Password Mistyping in Two-Factor-Authenticated Key Exchange

Vladimir Kolesnikov[1] and Charles Rackoff[2]

[1] Bell Labs, Murray Hill, NJ 07974,USA kolesnikov@research.bell-labs.com
[2] Dept. Computer Science, University of Toronto, Canada rackoff@cs.utoronto.ca

Abstract. We study the problem of Key Exchange (KE), where authentication is two-factor and based on both electronically stored long keys and human-supplied credentials (passwords or biometrics). The latter credential has low entropy and may be *adversarily* mistyped. Our main contribution is the first formal treatment of mistyping in this setting.
Ensuring security in presence of mistyping is subtle. We show mistyping-related limitations of previous KE definitions and constructions (of Boyen et al. [7, 6, 10] and Kolesnikov and Rackoff [16]).
We concentrate on the practical two-factor authenticated KE setting where *servers* exchange keys with *clients*, who use short passwords (memorized) and long cryptographic keys (stored on a card). Our work is thus a natural generalization of Halevi-Krawczyk [15] and Kolesnikov-Rackoff [16]. We discuss the challenges that arise due to mistyping. We propose the first KE definitions in this setting, and formally discuss their guarantees. We present efficient KE protocols and prove their security.

1 Introduction

The problem of securing communication over an insecure network is generally solved using *key exchange* (KE). KE provides partners with matching randomly chosen keys, which are used for securing their conversation. Of course, no adversary *Adv* should be able to mismatch players. Therefore, players must possess secrets with which they can authenticate themselves. The kind of secrets that are available to players determines the setting of KE. In the simplest KE setting players have a long shared random string. KE is more complicated if parties establish key pairs with the public keys securely published. Using weak and/or fuzzy credentials, such as passwords or biometrics, further complicates the design of KE. Finally, using a combination of credentials may make certain aspects of KE easier (such as incorporating password authentication), but increases the overall complexity of the solution, as discussed in [16].

Our setting. Two-factor authentication is critical and is used extensively in secure applications such as banking, VPN, etc. Stored long keys protect against online adversaries, but are vulnerable against theft. The extra layer of security is achieved with additional use of a theft-resistant credential, e.g. a short password or a biometric. Unfortunately, neither password nor biometric can be expected to be read reliably into the computer.

We give foundation to this setting by generalizing the work of Halevi-Krawczyk (HK) [15] and Kolesnikov-Rackoff (KR) [16]. Recall, they address the client-server setting where both long key and a short password are used for KE. The servers are incorruptible, but client's card or password can be compromised.

Motivated by real scenarios, we study the effects of password mistyping. Mistyping need not be random, but may be skewed by the adversary, e.g. by technical means or social engineering manipulation. We thus consider security against adversaries who can *arbitrarily* affect user's mistyping. This consideration is especially relevant in case biometric credentials are used for authentication, since, due to technology limitations, biometric readings are *expected* to be misread.

Mistyping opens subtle vulnerabilities and raises complex definitional issues. In the sequel, we use terms "password" and "mistype", although our work applies to passwords, biometrics, and other short noisy credentials, as noted in Sect. 5.

1.1 Our contributions and outline of work

Our main contribution is the first formal treatment of mistyping of passwords in KE that uses a combination of credentials.

We discuss recent definitions that consider mistyping-related settings and issues – robust fuzzy extractors of [7, 6, 10]. We point out a limitation of the definitions of [7, 6, 10] with respect to robust handling of biometric misreading/mistyping and discuss possible remedies. We demonstrate and correct a vulnerability of the definition and protocol of [16], which can only be exploited when users mistype. These observations further emphasize the subtleties of mistyping and the need for its formal treatment and deeper understanding.

In Sect. 3, we introduce our setting and the framework of [16] which we build upon. Then, with simple protocols we illustrate mistyping-related issues, discuss natural definitional approaches to handling mistyping and their shortcomings. Most of the mistyping-related subtleties we uncover arise due to the simultaneous use of both long keys and passwords. In Sect. 4, we formalize our discussion in a definition, and formally argue that it prevents attacks that exploit mistyping.

