G22.3033-003 Introduction to Cryptography November 29^{th}, 2001

Lecture 12

Lecturer: Yevgeniy Dodis *Scribe: Roberto Oliveira*

This lecture is on *public-key signature schemes (*PKS*)*, which are the public-key counterparts of the *message authentication codes (*MAC*)* that we studied in our previous lecture. After a high-level discussion on signatures and some basic definitions, we try to argue through somewhat natural examples (trapdoor signatures) that even though it is rather easy to devise somewhat secure signature schemes, the degree of security that we expect to achieve is not that easy to attain. That task is hard enough to make the "signature paradox" we discuss seem believable, but as it turns out, it is false. With our hope renewed, we then look at a "one-time secure" signature scheme, and show how to amplify it (and other one-time secure schemes) to arbitrary length secure signatures, based on a natural idea that doesn't quite work but that can be fixed by a new kind of hash function that we introduce in the last section.

1 Introduction

1.1 Motivation and intuition

The use of signatures as a form of authentication in the "real world" is very old and widespread. It is based on the assumption that it is very hard to emulate one's handwriting well enough or to modify a document so that differences cannot be detected. If one accepts that, signing is then a very efficient way of ensuring that a given public document bears one's approval.

"Physical" signatures must be reproducible by the signer (that is, every person must have a definite procedure for signing as often as needed) and recognizable by others (also by means of some definite procedure) . Their usefulness comes from the fact that lots of entities (e.g. the government, banks, credit card companies, family and friends) can recognize what one's signature looks like (i.e. one's signature is known by the *public*) and yet they cannot forge it.

In this class we discuss the computational counterpart of "physical" signatures, which are called *digital signatures*. Those are intended to provide the *sender* with the means to authenticate his/her *messages* (here understood to be any information he/she intends to make available to the world) in a way that *can be checked by anyone* but that *cannot be copied by others*.

The message authentication codes (MAC) that we studied last class do not quite fit into the niche of digital signatures. For they are *secret-key authentication schemes* and implicit in that concept is that all parties that share the secret information (which is necessary for authenticity verification) must be trustworthy if one does not want to lose all hope for security. It is then clear that anything that deserves the name *digital signature* should be a *public-key authentication scheme*: even people in which one does not trust completely should be able to check the authenticity of one's signature. That is, one does not want

to impose any restrictions on the parties that may want to verify the authenticity of one's signature. Of course, the signing algorithm must use secret information (that is, a *secret key*), which roughly corresponds to one's unique way of signing.

In the following subsection we define digital signatures and their security taking all those previous considerations into account. Subsequently, we shall attempt to build something that satisfies these definitions. This second task will not be easy, and we will find it necessary to settle for various intermediary objects on our way to *secure public-key signature schemes.*

1.2 Basic Definitions

In this subsection we define the notion of a *public-key signature scheme* as a public-key analog of MAC and then present a definition of security for it. $\mathcal{M}$ is the message space (e.g. $\mathcal{M} = \{0,1\}^k$ or $\mathcal{M} = \{0,1\}^*$).

Definition 1 (Public-Key Signature Scheme) *A* **Public-Key Signature Scheme** (PKS) *is a triple* (Gen,Sign,Ver) *of* PPT *algorithms:*

a) The key generating algorithm Gen *outputs the secret (private) and verification (public) keys:* $(SK, VK) \leftarrow \mathsf{Gen}(1^k)$.

b) The message signing algorithm Sign *is used to produce a signature for a given message:* $\sigma \leftarrow \mathsf{Sign}_{SK}(m)$, *for any* $m \in \mathcal{M}$.

c) The signature verification algorithm checks the correctness of the signature: $\mathsf{Ver}_{VK}(m, \sigma) \in \{accept, reject\}$

The correctness property must hold: $\forall m, \quad \mathsf{Ver}_{VK}(\mathsf{Sign}_{SK}(m)) = accept.$

Remark 2 *One can also consider* **stateful** PKS*; we shall encounter those towards the end of the lecture.*

