

Notes #7 (for Lecture 9)

We now want to talk about *public-key* signature schemes. Before we do this, it will be useful to discuss different security properties for families of hash functions. Recall that a family of hash functions satisfies the privately collision resistant property defined above if, without seeing the key or anything about the function except the parameter  $n$ , it is nearly impossible to find a pair of distinct inputs that will hash to the same string. We can prove (without any assumptions) that such families exist.

We will define a hash family to be “publicly collision resistant” if, even after seeing the key of the hash function, a polynomial time adversary cannot (except with negligible probability) find a pair of distinct inputs that hash to the same string. We will define a hash family to be “weakly publicly collision resistant” if a polynomial time adversary cannot (except with negligible probability) choose one input before seeing the key of the hash function, then see the key of the hash function, then choose a second input, such that the two inputs hash to the same string.

**Definitions:** (Nonuniform adversary setting)

By a family of hash functions  $H$  we mean that for a key length  $l(n)$  (that can be computed, in unary, in time polynomial in  $n$ ), we associate with every  $l(n)$ -bit key  $k$ , a function  $H_k : \{0, 1\}^* \rightarrow \{0, 1\}^n$ ; it must be the case that given  $k$  and  $x$ ,  $H_k(x)$  can be computed in time polynomial in  $n$  and  $|x|$ . (We assume that  $l(n)$  uniquely determines  $n$ .)

- We say  $H$  is *publicly collision resistant* if the following holds for every  $\{C_n\}$ .

Let  $\{C_n\}$  be a polynomial size family of circuits, such that  $C_n$  has  $l(n)$  input bits, and such that  $C_n$  outputs two binary strings  $s$  and  $t$ ; the lengths of  $s$  and  $t$  may depend upon the input to  $C_n$ . (Note that since from a strictly syntactic point of view  $C_n$  must output a string of fixed length, we will view this syntactic output as coding for  $s$  and  $t$  in some natural way.)

Let  $p(n)$  be the probability that, if a random  $l(n)$  bit string  $k$  is chosen and given to  $C_n$ , and  $C_n$  outputs  $s$  and  $t$ , then  $s \neq t$  and  $H_k(s) = H_k(t)$ .

Then  $p(n) \leq \frac{1}{n^c}$  for each  $c$  and sufficiently large  $n$ .

- We say  $H$  is *weakly publicly collision resistant* if the following holds for every  $\{C_n, s_n\}$ .

Let  $\{s_n\}$  be a polynomial size family of strings and let  $\{C_n\}$  be a polynomial size family of circuits, such that  $C_n$  has  $l(n)$  input bits, and such that  $C_n$  outputs a binary string  $t$ ; the length of  $t$  may depend upon the input to  $C_n$ .

Let  $p(n)$  be the probability that, if a random  $l(n)$  bit string  $k$  is chosen and given to  $C_n$ , and  $C_n$  outputs  $t$ , then  $s_n \neq t$  and  $H_k(s_n) = H_k(t)$ .

Then  $p(n) \leq \frac{1}{n^c}$  for each  $c$  and sufficiently large  $n$ .

(Note that in the uniform adversary model, the adversary is given  $1^n$ , computes (probabilistically) for polynomial in  $n$  steps, outputs  $s$ , sees  $k$ , outputs  $t$ .)

We do not know how to prove the existence of publicly collision resistant hash families merely by assuming that one-way functions or pseudo-random generators exist. The most usual assumption is

the stronger assumption that “claw-free families” exist. These can be proven to exist from certain assumptions about the computational difficulty of integer factorization; the reader can consult Chapter 2 of *Goldreich* for more information on this. In practice, if one wants to choose a random function from a publicly collision resistant hash family, one just uses a fixed, “standard” function such as  $MD5 : \{0, 1\}^* \rightarrow \{0, 1\}^{128}$  or  $SHA-1 : \{0, 1\}^* \rightarrow \{0, 1\}^{160}$  or  $SHA-2(256) : \{0, 1\}^* \rightarrow \{0, 1\}^{256}$  or  $SHA-3(256) : \{0, 1\}^* \rightarrow \{0, 1\}^{256}$ . The implied belief is that the function has been chosen at random from a suitable family, even though it is not really clear how it was chosen, or why. At this time,  $MD5$  has been badly broken, and  $SHA-1$  has been somewhat broken. ( $SHA$  stands for “secure hashing algorithm”. Both  $SHA-2$  and  $SHA-3$  come in 4 versions, enabling output sizes of 224, 256, 384 or 512 bits.  $SHA-3$  is the most recent of these, having been accepted as a standard by NIST only in October, 2012. The NIST web site contains complete details of these algorithms.)