In Sect. 5 we discuss applications of our work in biometric authentication.

In Sect. 6 we give efficient protocols; we prove their security in the full version.

1.2 Related work

The problem of key exchange has deservedly received a vast amount of attention. Password KE was first considered by Bellovin and Merritt [4]. Foundations – formal definitions and protocols – were laid in [3, 8, 13, 9], and other works.

The use of combined keys in authentication, where the client has a password and the public key of the server, was introduced by Gong et al. [14] and first formalized by Halevi and Krawczyk [15]. Kolesnikov and Rackoff [16] extended this setting by allowing the client to also share a long key with the server, and gave first definitions of KE in their (and thus in the Gong et al. and HK) setting.

Password mistyping in KE. Despite the large research effort, the definitional issues of KE password mistyping are formally approached only in the

UC definition of Canetti et al. [9]. In their password-only setting, mistyping is modelled by Environment $\mathcal{Z}$ providing players' inputs. Additional use of long key makes our setting significantly different (and more subtle with respect to mistyping) from that of [9]. Mistyping was also considered in different settings: related-key attacks on blockciphers [2] and signing authority delegation [17].

Biometric authentication and fuzzy extractors. A growing body of work, e.g. [12, 5, 10, 11], addresses the use of biometrics in cryptography. Boyen et al. [7, 6, 10] consider its application to KE. They introduce the notion of *robust fuzzy extractor* (RFE), and give generic constructions of biometric-based KE from RFE. While their setting is similar to ours, the problems solved by [7, 6, 10] are different. They give KE protocols that accept "close enough" secrets, thus enabling security and privacy of biometric authentication. They do not aim to give a formal KE definition that handles biometric/password misreading. Moreover, as shown in Sect. 2, their notion of RFE is insufficiently strong to guarantee security of their generic KE protocol in many practical settings. (However, instantiating their KE protocol with their RFE construction is secure, since the latter satisfies stronger requirements than required by the definition.)

2 Mistyping-related limitations in previous work

On robust fuzzy extractor (RFE) definition and KE protocol [7, 6, 10].

We first clarify underlying biometric technology limitations and assumptions. Biometrics are "fuzzy", i.e. each scan is likely to be different from, but "close" to the "true" scan. Error-correction [12] is then used to extract non-fuzzy keys usable in cryptography. However, error-correction cannot correct many misreading errors (up to 10%), since this would imply high false acceptance rate[3]. Thus misreading beyond error-correction ball occurs often, and must be considered.

We note a limitation of RFE definition [7, 6, 10], prohibiting its use with the generic KE construction (Sect. 3.3 of [7]) in many scenarios. Roughly, definition's domains of correctness and security guarantees coincide. That is, extracted randomness is only guaranteed to be good if the scan is within the *error-correction* distance t from the original. There are no guarantees on the randomness if this condition does not hold. This is, perhaps, due to the papers' implicit assumption that "natural" misreadings are almost always "close" and are corrected (i.e. FRR is negligible). However, as discussed above, this assumption often does not hold. Strengthening the randomness guarantees of RFE would increase its usability.

More specifically, a RFE (Gen, Rep) may exhibit the following vulnerability. Given the public helper string P, if the biometric w_0 is misread in a special way w' outside the error-correction ball, the extracted randomness $Rep(w', P)$ is predictable. Even more subtly, $Rep(w', P)$ and $Rep(w_0, P)$ could be related, but unequal. Clearly, KE protocols, including one of Sect. 3.3 of [7, 6], constructed from such RFE would not be secure. One solution is to require, for w' outside the

[3] In balanced optimized real-life systems, which compare scans directly, False Reject Rate (FRR) is usually 1..10%. Notably, NIST reports FRR of fingerprints 0.1..2%, iris 0.2..1% and face 10%. See [1] for comprehensive overview and references.

error-correction ball, that either $Rep(w', P) = \perp$ (property of RFE construction of [7, 6]) or that $Rep(w', P)$ is either equal to or independent from $Rep(w_0, P)$.

