PPT oracle machine $D^{(\cdot,\cdot)}$, we have $|p_D(n) - r_D(n)| < 1/n^c$ for all c and sufficiently large $n \in \mathcal{N}$, it suffices to show that for every PPT oracle machine $D^{(\cdot,\cdot)}$, we have $|p'_D(n) - r'_D(n)| < 1/n^c$ for all c and sufficiently large $n \in \mathcal{N}$.

Claim 4.3.21 *For all probabilistic oracle machines $D^{(\cdot,\cdot)}$ and all $n \in \mathbb{N}$, we have $q_D(n) = q^p_D(n) = q^r_D(n)$ and $|p'_D(n) - r'_D(n)| \le q_D(n)$.*

Proof *(Claim 4.3.21)* Consider the following experiments, parametrized by probabilistic oracle machine $D^{(\cdot,\cdot)}$ and $n \in \mathcal{N}$.

Experiment 3

(a) Choose $s \leftarrow_r \{0,1\}^n$.

(b) If $s \in \mathit{Fixed}(n)$, choose $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, run $D^{(f,A)}$ on input $0^{\ell_2(n)}||s$, and skip the remaining steps of this experiment.

(c) Let $\sigma = |\mathit{Siblings}_n(s)|$. Let $s_1, \ldots, s_\sigma$ denote the strings in $\mathit{Siblings}_n(s)$ in lexicographical order. Randomly choose $\sigma \cdot 2^{\ell_2(n)-n}$ *distinct* strings $z_{i,j} \in \{0,1\}^{\ell_2(n)}, 1 \leq i \leq \sigma, 1 \leq j \leq 2^{\ell_2(n)-n}$. Let $W = \{z_{i,j} : 1 \leq i \leq \sigma,\ 1 \leq j \leq 2^{\ell_2(n)-n}\}$.

(d) Randomly choose bijection $\rho_n : \{0,1\}^{\ell_2(n)-n} \times (\mathit{NotFixed}(n) - \mathit{Siblings}_n(s)) \rightarrow \{0,1\}^{\ell_2(n)-n} \times (\mathit{NotFixed}(n) - W)$. Define function $\pi_n : \{0,1\}^{\ell_2(n)-n} \times (\{0,1\}^n - \mathit{Siblings}_n(s)) \rightarrow \{0,1\}^{\ell_2(n)-n} \times (\{0,1\}^n - W)$ as follows: for all $y \in \{0,1\}^{\ell_2(n)-n}$ and all $x \in \{0,1\}^n - \mathit{Siblings}_n(s)$, if $x \in \mathit{NotFixed}(n)$ then $\pi_n(y,x) = \rho_n(y,x)$, and otherwise $\pi_n(y,x) = 0^{\ell_2(n)-n}||x$.

(e) If $s \notin \mathit{GoodQueries}(n)$, define function $A_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \rightarrow \{0,1\}$ as follows: for every z, $A_n(z) = 1$ if and only if there exists an $x \in \mathit{Good}(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $g(x, \pi_n(y, Q_G(x, \langle 0 \rangle)), \ldots, \pi_n(y, Q_G(x, \langle k-1 \rangle))) = z$.

If $s \in \mathit{GoodQueries}(n)$ define function $A_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \rightarrow \{0,1\}$ as follows: for every γ, $A_n(\gamma) = 1$ if and only if *either* there exists an $x \in \mathit{Good}(n) - \{Q^{-1}_{G,n}(s)\}$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that

$$g(x, \pi_n(y, Q_G(x, \langle 0 \rangle)), \ldots, \pi_n(y, Q_G(x, \langle k-1 \rangle))) = \gamma$$

or there exists a j, $1 \leq j \leq 2^{\ell)_2(n)-n}$, such that $\gamma = \hat{g}_n(Q^{-1}_{G,n}(s), z_{1,j}, \ldots, z_{\sigma,j})$.

(f) Define function $f_n : (\{0,1\}^n - \mathit{Siblings}_n(s)) \rightarrow \{0,1\}^{\ell_2(n)}$ as follows: for all $x \in (\{0,1\}^n - \mathit{Siblings}_n(s))$, $f_n(x) = \pi_n(0^{\ell(n)-n}x)$.

(g) For all $1 \leq i \leq \sigma$, define $f_n(s_i) = z_{i,1}$.

(h) For all $i \neq n$, choose $(f_i, A_i) \leftarrow (\mathcal{F}_i, \mathcal{A}_i)$. Define $f = \{f_m\}$ and $A = \{A_m\}$.

(i) Let h be such that $s_h = s$, and run $D^{(f,A)}$ on input $z = z_{h,1}$.

<u>Experiment 4</u>

This experiment is identical to Experiment 3, except we modify step (g) as follows:

(g) Choose $j \leftarrow_r \{1, \ldots, 2^{\ell_2(n)-n}\}$. For all $1 \leq i \leq \sigma$, define $f_n(s_i) = z_{i,j}$.

It is not difficult to verify that s, f, A, and the input to D in Experiment 1 are jointly distributed identically to s, f, A, and the input to D in Experiment 3. Similarly, s, f, A, and the input to D in Experiment 2 are jointly distributed identically to s, f, A, and the input to D in Experiment 4.

The intuition for the case $s \notin \mathit{Fixed}(n)$ is that even though we do not define π_n on $\{0,1\}^{\ell_2(n)-n} \times \mathit{Siblings}_n(s)$, we have in mind that for each $1 \le i \le \sigma$, $\pi_n(\{0,1\}^{\ell_2(n)-n} \times \{s_i\}) = \{z_{i,1}, \ldots z_{i,2^{\ell_2(n)-n}}\}$ without fixing a particular bijection between $\{0,1\}^{\ell_2(n)-n} \times \{s_i\}$ and $\{z_{i,1}, \ldots z_{i,2^{\ell_2(n)-n}}\}$; this information is sufficient for defining A_n. Of course, in order to define $f_n(s_i)$ for each $1 \le i \le \sigma$, we need to have a value in mind for $\pi_n(0^{\ell_2(n)-n}||s_i)$. In Experiment 3, this value is $z_{i,1}$. In Experiment 4, we view the input $z_{h,1}$ to D as the value of $\pi_n(y||s_h)$ for randomly chosen $y \in \{0,1\}^{\ell_2(n)-n}$. In this case, it suffices to view π_n on $\{0,1\}^{\ell_2(n)-n} \times \{s_h\}$ as a *randomly* chosen bijection between $\{0,1\}^{\ell_2(n)-n} \times \{s_h\}$ and $\{z_{h,1}, \ldots z_{h,2^{\ell_2(n)-n}}\}$, and hence we view $\pi_n(0^{\ell_2(n)-n}||s_h)$ as simply $z_{h,j}$ for randomly chosen j; for the sake of consistency with the behaviour of A_n, we view $\pi_n(0^{\ell_2(n)-n}||s_i)$ for all $1 \le i \le \sigma$ as $z_{i,j}$ for the same randomly chosen j.

Observe that Experiments 3 and 4 differ only in step (g), and this difference only affects the view of D when a query from $\mathit{Siblings}_n(s)$ is made to f. Indeed, step (g) of these experiments can even be deferred until D makes a query from $\mathit{Siblings}_n(s)$ to f. This means that so long as D has *not* made a query from $\mathit{Siblings}_n(s)$ to f, the joint distribution of $\mathit{Siblings}_n(s)$ and the view of D in Experiment 3 is identical to the joint distribution of $\mathit{Siblings}_n(s)$ and the view of D in Experiment 4. Equivalently, so long as D has *not* made query from $\mathit{Siblings}_n(s)$ to f, the joint distribution of $\mathit{Siblings}_n(s)$ and the view of D in Experiment 1 is identical to the joint distribution of $\mathit{Siblings}_n(s)$ and the view of D in Experiment 2. It follows that $q_D(n) = q^p_D(n) = q^r_D(n)$. It also follows that whenever D fails to make a query from $\mathit{Siblings}_n(s)$ to f, it has no information whatsoever to distinguish Experiment 1 from Experiment 2. We conclude that $|p'_D(n) - r'_D(n)| \le q_D(n)|$. $\square$

Now, as in the proof of Lemma 4.3.4, instead of considering only PPT oracle machine $D^{(\cdot,\cdot)}$, it is more convenient to consider computationally unbounded probabilistic oracle machines $D^{(\cdot,\cdot)}$ that make at most polynomially-many queries. Also as before, it suffices to give such machines $D^{(\cdot,\cdot)}$ oracles for (f_n, A_n) in Experiment 1, since for all $i \neq n$, we have that (f_i, A_i) is independent of s even given D's input $f(s)$, and hence such machines $D^{(\cdot,\cdot)}$ can sample (f_i, A_i) for $i \neq n$ by themselves without reducing the probability that they make oracle query s. Finally, as before, we modify Experiment 1 by replacing A_n with a pair of oracle that together are at least as strong as A_n. We also provide A_n with additional information as input. We now describe the entire modified experiment, parametrized by probabilistic oracle machine $D^{(\cdot,\cdot,\cdot)}$ and $n \in \mathcal{N}$.

<u>Experiment 1′</u>

(a) Choose $(f_n, A_n, \pi_n) \leftarrow (\mathcal{F}_n, \mathcal{A}_n, \Pi_n)$ and $s \leftarrow_r \{0,1\}^n$.

(b) Let $\alpha = \pi_n(0^{\ell_2(n)-n}||s)$.

(c) Define function $A_n^1 : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1,\perp\}$ as follows:

- If $s \notin \mathit{GoodQueries}(n)$: for all $\gamma \in \{0,1\}^{\ell_2(n)+(\ell_1(n)-n)+1}$, $A_n^1(\gamma) = \perp$.
- If $s \in \mathit{GoodQueries}(n)$: for all $\gamma \in \{0,1\}^{\ell_2(n)+(\ell_1(n)-n)+1}$, if $G^{f_n}(Q_{G,n}^{-1}(s)) = \gamma$ then $A_n^1(\gamma) = 1$, and otherwise $A_n^1(\gamma) = 0$.

(d) Define function $A_n^2 : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1} \to \{0,1\}$ as follows:

- If $s \notin \mathit{GoodQueries}(n)$: for all γ, $A_n^2(\gamma) = A_n(\gamma)$.
- If $s \in \mathit{GoodQueries}(n)$: for all γ, $A_n^2(\gamma) = 1$ if and only if there exists $x \in \mathit{Good}(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $y||x \neq 0^{\ell_2(n)-n}||Q_{G,n}^{-1}(s)$ and

$$g(x, \pi_n(y||Q_G(x,\langle 0\rangle)), \ldots, \pi_n(y||Q_G(x,\langle k-1\rangle))) = \gamma.$$

(e) Let $\sigma = |\mathit{Siblings}_n(s)|$. Let $s_1, \ldots, s_\sigma$ denote the strings in $\mathit{Siblings}_n(s)$ in lexicographical order. For $1 \le i \le \sigma$, let $z_i = \pi_n(0^{\ell_2(n)-n}||s_i)$. Let $\beta = (z_1, \ldots z_\sigma)$.

(f) Run $D^{(f_n, A_n^1, A_n^2)}$ on input (α, β).

Observe that for all $\gamma \in \{0,1\}^{\ell(n)+1}$, we have $A_n(\gamma) = 1$ if either $A_n^1(\gamma) = 1$ or $A_n^2(\gamma) = 1$, and otherwise we have $A_n(\gamma) = 0$. This means that D can compute A_n using the oracles it is given for A_n^1 and A_n^2.

Define $q'_D(n)$ to be the probability that D makes an oracle query from $\mathit{Siblings}_n(s)$ to f_n.

We need to show that for every computationally unbounded probabilistic oracle machine $D^{(\cdot,\cdot,\cdot)}$ that makes at most polynomially-many oracle queries, $q'_D(n)$ is negligible.

We begin by considering probabilistic oracle machines $E^{(\cdot,\cdot,\cdot)}$ that make no queries to A_n^2 and that query A_n^1 and f_n in a particular structured manner. We consider a conditioned version of Experiment $1'$; the relevance of this conditioning will become clear when we consider unrestricted probabilistic oracle machines $D^{(\cdot,\cdot,\cdot)}$ in Claim 4.3.23.

Claim 4.3.22 *Let $E^{(\cdot,\cdot,\cdot)}$ be a probabilistic oracle machine. Let function $m(n)$ be a bound on the number of queries made by $E^{(\cdot,\cdot,\cdot)}$ when its first input is of length $\ell_2(n)$. Suppose E makes no queries to its third oracle, and uses its first and second oracles in the following restricted "two-phase" manner: initially, E makes queries only to its second oracle; if, at some point, E receives response 1 to an oracle query, then E makes no further queries to its second oracle, and makes queries only to its first oracle. Suppose E is run according to Experiment $1'$. Then, for every $w \in \mathit{Good}(n)$, we have that conditioned on $s \in \mathit{GoodQueries}(n)$ and $w \neq Q_{G,n}^{-1}(s)$, the probability that $E^{(f_n, A_n^1, A_n^2)}$ makes a query from $\mathit{Siblings}_n(s)$ to f_n is at most $\frac{(m(n))^2 - m(n)}{2|\mathit{Good}(n)|-2}$.*

Proof *(Claim 4.3.22)* We proceed in a manner similar to the proof of Claim 4.3.6. Fix $w \in \mathit{Good}(n)$, and consider running $E^{(\cdot,\cdot,\cdot)}$ as in Experiment $1'$, conditioning on the case that $s \in$

$GoodQueries(n)$ and $w \neq Q_{G,n}^{-1}(s)$. Let $\mathcal{V}$ denote the set of $v \in Good(n) - \{w\}$ such that $\hat{g}_n(v, z_1, \ldots, z_\sigma) = \hat{g}_n(Q_{G,n}^{-1}(s), z_1, \ldots, z_\sigma)$. That is, $\mathcal{V}$ is the set of $v \in Good(n) - \{w\}$ such that $A_n^1(\hat{g}_n(v, z_1, \ldots, z_\sigma)) = 1$. We will abuse notation by using m to denote $m(n)$; that is, E makes at most m oracle queries.

Bu assumption, E's querying behaviour can be viewed as consisting of two phases. In the first phase E makes queries only to A_n^1. If some query to A_n^1 receives response 1, then E immediately enters a second phase where it makes queries only to f_n.

Note that before E begins making queries, we have that given the view of E, every $v \in Good(n) - \{w\}$ is equally likely to be the value of $Q_{G,n}^{-1}(s)$. Each query γ to A_n^1 whose response is 0 rules out (as potential values of $Q_{G,n}^{-1}(s)$) all v such that $\hat{g}_n(v, z_1, \ldots, z_\sigma) = \gamma$. But note that after such a query, the "un-ruled-out" values v (that is, all $v \in Good(n) - \{w\}$ such that query $\hat{g}_n(v, z_1, \ldots, z_\sigma)$ has *not* yet been made to A_n^1) are all equally likely to be the value of $Q_{G,n}^{-1}(s)$ given the view of E. Similarly, note that a query γ to A_n^1 whose response is 1 rules out (as potential values of $Q_{G,n}^{-1}(s)$) all v such that $\hat{g}_n(v, z_1, \ldots, z_\sigma) \neq \gamma$; immediately following such a query γ, all $v' \in Good(n) - \{w\}$ such that $\hat{g}_n(v', z_1, \ldots, z_\sigma) = \gamma$ (that is, all $v' \in \mathcal{V}$) are equally likely, given the view of E to be the value of $Q_{G,n}^{-1}(s)$. Once E begins querying f_n (and has so far not made a query from $Siblings_n(s)$ to f_n), each query x whose response is different from all the z_i rules out $Q_{G,n}^{-1}(x)$ as a potential value of $Q_{G,n}^{-1}(s)$; after such a query, all the $v \in \mathcal{V}$ such that no string in $\{Q_G(v, \langle i \rangle) : 0 \leq i \leq k-1 \text{ and } i \notin Small\}$ has yet been queried to f_n are equally likely, given the view of E, to be the value of $Q_{G,n}^{-1}(s)$.

For $1 \leq N \leq |Good(n)| - 1$ and $u \geq 0$, define $q_E^{(N,u)}$ to be the probability that if E has not yet made a query to A_n^1 whose response is 1, there are N "un-ruled-out" values $v \in Good(n) - \{w\}$, and E is allowed to make at most w additional oracle queries, then E makes a query from $Siblings_n(s)$ to f_n. The value we are ultimately interested in upper bounding is $q_E^{(|Good(n)|-1,m)}$.