Remark 3 *We shall adopt the convention that VK is a substring appended at the end of SK (i.e. SK contains in VK) whenever this is useful. Of course, this entails no loss of generality.*

As in the case of a MAC, we can consider the notions of existential or universal unforgeability against VK-only, random-message or chosen-message attacks. To assume that an adversary might be able to query the receiver of the messages to check the validity of given pairs (m, σ) makes no difference in the present case, as long as the adversary has the public key, he can test that by himself. Therefore, the natural counterpart of the standard notion of security for MAC in the present context is:

Definition 4 (Standard notion of security for PKS**)** *A* PKS $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *is said to be secure, that is, existentially unforgeable against chosen-message attack* (CMA) *if for all* PPT *A we have that*

$$\Pr(\mathsf{Ver}(m, \sigma) = accept \mid (SK, VK) \leftarrow \mathsf{Gen}(1^k), (m, \sigma) \leftarrow A^{\mathsf{Sign}_{SK}}(VK)) \leq \mathsf{negl}(k)$$

where A cannot query the oracle Sign_{SK} *on the message string m it outputs.*

2 Examples and problems

The purpose of this section is twofold. First, we show how signature schemes that are somewhat secure can be designed using trapdoor permutations. Second, by showing that these schemes fail to meet the security standards we set for signatures, we give evidence that designing secure signatures is hard, if possible at all. We also present a convincing but thankfully fallacious "proof" of the non-existence of secure signature schemes.

2.1 Examples: trapdoor signature schemes

The subject of this section is a smart way of building signature schemes from trapdoor permutations.

RSA Signature. The idea behind this scheme is the following. We use the *inverse* of the RSA function to produce the signature $\sigma = \mathsf{RSA}^{-1}(m)$ from the message m, and those interested in checking it just compute $\mathsf{RSA}(\sigma)$ and compare it with m. More precisely (using the notation in the definition of PKS):

a) $SK = (p, q, d)$ and $VK = (n, e)$ where p, q are random k-bit primes, $n = pq$ is the RSA modulus, $e \in \mathbb{Z}^*_{\varphi(n)}$ is the RSA exponent and $d = e^{-1} \bmod \varphi(n)$.

b) $\mathsf{Sign}_{SK}(m) = m^d \bmod n$ (where $m \in \mathbb{Z}^*_n$).

c) $\mathsf{Ver}_{VK}(\sigma) = [\sigma^e = m \bmod n]$ (where $\sigma \in \mathbb{Z}^*_n$).

A VK-only attack cannot result in universal forgery with non-negligible probability under the RSA assumption though the following reasoning. Because a successful universal forgery would enable one to find $\mathsf{RSA}^{-1}(m)$ of any (and, thus, of a random) m with probability ε, contrary to the RSA assumption. This implies that under the RSA assumption this scheme is universally unforgeable against VK-only attack.

Rabin Signature. The idea used in the previous scheme is again used, only substituting the modular squaring function for RSA. That is,

a) $SK = (p, q, g, h)$ and $VK = n$ where p, q are random k-bit primes, g generates $\mathbb{Z}_p$, h generates $\mathbb{Z}_q$ and $n = pq$ is the modulus.

b) $\mathsf{Sign}_{SK}(m) = \sqrt{m} \in \mathbb{Z}^*_n$ (where $m \in \mathbb{Z}^*_n$ and any of the four square roots will do).

c) $\mathsf{Ver}_{VK}(\sigma) = [\sigma^2 = m]$ $(\sigma \in \mathbb{Z}^*_n)$.

The same argument given for RSA proves that this PKS is universally unforgeable against VK-only attacks under the factoring assumption, and therefore even more believable than RSA.

Signature based on any Trapdoor Function. It turns out that the above constructions can be generalized for an arbitrary trapdoor function f with trapdoor information t (technically, a *family* of trapdoor functions with an efficient generation algorithm for (f,t))

a) $SK = (f,t)$ and $VK = (f)$.

b) $Sign_{SK}(m) = f^{-1}(m)$, computed using t.

c) $Ver_{VK}(\sigma) = [f(\sigma) = m]$.