Finally, although [7, 6, 10] consider adversarial substitution of P with P', they guarantee $Rep(w', P') = \perp$ only for w' in the error-correction ball. This vulnerability also can be resolved by separating the error-correction and security domains. We defer detailed definition, analysis and constructions as future work.

On the definition and construction of [16]. We present the following practical outside-of-the-model mistyping attack on the protocol (and thus also on the definition) of Kolesnikov and Rackoff [16]. Specifically, resistance to Denial of Access (DoA) attacks of the protocol of [16] is compromised if the honest client ever mistypes. Indeed, since their protocol is not challenge-response, client C's message can be replayed. This is not a problem if C always types the correct password (session keys of C and server S will be independent). However, if the password was mistyped, both the original and replayed message will cause S to register password failure, violating the intent of the DoA resistance. We stress that the KR protocol is otherwise secure against mistyping (and we prove it in Sect. 6). Our definitions and protocols address and correct the above insecurity.

Above limitations show subtleties of mistyping and the need to address them.

3 Pre-definition discussion

Our main contribution is a formal treatment of mistyping in the combined keys KE setting of Kolesnikov and Rackoff [16]. The KR setting is a generalization of the Halevi-Krawczyk setting [15], in which clients have a password and the public key of S. In KR setting, clients carry stealable cards capable of storing cryptographic keys – public key of S and long key ℓ shared by C and S. Addition of the cards allows better functionality and security than that of HK. KR definitions and protocol guarantee and achieve strong security when C's card is secure, and weaker, password-grade, security, when the card is compromised.

We stress that the definition of KR does not handle mistyping. That is, it is possible to construct KR-secure protocols that "break" if the client ever mistypes his password. Sect. 3.3 of [16] provides an example and a short informal discussion on mistyping, and leaves the problem open. In Sect. 3.2, we expand this discussion, present more subtle mistyping threats, and discuss approaches to handling them. This leads to the presentation of our definitions in Sect. 4.

Notation. We concentrate on the two-factor authentication setting, where a client (denoted C) exchanges keys with a server (S). Both long and short keys are used for KE. Let P be a player. We denote by P_i the i-th instance of P. We write P_i^Q to emphasize that P_i intends to do KE with (some instance of) player Q. Denote the adversary by Adv. Sometimes we distinguish the game and real-life adversary, and denote the latter Adv_{Real}. Denote C's password by pwd and long key by ℓ. S's public/ private keys are pk_S and sk_S. Password failure and the associated control symbol output by S is denoted by P$\perp$.

On the Style of Definition. We chose the game (Bellare-Pointcheval-Rogaway [3]) style, since this allowed using the intuitive definition of KR (only existing two-factor-authentication KE definition). Extending KR allowed reduction of

security claims of our definition/setting to those of KR. Further, the stronger and arguably more intuitive UC model unfortunately is sometimes too strict, ruling out some efficient protocols which appear to be good enough in practice.

Proposing a simulation-based (especially, UC) definition, and exploring the relationship between it and our definition would add confidence in both our and the UC treatment of the problem. We thus leave as an important next step the design, detailed analysis and comparison of a corresponding UC definition. We expect that our discussions of ideas and obstacles would aid in this future work.

3.1 Review of the framework of [16]

Our definition is an extension of the KE definition of KR (Def. 2 of [16]).

Recall, KR (and thus our) definition follows the common game-based paradigm. The real world and real adversary Adv_{Real} are abstracted as a game, played by the game adversary Adv. Game includes clients and servers – Interactive Turing Machines (ITM) running the KE protocol Π, communicating via channels controlled by Adv. Game rules mimic reality, and are designed so that Adv's wins correspond to real-life breaks. Π is defined secure if no polytime Adv is able to win above certain "allowed" probability. Definition is thus reduced to the design of the game. KR break down the real world into five intuitive games (KE_1, KE_2, KE_3, DOA and SID), which mimic possible real-life attack scenarios.

Game KE_1 is the core of the definition; it addresses password security when the long key is compromised. The difficulty of KE_1 design is in balancing the power given to Adv, since Adv_{Real}'s non-negligible advantage must be accounted exactly. It is achieved by "charging" Adv for each active attack (i.e. P$\perp$ output by S). The allowed Adv win probability is a function of the number of charges.