We will prove by strong induction on u that for all $1 \leq N \leq |Good(n)| - 1$ and $u \geq 0$, we have $q_E^{(N,u)} \leq u(u-1)/(2N)$.

It is clear that $q_E^{(N,0)} = 0$ for all $1 \leq N \leq |Good(n)| - 1$. We also have that $q_E^{(N,1)} = 0$ for all $1 \leq N \leq |Good(n)| - 1$, since E must make a query to A_n^1 whose response is 1 before making queries to f_n, and hence if E is allowed only a single query then it cannot query f_n.

Now consider $q_E^{(N,u)}$ for $u \geq 2$ and $1 \leq N \leq |Good(n)| - 1$. Let γ denote the next query to A_n^1 that will be made by E. Let V' be set of strings $v \in Good(n) - \{w\}$ such that $\hat{g}_n(v, z_1, \ldots, z_\sigma) = \gamma$. If $|V'| = 0$ or if E has previously made query γ, then $A_n^1(\gamma) = 0$ but no additional $v \in Good(n) - \{w\}$ will be ruled out by this query; in this case, E has simply "wasted a query", and the probability E makes a query from $Siblings_n(s)$ to f_n is $q_E^{(N,u-1)}$, which by induction is at most $(u-1)(u-2)/(2N) < u(u-1)/(2N)$. So suppose $|V'| > 0$ and E has not previously made query γ. Observe that given the view of E before making query γ, the probability that

$Q^{-1}_{G,n}(s) \in V'$ is $|V'|/N$. Now, if $Q^{-1}_{G,n}(s) \in V'$, then $A^1_n(\gamma) = 1$, and hence E will stop querying A^1_n and henceforth only query f_n. In this case, the probability that E makes a query from $Siblings_n(s)$ to f_n is at most $(u-1)/|V'|$: we say a string $v \in V'$ is *covered* if E makes a query from $\{Q_G(v, \langle i \rangle) : 0 \le i \le k-1 \text{ and } i \notin Small\}$ to f_n; then, making a query from $Siblings_n(s)$ is equivalent to covering $Q^{-1}_{G,n}(s)$; but each query made by E to f_n covers at most one string $v \in V'$, and as long as $Q^{-1}_{G,n}(s)$ has not yet been covered, each of the uncovered strings in V' are equally likely (given the view of E) to be $Q^{-1}_{G,n}(s)$. On the other hand, if $Q^{-1}_{G,n}(s) \notin V'$, then $A^1_n(\gamma) = 0$, and hence E will continue making queries to A^1_n. However, since $A^1_n = 0$, all $v \in V'$ are ruled out as potential values of $Q^{-1}_{G,n}(s)$. It follows that, in this case, the probability that E makes a query from $Siblings_n(s)$ to f_n is at most $q_E^{(N-|V'|,u-1)}$; by induction we have $q_E^{(N-|V'|,u-1)} \le (u-1)(u-2)/(2N-2|V'|)$. We then have that

$$\begin{aligned} q_E^{(N,u)} &\le \frac{|V'|}{N} \cdot \frac{u-1}{|V'|} + \frac{N-|V'|}{N} \cdot \frac{(u-1)(u-2)}{2(N-|V'|)} \\ &= \frac{u-1}{N} + \frac{(u-1)(u-2)}{2N} \\ &= \frac{2(u-1)+(u-1)(u-2)}{2N} \\ &= \frac{u(u-1)}{2N}, \end{aligned}$$

as required.

We then have that $q_E^{(|Good(n)|-1,m)} \le m(m-1)/(2|Good(n)|-2) = (m^2-m)/(2|Good(n)|-2)$. □

To complete the proof of Lemma 4.3.20, we consider probabilistic oracle machines $D^{(\cdot,\cdot,\cdot)}$ that have no restrictions in the manner in which they query their oracles. It suffices to prove the following claim (recall that by Claim 4.3.18, we have $|Fixed(n)|/2^n < 1/n^d$ for all d and sufficiently large n).

Claim 4.3.23 *Let $D^{(\cdot,\cdot,\cdot)}$ be a probabilistic oracle machine. Let function $m(n)$ be a bound on the number of oracle queries made by $D^{(\cdot,\cdot,\cdot)}$ when its first input is of length $\ell_2(n)$. Then, for all $n \in \mathcal{N}$, $q'_D(n) \le \frac{k(m(n))^2+(7k)m(n)+4k}{2^{n+1}} + \frac{2m(n)}{2^n} + \frac{|Fixed(n)|}{2^n}$.*

Proof *(Claim 4.3.23)* We proceed in a manner based on ideas from the proof of Claim 4.3.7. We define a probabilistic machine $E^{(\cdot,\cdot,\cdot)}$ that, when run according to Experiment $1'$, simulates D "almost" according to Experiment $1'$, but uses its own oracles in the restricted two-phase manner described in Claim 4.3.22. We will argue that it is unlikely that D will make a query that "exposes" the fact that it is being simulated only "almost" according to Experiment $1'$ rather than perfectly according to Experiment 1. Then, to bound $q'_D(n)$, we will show that whenever the simulation of D makes a query from $\mathit{Siblings}_n(s)$ so does E, and we will use Claim 4.3.22 to bound the probability that E makes such a query.

A key difference from the proof of Claim 4.3.7 is that in our definition of E, we need to do cases based on whether $s \in \mathit{GoodQueries}(n)$. E will use its oracle A^1_n to determine if $s \in \mathit{GoodQueries}(n)$.

On input (α, β) where $\alpha \in \{0,1\}^{\ell_2(n)}$ and with access to oracles $f_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$, $A^1_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$, and $A^2_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$, E behaves as follows. E first checks if $\alpha \in \{0^{\ell_2(n)-n}\} \times \mathit{Fixed}(n)$; if so E simply halts. Otherwise, E makes query $0^{\ell_1(n)+(\ell_2(n)-n)+1}$ to A^1_n.

If the response to this first query is $\perp$, then E randomly selects $s' \in (\mathit{NotFixed}(n) - \mathit{GoodQueries}(n))$, and randomly selects a permutation $\hat{\rho}_n : \{0,1\}^{\ell_2(n)-n} \times \mathit{NotFixed}(n) \to \{0,1\}^{\ell_2(n)-n} \times \mathit{NotFixed}(n)$ such that $\hat{\rho}_n(0^{\ell_2(n)-n}, s') = \alpha$. Then, E defines function $\hat{\pi} : \{0,1\}^{\ell_2(n)} \to \{0,1\}^{\ell_2(n)}$ as follows: for all $y \in \{0,1\}^{\ell_2(n)-n}$ and all $x \in \{0,1\}^n$, if $x \in \mathit{NotFixed}(n)$ then $\hat{\pi}_n(y||x) = \hat{\rho}_n(y,x)$, and otherwise $\hat{\pi}_n(y||x) = 0^{\ell_2(n)-n}||x$. E simulates $D^{(\cdot,\cdot,\cdot)}$ on input (α,β) and with access to oracles $\hat{f}_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$, $\hat{A}^1_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$, and $\hat{A}^2_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ defined as follows. Whenever D makes query x to its oracle $\hat{f}_n$, E provides response $\hat{\pi}_n(0^{\ell_2(n)-n}||x)$ to D, and makes query x to f_n. Whenever D makes a query γ its oracle $\hat{A}^1_n$, E provides response $\perp$. Whenever D makes a query γ to its oracle $\hat{A}^2_n$, E checks if there exists $x \in \mathit{Good}(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $g(x, \hat{\pi}_n(y||Q_G(x,\langle 0\rangle)), \ldots, \hat{\pi}_n(y||Q_G(x,\langle k-1\rangle))) = \gamma$; if so, E gives 1 to D as the response to query γ, and otherwise E gives 0 to D as the response to query γ.

If the response to the first query made by E is not $\perp$, then E proceeds as follows. Say $\beta = z_1, \ldots z_\sigma$, where $\sigma = k - |\mathit{Small}|$. E randomly selects $w \in \mathit{Good}(n)$, and lets $s'_1, \ldots s'_\sigma$ denote the members of $\{Q_G(w,\langle i\rangle) : i \notin \mathit{Small}\}$ in lexicographical order. Then, E randomly selects a permutation $\hat{\rho}_n : \{0,1\}^{\ell_2(n)-n} \times \mathit{NotFixed}(n) \to \{0,1\}^{\ell_2(n)-n} \times \mathit{NotFixed}(n)$ such that $\hat{\rho}_n(0^{\ell_2(n)-n}, s'_i) = z_i$ for $1 \le i \le \sigma$. E defines function $\hat{\pi} : \{0,1\}^{\ell_2(n)} \to \{0,1\}^{\ell_2(n)}$ as follows: for all $y \in \{0,1\}^{\ell_2(n)-n}$ and all $x \in \{0,1\}^n$, if $x \in \mathit{NotFixed}(n)$ then $\hat{\pi}_n(y||x) = \hat{\rho}_n(y,x)$, and otherwise $\hat{\pi}_n(y||x) = 0^{\ell_2(n)-n}||x$. E simulates $D^{(\cdot,\cdot,\cdot)}$ on input (α,β) and with access to oracles $\hat{f}_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$, $\hat{A}^1_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$, and $\hat{A}^2_n : \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ whose behaviour we now define.

Whenever D makes a query x to its oracle $\hat{f}_n$, E provides response $\hat{\pi}_n(0^{\ell_2(n)-n}||x)$ to D. If $x \notin \mathit{GoodQueries}(n)$, then E takes no further action for query x. Otherwise, E checks if $A^1_n(\hat{g}_n(Q^{-1}_{G,n}(x), z_1, \ldots, z_\sigma)) = 1$: if E has previously made query $\gamma = \hat{g}_n(Q^{-1}_{G,n}(x), z_1, \ldots, z_\sigma)$ to A^1_n, it checks the response it received to that query; if E has previously received response 1 to a query to A_n^1 different from γ, then it "knows" that $A_n^1(\gamma) \neq 1$ without needing to make any query; otherwise, E makes query γ to A_n^1. If $A_n^1(\gamma) = 1$, E makes query x to its oracle f_n.

Whenever D makes a query γ to its oracle $\hat{A}_n^1$, E first checks if it has previously made query γ to (its own oracle) A_n^1. If so, E gives the (previously obtained) value $A_n^1(\gamma)$ to D as the response to query γ. If not, E checks if any previous query (of its own) to A_n^1 had response 1; if so, E gives 0 to D as the response to query γ. Otherwise, E makes query γ to A_n^1, and then gives $A_n^1(\gamma)$ to D as the response to query γ.

Whenever D makes a query γ to its oracle $\hat{A}_n^2$, E checks if there exists $x \in Good(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $\alpha \notin \{\hat{\pi}_n(y||Q_G(x,\langle i\rangle)) : 0 \leq i \leq k-1\}$ and

$$g(x, \hat{\pi}_n(y||Q_G(x,\langle 0\rangle)), \ldots, \hat{\pi}_n(y||Q_G(x,\langle k-1\rangle))) = \gamma;$$

if so, E gives 1 to D as the response to query γ, and otherwise E gives 0 to D as the response to query γ. If there is a *unique* string $x \in Good(n)$ such that $\alpha \notin \{\hat{\pi}_n(y||Q_G(x,\langle i\rangle)) : 0 \leq i \leq k-1\}$ and

$$g(x, \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(x,\langle 0\rangle)), \ldots, \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(x,\langle k-1\rangle))) = \gamma,$$

E checks if $A_n^1(\hat{g}_n(x, z_1, \ldots, z_\sigma)) = 1$ (using the same approach it uses to evaluate A_n^1 when answering queries to $\hat{f}_n$), and if so, E chooses $j \in \{0, \ldots, k-1\} - Small$ arbitrarily and makes query $Q_G(x, \langle j\rangle)$ to f_n.

Observe that when the response to E's initial oracle query is $\perp$, the total number of oracle queries made by E is at most one more than the number of oracle queries made by simulation of D, that is, at most $m(n)+1$. Also, when the response to E's initial oracle query is not $\perp$, then by exactly the reasoning given in the proof of Claim 4.3.7, the number of oracle queries made by E (not counting the initial query) is at most one more than the number of oracle queries made by the simulation of D; it follows that in this case, the total number of oracle queries made by E is at most $m(n)+2$.

Consider running $E^{(\cdot,\cdot,\cdot)}$ according to Experiment $1'$, and let s and α be as chosen in this experiment. We claim that, as in the proof of Claim 4.3.7, E simulates $D^{(\cdot,\cdot,\cdot)}$ "almost according" to Experiment $1'$ with the same choice of α and s. Specifically, suppose E defined function π'_n to be the same as $\hat{\pi}_n$ except for the following changes: in the case that the response to the initial query is $\perp$, the values of $\hat{\pi}_n$ at points $0^{\ell_2(n)-n}||s$ and $0^{\ell_2(n)-n}||s'$ are interchanged (so $\pi'_n(0^{\ell_2(n)-n}||s) = \hat{\pi}_n(0^{\ell_2(n)-n}||s') = \alpha$ and $\pi'_n(0^{\ell_2(n)-n}||s') = \hat{\pi}_n(0^{\ell_2(n)-n}||s)$); in the case that the response to the initial query is not $\perp$, the values of $\hat{\pi}_n$ at points $0^{\ell_2(n)-n}||s_i$ and $0^{\ell_2(n)-n}||s'_i$ are interchanged for all $1 \leq i \leq \sigma$ (so $\pi'_n(0^{\ell_2(n)-n}||s_i) = \hat{\pi}_n(0^{\ell_2(n)-n}||s'_i) = z_i$ for all $1 \leq i \leq \sigma$, and $\pi'_n(0^{\ell_2(n)-n}||s'_i) = \hat{\pi}_n(0^{\ell_2(n)-n}||s_i)$ for all $1 \leq i \leq \sigma$). Further, suppose E used π'_n in place of $\hat{\pi}_n$ to answer the oracle queries of D. Then, when E is run according to Experiment $1'$ it would simulate D according to Experiment $1'$ with the same choice of α and s. Of course, E is not given s, so it cannot actually construct π'_n. However, since π'_n and $\hat{\pi}_n$ are identical except at a small number of points, there are only a small number of possible queries made by D that receive a different response from E when E answers using $\hat{\pi}_n$ instead of π'_n.

First consider the case that the first query made by E receives response $\perp$, that is, the case $s \in NotFixed(n) - GoodQueries(n)$.

This case occurs with probability exactly $(2^n - |Fixed(n)| - |GoodQueries(n)|)/2^n$. In this case, we have $Siblings_n(s) = \{s\}$. By definition of E, we also have $s' \notin NotFixed(n) - GoodQueries(n)$. But then it is easy to see by the way that E answers oracle queries in this case that all queries made by D to $\hat{A}^1_n$ and $\hat{A}^2_n$ receive the same response whether E answers using π'_n or $\hat{\pi}_n$. This means that the only queries made by D that receive a different response from E when E answers using $\hat{\pi}_n$ instead of π'_n are queries s and s' to $\hat{f}_n$ (of course, if it happens to be the case that $s = s'$, then even these queries receive the same response whether E answers using π'_n and $\hat{\pi}_n$).

Now, it is easy to see that the probability that D makes query s' when $s \neq s'$ (and hence s' is uniformly distributed over $(NotFixed(n) - GoodQueries(n)) - \{s\}$) is at most $m(n)/(2^n - |Fixed(n)| - |GoodQueries(n)| - 1)$. The probability that $s' \neq s$ is exactly $(2^n - |Fixed(n)| - |GoodQueries(n)| - 1)/(2^n - |Fixed(n)| - |GoodQueries(n)|)$. Then, the probability that both $s' \neq s$ *and* D makes query s' is at most $m(n)/(2^n - |Fixed(n)| - |GoodQueries(n)|)$.

Also, note that whenever D makes query s, so does E. But it is easy to see that the probability that E makes query s is at most $m(n)/(2^n - |Fixed(n)| - |GoodQueries(n)|)$, since the only information that E gets from oracle A^1_n is that $s \in NotFixed(n) - GoodQueries(n)$, E makes no queries to A^2_n, and E makes at most $m(n)$ queries to f_n. This means that the probability that D makes query s is at most $m(n)/(2^n - |Fixed(n)| - |GoodQueries(n)|)$.