We can, however, use one-way functions to construct *weakly* publicly collision resistant hash families, which in turn can be used to construct secure public-key signature schemes. However, neither of these constructions are used in practice. Instead, one uses something like  $SHA-3(256)$  for collision-resistance, and something like the  $DSS$  (for Digital Signature Standard) for signatures.

**Theorem:** (Naor and Yung, Rompel) If one-way functions exist, then weakly publicly collision resistant hash families exist.

**Proof:** Difficult and omitted.

**Definition:** A *public-key signature scheme*  $\mathcal{S}$  consists of the following.

- A generating function  $GEN$ .  $GEN$  has as input a string  $1^n$  together with random bits, and should be computable in time polynomial in  $n$ . The output of  $GEN$  is a pairs of strings  $pub$  (a public key) and  $pri$  (a private key). We assume that the lengths of  $pub$  and  $pri$  depend only on  $n$ , and that  $n$  is determined by either of these lengths.
- A signing algorithm  $SIGN$  that has as input a key  $pri$  (generated from security parameter  $n$ ) and a message  $m \in \{0, 1\}^*$ .  $SIGN$  should be computable in time polynomial in the lengths of the inputs; we allow  $SIGN$  to be probabilistic (that is, to have random bits as input). We write  $SIGN_{pri}(m)$  for  $SIGN(pri, m)$ . The length of  $SIGN_{pri}(m)$  should depend only on the security parameter  $n$ , and not on the length of the message being signed. (Although it is no loss of generality to assume that  $|SIGN_{pri}(m)| = n$ , it will be convenient not to insist on this.)
- A verifying function  $VER$  that has as input a key  $pub$ , a message  $m$  and a supposed signature  $\sigma$ , and outputs a single bit.  $VER$  should be computable in time polynomial in the lengths of the inputs. It should be the case that for every  $n$ , and for every pair  $(pub, pri)$  that can be output by  $GEN$  on  $1^n$ , and for every message  $m$ , if  $\sigma = SIGN_{pri}(m)$ , then  $VER(pub, m, \sigma) = 1$ .

**Definition:** (Nonuniform adversary setting)

A signature scheme  $\mathcal{S}$  is *secure* if the following holds for every adversary  $A$ :

Let  $A = \{A_n\}$  be a polynomial size family of circuits.  $A_n$  has as input a string  $pub$ ;  $A_n$  creates a binary string  $m_0$  and sees an  $n$ -bit string  $\sigma_0$ ;  $A_n$  then creates a binary string  $m_1$  and sees an  $n$ -bit string  $\sigma_1$ ; this continues for some (polynomial in  $n$ ) number of stages; (if signing is probabilistic, then  $A_n$  may choose to create the same binary string more than once); at the end,  $A_n$  outputs a string  $m$  and an  $n$ -bit string  $\sigma$ , such that  $m$  is different from every  $m_i$ .

Consider the following experiment. A pair  $(pub, pri)$  is randomly generated from  $1^n$  using  $GEN$ ; then  $A_n$  is run on  $pub$ , and for each  $m_i$  that is created, we give  $\sigma_i = SIGN_{pri}(m_i)$  to  $A_n$ ; eventually

$A_n$  outputs  $m$  (different from every  $m_i$ ) and  $\sigma$ .

Let  $p(n)$  be the probability that  $VER(pub, m, \sigma) = 1$ .

Then  $p(n) \leq \frac{1}{n^c}$  for each  $c$  and sufficiently large  $n$ .

**Theorem:** (Goldwasser, Micali, Rivest) If one-way functions exist, then (deterministic) secure signature schemes exist.

**Proof:** The rather complicated construction is outlined below.

The construction proceeds in a number of stages. We will explain each stage below for security parameter  $n$ .