KE_2 models Adv_{Real} posing as S to C. KE_3 models KE with uncompromised card. In both cases, Adv is allowed only negligible success, which is easy to model. DOA models a "denial of access" attack formalized by KR, which requires that Adv is not able to cut C's access to S by exhausting allowed password failures. Finally, SID is a game preventing technicality-based insecure protocols.

We stress that a good model need not mimic the world *exactly*. E.g., Adv's ability to mistype or to know whether S failed may be different from Adv_{Real}'s, as long as Adv can win in *some* way (only) against bad protocols.

Mistyping in KR definitions. In KR games, client ITMs are always instantiated with correct password, which limits Adv's ability to emulate mistyping. Many real-life attacks that exploit mistyping cannot be carried in the game, allowing vulnerable protocol to withstand Adv's attacks and be defined secure. In Sect. 3.2, we discuss vulnerabilities, some natural "fixes" and their limitations.

3.2 Natural definitional approaches to mistyping (that don't work)

To better expose subtle definitional issues and the limitations of some natural approaches, we build presentation incrementally. We propose several mistyping-vulnerable protocols, each progressively more "tricky", and show that they are KR-secure. We then discuss corresponding natural "fixes" of the KR definition – ways of allowing Adv to modify or substitute client's password, so as to mimic

real-life mistyping and allow *Adv* to carry the real-world attacks. We show that ultimately they are insufficient and conclude that, for technical reasons, direct mimicking of mistyping in the games does not result in a good model. For readability, we keep discussion brief and informal (but readily formalizeable).

Mistyping vulnerabilities by example. Let Π be a KR-secure KE protocol. Π_1, Π_2, Π_3 below are KR-secure, but fail in progressively more subtle ways.

Π_1 *(S leaks long key upon mistyping).* Let Π_1 be a protocol as Π, except that in Π_1 S reveals the long key ℓ in a message, once password failure P$\perp$ occurred.

Clearly, Π_1 is "bad". But, it is easy to see that Π_1 is secure by KR definition. Since instances of C never mistype in the game, KR *Adv* cannot cause P$\perp$ without possession of ℓ. Thus, *Adv* cannot gain from S revealing ℓ, and Π_1 is KR-secure.

Π_2 *(S leaks password upon repeated mistyping).* Let pwd be C's password.Let Π_2 be a protocol as Π, except that in Π_2, S reveals pwd once $pwd+1$ was tried twice. (Limited global state can be communicated among instances of S with the help of *Adv*, thus allowing Π_2 [16]; see full version for detailed discussion.)

At the first glance, it may appear that Π_2 is "good". Indeed, the advantage *Adv* gets from causing the leak is canceled by the effort to obtain it – a redundant password attempt for each attempt of causing the leak (this is the reason why Π_2 is KR-secure). However, this leak can be caused by real-life honest C mistakenly entering $pwd+1$ twice. This is not an unusual situation, and the resulting password compromise is clearly unacceptable.

Π_3 *(S leaks a small hint about a password upon repeated mistyping).*Let pwd be C's password. Let Π_3 be a protocol as Π, except that in Π_3, S reveals whether $pwd = 0$ once $pwd+1$ was tried 4 times. Π_3 is bad for the same reason as Π_2.

Definitional approaches. We consider strengthening *Adv* of KR by mimicking powers of real-life adversary. Our goal is to disallow above "bad" protocols.

Allowing *Adv* to specify the password of C's instances disqualifies Π_1.Indeed, *Adv* wins the game where he is not given ℓ, as follows. He instantiates C with a wrong password, causing P$\perp$ and leak of ℓ, which *Adv* uses to win.

To disqualify Π_2, *Adv* needs more than simple substitution of C's password. *Adv* needs the power to specify a "mistyping function" applied to the password given to C (idea also considered in [17]). That is, *Adv* specifies a map $F : D \mapsto D$, and C is instantiated with password $F(p)$. (Not every map F is allowed [16].)