Note that E simulates D according to Experiment $1'$ until D makes query s or query $s' \neq s$. This means the probability that D makes query s in Experiment $1'$ is at most the probability that the simulation of D makes query s or query $s' \neq s$. That is, conditioned on $s \in NotFixed(n) - GoodQueries(n)$, the probability that D makes query s in Experiment $1'$ is at most $2m(n)/(2^n - |Fixed(n)| - |GoodQueries(n)|)$.

Now consider the case that the first query made by E receives a response different from $\perp$, that is, the case $s \in GoodQueries(n)$.

This case occurs with probability exactly $|GoodQueries(n)|/2^n$. In this case, there are at most $2+2\sigma$ possible queries made by D that receive a different response from E when E answers using $\hat{\pi}_n$ instead of π'_n (where we assume, for now, that $w = Q^{-1}_{G,n}(s'_1) \neq Q^{-1}_{G,n}(s_1)$ and hence $s_i \neq s'_i$ for all i):

1. **Query s_i to $\hat{f}_n$, for $1 \leq i \leq \sigma$.** When E uses $\hat{\pi}_n$, it gives answer $\hat{\pi}_n(0^{\ell_2(n)-n}||s_i)$ instead of z_i.

2. **Query s'_i to $\hat{f}_n$, for $1 \leq i \leq \sigma$.** When E uses $\hat{\pi}_n$, it gives answer z_i instead of $\hat{\pi}_n(0^{\ell_2(n)-n}||s_i)$.

3. **Query**

$$\begin{aligned}\gamma_s &= \hat{g}_n(Q_{G,n}^{-1}(s), \hat{\pi}_n(0^{\ell_2(n)-n}||s_1), \ldots, \hat{\pi}_n(0^{\ell_2(n)-n}||s_\sigma)) \\ &= \hat{g}_n(Q_{G,n}^{-1}(s), \pi'_n(0^{\ell_2(n)-n}||s'_1), \ldots, \pi'_n(0^{\ell_2(n)-n}||s'_\sigma))\end{aligned}$$

to $\hat{A}_n^2$. When E uses $\hat{\pi}_n$, it gives answer 1. When E uses π'_n, it gives answer 1 if and only if there exists $x \in \mathit{Good}(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $\alpha \notin \{\pi'_n(y||Q_G(x, \langle i \rangle) : 0 \leq i \leq k-1\}$ (that is, $y||x \neq 0^{\ell_2(n)-n}||Q_{G,n}^{-1}(s)$) and $g(x, \pi'_n(y||Q_G(x,\langle 0\rangle)), \ldots, \pi'_n(y||Q_G(x, \langle k-1\rangle))) = \gamma_s$.

4. **Query**

$$\begin{aligned}\delta_w &= \hat{g}_n(w, \hat{\pi}_n(0^{\ell_2(n)-n}||s_1), \ldots, \hat{\pi}_n(0^{\ell_2(n)-n}||s_\sigma)) \\ &= \hat{g}_n(w, \pi'_n(0^{\ell_2(n)-n}||s'_1), \ldots, \pi'_n(0^{\ell_2(n)-n}||s'_\sigma))\end{aligned}$$

to $\hat{A}_n^2$. When E uses $\hat{\pi}_n$, it gives answer 1 if and only if there exists $x \in \mathit{Good}(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $\alpha \notin \{\hat{\pi}_n(y||Q_G(x, \langle i \rangle) : 0 \leq i \leq k-1\}$ (that is, $y||x \neq (0^{\ell_2(n)-n}||w)$) and $g(x, \hat{\pi}_n(y||Q_G(x,\langle 0\rangle)), \ldots, \hat{\pi}_n(y||Q_G(x, \langle k-1\rangle))) = \delta_w$. When E uses π'_n, it gives answer 1.

Staying consistent with the terminology from the proof of Claim 4.3.7, we will say that $\gamma_s = \hat{g}_n(Q_{G,n}^{-1}(s), \hat{\pi}_n(0^{\ell_2(n)-n}||s_1), \ldots, \hat{\pi}_n(0^{\ell_2(n)-n}||s_\sigma))$ is *bad* if there does *not* exist $x \in \mathit{Good}(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $y||x \neq 0^{\ell_2(n)-n}||Q_{G,n}^{-1}(s)$ and

$$g(x, \pi'_n(y||Q_G(x,\langle 0\rangle)), \ldots, \pi'_n(y||Q_G(x, \langle k-1\rangle))) = \gamma_s.$$

We will say that $\delta_w = \hat{g}_n(w, \pi'_n(0^{\ell_2(n)-n}||s'_1), \ldots, \pi'_n(0^{\ell_2(n)-n}||s'_\sigma))$ is *bad* if there does *not* exist $x \in \mathit{Good}(n)$ and $y \in \{0,1\}^{\ell_2(n)-n}$ such that $y||x \neq 0^{\ell_2(n)-n}||w$ and

$$g(x, \hat{\pi}_n(y||Q_G(x,\langle 0\rangle)), \ldots, \hat{\pi}_n(y||Q_G(x, \langle k-1\rangle))) = \delta_w.$$

We will say that D makes a *bad* query if it makes one of the following queries: s_i to $\hat{f}_n$ for $1 \leq i \leq \sigma$, s'_i to $\hat{f}_n$ for $1 \leq i \leq \sigma$, bad γ_s to $\hat{A}_n^2$, or bad δ_w to $\hat{A}_n^2$.

Observe that E simulates D according to Experiment $1'$ until D makes a bad query. This means that the probability that D makes a query from $\mathit{Siblings}_n(s)$ in Experiment $1'$ must be at most the probability that the simulation of D makes a query from $\mathit{Siblings}_n(s)$ or a bad query, which is simply the probability that the simulation of D makes a bad query. We now bound the probability that the simulation of D makes a bad query.

We first claim that whenever the simulation of D makes a query from $Siblings_n(s)$ to $\hat{f}_n$ or queries bad γ_s to $\hat{A}_n^2$, E makes a query from $Siblings_n(s)$ to f_n. To see this, first suppose that the simulation of D makes a query $x \in Siblings_n(s)$ to $\hat{f}_n$. Then, since $A_n^1(\hat{g}_n(Q_{G,n}^{-1}(x), z_1, \ldots, z_\sigma)) = 1$ for such x, E will make query x to f_n. Now suppose the simulation of D queries bad γ_s to $\hat{A}_n^2$. Since γ_s is bad, there is no $x \in Good(n)$ such that $x \neq Q_{G,n}^{-1}(s)$ and

$$g(x, \pi'_n(0^{\ell_2(n)-n}||Q_G(x, \langle 0\rangle)), \ldots, \pi'_n(0^{\ell_2(n)-n}||Q_G(x, \langle k-1\rangle))) = \gamma_s.$$

Then, since π'_n and $\hat{\pi}_n$ agree everywhere on the set

$$\{0^{\ell_2(n)-n}||Q_G(v, \langle i\rangle) : v \neq Q_{G,n}^{-1}(s), v \neq Q_{G,n}^{-1}(s'_1), 0 \leq i \leq k-1\},$$

there is no $x \in Good(n)$ such that $x \neq Q_{G,n}^{-1}(s)$, $x \neq Q_{G,n}^{-1}(s'_1)$ and

$$g(x, \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(x, \langle 0\rangle)), \ldots, \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(x, \langle k-1\rangle))) = \gamma_s.$$

Also, recall that since $s \in Siblings_n(s)$, we have $\alpha = z_i$ for some i, and hence for some i we have $\alpha = \hat{\pi}_n(0^{\ell_2(n)-n}||s'_i) = \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(Q_{G,n}^{-1}(s'_1), \langle i\rangle))$. This means that there is a unique string $x \in Good(n)$ – in particular, $x = Q_{G,n}^{-1}(s)$ – such that $\alpha \notin \{\hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(x, \langle i\rangle)) : 0 \leq i \leq k-1\}$ and

$$g(x, \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(x, \langle 0\rangle)), \ldots, \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(x, \langle k-1\rangle))) = \gamma_s.$$

Then, since $A_n^1(\hat{g}_n(Q_{G,n}^{-1}(s), z_1, \ldots, z_\sigma)) = 1$, we have by definition of E's behaviour when responding to queries to $\hat{A}_n^2$ that E will make query $Q_G(Q_{G,n}^{-1}(s), \langle j\rangle)$ to f_n for some $j \in \{0, \ldots, k-1\} - Small$; for such j, we have $Q_G(Q_{G,n}^{-1}(s), \langle j\rangle) \in Siblings_n(s)$. It follows that (still conditioning on $s \in GoodQueries(n)$ and $Q_{G,n}^{-1}(s'_1) \neq Q_{G,n}^{-1}(s)$), the probability that the simulation of D makes a query from $Siblings_n(s)$ to $\hat{f}_n$ or queries bad γ_s to $\hat{A}_n^2$ is at most the probability that E makes a query from $Siblings_n(s)$ to f_n.

Now, what is the probability that E makes a query from $Siblings_n(s)$ to f_n (again, still conditioning on $s \in GoodQueries(n)$ and $Q_{G,n}^{-1}(s'_1) \neq Q_{G,n}^{-1}(s)$)? Observe that E makes no queries to its oracle A_n^2, and uses its oracles A_n^1 and f_n in the two-phase manner required in the statement of Claim 4.3.22. Then, since E makes at most $m(n) + 2$ oracle queries, we have by Claim 4.3.22 that the probability that E makes a query from $Siblings_n(s)$, conditioned on $s \in GoodQueries(n)$ and $Q_{G,n}^{-1}(s'_1) \neq Q_{G,n}^{-1}(s)$, is at most $\frac{(m(n))^2+3m(n)+2}{2|Good(n)|-2}$.

We next upper-bound the probability that the simulation of D makes at least one bad query *and* that the first such bad query is either s'_i to $\hat{f}_n$ for $1 \leq i \leq \sigma$ or bad δ_w to $\hat{A}_n^2$. When upper-bounding the probability of this event, we will assume that E answers oracle queries using π'_n rather than $\hat{\pi}_n$, since this only changes the answers to bad queries, and hence will not change the probability that at least one bad query is made by the simulation of D, nor will it change the *first* bad query made by D. Under this assumption, observe that the response to each oracle query made by the simulation of D is completely determined by π'_n and α (recall that s itself is completely determined by α and π'_n since $\pi'_n(0^{\ell_2(n)-n}||s) = \alpha$; similarly, $s_1, \ldots, s_\sigma$ and $z_1, \ldots, z_\sigma$ are completely determined by π'_n and s, and hence by π'_n and α).

Note that given π'_n, α, and s, we have that $w = Q_{G,n}^{-1}(s'_1)$ is simply a random string in $\mathit{Good}(n)$ that is different from $Q_{G,n}^{-1}(s)$. That is, given s along with the entire view of the simulation of D, we have that w is a random string in $\mathit{Good}(n)$ that is different from $Q_{G,n}^{-1}(s)$. Now, fix π'_n and α (and hence s), and, conditioned on these values, consider the probability that the simulation of D makes at least one bad query and that the first such bad query is either either s'_i to $\hat{f}_n$ for $1 \le i \le \sigma$ or bad δ_w to $\hat{A}^2_n$. Define $\mathit{Bad} \subseteq \mathit{Good}(n) - \{Q_{G,n}^{-1}(s'_1)\}$ to be the set of strings v such that if $w = v$, then $\delta_w = g(w, \pi'_n(0^{\ell_2(n)-n}||Q_G(w, \langle 0\rangle)), \ldots, \pi'_n(0^{\ell_2(n)-n}||Q_G(w, \langle k-1\rangle)))$ is bad (note that fixing w also fixes $\hat{\pi}_n$, allowing us to determine if δ_w is bad).

We claim that for all distinct $v_1, v_2 \in \mathit{Bad}$, we have $\delta_{v_1} \ne \delta_{v_2}$. Suppose not, that is, suppose $v_1, v_2 \in \mathit{Bad}$ are distinct strings such that $\delta_{v_1} = \delta_{v_2}$. Now, if $w = v_1$, we have that for all $0 \le i \le k-1$, $\hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(v_2, \langle i\rangle)) = \pi'_n(0^{\ell_2(n)-n}||Q_G(v_2, \langle i\rangle))$, since v_2 is neither w nor $Q_{G,n}^{-1}(s)$, and hence we must have that $g(v_2, \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(v_2, \langle 0\rangle)), \ldots, \hat{\pi}_n(0^{\ell_2(n)-n}||Q_G(v_2, \langle k-1\rangle))) = g(v_2, \pi'_n(0^{\ell_2(n)-n}||Q_G(v_2, \langle 0\rangle)), \ldots, \pi'_n(0^{\ell_2(n)-n}||Q_G(v_2, \langle k-1\rangle))) = \delta_{v_2} = \delta_{v_1} = \delta_w$. But this means that δ_w is not bad, contradicting $v_1 \in \mathit{Bad}$. Now, defining $\mathit{Bad}^\delta = \{\delta_v : v \in \mathit{Bad}\}$, we have $|\mathit{Bad}^\delta| = |\mathit{Bad}|$.

First condition on the case $w \notin \mathit{Bad}$. Then, w is uniformly distributed over $(\mathit{Good}(n) - \mathit{Bad}) - \{Q_{G,n}^{-1}(s)\}$. We will say that $v \in (\mathit{Good}(n) - \mathit{Bad}) - \{Q_{G,n}^{-1}(s)\}$ is *covered* by the simulation of D if D makes a query in $\{Q_G(v, \langle i\rangle) : 0 \le i \le k-1 \text{ and } i \notin \mathit{Small}\}$ to $\hat{f}_n$. Observe that each query made by D can cover at most one $v \in (\mathit{Good}(n) - \mathit{Bad}) - \{Q_{G,n}^{-1}(s)\}$. It follows that the probability that w is covered is at most $m(n)/(|\mathit{Good}(n)| - |\mathit{Bad}| - 1)$.

Now condition on the case $w \in \mathit{Bad}$. Then, w is uniformly distributed over Bad, and δ_w is uniformly distributed over Bad^δ. We will say that $v \in \mathit{Bad}$ is *covered* by the simulation of D if D makes a query in $\{Q_G(v, \langle i\rangle) : 0 \le i \le k-1 \text{ and } i \notin \mathit{Small}\}$ to $\hat{f}_n$ or D queries δ_v to $\hat{A}^2_n$. Observe that each query by D can cover at most one $v \in \mathit{Bad}$. It follows that the probability that w is covered is at most $m(n)/|\mathit{Bad}|$.

Removing conditioning on w, π'_n, and α, we have that the probability that the simulation of D makes at least one bad query and that the first such query is either s'_i to $\hat{f}_n$ for $1 \le i \le \sigma$ or bad δ_w to $\hat{A}^2_n$ is at most

$$\frac{|\mathit{Good}(n)| - |\mathit{Bad}| - 1}{|\mathit{Good}(n)| - 1} \cdot \frac{m(n)}{|\mathit{Good}(n)| - |\mathit{Bad}| - 1} + \frac{|\mathit{Bad}|}{|\mathit{Good}(n)| - 1} \cdot \frac{m(n)}{|\mathit{Bad}|} = \frac{2m(n)}{|\mathit{Good}(n)| - 1}.$$

Putting everything together, and recalling that we analyzed the case $s \in \mathit{GoodQueries}(n)$ under the assumption that $Q_{G,n}^{-1}(s'_1) \ne Q_{G,n}^{-1}(s)$ (an event that fails to hold with probability

$1/|Good(n)|$), we have that conditioned on $s \in GoodQueries(n)$, the probability that D makes a query from $Siblings_n(s)$ in Experiment $1'$ is at most

$$\frac{|Good(n)|-1}{|Good(n)|}\left(\frac{(m(n))^2+3m(n)+2}{2|Good(n)|-2}+\frac{2m(n)}{|Good(n)|-1}\right)+\frac{1}{|Good(n)|},$$

which is at most $\frac{(m(n))^2+7m(n)+4}{2|Good(n)|}$.