First, assume that we have a signature scheme  $\mathcal{S}$  which is secure with respect to the signing of messages that have length exactly  $n$ ; that is, adversaries for  $\mathcal{S}$  are only allowed to see signatures of messages of length  $n$ , and must try to forge a message of length  $n$ ; say that the algorithms of  $\mathcal{S}$  are  $GEN$ ,  $SIGN$ ,  $VER$ . We wish to construct a signature scheme  $\mathcal{S}'$  that will be secure for signing messages of arbitrary lengths.

One way we can do this is by using a publicly collision resistant hash family  $H$ . To generate a key pair for  $\mathcal{S}'$ , we generate a key pair  $(pub, pri)$  for  $\mathcal{S}$  and a key  $k$  for  $H$  (assuming security parameter  $n$ ); the public key for  $\mathcal{S}'$  will then be  $[pub, k]$  and the private key will be  $[pri, k]$ . The signature of a string  $m$  in  $\mathcal{S}'$  will be  $\sigma' = SIGN_{pri}(H_k(m))$ . We verify  $\sigma'$  in  $\mathcal{S}'$  by checking that  $VER(pub, H_k(m), \sigma')$  holds. We leave it as an exercise to prove that this is secure. The only problem with the above construction is that it assumes a publicly collision resistant hash family, and we don't know how to prove that these exist by only assuming the existence of a one-way function.

We will therefore give an alternative way of constructing  $\mathcal{S}'$  from  $\mathcal{S}$  that only uses a weakly publicly collision resistant hash family,  $H$ . We generate a key pair for  $\mathcal{S}'$  by choosing a key pair  $(pub, pri)$  for  $\mathcal{S}$ , and using the same pair for  $\mathcal{S}'$ . The signature of message  $m$  in  $\mathcal{S}'$  will be computed as follows. First we choose a random key  $k$  for  $H$  (assuming security parameter  $n$ ). The signature for  $m$  in  $\mathcal{S}'$  will then be the pair  $\sigma' = (k, SIGN_{pri}[k, H_k(m)])$ . We verify  $\sigma' = (k, \sigma)$  in  $\mathcal{S}'$  by checking that  $VER(pub, [k, H_k(m)], \sigma)$  holds. We leave the proof of security as an exercise.

Actually, we have cheated in two ways here. For one thing, we assumed that the length of messages being signed by  $\mathcal{S}$  was not  $n$ , but rather  $n$  plus the length of a key for  $H$  (on security parameter  $n$ ). This is not a problem, as the construction for  $\mathcal{S}$  (see below) can be easily modified to sign messages of this particular length. (Alternatively, one can choose  $H$  so that on security parameter  $n$ ,  $|k| + |H_k(m)| = n$ .)

A more serious problem is that the signing process we have described for  $\mathcal{S}'$  is in reality a *probabilistic* algorithm, whereas our theorem specifies that it should be deterministic. We can fix this as follows. We create from  $\mathcal{S}'$  a deterministic scheme  $\mathcal{S}''$ . We let the private key for  $\mathcal{S}''$  contain, in addition to the private key of  $\mathcal{S}'$ , an  $n$  bit seed  $s$  for a pseudo-random function generator  $F'$ , such that  $F'_s : \{0, 1\}^* \rightarrow \{0, 1\}^{l(n)}$  where  $l(n)$  is length of a key for  $H$  (on security parameter  $n$ ). Then instead of using a random  $k$  when we sign  $m$  as in  $\mathcal{S}'$ , we will use  $k = F'_s(m)$ . Again, we leave the proof of security as an exercise.

We now want to show how to create a signature scheme that is secure for signing messages of length  $n$ . (Note that not all of this construction was presented in class.)

First we construct a signature scheme  $\mathcal{S}^1$  that is only for signing a single,  $n$  bit message. That is, the adversary gets to see one  $n$  bit message of his choice signed, and then must try to forge the signature of a new  $n$  bit message. We assume we have a one-way function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ .

To generate a key pair, we choose  $2n$  random  $n$  bit strings:  $x_1^0, x_1^1, x_2^0, x_2^1, \dots, x_n^0, x_n^1$ . We then compute  $y_i^b = f(x_i^b)$  for  $b \in \{0, 1\}$  and  $1 \leq i \leq n$ . We then assign  $pub = (y_1^0, y_1^1, y_2^0, y_2^1, \dots, y_n^0, y_n^1)$  and  $pri = (x_1^0, x_1^1, x_2^0, x_2^1, \dots, x_n^0, x_n^1)$ .