While Π_3 is bad for the same reason as Π_2 (real-life C's mistyping leaks a password hint), it is harder to disqualify Π_3 due to the small size of the leak. It turns out that Π_3 is an important example, showing that allowing *Adv* to influence C's input is insufficient. We continue this discussion below in Sect. 4.

4 Mistyping-secure KE definition

Π_3, the last example of Sect. 3.2 is a (otherwise secure) protocol where S leaks a small password hint after four certain *repeated* mistypings. A repeated mistyping does not help *Adv* (he is checking already checked password). Since in KR definition, *Adv* is charged for each (even repeated) mistyping, the cost of mistyping outweighs the benefit of the leak, and *Adv* is not able to exploit the vulnerability.

This leads to our main idea – to allow *Adv* to run mistyped KE executions "for free". This way, *Adv* will be able to win whenever a non-negligible amount of information is leaked due to mistyping. It turns out that this additional power, applied properly, results in a good (i.e. sufficiently, but not too strong, and easy to use) definition, presented in this section.

Our extension of KR definition. We would like to give *Adv* the ability to observe and actively participate "for free" in mistyped KE sessions. This is not possible with the approaches we previously discussed, including that of [16]. This is because there *Adv* always learns whether S accepted the password, allowing *Adv* to verify a password guess, for which *Adv* must be charged. Our idea is to withhold failure information from *Adv* (and not charge him in case of P$\perp$) by default, thus allowing "free" mistypings. If *Adv* wants to obtain failure information, it is given to him upon special "check" request. Since this gives him information about the password, he is charged one attempt, if the check reveals P$\perp$. Note, this cost structure is a simple generalization of the one used in [16]. This amendment of KR is sufficient to handle mistyping.

Another advantage of this approach is allowing to mimic mistyping without *Adv* creating instances with substituted password. Indeed, *Adv* can make a password guess, and, based on it, emulate any mistyping sequence of C. As shown in Sect. 4.1, this guarantees security, since a "free" mistyping-dependent leak would confirm *Adv*'s guess, allowing him to win. On the other hand, C's input substitution, especially using a mistyping map, is technically complex, and makes the definition less usable, since proofs would have to consider all such maps.

We now present our definition. Let n be a security parameter, and $D = \{0,1\}^m$ is the password domain. (In general, m can be a function of n; interesting cases are when m is constant or logarithmic in n.) All players (*Adv*, C, S) are p.p.t. machines. As does [16], we use session IDs (SID) to partner instances of players, and impose the following correctness requirement. In the absence of adversary, all sessions terminate and intended parties output same *sid* and key.

Definition 1. *We say that an instance C_i^S of a client C and an instance S_j^C of a server S are* partners*, if they have output the same session id sid.*

We start by presenting KE games, which model attacks of a real-life adversary Adv_{Real}. The first game models the setting where Adv_{Real} obtained C's long key, is attacking a server, and is allowed a limited number of password tries.

Game $\mathbf{KE}_1$. *Adv deterministically chooses active attack threshold $q \in 1..|D|$ (based on security parameter n) and creates an (honest) server S. Adv chooses S's name; then S's public/private keys are set up, and the public key revealed to Adv. Adv then runs players by executing steps 1-7 multiple times, in any order:*

1. *Adv creates an honest client C. Adv is allowed to pick any unused name for the client; the client C is registered with S, and long key ℓ and password pwd are set up and associated with C. Only one honest client can be created. Adv is given the long key ℓ, but not pwd.*
2. *Adv creates a corrupt client B^i. Adv is allowed to initialize him in any way, choosing any unused name, long key and password for him.*

3. *Adv creates an instance C_i of the honest client C. C_i is given (secretly from Adv) as input: his name C, the partner server's name S, the public key of S, the long key and the password of C.*
4. *Adv creates an instance S_j of the honest server S. S_j is given (secretly from Adv) as input: his name S, the private key of S, his partner's name (C or B^i) and that client's long key and password.*
5. *Adv delivers a message m to an honest party instance. The instance immediately responds with a reply (by giving it to Adv) and/or, terminates and outputs the result (a sid and either the session key, the* failure symbol $\perp$*, or, in case of the server instance, the* password failure symbol $P\perp$*) according to the protocol. Adv learns only the sid part of the output.*
6. *Adv "checks" any completed honest instance – then he is notified whether the instance output $P\perp$, $\perp$, or a session key. Adv gets charged one attempt, if he checked S^C and it output $P\perp$.*
   When Adv accumulates q charges, he becomes restricted – he can neither deliver messages to any instances S_j^C nor check any instances.
7. *Adv "opens" any successfully completed and checked honest instance – then he is given the session key output of that instance.*