Now, recalling that conditioned on $s \in NotFixed(n) - GoodQueries(n)$ we have that the probability that D makes a query from $Siblings_n(s)$ in Experiment $1'$ is at most

$$\frac{2m(n)}{2^n - |Fixed(n)| - |GoodQueries(n)|},$$

and noting that conditioned on $s \in Fixed(n)$ we have no bound (other than the trivial bound of 1) on the probability that D makes a query from $Siblings_n(s)$ in Experiment $1'$, we have that $q'_D(n)$ is at most:

$$\begin{aligned}
& \frac{|GoodQueries(n)|}{2^n}\cdot\frac{(m(n))^2+7m(n)+4}{2|Good(n)|} \\
+ \quad & \frac{2^n-|Fixed(n)|-|GoodQueries(n)|}{2^n}\cdot\frac{2m(n)}{2^n-|Fixed(n)|-|GoodQueries(n)|} \\
+ \quad & \frac{|Fixed(n)|}{2^n} \\
= \quad & \frac{|GoodQueries(n)|}{2^n}\cdot\frac{(m(n))^2+7m(n)+4}{2|Good(n)|}+\frac{2m(n)}{2^n}+\frac{|Fixed(n)|}{2^n} \\
\leq \quad & \frac{k(m(n))^2+(7k)m(n)+4k}{2^{n+1}}+\frac{2m(n)}{2^n}+\frac{|Fixed(n)|}{2^n}
\end{aligned}$$

where the final inequality uses the fact that $|GoodQueries(n)| \leq k|Good(n)|$.

This completes the proof of Claim 4.3.23. □

This completes the proof of Lemma 4.3.20.

□

This completes the proof of Theorem 4.3.1.

□

4.4 Constructions with long seeds

In Section 4.3, we saw that black-box constructions $G^{(\cdot)}$ making constantly-many non-adaptive oracle queries, where the seed length of $G^{(\cdot)}$ is not too much longer than the length of each oracle query, cannot achieve even a single bit more stretch than their oracle. In this section, we consider constructions whose seed length is allowed to be much longer than the length of each oracle query, but where the oracle queries are collectively chosen in a manner that depends only on a portion of the seed whose length is not more than $O(\log n)$ bits longer than the length n of each oracle query. Recall that such constructions making even a single query to a given pseudo-random generator can achieve stretch that is $O(\log n)$ bits longer than the stretch of the given generator [GL89]. Further, recall that such constructions making k adaptive queries can achieve stretch that is $O(\log n)$ bits longer than k times the stretch of the given generator. We show that such constructions making constantly-many non-adaptive queries cannot achieve stretch that is $\omega(\log n)$ bits longer than the stretch of the given generator.

Theorem 4.4.1 *Let $k \in \mathbb{N}$, $c \in \mathbb{R}^+$, and $m(n) \in \omega(\log n)$. Let $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that $\ell_1(n) \le n + c\log n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \to \{0,1\}^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}$ be a non-adaptive oracle construction of a number generator that makes k queries of length n to a number generator mapping n bits to $\ell_2(n)$ bits, such that for all $r \in \{0,1\}^{\ell_0(n)}$ and $x \in \{0,1\}^{\ell_1(n)}$, the queries made by $G^{(\cdot)}$ on input $(r||x)$ depend only on x. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

As is the case for Theorem 4.3.1, the approach we use to prove Theorem 4.4.1 does not seem to extend to the case of polynomially-many (or even $\omega(1)$-many) queries. However, a similar approach does work for polynomially-many queries when we place a restriction on the many-oneness of the number generator's querying function. We state this restriction in Section 4.5.

We give an overview of the proof of Theorem 4.4.1 in Section 4.4.1, and we give the proof details in Section 4.4.2.

4.4.1 Proof overview for Theorem 4.4.1

As in the proof of Theorem 4.3.1, it suffices to define a joint distribution $(\mathcal{F}, \mathcal{A})$ over pairs of functions, such that with probability one over $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, A breaks the pseudo-randomness of G^f but f is pseudo-random even with respect to adversaries that have oracle access to f and A. Unlike the previous proof, we actually define distributions $\mathcal{F}$ and $\mathcal{A}$ that are independent – in fact, we define $\mathcal{A}$ to be a degenerate distribution that assigns all probability to a fixed function A. We define a set $\mathit{Good}(n) \subseteq \{0,1\}^{\ell_1(n)}$ in a careful manner very similar to the proof of Theorem 4.3.1, but taking into account the fact that the queries of $G^{(\cdot)}$ depend only on the rightmost $\ell_1(n)$ bits of its seed. The goal is to ensure that $\mathit{Good}(n)$ is sufficiently large and has the property that for every string $x \in \mathit{Good}(n)$, every $r \in \{0,1\}^{\ell_0(n)}$, and every $f \in \mathcal{F}$, A accepts $G^f(r||x)$. Simultaneously, we need to ensure that the total number of strings accepted by A is sufficiently smaller than $2^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}$ and that $f \leftarrow \mathcal{F}$ is pseudo-random with probability one even with respect to adversaries that have oracle access to f and A.

If we define $\mathcal{F}$ in a very straightforward way (e.g. as the uniform distribution over all 1-1 functions), the total number of strings that A will need to accept (in order to accept $G^f(r||x)$ for every $f \in \mathcal{F}$, every r, and every $x \in \mathit{Good}(n)$) could be too large. The problem is that when deciding whether to accept a given input, A is existentially quantifying over over a set that is (much) larger than the set of its possible inputs. We need to minimize the number of different $f \in \mathcal{F}$ (while, of course, still ensuring that $f \leftarrow \mathcal{F}$ is pseudo-random with probability one even with respect to adversaries that have oracle access to f and A). At the same time, we need to add some structure to the $f \in \mathcal{F}$ to, intuitively, reduce the amount of new information contained in the responses to the oracle queries made by G^f when run on each $r||x$ where $x \in \mathit{Good}(n)$. The idea is that rather than existentially quantifying over every r, every $x \in \mathit{Good}(n)$, and every $f \in \mathcal{F}$ when deciding whether to accept a particular input z, A will instead existentially quantify over every r, every $x \in \mathit{Good}(n)$, and every possible value for the (small amount of) new information (that is, the information not already determined by x) contained in the responses to oracle queries made by $G^{(\cdot)}$ when run on input $r||x$.

Similarly to the proof of Theorem 4.3.1, our procedure for constructing the set $\mathit{Good}(n)$ ensures that for every distinct $x, x' \in \mathit{Good}(n)$, each query q made by G, when run on an input whose rightmost bits are x, is either in some small set $\mathit{Fixed}(n)$ or is distinct from every query q' made by G when run on every input whose rightmost bits are x'. This allows us to follow a two-step approach to defining $\mathcal{F}$. We first define a permutation h on $\{0,1\}^n$ that, for each $x \in \mathit{Good}(n)$, maps the queries $q \notin \mathit{Fixed}(n)$ made by G, when run on an input whose rightmost bits are x, to strings that differ in at most a small number of bits, and, in particular, have a common $(m(n)/2)$-bit suffix. Roughly speaking, sampling $f \leftarrow \mathcal{F}$ proceeds as follows. We randomly select a function $f' : \{0,1\}^n \leftarrow \{0,1\}^{\ell_2(n)}$ that is the identity on its first $n - m(n)/2$ input bits, and is 1-1 on its last $m(n)/2$ input bits, mapping them to $\ell_2(n) - n + m(n)/2$ output bits. We then define $f = f' \circ h$. The actual definition of $\mathcal{F}$ that we use in the proof also ensures that for every $q \in \mathit{Fixed}(n)$, the value $f(q)$ is independent of the choice $f \leftarrow \mathcal{F}$ (that is, $f_1(q) = f_2(q)$ for all $f_1, f_2 \in \mathcal{F}$).

Intuitively, this approach ensures that $f \leftarrow \mathcal{F}$ has "just enough" randomness. At the same time, this approach ensures that for every r and every $x \in \mathit{Good}(n)$, the responses to oracle queries made by $G^f(r||x)$ collectively contain at most $\ell_2(n) - n + m(n)/2$ bits of information that depend on the choice $f \leftarrow \mathcal{F}$.

We remark that it is crucial for this proof that $2^{m(n)/2}$ is super-polynomial. It is for this reason that we cannot adapt the current proof in order to obtain a significantly simpler proof of Theorem 4.3.1; in Theorem 4.3.1, the corresponding value of $m(n)$ (the additional stretch achieved by $G^{(\cdot)}$) is exactly 1.

4.4.2 Proof of Theorem 4.4.1

We will describe distributions $\mathcal{F} = \{\mathcal{F}_n\}$ and an adversary $A = \{A_n\}$ such that when f is chosen according to $\mathcal{F}$, it is pseudo-random with high probability even with respect to adversaries that are given oracle access to f and A, but A breaks G^f for all $f \in \mathcal{F}$.

Let $Q_G : \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log k} \to \{0,1\}^n$ be the $\ell_1(n)$-restricted querying function of $G^{(\cdot)}$. We assume without loss of generality that $G^{(\cdot)}$ always makes k *distinct* queries. We also assume without loss of generality that Q_G encodes the queries of $G^{(\cdot)}$ in lexicographical order: specifically, we assume that for every $x \in \{0,1\}^{\ell_1(n)}$ and every $0 \leq i < j < k$, we have $Q_G(x, \langle i \rangle) < Q_G(x, \langle j \rangle)$. For all $n > 0$, define $Q_{G,n}$ to be Q_G restricted to inputs of length $\ell_1(n) + \log k$; that is, $Q_{G,n}$ is the $\ell_1(n)$-restricted querying function of $G^{(\cdot,\cdot)}$ for security parameter n.

Also, for all $n > 0$, define $g : \{0,1\}^{\ell_0(n)+\ell_1(n)} \times \{0,1\}^{k \cdot \ell_2(n)} \to \{0,1\}^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)m(n)}$ to be a function that represents the computation of $G^{(\cdot)}$ *after* it has made its oracle queries. Specifically, for all $r \in \{0,1\}^{\ell_0(n)}$, $x \in \{0,1\}^{\ell_1(n)}$, and all $O : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$, we have

$$G^O(r||x) = g(r||x, O(Q_G(x, \langle 0 \rangle)), O(Q_G(x, \langle 1 \rangle)), \ldots, O(Q_G(x, \langle k-1 \rangle)))$$

We now give a procedure that iteratively defines a sequence of sets $\mathcal{N}_k \subseteq \mathcal{N}_{k-1} \cdots \subseteq \mathcal{N}_0 \subseteq \mathbb{N}$ and, at the same time, for each $0 \leq i \leq k$ and for each $n \in \mathcal{N}_i$, defines a set $\mathit{Good}_i(n) \subseteq \{0,1\}^{\ell_1(n)}$. The procedure also defines a set $\mathit{Small} \subseteq \{0, \ldots k-1\}$, a sequence of polynomials $p_{i+1}(n)$ for $i \notin \mathit{Small}$. We need the following properties:

(i) For all $0 \leq i \leq k$, $\mathcal{N}_i$ is of infinite size.

(ii) For all $0 \leq i \leq k$, there exists a polynomial $p(n)$ such that for all $n \in \mathcal{N}_i$, $|\mathit{Good}_i(n)| \geq 2^{\ell_1(n)}/p(n)$.

(iii) For all $0 \leq i \leq k$, all $n \in \mathcal{N}_i$, all $x, x' \in \mathit{Good}_i(n)$, all $0 \leq j < i$ and all $0 \leq j' < k$, if $x \neq x'$ and $j, j' \notin \mathit{Small}$, then $Q_G(x, \langle j \rangle) \neq Q_G(x', \langle j' \rangle)$.

Initially, $\mathit{Small} = \emptyset$. We define $\mathcal{N}_0 = \mathbb{N}$ and $\mathit{Good}_0(n) = \{0,1\}^{\ell_1(n)}$ for all $n \in \mathbb{N}$. We then proceed as follows:

For every $1 \leq i \leq k$ do:

1. For each $n \in \mathcal{N}_{i-1}$, define $\mathit{Image}_{i-1}(n) = \{Q_G(x, \langle i-1 \rangle) : x \in \mathit{Good}_{i-1}(n)\}$.
2. If there exists a polynomial $p(n)$ such that $|\mathit{Image}_{i-1}(n)| \geq 2^n/p(n)$ for infinitely many $n \in \mathcal{N}_{i-1}$ then:

2.1 Define $p_i(n)$ to be such a polynomial.

2.2 Define $\mathcal{N}_i$ to be the maximal subset of $\mathcal{N}_{i-1}$ such that $|Image_{i-1}(n)| \geq 2^n/p_i(n)$ for all $n \in \mathcal{N}_i$.

2.3 For each $n \in \mathcal{N}_i$ do:

2.3.1 Define $Image'_{i-1}(n) = Image_{i-1}(n)$.

2.3.2 While $Image'_{i-1}(n) \neq \emptyset$ do:

2.3.2.1 Let $y \in Image'_{i-1}(n)$ be lexicographically first.

2.3.2.2 Let $x \in Good_{i-1}(n)$ be the lexicographically first string such that $Q_G(x, \langle i-1 \rangle) = y$.

2.3.2.3 Add x to $Good_i(n)$.

2.3.2.4 For every $y \in \{Q_G(x, \langle j \rangle) : i-1 \leq j < k\} \cap Image'_{i-1}(n)$, remove y from $Image'_{i-1}(n)$.

3. Else:

3.1 Define $\mathcal{N}_i = \mathcal{N}_{i-1}$.

3.2 Define $Good_i(n) = Good_{i-1}(n)$.

3.3 Add $i-1$ to $Small$.

Now, define $\mathcal{N} = \mathcal{N}_k$, and for all $n \in \mathcal{N}$, define $Good(n)$ as follows: if $|Good_k(n)| < 2^{n-\log k-1}$, then $Good(n) = Good_k(n)$; otherwise, $Good(n)$ is the set of the lexicographically first $2^{n-\log k-1}$ strings in $Good_k(n)$.

We note that the set $Good(n)$ is computable given input $n \in \mathcal{N}$. It is not hard to see that a Turing machine that has set $Small$ and polynomials p_{i+1} for $i \in (\{0, \ldots k-1\} - Small)$ hardcoded can use the ideas from the above procedure to compute $Good(n)$.

It is easy to see that property (i) is satisfied. The fact that property (iii) is satisfied can be shown by induction on i, $0 \leq i \leq k$, where we use the fact that Q_G encodes queries in lexicographical order and the fact that step 2.3.2.1 in the procedure selects elements of $Image'_{i-1}$ in lexicographical order, and hence step 2.3.2.4 prevents any potential violations of property (iii). We now show that property (ii) is satisfied.

Claim 4.4.2 *For all $0 \leq i \leq k$, there exists a polynomial $p(n)$ such that for all $n \in \mathcal{N}_i$, $|Good_i(n)| \geq 2^{\ell_1(n)}/p(n)$.*

Proof We will use induction on i. The base case is trivial since $Good_0(n) = \{0,1\}^{\ell_1(n)}$. So fix $0 \leq i < k$, and suppose there exists a polynomial $p(n)$ such that for all $n \in \mathcal{N}_i$, $|Good_i(n)| \geq 2^{\ell_1(n)}/p(n)$. If $i \in Small$, then $Good_{i+1}(n) = Good_i(n)$ and $\mathcal{N}_{i+1} = \mathcal{N}_i$ so we are done. So suppose $i \notin Small$; then, by definition of $\mathcal{N}_{i+1}$ and p_{i+1} we have $|Image_i(n)| \geq 2^n/p_{i+1}(n)$ for all $n \in \mathcal{N}_{i+1}$. Now, observe that by the procedure used to construct $Good_{i+1}(n)$, we have

$|Good_{i+1}(n)| \geq |Image_i(n)|/k$ for all $n \in \mathcal{N}_{i+1}$. It follows that for all $n \in \mathcal{N}_{i+1}$, we have $|Good_{i+1}(n)| \geq 2^n/(kp_{i+1}(n)) = 2^{n+c\log n}/(kn^c p_{i+1}(n)) \geq 2^{\ell_1(n)}/(kn^c p_{i+1}(n))$.

□

Claim 4.4.3 *There exists a polynomial $p(n)$ such that for all $n \in \mathcal{N}$, $|Good(n)| \geq 2^{\ell_1(n)}/p(n)$.*

Proof Follows immediately from Claim 4.4.2 and from the definitions of $Good(n)$ and $\mathcal{N}$. □

Claim 4.4.4 *For all $n \in \mathcal{N}$, all $x, x' \in Good(n)$, and all $0 \leq j, j' < k$, if $x \neq x'$ and $j, j' \notin Small$, then $Q_G(x, \langle i\rangle) \neq Q_G(x', \langle j\rangle)$.*

Proof Follows immediately from property (iii) of the above procedure and from the definitions of $Good(n)$ and $\mathcal{N}$. □

For all $n \in \mathcal{N}$, define $Fixed(n) = \{Q_G(x, \langle i\rangle) : i \in Small \text{ and } x \in Good(n)\}$. For all $n \notin \mathcal{N}$, define $Fixed(n) = \emptyset$.