To sign the  $n$  bit message  $m = b_1 b_2 \dots b_n$ , we compute  $SIGN_{pri}^1(m) = (x_1^{b_1}, x_2^{b_2}, \dots, x_n^{b_n})$ .

To verify, we do the obvious thing.  $VER^1(pub, m, \sigma) = 1$  if and only if  $\sigma$  consists of  $n$ ,  $n$ -bit strings  $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$  and  $f(\sigma_1) = y_1^{b_1}, f(\sigma_2) = y_2^{b_2}, \dots, f(\sigma_n) = y_n^{b_n}$ . It is not hard to show that  $\mathcal{S}^1$  is secure in the desired sense.

We will now construct a signature scheme  $\mathcal{S}^2$  that is secure for signing (any number of)  $n$ -bit messages. We assume we have the above scheme  $\mathcal{S}^1$ , as well as a pseudo-random function generator  $F$  and a weak publicly collision resistant family  $H$ .

We generate a key pair as follows. First we choose a random  $n$ -bit seed  $s$  for  $F$ ; then we choose a key pair  $(pub, pri)$  for  $\mathcal{S}^1$ ; then we choose a random key  $k$  for  $H$ . The public key for  $\mathcal{S}^2$  will then consist of  $pub$  and  $k$ ; the private key will consist of  $s, pri$  and  $k$ .

Signing an  $n$  bit message will be complicated. First, imagine the depth  $n+1$  binary tree, where we identify each node with a binary string of length  $\leq n+1$ ; the root is  $\lambda$ , the empty string; the children of  $\alpha$  are  $\alpha 0$  and  $\alpha 1$ . We want to identify with each node  $\alpha$  a pair  $(pub_\alpha, pri_\alpha)$ . We do this by setting  $pub_\lambda = pub$  and  $pri_\lambda = pri$ . For the other nodes  $\alpha$ , we would like to generate values *randomly*, using the generating algorithm for  $\mathcal{S}^1$ . Instead we generate them *pseudo-randomly* using  $F_s$ . We assume that  $F_s(\alpha)$  is long enough, and we use  $F_s(\alpha)$  as the random bits needed to generate  $(pub_\alpha, pri_\alpha)$  for  $\mathcal{S}^1$ . The idea is that the message to be signed will determine a path through the tree, and we will sign the message by giving a chain of signatures, at each node in the path signing a hash of the public information at the two nodes beneath it.

For each  $\alpha$  of length  $\leq n$ , let  $\sigma_\alpha = pub_{\alpha 0}, pub_{\alpha 1} SIGN_{pri_\alpha}^1(H_k[pub_{\alpha 0}, pub_{\alpha 1}])$ . If we want to sign an  $n$  bit message  $m$  in  $\mathcal{S}^2$ , let  $m_i$  be the  $i$  bit prefix of  $m$ , for  $0 \leq i \leq n$ ; the signature of  $m$  is defined to be the sequence  $\sigma_{m_0}, \sigma_{m_1}, \dots, \sigma_{m_n}$ . The Verification algorithm works in the obvious way.

We will now informally discuss why  $\mathcal{S}^2$  is secure.

Imagine that the values at each node were generated using randomly, rather than pseudo-randomly, generated bits. Each node is used in exactly one way, namely to sign (using  $\mathcal{S}^1$ ) the public information of its children; the information at each node is randomly generated, and only used to sign (in  $\mathcal{S}^1$ ) exactly one message, although that same signature may appear in the signatures for many different  $m$  in  $\mathcal{S}^2$ . Therefore, because of the security of  $\mathcal{S}^1$ , to forge a signature for a new message in  $\mathcal{S}^2$ , it will be necessary, for some  $\alpha$  of length at most  $n$ , to compute  $[pub'_{\alpha 0}, pub'_{\alpha 1}] \neq [pub_{\alpha 0}, pub_{\alpha 1}]$  such that  $H_k[pub'_{\alpha 0}, pub'_{\alpha 1}] = H_k[pub_{\alpha 0}, pub_{\alpha 1}]$ . Note that  $[pub_{\alpha 0}, pub_{\alpha 1}]$  is generated completely independently of  $k$ . Therefore, an algorithm for finding such a  $[pub'_{\alpha 0}, pub'_{\alpha 1}]$  would break the weak public collision resistance of  $H$ .  $\square$