Then Adv asks for a challenge on an instance S_j^C of the server S. S_j^C, who has been instantiated to talk to the honest client C, must have completed, been checked by Adv, and output a session key. The challenge is, equiprobably, either the key output by S_j^C or a random string of the same length. Adv must not have opened S_j^C or a partner of S_j^C, and is not allowed to do it in the future.

Then Adv continues to run the game as before (execute steps 2-7). Finally, Adv outputs a single bit b which denotes Adv's guess at whether the challenge string was random. Adv wins if he makes a correct guess, and loses otherwise. Adv cannot "withdraw" from a challenge, and must produce his guess.

Note that we handle *sid* differently from [16]. Here we insist that parties always output *sid*, while previously *sid* was only output if a party did not fail. We need this change, since KE_1's interface needs to be the same for cases when an instance failed and did not fail. Outputting a *sid* only if KE succeeded (and letting it known to *Adv* for free) helps *Adv* determine whether $\text{P}\perp$ occurred.

In all other KE games (KE_2, KE_3, SID and DOA) below, password mistyping and even the knowledge of *pwd* should not help *Adv*. We thus choose to reveal the password to *Adv* and remove restrictions on the number of $\text{P}\perp$'s (thus removing the definition of q). We also allow *Adv* to specify C_i's password at its instantiations. These games are presented by modifying the above KE_1. All of the above four modifications are included in all games below.

KE_2 models the setting where *Adv* stole C's *pwd* and ℓ, but is attacking C.

Game KE_2 *is derived from KE_1 as noted in the previous paragraphs; further, Adv is given ℓ and must challenge an honest client instance C_i^S.*

KE_3 models the setting where *Adv* only stole C's *pwd*, and is attacking S.

Game KE_3 *derived from KE_1 as noted above, but Adv is not given ℓ.*

SID enforces non-triviality, preventing improper partnering (e.g. players unnecessarily outputting same *sid*). Recall, *Adv* is not allowed to challenge parties whose partner has been opened; SID ensures that *Adv* is not unfairly restricted.

Game SID *is derived from KE_1 as noted above; further, Adv does not ask for (nor answers) the challenge. Adv wins if any two honest partners output different session keys.*

Note, SID allows for one (or both) of the partners to output a failure symbol. *Adv* only wins if two successfully completed parties output different session keys.

Finally, game DOA models resistance to the Denial of Access (DoA) attacks. This game prevents vulnerabilities due to mistyping (see Sect. 4.1).

Game DOA *is derived from KE_1 as noted above; further, Adv does not ask for (nor answers) the challenge. Adv wins if the number of $P\perp$'s is greater than the number of client instances where he substituted the password.*

Definition 2. *We say that a key exchange protocol Π is* secure in the Combined Keys model with mistyping*, if for every polytime adversaries Adv_1, Adv_2, Adv_3, Adv_{sid} and Adv_{doa} playing games KE_1, KE_2, KE_3, SID and DOA, their probabilities of winning (over the randomness used by the adversaries, all players and generation algorithms) is at most only negligibly (in n) better than:*

- $1/2 + \frac{q}{2|D|}$*, for KE_1,*
- $1/2$*, for KE_2 and KE_3,*
- 0*, for SID and DOA.*

The definition for the HK setting (where C does not have ℓ) is extracted from Def. 2 by removing all uses of ℓ and the games where *Adv* doesn't know ℓ.

4.1 Why this is a good definition

First, since *Adv* is not weaker than *Adv* of [16], Def. 2 enforces basic security properties of the protocols. We additionally need to argue that the definition is not too strict and that it prevents mistyping-caused leaks in protocols. The former property is intuitive, and we support it by proposing an efficient protocol and proving its security w.r.t. Def. 2 (Sect. 6). The latter property, on the other hand, requires significantly more careful consideration, presented in this section.