Claim 4.4.5 *$|Fixed(n)| < \frac{2^n}{n^d}$ for all d and sufficiently large n.*

Proof By the procedure within which $Small$ is defined, we have $|\{Q_G(x, \langle i\rangle) : x \in Good_i(n)\}| < \frac{2^n}{n^d}$ for all $i \in Small$, all d, and sufficiently large $n \in \mathcal{N}$. Then, since $Good_i(n) \subseteq Good(n)$ for all $0 \leq i \leq k$ and all $n \in \mathcal{N}$, and since $|Small| \leq k$, we have

$$|Fixed(n)| = \left| \bigcup_{i \in Small} \{Q_G(x, \langle i\rangle) : x \in Good(n)\} \right| < \frac{2^n}{n^d}$$

for all d and sufficiently large $n \in \mathcal{N}$. To finish the proof, it suffices to note that for all $n \notin \mathcal{N}$, $|Fixed(n)| = 0$. □

We next define, for all $n \in \mathcal{N}$, a permutation $h_n : \{0,1\}^n \to \{0,1\}^n$, used to rearrange the image of $Q_{G,n}$. Fix $n \in \mathcal{N}$. Then proceed as follows:

1. For $0 \leq j < |Fixed(n)|$ do:
1.1 Let y be the lexicographically j-th string in $Fixed(n)$.
Define $h_n(y) = 0^{n-\log|Fixed(n)|}\langle j\rangle_{\log|Fixed(n)|}$.
2. For $0 \leq j < |Good(n)|$ do:
2.1 Let x be the lexicographically j-th string in $Good(n)$.
2.2 For $0 \leq i < k$ such that $Q_G(x, \langle i\rangle) \notin Fixed(n)$ do:
2.2.1 Define $h_n(Q_G(x, \langle i\rangle)) = 1\langle i\rangle_{\log k}\langle j\rangle_{n-\log k-1}$.
3. For every $y \in \{0,1\}^n$ on which h_n is still undefined, define $h_n(y)$ arbitrarily subject to the restriction that h_n is 1-1.

Using Claim 4.4.4, it is easy to see that for all $n \in \mathcal{N}$, h is a well-defined permutation.

For every $n \notin \mathcal{N}$, define $h_n : \{0,1\}^n \to \{0,1\}^n$ to be the identity function.

For all $n \in \mathcal{N}$ and all $p \geq n$, define $FixedImage^p(n)$ to be the set of p-bit strings whose n-bit prefix is in the set $h_n(\mathit{Fixed}(n))$. For all $n \notin \mathcal{N}$ and all $p \geq n$, define $FixedImage^p(n) = \emptyset$.

We now describe distribution $\mathcal{F}$ and adversary $A = \{A_n\}$.

Distribution $\mathcal{F} = \{\mathcal{F}_n\}$: In order to define distribution $\mathcal{F}$, we first define distributions $\mathcal{F}' = \{\mathcal{F}'_n\}$, $\mathcal{F}'' = \{\mathcal{F}''_n\}$, and $\mathcal{F}''' = \{\mathcal{F}'''_n\}$ as follows. For each $n \geq 0$, define $\mathcal{F}''_n$ to be the uniform distribution over the set of 1-1 functions $f''_n : \{0,1\}^{m(n)/2} \to \{0,1\}^{m(n)/2+(\ell_2(n)-n)}$. For each $n \geq 1$, define $\mathcal{F}'''_n$ to be the uniform distribution over the set of 1-1 functions

$$f'''_n : \left(\{0\}||\{0,1\}^{n-1} - FixedImage^n(n)\right) \to \left(\{0\}||\{0,1\}^{\ell_2(n)-1} - FixedImage^{\ell_2(n)}(n)\right).$$

That is, $\mathcal{F}'''_n$ is the uniform distribution over 1-1 functions that map n-bit strings that start with 0 but are not in $h_n(\mathit{Fixed}(n))$ to $\ell_2(n)$-bit strings that start with 0 but whose n-bit prefix is not in $h_n(\mathit{Fixed}(n))$. For each $n \geq 1$, define $\mathcal{F}'_n$ to be the distribution over the set of 1-1 functions $f'_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$ obtained by first sampling $f''_n \leftarrow \mathcal{F}''_n$ and $f'''_n \leftarrow \mathcal{F}'''_n$, and then defining f'_n as follows: for all $x \in \{0,1\}^{n-1}$ such that $0||x \notin FixedImage^n(n)$, $f'_n(0||x) = f'''_n(0||x)$; for all $x \in \{0,1\}^{n-1}$ such that $0||x \in FixedImage^n(n)$, $f'_n(0||x) = 0||x||0^{\ell_2(n)-n}$; for all $x_1 \in \{0,1\}^{n-m(n)/2-1}$ and $x_2 \in \{0,1\}^{m(n)/2}$, $f'_n(1||x_1||x_2) = 1||x_1||f''_n(x_2)$. Also, for each $n \in \mathbb{N}$, define $\mathcal{F}_n$ to be the distribution over the set of 1-1 functions $f_n : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$ obtained by first sampling $f'_n \leftarrow \mathcal{F}'_n$ and then defining f_n as follows: $f_n = f'_n \circ h_n$.

Adversary $A = \{A_n\}$: For each $n \in \mathcal{N}$, define adversary A_n as follows. On input $y \in \{0,1\}^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}$, A_n accepts if and only if there exists $0 \leq j < |\mathit{Good}(n)|$, $r \in \{0,1\}^{\ell_0(n)}$, and $z \in \{0,1\}^{m(n)/2+(\ell_2(n)-n)}$ such that, letting x be the lexicographically j-th string in $\mathit{Good}(n)$, defining $v \in \{0,1\}^{n-\log k-m(n)/2-1}$, $w \in \{0,1\}^{m(n)/2}$ so that $\langle j\rangle_{n-\log k-1} = vw$, and, for $0 \leq i < k$, defining $u_i \in \{0,1\}^n$ so that if $Q_G(x,\langle i\rangle) \in \mathit{Fixed}(n)$ then $u_i = h_n(Q_G(x,\langle i\rangle))||0^{\ell_2(n)-n}$ and otherwise $u_i = 1\langle i\rangle vz$, we have that $g(r||x, u_0, u_1, \ldots, u_{k-1}) = y$. For each $n \notin \mathcal{N}$, define A_n to reject every input.

Claim 4.4.6 *With probability 1 over $f \leftarrow \mathcal{F}$, the adversary $A = \{A_n\}$ breaks the pseudo-randomness of G^f.*

Proof Fix $f \in \mathcal{F}$.

Let $p(n)$ be a polynomial such that for all $n \in \mathcal{N}$, $|Good(n)| > 2^{\ell_1(n)}/p(n)$; such a $p(n)$ exists by Claim 4.4.3.

Fix $n \in \mathcal{N}$.

Observe than when $r \in \{0,1\}^{\ell_0(n)}$ and $x \in \{0,1\}^{\ell_1(n)}$ are randomly chosen, A_n accepts $G^f(r||x)$ if $x \in Good(n)$. It follows that A_n accepts pseudo-randomly generated strings with probability at least $\frac{2^{\ell_1(n)}}{p(n)}/2^{\ell_1(n)} = \frac{1}{p(n)}$.

Consider the probability that A_n accepts randomly chosen $y \in \{0,1\}^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}$. Observe that A_n accepts at most $|Good(n)| \cdot 2^{\ell_0(n)+m(n)/2+(\ell_2(n)-n)}$ strings. This means that A_n accepts randomly chosen strings with probability at most $\frac{|Good(n)| \cdot 2^{\ell_0(n)+m(n)/2+(\ell_2(n)-n)}}{2^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}} \leq \frac{1}{2^{m(n)/2}}$, since $|Good(n)| \leq 2^{\ell_1(n)}$. Then, since $m(n) \in \omega(\log n)$, we have that A_n accepts randomly chosen strings with probability at most $1/2^{\omega(\log n)}$. □

To finish the proof, it remains to consider the pseudo-randomness of f chosen according to $\mathcal{F}$ with respect to probabilistic polynomial-time adversaries that have oracle access to f and A. While we only need to show that at least one $f \in \mathcal{F}$ is pseudo-random, we will actually show that almost all $f \in \mathcal{F}$ are pseudo-random.

Claim 4.4.7 *With probability 1 over $f \leftarrow \mathcal{F}$, we have that for every PPT oracle machine $D^{(\cdot,\cdot)}$,*

$$\left| \Pr_{s \leftarrow_r \{0,1\}^n} \left[D^{(f,A)}\left(f(s)\right) = 1 \right] - \Pr_{z \leftarrow_r \{0,1\}^{\ell_2(n)}} \left[D^{(f,A)}\left(z\right) = 1 \right] \right| < \frac{1}{n^d}$$

for all d and sufficiently large n.

By Theorem 4.2.1, we have that in order to prove Claim 4.4.7, it suffices to prove the following.

Claim 4.4.8 *For every PPT oracle machine $D^{(\cdot,\cdot)}$, we have*

$$\left| \Pr_{\substack{f \leftarrow \mathcal{F} \\ s \leftarrow_r \{0,1\}^n}} \left[D^{(f,A)}\left(f(s)\right) = 1 \right] - \Pr_{\substack{f \leftarrow \mathcal{F} \\ z \leftarrow_r \{0,1\}^{\ell_2(n)}}} \left[D^{(f,A)}\left(z\right) = 1 \right] \right| < \frac{1}{n^d}$$

for all d and sufficiently large n.

To prove Claim 4.4.8, we will actually consider stronger probabilistic adversaries that are computationally unbounded but make only polynomially-many queries to f. Giving such adversaries oracle access to A is unnecessary, since a computationally unbounded adversary can compute A for itself[3].

[3]Note that in order for a computationally unbounded machine to compute A, it suffices to have the set *Small* and the sequence of polynomials p_{i+1} for $i \notin$ *Small* hardcoded into the machine; these objects can be hardcoded since they are of finite size.

For all probabilistic oracle machines $D^{(\cdot)}$ and all $n \in \mathbb{N}$: define $p_D(n)$ to be the probability that when $f \leftarrow \mathcal{F}$ and $s \leftarrow_r \{0,1\}^n$, D^f accepts $f(s)$; define $r_D(n)$ to be the probability that when $f \leftarrow \mathcal{F}$ and $z \leftarrow_r \{0,1\}^{\ell_2(n)}$, D^f accepts z; and define $q_D(n)$ to be the probability that when $f \leftarrow \mathcal{F}$, $s \leftarrow_r \{0,1\}^n$, and D^f is run on input $f(s)$, either D makes oracle query s, or the first bit of $h_n(s)$ is 1 and D makes an oracle query $\alpha \in \{0,1\}^n$ such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$.

Claim 4.4.8 is immediate from the following two claims.

Claim 4.4.9 *Let $D^{(\cdot)}$ be a probabilistic oracle machine. Then $|p_D(n) - r_D(n)| < q_D(n) + 1/n^d$ for all d and sufficiently large n.*

Claim 4.4.10 *Let $D^{(\cdot)}$ be a probabilistic oracle machine that makes at most polynomially-many oracle queries. Then $q_D(n) < 1/n^d$ for all d and sufficiently large n.*

We first prove Claim 4.4.9.

Proof *(Claim 4.4.9)* Fix probabilistic oracle machine $D^{(\cdot)}$. For each $n \in \mathbb{N}$, consider the following experiments.

Experiment 1

(a) Choose $s \leftarrow_r \{0,1\}^n$.

(b) Choose $f \leftarrow \mathcal{F}$.

(c) Run D^f on input $f(s)$.

Observe that the probability that D accepts in the above experiment is $p_D(n)$, observe that $q_D(n)$ is the probability that either D makes oracle query s, or the first bit of $h_n(s)$ is 1 and D makes an oracle query $\alpha \in \{0,1\}^n$ such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$.

Experiment 2

(a) Choose $z_1 \leftarrow_r \{0,1\}^n$.

(b) If $z_1 \in \mathit{FixedImage}^n(n)$, let $z_2 = 0^{\ell_2(n)-n}$; otherwise, choose $z_2 \leftarrow_r \{0,1\}^{\ell_2(n)-n}$.

(c) Choose $f \leftarrow \mathcal{F}$.

(d) If $z_1||z_2$ is in the image of f, let $s \in \{0,1\}^n$ be the string such that $f(s) = z_1||z_2$ (since f is 1-1, there can be at most one such string s). Otherwise, if $z_1||z_2$ is *not* in the image of f: if the leftmost bit of z_1 is 0, randomly select $t \in \{0,1\}^{n-1}$ such that $0||t \notin \mathit{FixedImage}^n(n)$, and let $s = h_n^{-1}(0||t)$; if the leftmost bit of z_1 is 1, let

$v \in \{0,1\}^{n-m(n)/2-1}$ be such that $1||v$ is a prefix of z_1, choose $w \leftarrow_r \{0,1\}^{m(n)/2}$, and let $s = h_n^{-1}(1||v||w)$.

(e) Run D^f on input $z_1||z_2$.

Define $r'_D(n)$ to be the probability that D accepts. Define $q^r_D(n)$ to be the probability that either D makes oracle query s, or the first bit of $h_n(s)$ is 1 and D makes an oracle query $\alpha \in \{0,1\}^n$ such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$.

Observe that in Experiment 2, we have that each string $z \in \left(\{0,1\}^{\ell_2(n)} - \mathit{FixedImage}^{\ell_2(n)}(n)\right)$ has probability exactly $1/2^{\ell_2(n)}$ of being the input to D – that is, such strings are chosen as the input to D with the same probability in Experiment 2 as in the experiment that gives rise to $r_D(n)$. Furthermore, in both Experiment 2 and in the experiment that gives rise to $r_D(n)$, the input to D is chosen independently of the choice $f \leftarrow \mathcal{F}$. It follows that $|r'_D(n) - r_D(n)| \leq |\mathit{FixedImage}^{\ell_2(n)}(n)|/2^{\ell_2(n)} = |h_n(\mathit{Fixed}(n))|/2^n = |\mathit{Fixed}(n)|/2^n$, where the first equality is by definition of $\mathit{FixedImage}^{\ell_2(n)}(n)$ and the second equality uses the fact that h_n is a permutation.

Then, we have that $|p_D(n) - r_D(n)| \leq |p_D(n) - r'_D(n)| + |\mathit{Fixed}(n)|/2^n$ for all $n \in \mathbb{N}$. However, by Claim 4.4.5, we have that $|\mathit{Fixed}(n)|/2^n < 1/n^d$ for all d and sufficiently large n. This means that $|p_D(n) - r_D(n)| < |p_D(n) - r'_D(n)| + 1/n^d$ for all d and sufficiently large n. Then, to complete the proof of Claim 4.4.9, it suffices to show that $|p_D(n) - r'_D(n)| \leq q_D(n)$ for all n.

Fix $n \in \mathbb{N}$.

We claim that D's view in Experiment 1 and D's view in Experiment 2 are distributed identically *until* either D makes oracle query s, or the first bit of $h_n(s)$ is 1 and D makes an oracle query $\alpha \in \{0,1\}^n$ such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$.

We begin by noting that the joint distribution of s and the input to D is identical in the two experiments. To see this, first observe by definition of $\mathcal{F}$ and by step (b) in Experiment 2 that in the two experiments, the input to D is chosen identically: each $z \in \left(\{0,1\}^{\ell_2(n)} - \mathit{FixedImage}^{\ell_2(n)}(n)\right)$ has probability exactly $1/2^{\ell_2(n)}$ of being the input to D; each $z \in \mathit{FixedImage}^{\ell_2(n)}(n)$ whose $(\ell_2(n) - n)$-bit suffix is $0^{\ell_2(n)-n}$ has probability exactly $1/|\mathit{Fixed}(n)|$ of being to the input to D; and all other z have probability 0 of being the input to D. Next observe that the distribution of $h_n(s)$ conditioned on the input z to D is identical in both experiments: when $z \in \mathit{FixedImage}^{\ell_2(n)}(n)$, $h_n(s)$ is the n-bit prefix of z; when the first bit of z is 0 but $z \notin \mathit{FixedImage}^{\ell_2(n)}(n)$, $h_n(s)$ is a randomly chosen n-bit string x such that the first bit of x is 0 and $x \notin \mathit{FixedImage}^n(n)$; when the first bit of z is 1, $h_n(s)$ is a randomly chosen n-bit string whose $(n - m(n)/2)$-bit prefix is identical to the $(n - m(n)/2)$-bit prefix of z. Now, since h_n is a fixed permutation, we also have that the distribution of s conditioned on the input z to D is identical in both experiments.