Note, KE_1 is the only game where we need to be careful with not giving *Adv* too much power w.r.t. mistyping. In other games, unlimited ability of *Adv* to substitute C's input should not help him win against a secure protocol. At the same time, such *Adv* directly models real-life adversary. Therefore, this simple allowance resolves mistyping problems w.r.t. other games we consider.

KE_1 is the core of the definition, and most of the definitional subtleties appear in KE_1. We start with the discussion of the details and ideas about this game.

Why KE_1 is a good model. Often, when a definition is proposed, a proof is provided, demonstrating the relationship between the new and previous definitions. This adds confidence in the proposed definition. We introduce the first definition in our setting; thus there is no previous definition to relate it to.

Our approach. Instead, we prove that if a protocol Π is secure by Def. 2, Adv of the game KE_1 cannot tell the difference between the following two executions, if he is not allowed to see the outputs of S. In one execution, selected (by Adv) client instances are instantiated with a mistyping sequence Adv chooses, and in the other they are instantiated with the password pwd of C. We stress that Adv is active during these executions; he can perform (almost) all the actions Adv of KE_1 can. This provides an informal "reduction" to the definition of [16], in the following sense. Assume the definition of [16] is "good", i.e. accurately identifies insecure protocols in its "no-mistyping" model. Then Def. 2 is "good" in the general setting, where clients are allowed to mistype.

Indeed, suppose Π is "bad". Due to the indistinguishability of the above executions, anything that Π leaks due to mistyping can also be seen and exploited without mistyping by Adv of KE_1 of [16]. Then Π will be insecure by definition of [16], since, by assumption, it is a good definition. Since KE_1 Adv of Def. 2 is at least as strong as that of [16], Π will also be insecure by Def. 2. From another angle, if active Adv cannot distinguish the above executions, then he is not learning anything from the mistypings, other than what may be inferred from the corresponding sequence of P$\perp$'s, but the latter is unavoidable anyway.

This reduction is informal, and serves only as evidence that our definition is good. By the nature of definitional work, it is not possible to "prove" definitions.

Formal theorem statement and proof of indistinguishability of the above executions is in full version. Proof idea is that some passwords used in the mistyped execution must be unequal to C's pwd. Ability to distinguish executions gives a free hint of what pwd is not, allowing corresponding KE_1 Adv to win.

On DoA protection. As mentioned in Sect. 2, the definition of [16] does not model (and fails to guarantee) DoA resistance when honest users mistype. We need that a replayed client's flow must not cause S output P$\perp$. Therefore, C must send at least one message that is dependent on S's message. Thus, the one-round, two-independent-flow protocols are not possible if DoA is desired.

We change the DOA game accordingly. Adv knows pwd, and is now allowed to instantiate clients with passwords of his choice. Adv wins DOA, if the number of P$\perp$ is greater than the number of client instances with substituted password.

5 Application to biometric authentication

We note that our definitions and protocols are directly applicable to biometric-based authentication. For example, fuzzy extractors [11] can be naturally used in our two-factor authentication setting, as follows. The storage card now additionally contains the public data pub_C of C's biometric b_C. The (potentially short) randomness extracted from b_C plays the role of the password. To authenticate, C first reconstructs the password using extractor's recovery procedure $Rec(pub_C, b'_C)$, and then uses it as prescribed by a KE protocol. Misreading b'_C of b_C can cause variety in the output of Rec and thus effect mistypings in the protocol. Still, our definitions (in-particular, mistyping-security property) and properties of fuzzy extractors guarantee security of this construction, even if

Adv captured the card with the long key and pub_C. (In the HK setting, where C only has pk_S, we also can use our definition and above protocol – but pub_C is now sent by S to C authenticated by S's signature, as part of the protocol.)