We next argue that distribution of query responses conditioned on s and on the input z to D is identical in both experiments at least until either D makes oracle query s, or the first bit of $h_n(s)$ is 1 and D makes an oracle query $\alpha \in \{0,1\}^n$ such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$.

First note that when the input z is in $FixedImage^{\ell_2(n)}(n)$, the distribution of query responses conditioned on s and on the input z to D is identical in the two experiments no matter which queries are made by D (since in this case, the joint distribution of f, s, and z is identical in the two experiments). That is, in this case, the joint distribution of s and the view of D is identical in the two experiments.

Now consider the case when when the first bit of z is 0 but $z \notin FixedImage^{\ell_2(n)}(n)$. In this case, until D makes query s, we have that in both experiments: each query α such that $h_n(\alpha) \in FixedImage^n(n)$ receives response $h_n(\alpha)||0^{\ell_2(n)-n}$; each new query α such that the leftmost bit of $h_n(\alpha)$ is 0 and such that $h_n(\alpha) \notin FixedImage^n(n)$ receives as a response a randomly chosen $\ell_2(n)$-bit string y such that y is different from every response seen so far, y is different from z, the first bit of y is 0, and $y \notin FixedImage^{\ell_2(n)}(n)$; each query α such that $h_n(\alpha) = 1||v||w$ for some $v \in \{0,1\}^{n-m(n)/2-1}$ and $w \in \{0,1\}^{m(n)/2}$, and such that for no previous query α' is it the case that $h_n(\alpha') = 1||v'||w$ for some $v' \in \{0,1\}^{n-m(n)/2-1}$, receives as a response $1||v||u$ where u is a randomly chosen $(\ell_2(n) - n + m(n)/2)$-bit string that is different from the suffix of every previous response for queries α' such that $h_n(\alpha')$ has leftmost bit 1; and, finally, each query α such that $h_n(\alpha) = 1||v||w$ for some $v \in \{0,1\}^{n-m(n)/2-1}$ and $w \in \{0,1\}^{m(n)/2}$, and such that for some previous query α' we have $h_n(\alpha') = 1||v'||w$ for some $v' \in \{0,1\}^{n-m(n)/2-1}$, receives response $1||v||u$ where u is identical to $(\ell_2(n) - n + m(n)/2)$-bit suffix of the response to query α'. That is, in this case, until D makes query s, the joint distribution of s and the view of D is identical in the two experiments.

Finally, consider the case where the first bit of z is 1. Recall that in this case, the first bit of $h_n(s)$ is 1. In this case, until D makes a query α such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$, we have that in both experiments: each query α such that $h_n(\alpha) \in FixedImage^n(n)$ receives response $h_n(\alpha)||0^{\ell_2(n)-n}$; each new query α such that the leftmost bit of $h_n(\alpha)$ is 0 and such that $h_n(\alpha) \notin FixedImage^n(n)$ receives as a response a randomly chosen $\ell_2(n)$-bit string y such that y is different from every response seen so far, the first bit of y is 0, and $y \notin FixedImage^{\ell_2(n)}(n)$; each query α such that $h_n(\alpha) = 1||v||w$ for some $v \in \{0,1\}^{n-m(n)/2-1}$ and $w \in \{0,1\}^{m(n)/2}$, and such that for no previous query α' is it the case that $h_n(\alpha') = 1||v'||w$ for some $v' \in \{0,1\}^{n-m(n)/2-1}$, receives as a response $1||v||u$ where u is a randomly chosen $(\ell_2(n) - n + m(n)/2)$-bit string that is different from the suffix of z and also different from the suffix of every previous response for queries α' such that $h_n(\alpha')$ has leftmost bit 1; and, finally, each query α such that $h_n(\alpha) = 1||v||w$ for some $v \in \{0,1\}^{n-m(n)/2-1}$ and $w \in \{0,1\}^{m(n)/2}$, and such that for some previous query α' we have $h_n(\alpha') = 1||v'||w$ for some $v' \in \{0,1\}^{n-m(n)/2-1}$, receives response $1||v||u$ where u is identical to $(\ell_2(n) - n + m(n)/2)$-bit suffix of the response to query α'. That is, in this case, the first bit of $h_n(s)$ is 1, and until D makes a query α such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$, the joint distribution of s and the view of D is identical in the two experiments.

We conclude that in Experiments 1 and 2, the joint distribution of s and the view of D is identical until either D makes oracle query s, or the first bit of $h_n(s)$ is 1 and D makes an oracle query $\alpha \in \{0,1\}^n$ such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$. It follows that $q^r_D(n) = q_D(n)$. It also follows that whenever *it is not the case* that either D makes oracle query s, or the first bit of $h_n(s)$ is 1 and D makes an oracle query $\alpha \in \{0,1\}^n$ such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$, then D has no information whatsoever to distinguish Experiment 1 from Experiment 2. Then we must have $|p_D(n) - r'_D(n)| \leq q_D(n)$.

□

We conclude by proving Claim 4.4.10.

Proof *(Claim 4.4.10)* We will use ideas from the proof of Impagliazzo and Rudich [IR89] that randomly chosen functions are one-way with probability 1.

Fix probabilistic oracle machine $D^{(\cdot)}$. Let $p(n)$ be a polynomial that bounds the number of queries made by D on inputs of length $\ell_2(n)$.

Fix $n \in \mathbb{N}$ and consider the probability that when $f \leftarrow \mathcal{F}$, $s \leftarrow_r \{0,1\}^n$, and D^f is run on input $f(s)$, D makes an oracle query α such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$.

Define T_1 to be the set of strings $t \in \{0,1\}^n$ such that the leftmost bit of $h_n(t)$ is 0 but $h_n(t) \notin \mathit{FixedImage}^n(n)$. Define T_2 to be the set of strings $t \in \{0,1\}^n$ such that the leftmost bit of $h_n(t)$ is 1. Define T_3 to be the set of strings $t \in \{0,1\}^n$ such that $h_n(t) \in \mathit{FixedImage}^n(n)$.

First condition on the case that $s \in T_1$. Note that in this case, the leftmost bit of $h_n(s)$ is 0. Recall that by definition of $\mathcal{F}$, we have that f on T_1 is a *randomly* chosen 1-1 function with range $\left(\{0\}||\{0,1\}^{\ell_2(n)-1} - \mathit{FixedImage}^{\ell_2(n)}(n)\right)$. Furthermore, the behaviour of f on T_1 is chosen independently from its behaviour on strings not in T_1. Then, the probability that D makes an oracle query s is at most $p(n)/(2^{n-1} - |\mathit{Fixed}(n)|)$.

Next condition on the case that $s \in T_2$. Recall that by definition of $\mathcal{F}$, we have that f on strings $t \in T_2$ first computes $w = h_n(t)$, uses the $(n-m(n)/2)$-bit prefix of w as the prefix of the output, and then applies randomly chosen 1-1 function $f''_n : \{0,1\}^{m(n)/2} \rightarrow \{0,1\}^{m(n)/2+(\ell_2(n)-n)}$ (chosen when the choice $f \leftarrow \mathcal{F}$ is made) to the $(m(n)/2)$-bit suffix of w to obtain the $(\ell_2(n)-n+m(n)/2)$-bit suffix of the output. Furthermore, the behaviour of f on T_2 is chosen independently from its behaviour on strings not in T_2. Then, since h_n is a fixed permutation, the probability that D makes an oracle query α such that the $(m(n)/2)$-bit suffix of $h_n(\alpha)$ is identical to the $(m(n)/2)$-bit suffix of $h_n(s)$ is at most $p(n)/(2^{m(n)/2})$.

Now, note that the probability that $s \in T_1$ is at most $(2^{n-1} - |Fixed(n)|)/2^n$, the probability that $s \in T_2$ is exactly $1/2$, and the probability that $s \in T_3$ is exactly $|Fixed(n)|/2^n$.

We then have $q_D(n) \leq \frac{2^{n-1}-|Fixed(n)|}{2^n} \cdot \frac{p(n)}{2^{n-1}-|Fixed(n)|} + \frac{1}{2} \cdot \frac{p(n)}{2^{m(n)/2}} + \frac{|Fixed(n)|}{2^n}$. That is, we have $q_D(n) \leq \frac{p(n)}{2^n} + \frac{p(n)}{2^{m(n)/2+1}} + \frac{|Fixed(n)|}{2^n}$; by Claim 4.4.5 and since $m(n) \in \omega(\log n)$, this is at most $1/n^d$ for all d and sufficiently large n.

□

4.5 Moving beyond constantly-many queries

In this section we consider extending Theorem 4.3.1 and Theorem 4.4.1 to the case of polynomially-many queries. We are able to do this for a restricted class of constructions. We begin by defining the restriction we need to place on the querying function of the construction.

Definition 19 (Many-oneness bounded almost everywhere) Let $\ell(n)$ and $q(n)$ be polynomials, and let $f : \{0,1\}^{\ell(n)} \rightarrow \{0,1\}^n$ be a function. f has *many-oneness bounded by $q(n)$ almost everywhere* if for all c and sufficiently large n, there are fewer than $2^n/n^c$ strings $y \in \{0,1\}^n$ such that $|f^{-1}(y)| > q(n)$.

Theorem 4.5.1 *Let $p(n)$, $q(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that $\ell_1(n) \leq n+O(\log n)$ and $\ell_2(n) > n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \rightarrow \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ be a non-adaptive oracle construction of a number generator, making $p(n)$ queries of length n to an oracle mapping n bits to $\ell_2(n)$ bits, such that the querying function of $G^{(\cdot)}$ has many-oneness bounded by $q(n)$ almost everywhere. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

Theorem 4.5.2 *Let $c \in \mathbb{R}^+$ and $m(n) \in \omega(\log n)$. Let $p(n)$, $q(n)$, $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that $\ell_1(n) \leq n+c\log n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \rightarrow \{0,1\}^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}$ be a non-adaptive oracle construction of a number generator that makes $p(n)$ queries of length n to a number generator mapping n bits to $\ell_2(n)$ bits, such that $G^{(\cdot)}$ has an $\ell_1(n)$-restricted querying function whose many-oneness is bounded by $q(n)$ almost everywhere. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

The proofs of Theorem 4.5.1 and Theorem 4.5.2 follow the same basic structure as the proofs of Theorem 4.3.1 and Theorem 4.4.1, respectively, but the procedure used to define the set $\mathit{Good}(n)$ in each proof is simpler as a result of the restriction on the many-oneness of the querying function. For both Theorem 4.5.1 and Theorem 4.5.2, the procedure begins by defining $\mathit{Fixed}(n) \subseteq \{0,1\}^n$ to be the set of strings in the image of the querying function Q_G whose many-oneness is *not* bounded by $q(n)$. Then, since the remaining strings in the image of Q_G have bounded many-oneness, it is easy to define a large set $\mathit{Good}(n) \subseteq \{0,1\}^{\ell_1(n)}$ such that for all distinct $x, x' \in \mathit{Good}(n)$ and all $0 \leq i, j < p(n)$, either $Q_G(x, \langle i \rangle) \in \mathit{Fixed}(n)$ or $Q_G(x, \langle i \rangle) \neq Q_G(x', \langle j \rangle)$. The idea is to proceed as follows: initially, every $x \in \{0,1\}^{\ell_1(n)}$ is a candidate for inclusion in $\mathit{Good}(n)$; while there are candidates remaining, select an arbitrary candidate x, add it to $\mathit{Good}(n)$, and remove from consideration as candidates all x' such that for some $0 \leq i, j < p(n)$, we have $Q_G(x, \langle i \rangle) \notin \mathit{Fixed}(n)$ and $Q_G(x, \langle i \rangle) = Q_G(x', \langle j \rangle)$. For every x added to $\mathit{Good}(n)$ by this procedure, at most $p(n)(q(n)-1)$ are removed from consideration, and hence at the end of this procedure $\mathit{Good}(n)$ has size at least $2^{\ell_1(n)}/(p(n)(q(n)-1)+1)$. Further details about these proofs are omitted for the sake of conciseness.

4.6 Goldreich-Levin-like constructions

In this section, we consider constructions where the seed has a public portion that is always included in the output, such that the oracle queries are chosen *non-adaptively* based only on the non-public portion of the seed. We further require that the computation of each individual output bit depends only on the seed and on the response to a single oracle query. We begin by formalizing this class of constructions.

Definition 20 (Bitwise-nonadaptive construction) Let $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials, and let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \to \{0,1\}^{\ell_0(n)+\ell_2(n)}$ be a non-adaptive oracle machine. We say that $G^{(\cdot)}$ is *bitwise-nonadaptive* if there exist uniformly-computable functions

$$Q_G = \left\{ Q_{G,n} : \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log \ell_2(n)} \to \{0,1\}^n \right\}$$

and

$$B = \left\{ B_n : \{0,1\}^{\ell_0(n)} \times \{0,1\}^{\ell_1(n)} \times \{0,1\}^n \times \{0,1\}^{\log \ell_2(n)} \to \{0,1\} \right\}$$

such that for all n, all $r \in \{0,1\}^{\ell_0(n)}$, all $x \in \{0,1\}^{\ell_1(n)}$, and all permutations $\pi : \{0,1\}^n \to \{0,1\}^n$, we have $G^\pi(r||x) = r||b_0||b_1||\ldots||b_{\ell_2(n)-1}$ where $b_i = B_n(r, x, \langle i \rangle, \pi(Q_{G,n}(x, \langle i \rangle)))$ for $0 \leq i \leq \ell_2(n) - 1$.

Observe that the Goldreich-Levin-based pseudo-random generator $G^\pi(r||x) = r||\pi(x)||\langle r, x \rangle$ is bitwise-nonadaptive.

We show that fully black-box bitwise-nonadaptive constructions $G^{(\cdot)}$ making queries to a one-way permutation, such that the non-public portion of the seed of $G^{(\cdot)}$ is no more that $O(\log n)$ bits longer than the length n of each oracle query, cannot achieve linear stretch.

Theorem 4.6.1 *Let $\alpha > 1$, and let $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that $\ell_1(n) < n + O(\log n)$ and $\ell_2(n) \geq \alpha \cdot \ell_1(n)$. Let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \to \{0,1\}^{\ell_0(n)+\ell_2(n)}$ be a bitwise-nonadaptive number generator that makes queries to a permutation on $\{0,1\}^n$. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the one-wayness of its oracle.*

To prove Theorem 4.6.1, we proceed in a manner similar to the proof of Theorem 4.4.1, building up a set *Good*$'(n)$ whose purpose is similar to the set *Good*(n) in that proof. The fact that each output bit of G depends only a single oracle query simplifies the construction of *Good*$'(n)$. Specifically, when constructing *Good*$'(n)$, we can ignore some of the "more difficult to deal with" queries made by $G^{(\cdot)}$, since we can later define adversary A to also ignore these queries simply by ignoring the corresponding output bits. This is what allows us to handle linearly-many queries in the current setting, even though we could only handle constantly-many queries in the proof of Theorem 4.4.1.

4.6.1 Proof of Theorem 4.6.1

Let $\alpha > 1$, let $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that we have $\ell_1(n) < n + O(\log n)$ and $\ell_2(n) \geq \alpha \cdot \ell_1(n)$, and let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \to \{0,1\}^{\ell_0(n)+\ell_2(n)}$ be a bitwise-nonadaptive number generator. For all $n > 0$, let $Q_{G,n} : \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log(\ell_2(n))} \to \{0,1\}^n$ and $B_n : \{0,1\}^{\ell_0(n)} \times \{0,1\}^{\ell_1(n)} \times \{0,1\}^n \times \{0,1\}^{\log(\ell_2(n))} \to \{0,1\}$ be the functions whose existence is guaranteed by the bitwise-nonadaptiveness of $G^{(\cdot)}$. We will describe distributions $\Pi = \{\Pi_n\}$ and an adversary $A = \{A_n\}$ such that when π is chosen according to Π, it is one-way with high probability with respect to adversaries that are given oracle access to A and π, but A breaks G^π for all $\pi \in \Pi$.

We will need the following definition.

Definition 21 Let $f : \{0,1\}^m \to \{0,1\}^n$ be a function.

1. The *image* of f, denoted $Im(f)$, is the set $\{f(x) : x \in \{0,1\}^m\}$.
2. For all $y \in \{0,1\}^n$, the *pre-image size* of y, denoted $s^f(y)$, is the size of the set $\{x : f(x) = y\}$.
3. For all $x \in \{0,1\}^m$, the *range-pre-image size* of x, denoted $t^f(x)$, is the pre-image size of $f(x)$ (that is, $t^f(x) = s^f(f(x))$).

4. For all $0 \leq p \leq 1$, the *p-median range-pre-image size* of f, denoted $Med_p(f)$, is the smallest $v \in \mathbb{N}$ such that for at least $p2^m$ strings $x \in \{0,1\}^m$, we have $t^f(x) \leq v$.

Now, define k to be the smallest integer such that $k > (\frac{2\alpha}{\alpha-1})^2$. For all $n > 0$ and $0 \leq i \leq k-1$, define $q_i(n) = Med_{i/k}(Q_{G,n})$.

Define m to be the largest integer such that $0 \leq m \leq k-1$ and $q_m(n) \leq n^c$ for some c and infinitely many n; note that such an m must exist, since $q_0(n) = 0$ for all n. Fix c such that $q_m(n) \leq n^c$ for infinitely many n, and define infinite set $\mathcal{N}$ to be the set of all n such that $q_m(n) \leq n^c$.

For each $n \in \mathcal{N}$, define $Small(n) = \{x \in \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log(\ell_2(n))} : t^{Q_{G,n}}(x) \leq q_m(n)\}$. By definition of $q_m(n)$, we have that $|Small(n)| \geq \frac{m}{k}2^{\ell_1(n)+\log(\ell_2(n))}$. If $m < k-1$, then for each $n \in \mathcal{N}$, define $Big(n) = \{x \in \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log(\ell_2(n))} : t^{Q_{G,n}}(x) \geq q_{m+1}(n)\}$; if $m = k-1$, then for each $n \in \mathcal{N}$, define $Big(n) = \emptyset$. Observe that for all $n \in \mathcal{N}$, we have that $|Big(n)| \geq \frac{k-(m+1)}{k}2^{\ell_1(n)+\log(\ell_2(n))}$. It follows that for all $n \in \mathcal{N}$, we have that $|Small(n) \cup Big(n)| \geq \frac{k-1}{k}2^{\ell_1(n)+\log(\ell_2(n))}$.

For each $n \notin \mathcal{N}$, define $Small(n) = Big(n) = \emptyset$.

Claim 4.6.2 *$|Q_{G,n}(Big(n))| \leq \frac{2^n}{n^d}$ for all d and sufficiently large n.*

Proof If $m = k-1$ then $Big(n) = \emptyset$ for all n, which means the claim holds trivially. So assume $m < k-1$. Note that by definition of m, we have $q_{m+1}(n) > n^{c'}$ for all c' and sufficiently large n.

Since we have by assumption that $\ell_1(n) < n + O(\log n)$, let b be such that $\ell_1(n) < n + b\log n$ for sufficiently large n.

Suppose for the sake of contradiction that the claim is false, that is, suppose $|Q_{G,n}(Big(n))| > \frac{2^n}{n^d}$ for some d and infinitely many n. Since $Big(n) = \emptyset$ for $n \notin \mathcal{N}$, it follows that $|Q_{G,n}(Big(n))| > \frac{2^n}{n^d}$ for some d and infinitely many $n \in \mathcal{N}$. But for sufficiently large n, we have $q_{m+1}(n) > b \cdot n^{d+1} \cdot \ell_2(n)$; for such $n \in \mathcal{N}$, each $y \in Q_{G,n}(Big(n))$ is such that $s^{Q_{G,n}}(y) > q_{m+1}(n) > b \cdot n^{d+1}\ell_2(n)$; it follows that for infinitely many $n \in \mathcal{N}$, we have:

$$\begin{aligned}
|Big(n)| &\geq b \cdot n^{d+1} \cdot \ell_2(n)|Q_{G,n}(Big(n))| \\
&> b \cdot n^{d+1} \cdot \ell_2(n)\frac{2^n}{n^d} \\
&= b \cdot n \cdot \ell_2(n) \cdot 2^n \\
&= 2^{n+\log b+\log(\ell_2(n))+\log n} \\
&> 2^{\ell_1(n)+\log(\ell_2(n))+\log n}
\end{aligned}$$

This contradicts $Big(n) \subseteq \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log(\ell_2(n))}$. $\square$

For each $n \in \mathcal{N}$, define $Good(n)$ to be the set of $x \in \{0,1\}^{\ell_1(n)}$ such that for at least $1 - \frac{1}{\sqrt{k}}$ of the $i \in \{0,1\}^{\log(\ell_2(n))}$, we have $(x,i) \in Small(n) \cup Big(n)$. By Markov's inequality, we have $|Good(n)| \geq (1 - \frac{1}{\sqrt{k}})2^{\ell_1(n)}$ for all $n \in \mathcal{N}$. Now, by definition of k, we have $(1 - \frac{1}{\sqrt{k}}) > \frac{\alpha+1}{2\alpha}$. This means that for all $n \in \mathcal{N}$, $|Good(n)| > \frac{\alpha+1}{2\alpha}2^{\ell_1(n)}$, and each $x \in Good(n)$ is such that for at least a $\frac{\alpha+1}{2\alpha}$ fraction of the $i \in \{0,1\}^{\log(\ell_2(n))}$, we have that $(x,i) \in Small(n) \cup Big(n)$. That is, each $x \in Good(n)$ is such that for at least $\frac{\alpha+1}{2\alpha}\ell_2(n)$ strings $i \in \{0,1\}^{\log(\ell_2(n))}$, we have that $(x,i) \in Small(n) \cup Big(n)$. Then, since $\ell_2(n) \geq \alpha \cdot \ell_1(n)$, each $x \in Good(n)$ is such that for at least $\frac{\alpha+1}{2}\ell_1(n)$ strings $i \in \{0,1\}^{\log(\ell_2(n))}$, we have that $(x,i) \in Small(n) \cup Big(n)$.

Now, we give a procedure that for each $n \in \mathcal{N}$, defines a set $Good'(n) \subseteq Good(n)$ and functions $h_n : \{0,1\}^n \to \{0,1\}^n$ and $\delta_n : \{0,1\}^{\ell_1(n)+\log(\ell_2(n))} \to \{0,1\}^{\log(\ell_2(n))}$ satisfying the following properties:

(i) $h_n : \{0,1\}^n \to \{0,1\}^n$ is a well-defined permutation.

(ii) For all $x \in Good'(n)$, for the set I of the $\frac{\alpha+1}{2}\ell_1(n)$ lexicographically first $i \in \{0,1\}^{\log(\ell_2(n))}$ such that $(x,i) \in Small(n) \cup Big(n)$, and for all $i \in I$, if $(x,i) \in Small(n)$ then we have $h_n(Q_{G,n}(x,i)) = 1||\delta_n(x,i)||\mathsf{suffix}(x)$, where $\mathsf{suffix}(x)$ is the $\ell_1(n) - \log(\ell_2(n)) - 1$ bit suffix of x.

(iii) $Good'(n)$ is of size at least $2^{\ell_1(n)}/(\alpha \cdot \ell_1(n) \cdot n^c + \frac{4\alpha}{\alpha+1}\ell_2(n))$.

Fix $n \in \mathcal{N}$ and consider the following procedure. Initially, $Good'(n) = \emptyset$, $Remainder(n) = Good(n)$, and h_n and δ_n are undefined everywhere. Then proceed as follows:

1. While $Remainder(n) \neq \emptyset$ do:

1.1 Select the lexicographically first $x \in Remainder(n)$. Say $x = vw$, where $v \in \{0,1\}^{\log(\ell_2(n))+1}$ and $w \in \{0,1\}^{\ell_1(n)-\log(\ell_2(n))-1}$.

1.2 Define I to be the set of the $\frac{\alpha+1}{2}\ell_1(n)$ lexicographically first $i \in \{0,1\}^{\log(\ell_2(n))}$ such that $(x,i) \in Small(n) \cup Big(n)$. Let $i_0, i_1, \ldots, i_{\frac{\alpha+1}{2}\ell_1(n)-1}$ denote the strings in I in lexicographic order.

1.3 For $0 \leq j \leq \frac{\alpha+1}{2}\ell_1(n) - 1$ such that $(x, i_j) \in Small(n)$ do:

1.3.1 Define $\delta_n(x, i_j)$ to be the lexicographically first $i \in I$ such that $Q_{G,n}(x,i) = Q_{G,n}(x,i_j)$.

1.3.2 If $\delta_n(x, i_j) = i_j$ then:

1.3.2.1 Define $h_n(Q_{G,n}(x,i_j)) = 1||i_j||w$.

1.3.2.2 For every $x' \in Remainder(n)$ such that there exists $k \in \{0,1\}^{\log(\ell_2(n))}$ for which $(x',k) \in Small(n)$ and $Q_{G,n}(x',k) = Q_{G,n}(x,i_j)$, remove x' from $Remainder(n)$.

1.4 For every $x' \in Remainder(n)$ such that $x' = v'w$ for some $v' \in \{0,1\}^{\log(\ell_2(n))+1}$, remove x' from $Remainder(n)$.

1.5 Add x to $Good'(n)$.

2. For $0 \leq j \leq |Q_{G,n}(Big(n))| - 1$ do:

2.1 Let y be the lexicographically j-th string in $Q_{G,n}(Big(n))$.
Define $h_n(y) = 0^{n-\log|Q_{G,n}(Big(n))|} || \langle j \rangle$.

3. For every $y \in \{0,1\}^n$ on which h_n is still undefined, define $h_n(y)$ arbitrarily subject to the restriction that h_n is 1-1.

For all $n \notin \mathcal{N}$, define $h_n : \{0,1\}^n \to \{0,1\}^n$ to be the identity.

It is easy to see that for all $n \in \mathcal{N}$, the above procedure constructs $Good'$, h_n, and δ_n satisfying properties (i) and (ii). We now show that property (iii) is also satisfied.

Claim 4.6.3 *For all $n \in \mathcal{N}$, when the above procedure terminates, $Good'(n)$ is of size at least $2^{\ell_1(n)}/(\alpha \cdot \ell_1(n) \cdot n^c + \frac{4\alpha}{\alpha+1}\ell_2(n))$.*

Proof Observe that on each iteration of the outer loop, exactly one element is added to $Good'(n)$. On the other hand, observe that on each iteration of the outer loop, at most $\frac{\alpha+1}{2}\ell_1(n) \cdot q_m(n) + 2\ell_2(n) \leq \frac{\alpha+1}{2}\ell_1(n) \cdot n^c + 2\ell_2(n)$ elements are removed from $Remainder(x)$. This means that at least $|Good(n)|/(\frac{\alpha+1}{2}\ell_1(n) \cdot n^c + 2\ell_2(n)) \geq (\frac{\alpha+1}{2\alpha}2^{\ell_1(n)})/(\frac{\alpha+1}{2}\ell_1(n) \cdot n^c + 2\ell_2(n)) = 2^{\ell_1(n)}/(\alpha \cdot \ell_1(n) \cdot n^c + \frac{4\alpha}{\alpha+1}\ell_2(n))$ iterations of the outer loop occur. □

We now describe distribution Π and adversary $A = \{A_n\}$.

Distribution $\Pi = \{\Pi_n\}$: In order to define distribution Π, we first define distributions $\Pi' = \{\Pi'_n\}$, $\Pi'' = \{\Pi''_n\}$, and $\Pi''' = \{\Pi'''_n\}$ as follows. For each $n \geq 0$, define Π''_n to be the uniform distribution over the set of permutations $\pi''_n : \{0,1\}^n \to \{0,1\}^n$. For each $n \geq 0$, define Π'''_n to be the uniform distribution over the set of permutations $\pi'''_n : \{0,1\}^n \to \{0,1\}^n$ such that π'''_n is the identity on strings $y \in \{0,1\}^n$ that satisfy $0y \in h_{n+1}(Q_{G,n+1}(Big(n+1)))$. For each $n \geq 0$, define Π'_n to be the distribution over the set of permutations $\pi'_n : \{0,1\}^n \to \{0,1\}^n$ obtained by first sampling $\pi''_{\log^2 n} \in \Pi''_{\log^2 n}$ and $\pi'''_{n-1} \in \Pi'''_{n-1}$, and then defining π'_n as follows: for all $x \in \{0,1\}^{n-1}$, $\pi'_n(0x) = 0\pi'''_{n-1}(x)$; for all $x_1 \in \{0,1\}^{n-\log^2 n-1}$ and $x_2 \in \{0,1\}^{\log^2 n}$, $\pi'_n(1x_1x_2) = 1x_1\pi''_{\log^2}(x_2)$. Also, for each $n \in \mathbb{N}$, define Π_n to be the distribution over the set of permutations $\pi_n : \{0,1\}^n \to \{0,1\}^n$ obtained by first sampling $\pi'_n \in \Pi'_n$ and then defining π_n as follows: $\pi_n = \pi'_n \circ h_n$.

Adversary $A = \{A_n\}$: For every $n \in \mathcal{N}$, we define adversary A_n as follows. On input $r, b_0, b_1, \ldots, b_{\ell_2(n)-1}$, where $r \in \{0,1\}^{\ell_0(n)}$ and each $b_i \in \{0,1\}$, A_n accepts if and only if there exists $x \in Good'(n)$ and $z \in \{0,1\}^{\log^2 n}$ such that, defining $u \in \{0,1\}^{\log(\ell_2(n))+1}$, $v \in$

$\{0,1\}^{\ell_1(n)-\log(\ell_2(n))-\log^2 n-1}$, $w \in \{0,1\}^{\log^2 n}$ so that $x = uvw$, we have that the $\frac{\alpha+1}{2}\ell_1(n)$ lexicographically first $\langle i\rangle \in \{0,1\}^{\log(\ell_2(n))}$ for which $(x,\langle i\rangle) \in \mathit{Small}(n) \cup \mathit{Big}(n)$ are such that if $(x,\langle i\rangle) \in \mathit{Small}(n)$ then $b_i = B_n(r,x,\langle i\rangle, 1||\delta_n(x,\langle i\rangle)||v||z)$ and if $(x,\langle i\rangle) \in \mathit{Big}(n)$ then $b_i = B_n(r,x,\langle i\rangle, h_n(Q_{G,n}(x,\langle i\rangle)))$. For each $n \notin \mathcal{N}$, define A_n to reject every input.

Claim 4.6.4 *With probability 1 over the choice of $\pi \in \Pi$, the adversary $A = \{A_n\}$ breaks the pseudo-randomness of G^π.*

Proof Fix $\pi \in \Pi$. Fix $n \in \mathcal{N}$.

Observe than when $r \in \{0,1\}^{\ell_0(n)}$ and $x \in \{0,1\}^{\ell_1(n)}$ are randomly chosen, A_n accepts $G^\pi(r,x)$ if $x \in \mathit{Good}'(n)$. It follows that A_n accepts pseudo-randomly generated strings with probability at least $1/(\alpha \cdot \ell_1(n) \cdot n^c + \frac{4\alpha}{\alpha+1}\ell_2(n))$.