However, we note that our definitions do not handle the general case, where b_C is used directly as input to C. That is, S knows "acceptance set" of C (AS_C), and accepts if C's submitted password/biometric $b_C \in \mathrm{AS}_C$. We anticipate that a natural extension of our definition would handle this case. In particular, the correctness requirement should be amended w.r.t. AS_C, and *Adv*'s allowed success rate may be dependent on AS as well. We leave this definition as future work, to be performed either as extension of our definition, or in the UC framework.

6 Mistyping-Secure KE Protocols

WLOG, assume protocol messages are formed properly (i.e. values drawn from appropriate domains, etc.). Let n be a security parameter, $E = (Gen, Enc, Dec)$ be a CCA2 secure public key encryption scheme, $F : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n$ be a PRFG, and $MAC : \{0,1\}^n \times \{0,1\}^* \mapsto \{0,1\}^n$ be a message authentication code. Let $N_C \in \{0,1\}^n$ be the name of client C. (Shorter names may be used.)

Although KR definitions do not handle mistyping, their protocol resists all mistyping-related attacks, except for (perhaps, unimportant in some settings) DoA resistance. We first prove this fact. Constr. 1 is the protocol of [16], only with updated handling of *sid*, to satisfy the syntactic requirements of Def. 2.

Construction 1 *(KE with mistyping, no DoA resistance [16])*

S^C		C^S
choose $r \in_R \{0,1\}^n$		*choose* $k \in_R \{0,1\}^n$,
		set α=$Enc_{pk_S}(N_C$,pwd,$k)$
	$r \rightarrow \cdots \leftarrow \alpha$,$MAC_\ell(\alpha)$	
set $sid = (r, \alpha)$,		*set* $sid = (r, \alpha)$,
verify $MAC_\ell(\alpha)$ *and* N_C*;*		*output*
if fail, output $(sid,\perp)$*,halt*		$(sid, K = F_k(r))$
verify pwd;		
if fail, output $(sid,P\perp)$*,halt*		
else output $(sid, K = F_k(r))$		

Theorem 1. *Constr. 1 satisfies Def. 2, except for the success rate in game DoA.*

We now present a fully secure protocol in our model, derived from Constr. 1.

Construction 2 *is a challenge-response version of Constr. 1, where* C^S *replies with* $(\alpha, MAC_\ell(r, \alpha))$ *to message* r*.*

Theorem 2. *Constr. 2 is secure by Def. 2.*

We note that Constr. 2 can be modified to allow S to send confirmation to C whether he accepted, failed or password-failed. See full version for details.

Proofs of security of Theorems 1 and 2 are presented in the full version.

Achnowledgements: We thank Shai Halevi, Hugo Krawczyk, and anonymous referees for valuable comments.

References

1. http://en.wikipedia.org/wiki/Biometrics#Performance,Retrieved 02/10/08.
2. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506, 2003.
3. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, 2000.
4. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secureagainst dictionary attacks. In *SP '92: Proceedings of the 1992 IEEE Symposium on Security and Privacy*, page 72, Washington, DC, USA, 1992. IEEE Computer Society.
5. Xavier Boyen. Reusable cryptographic fuzzy extractors. In *CCS*, pages 82–91. ACM Press, 2004.
6. Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Secure remote authentication using biometric data (revised version). Available at http://www.cs.stanford.edu/~xb/eurocrypt05b/.
7. Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Secure remote authentication using biometric data. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 147–163, 2005.
8. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-hellman. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer, 2000.
9. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, 2005.
10. Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *LNCS*, pages 147–163, 2006.
11. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Cryptology ePrint Archive, Report 2003/235, 2003. http://eprint.iacr.org/.
12. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540, 2004.
13. Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 408–432, London, UK, 2001. Springer.
14. Li Gong, T. Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.
15. Shai Halevi and Hugo Krawczyk. Public-key cryptography and password protocols. *ACM Trans. Inf. Syst. Secur.*, 2(3):230–268, 1999.
16. Vladimir Kolesnikov and Charles Rackoff. Key exchange using passwords and long keys. In *Theory of Cryptography, TCC 2006*, volume 3876 of *LNCS*, pages 100–119. Springer, 2006.
17. Philip MacKenzie and Michael Reiter. Delegation of cryptographic servers for capture-resilient devices. *Distributed Computing*, 16(4):307–327, 2003.