Now consider the probability that A_n accepts randomly chosen $r, b_0, b_1, \ldots, b_{\ell_2(n)-1}$, where $r \in \{0,1\}^{\ell_0(n)}$ and each $b_i \in \{0,1\}$. Observe that for each $r' \in \{0,1\}^{\ell_0(n)}$, each $x \in \mathit{Good}'(n)$, and each $z \in \{0,1\}^{\log^2 n}$, there are exactly $2^{\ell_2(n)-\frac{\alpha+1}{2}\ell_1(n)}$ strings $b'_0, b'_1, \ldots, b'_{\ell_2(n)-1}$ such that, defining $u \in \{0,1\}^{\log(\ell_2(n))+1}$, $v \in \{0,1\}^{\ell_1(n)-\log(\ell_2(n))-\log^2 n-1}$, $w \in \{0,1\}^{\log^2 n}$ so that $x = uvw$, we have that the $\frac{\alpha+1}{2}\ell_1(n)$ lexicographically first $\langle i\rangle \in \{0,1\}^{\log(\ell_2(n))}$ for which $(x,\langle i\rangle) \in \mathit{Small}(n) \cup \mathit{Big}(n)$ are such that if $(x,\langle i\rangle) \in \mathit{Small}(n)$ then $b'_i = B_n(r',x,\langle i\rangle, 1||\delta_n(x,\langle i\rangle)||v||z)$ and if $(x,\langle i\rangle) \in \mathit{Big}(n)$ then $b'_i = B_n(r',x,\langle i\rangle, h_n(Q_{G,n}(x,\langle i\rangle)))$. This means that for each $r' \in \{0,1\}^n$, there are at most $|\mathit{Good}'(n)|2^{\log^2 n+\ell_2(n)-\frac{\alpha+1}{2}\ell_1(n)}$ strings $b'_0, b'_1, \ldots, b'_{\ell_2(n)-1}$ such that A_n accepts $r', b'_0, b'_1, \ldots, b'_{\ell_2(n)-1}$. But $|\mathit{Good}'(n)|2^{\log^2 n+\ell_2(n)-\frac{\alpha+1}{2}\ell_1(n)} \le 2^{\ell_1(n)}2^{\log^2 n+\ell_2(n)-\frac{\alpha+1}{2}\ell_1(n)} = 2^{\log^2 n+\ell_2(n)-\frac{\alpha-1}{2}\ell_1(n)}$. It follows that A_n accepts a randomly chosen string with probability at most $1/2^{\frac{\alpha-1}{2}\ell_1(n)-\log^2 n}$. Since $\alpha > 1$ and since $\ell_1(n)$ is a polynomial, we have that for sufficiently large $n \in \mathcal{N}$, A_n accepts a randomly chosen string with probability less than $1/2^{\frac{\alpha-1}{4}\ell_1(n)}$. $\square$

To finish the proof, it remains to consider the one-wayness of π chosen according to Π with respect to probabilistic polynomial-time adversaries that have oracle access to π and A. We will actually consider stronger adversaries that are computationally unbounded but make only polynomially-many queries to π. Giving such adversaries oracle access to A is unnecessary, since a computationally unbounded adversary can compute A for itself[4].

While we only need to show that at least one $\pi \in \Pi$ is one-way, we will actually show that almost all $\pi \in \Pi$ are one-way. Our proof is based on the proof of Impagliazzo and Rudich [IR89] that randomly chosen functions are one-way with probability 1.

Claim 4.6.5 *Suppose $\pi \leftarrow \Pi$ is randomly chosen. Then with probability 1, π is one-way with respect to computationally unbounded Turing machines that make only polynomially-many*

[4]Note that in order for a computationally unbounded machine to compute A, it suffices to have the constants m and c, used in the definition of $\mathcal{N}$, hardcoded into the machine.

queries to π*.*

Proof For each $\pi \in \Pi$, define permutation $P_\pi : \{0,1\}^* \to \{0,1\}^*$ as follows: for all $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, $P_\pi(x) = \pi(h_n^{-1}(x))$.

We claim that for each $\pi \in \Pi$, if P_π is one-way with respect to computationally unbounded Turing machines that make only polynomially-many oracle queries, then so is π. To see this, fix $\pi \in \Pi$ and suppose M is an oracle Turing machine that makes polynomially-many queries to its oracle and breaks the one-wayness of π. Using the fact that $h = \{h_n\}$ is a uniformly computable function, we define oracle Turing machine M' as follows. Given an oracle for P_π and an input $y \in \{0,1\}^*$, M' simulates M on input y. For each oracle query z made by M, M' uses $P_\pi(h_{|z|}(z))$ as the response to the query (note that this means the response to each query z is $\pi(z)$). Eventually M outputs a string x. Then, M' outputs $h_n(x)$. Observe that M' outputs $P_\pi^{-1}(y)$ if and only if the simulation of M outputs $\pi^{-1}(y)$, and hence M' breaks the one-wayness of P_π.

Then, by the definitions of Π and Π', it follows that in order to show $\pi \leftarrow \Pi$ is one-way with probability 1, it suffices to show that $\pi' \leftarrow \Pi'$ is one-way with probability 1.

Consider how well an adversary can invert randomly chosen $\pi' \leftarrow \Pi'$. Let M be an oracle Turing machine that makes polynomially-many queries to its oracle; let $p(n)$ be a polynomial that bounds the number of queries made by M on inputs of length n. For each n, consider the probability that when $\pi' \in \Pi'$ and $x \in \{0,1\}^n$ are randomly chosen, $M^{\pi'}(\pi'(x))$ outputs x. Note that M is *not* given any pre-computation on π', and hence advice-based techniques for inverting permutations (such as the techniques of [DTT10]) are not relevant in this setting. Without loss of generality, say that the string that M outputs is one of the oracle queries it makes. Then it suffices to consider the probability that on input $\pi'(x)$, M queries x. Now, recall that by the definition of Π', we have that π' is a *randomly* chosen permutation on the set $\{0y \in \{0,1\}^n : 0y \notin h_n(Q_{G,n}(Big(n)))\}$, π' is the identity on the set $\{0y \in \{0,1\}^n : 0y \in h_n(Q_{G,n}(Big(n)))\}$, and π' on inputs of length n with leftmost bit 1 is the identity on its leftmost $n - \log^2 n$ input bits and is a *randomly* chosen permutation on its rightmost $\log^2 n$ input bits. Then conditioned on the case that the leftmost bit of x is 0 but $x \notin h_n(Q_{G,n}(Big(n)))$, the probability that M queries x is at most $\frac{p(n)}{2^{n-1}-|Q_{G,n}(Big(n))|}$. It follows that conditioned on the case that the leftmost bit of x is 1, the probability that M queries x is at most $\frac{p(n)}{2^{n-1}} + \frac{|Q_{G,n}(Big(n))|}{2^{n-1}}$. We also have that conditioned on the case that the leftmost bit of x is 1, the probability that M queries x is at most $\frac{p(n)}{2^{\log^2 n}}$. Then, without conditioning on x, the probability that $M^{\pi'}(\pi'(x))$ outputs x is at most $\frac{p(n)}{2^n} + \frac{p(n)}{2^{\log^2 n+1}} + \frac{|Q_{G,n}(Big(n))|}{2^n}$; by Claim 4.6.2, this is at most $1/n^d$ for all d and sufficiently large n. It follows that for all d and sufficiently large n, the measure of $\pi' \in \Pi'$ such that

$$\Pr_{x\in\{0,1\}^n}\left[M^{\pi'}(\pi'(x)) = x\right] \geq 1/n^d$$

is less than $1/n^2$. Then, for all d, we have by the Borel-Cantelli lemma that the measure of $\pi' \in \Pi'$ such that

$$\Pr_{x\in\{0,1\}^n}\left[M^{\pi'}(\pi'(x)) = x\right] \geq 1/n^d$$

for infinitely many n is 0. That is, the measure of $\pi' \in \Pi'$ such that M breaks the one-wayness of π' is 0.

Since there are only countably many Turing machines, the measure of $\pi' \in \Pi'$ such that there exists a Turing machine that makes only polynomially-many oracle queries and breaks the one-wayness of π' is 0.

□

4.7 Open problems

It remains to consider more general classes of constructions.

Queries chosen based on a long seed Our results for constructions whose seed is significantly longer than the length of each oracle query (Theorems 4.4.1 and 4.5.2) are restricted to constructions whose queries depend only a portion of the seed whose length is close to to the length of each query. Can we remove this restriction? Of course, such unrestricted constructions *can* obtain stretch that is significantly longer than the stretch of the given oracle, simply by dividing their seed into portions whose length is equal to the length of each oracle query, and then querying the oracle on each such portion. The goal is to show that the stretch obtained by following this approach is the best that can be achieved by non-adaptive black-box constructions.

Constructions making polynomially-many queries For our results about constructions making polynomially-many queries (Theorems 4.5.1 and 4.5.2), can we remove the restriction on the many-oneness of the querying function? Such a restriction does not seem necessary – indeed, it is hard to imagine how a construction would benefit by violating this restriction. Nevertheless, removing this restriction has turned out to be a difficult problem so far. One possible explanation for this difficulty is that our proofs never use the fact that the constructions $G^{(\cdot)}$ that we are interested in are efficiently computable (and, in particular, have efficiently computable querying functions). It would be interesting to show that, in fact, there exists an *inefficient* non-adaptive black-box construction that has a complicated and inefficient querying function violating the many-oneness restriction and obtains more stretch than the bounds given in Theorem 4.5.1 or Theorem 4.5.2.

Weakening the black-box requirement All of our impossibility results are for *fully* black-box constructions. Can we extend these results to weaker versions of black-box constructions, such as *semi*-black-box and *mildly* black-box constructions?

Definition 22 (Semi-black-box reduction [IR89, RTV04]) Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ be a number generator whose construction has access to an oracle for a length-increasing function mapping $\ell'_1(n)$ bits to $\ell'_2(n)$ bits. There is a *semi-black-box* reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle if for every function $f : \{0,1\}^{\ell'_1(n)} \to \{0,1\}^{\ell'_2(n)}$ and every probabilistic polytime oracle Turing machine $A^{(\cdot)} : \{0,1\}^{\ell_2(n)} \to \{0,1\}$, if A^f breaks the pseudo-randomness of G^f then there exists a probabilistic polytime oracle Turing machine $M^{(\cdot)}$ such that M^f breaks the pseudo-randomness of f.

Definition 23 (Mildly black-box reduction [RTV04]) Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ be a number generator whose construction has access to an oracle for a length-increasing function mapping $\ell'_1(n)$ bits to $\ell'_2(n)$ bits. There is a *mildly black-box* reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle if for every function $f : \{0,1\}^{\ell'_1(n)} \to \{0,1\}^{\ell'_2(n)}$ and every probabilistic polytime Turing machine $A : \{0,1\}^{\ell_2(n)} \to \{0,1\}$, if A breaks the pseudo-randomness of G^f then there exists a probabilistic polytime oracle Turing machine $M^{(\cdot)}$ such that M^f breaks the pseudo-randomness of f.

We believe that in order to extend our results to the case of semi-black-box constructions, it may suffice to adapt our existing proofs. Extending our results to the case of mildly black-box constructions seems much more difficult.

Slightly-adaptive constructions Existing constructions that obtain the best increase in stretch use their oracle in a highly adaptive manner. Roughly speaking, these existing constructions query their oracle on inputs formed entirely of previous query response bits; furthermore, such response bits are never used to form more than a single query. Can our impossibility results be extended to constructions that use a much more restricted form of adaptivity?

For example, consider constructions that have a small number of rounds (e.g., two rounds) of adaptivity, where the construction queries non-adaptively *within* each round and receives responses to these queries at the end of each round. This kind of "slight" adaptivity has been considered by Naor and Reingold [NR99] in the context of constructing pseudo-random *function* generators. They define an object called a *pseudo-random synthesizer* (which can be thought of as a strong version of a pseudo-random number generator) and show how to construct a pseudo-random function generator from a pseudo-random synthesizer using $O(\log n)$ rounds of adaptivity.

Also, consider constructions that are computed in small space (e.g., logspace), which implicitly restricts adaptivity since the construction cannot "remember" more than a few response bits from previous queries. Formally, we can model such constructions as logspace Turing machines with a special write-only query tape and a special read-only query response tape, where the special tapes are erased between queries.

Fully-adaptive constructions Can we show that existing adaptive black-box constructions are optimal, in the sense that they achieve the maximum possible stretch relative to the number of queries they make? For example, the work of Bronson [Bro08] was motivated by the problem of showing that black-box constructions of pseudo-random function generators (which can be viewed as pseudo-random number generators of exponential stretch) from pseudo-random number generators must make super-logarithmically-many oracle queries, which would match the best known upper bound.

Constructions from one-way permutations Theorems 4.3.1, 4.4.1, 4.5.1 and 4.5.2 show that in certain settings, non-adaptive black-box constructions of pseudo-random number generators from pseudo-random number generators of smaller stretch cannot obtain as much stretch as adaptive black-box constructions. Does the same distinction between adaptivity and non-adaptivity hold in these settings for black-box constructions of pseudo-random number generators from one-way permutations? Theorems 4.3.1, 4.4.1, 4.5.1 and 4.5.2 can indeed be extended to non-adaptive black-box constructions of pseudo-random number generators from one-way permutations. However, the one-way permutation analogues of Theorems 4.3.1, 4.4.1, and 4.5.1 are not sufficient for getting a distinction between adaptive and non-adaptive constructions, since the adaptive versions of these analogues are not known to be false.

Bibliography

[ADW09] Joel Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage resilient public-key cryptography in the bounded retrieval model. In *Advances in Cryptology — CRYPTO 2009*, pages 36–54, Berlin, Heidelberg, 2009. Springer.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC '09: Proceedings of the 6th Theory of Cryptography Conference*, pages 474–495, Berlin, Heidelberg, 2009. Springer.

[BB03] David Brumley and Dan Boneh. Remote timing attacks are practical. In *SSYM'03: Proceedings of the 12th conference on USENIX Security Symposium*, Berkeley, CA, USA, 2003. USENIX Association.

[Ber05] Daniel J. Bernstein. Cache-timing attacks on AES. `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, April 2005. Revised version of earlier 2004-11 version.

[BJP11] Josh Bronson, Ali Juma, and Periklis Papakonstantinou. Limits on the stretch of non-adaptive constructions of pseudo-random generators. In *TCC '11: Proceedings of the 8th Theory of Cryptography Conference*, 2011.

[BKKV10] Zvika Brakerski, Yael Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS '10: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, Washington, DC, USA, 2010. IEEE Computer Society.

[Bro08] Josh Bronson. Constructing pseudorandom function generators. Master's thesis, University of Toronto, 2008.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, pages 361–381, 2010.

[DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana Lopez-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS '10: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, Washington, DC, USA, 2010. IEEE Computer Society.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 621–630, New York, NY, USA, 2009. ACM.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS '08: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 293–302, Washington, DC, USA, 2008. IEEE Computer Society.

[DTT10] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 649–665. Springer Berlin / Heidelberg, 2010.

[FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *TCC 2010*, pages 343–360, 2010.

[FRR+10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting against computationally bounded and noisy leakage. In *EUROCRYPT 2010*, 2010.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, New York, NY, USA, 2009. ACM.

[GGKT05] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput*, 35(1):217–246, 2005.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *Advances in Cryptology — CRYPTO 2008*, pages 39–56, Berlin, Heidelberg, 2008. Springer.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC'89*, pages 25–32, Berlin, 1989. ACM.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2: Basic Applications*. Cambridge University Press, 2004.

[GR10] Shafi Goldwasser and Guy Rothblum. Securing computation against continuous leakage. These proceedings, 2010.

[HRV10] Iftach Haitner, Omer Reingold, and Salil Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. In *STOC '10*, pages 437–446, New York, NY, USA, 2010. ACM.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC '89*, 1989.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology — CRYPTO 2003*, pages 463–481, Berlin, Heidelberg, 2003. Springer.

[JV10] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *Advances in Cryptology – CRYPTO 2010*, pages 41–58. Springer Berlin / Heidelberg, 2010.

[JV11] Ali Juma and Yevgeniy Vahlis. Leakage-resilient authentication. In progress, 2011.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[Kuh03] Markus G. Kuhn. Compromising emanations: eavesdropping risks of computer displays. Technical Report UCAM-CL-TR-577, University of Cambridge, 2003.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009.

[Lu06] Chi-Jen Lu. On the complexity of parallel hardness amplification for one-way functions. In *TCC '06*, LNCS, pages 462–481, 2006.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *TCC '04: Proceedings of the 1st Theory of Cryptography Conference*, pages 278–296, Berlin, Heidelberg, 2004. Springer.

[MTVY11] Tal Malkin, Isamu Teranishiy, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *TCC '11: Proceedings of the 8th Theory of Cryptography Conference*, 2011.

[MV11] Eric Miles and Emanuele Viola. On the complexity of increasing the stretch of pseudorandom generators. In *TCC '11: Proceedings of the 8th Theory of Cryptography Conference*, 2011.

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336 – 375, 1999.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology — CRYPTO 2009*, pages 18–35, Berlin, Heidelberg, 2009. Springer.

[OST06] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In *Topics in Cryptology - CT-RSA 2006*, pages 1–20, Berlin, Heidelberg, 2006. Springer.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *Advances in Cryptology – EUROCRYPT 2009*, pages 462–482, Berlin, Heidelberg, 2009. Springer-Verlag.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-SMART '01: Proceedings of the International Conference on Research in Smart Cards*, pages 200–210, London, UK, 2001. Springer-Verlag.

[RTV04] O. Reingold, L. Trevisan, and S. Vadhan. Notions of reducibility between cryptographic primitives. In *TCC '04*, LNCS, pages 1–20, 2004.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT 2010*, 2010.

[Vio05] E. Viola. On constructing parallel pseudorandom generators from one-way functions. In *CCC '05*, pages 183–197. IEEE Computer Society, 2005.