The previous two examples can be easily put into that framework. We now present a general theorem on the unforgeability of trapdoor signatures.

Theorem 5 ((In)Security of Trapdoor Signatures against VK-only attack) *If f is a trapdoor family, then the corresponding trapdoor signature scheme is universally unforgeable against VK-only attack, but existentially forgeable against VK-only attack.*

Proof: We prove the first assertion by contradiction. Suppose that f is a trapdoor but the corresponding signature scheme does not have the desired property. That means that there exists some PPT A such that with non-negligible probability $\varepsilon = \varepsilon(k)$

$$\Pr(\mathsf{Sign}_{SK}(m) = \sigma \mid (SK, VK) \leftarrow \mathsf{Gen}(1^k), m \leftarrow \mathcal{M}_k, \sigma \leftarrow A(m)) = \varepsilon$$

Since $\mathsf{Sign}_{SK} = f^{-1}$ and $\mathsf{Ver}_{VK} = f$, we can rewrite this as $\Pr(m = f(\sigma) \mid m \leftarrow \mathcal{M}_k, \sigma \leftarrow A(m)) = \varepsilon$. Moreover, since f is a permutation, if $x \in \mathcal{M}_k$ is random then $m = f(x)$ is random, and therefore $\Pr(f(x) = f(\sigma) | x \leftarrow M_k, \sigma \leftarrow A(f(x))) = \varepsilon$. Therefore, A inverts f with non-negligible probability, which contradicts the fact that f is a trapdoor permutation.

For the second assertion, notice that a PPT adversary B who on input VK outputs $(f(\sigma), \sigma)$ (for some "signature" σ he picks) always succeeds in his attack (i.e., σ is a signature of "message" $f(\sigma)$). □

Also, it turns out that in some special cases a trapdoor signature may be universally forgeable under the CMA. Consider, for example, trapdoor families for which for all k, $\mathcal{M}_k$ is a group and $f : \mathcal{M}_k \to \mathcal{M}_k$ is a group homomorphism. In that case, it is easy for an adversary to find out $\sigma = \mathsf{Sign}_{SK}(m)$ for any $m \in \mathcal{M}_k$ without a direct oracle query for $\mathsf{Sign}_{SK}(m)$. Indeed, for $m = 1$ (i.e. the identity element) we have that $\sigma = 1$, and for $m \neq 1$ it suffices to pick any $m_1 \in \mathcal{M}_k$ that is not 1 or m, then compute $m_2 = \frac{m}{m_1}$ (which also different from the identity and from m). The adversary can find out $\sigma_1 = \mathsf{Sign}_{SK}(m_1)$ and $\sigma_2 = \mathsf{Sign}_{SK}(m_2)$ by oracle calls and compute $\sigma = \sigma_1\sigma_2$. This shows that such trapdoor families, RSA and Rabin among them, are universally forgeable against chosen-message attacks.

It should be clear by now that the design of "secure" signature schemes is a difficult problem and at this point one might be inclined to believe that it has no solution.

2.2 A "Signature Paradox"

Those who subscribe to the pessimism expressed in the final statement of the previous subsection will not have a hard time accepting the following "theorem", which would be a far-reaching generalization of the second statement of the theorem on unforgeability of trapdoor signatures, were it only true.

Theorem 6 (The fallacious "Signature Paradox") *If* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *is universally unforgeable against* VK*-only attack, then it is existentially forgeable against chosen message attacks.*

Corollary 7 ("No secure signatures") *There do not exist signature schemes that are existentially unforgeable against* CMA.

Proof: The corollary follows form the "theorem" because existential unforgeability against CMA implies universal unforgeability against VK-only attack. Therefore, a scheme cannot satisfy the former property without also satisfying the latter, and the theorem says that universal unforgeability against VK-only attack implies existential forgery against CMA.

As for the "theorem", its "proof" goes as follows. The way one proves that some $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is universally unforgeable under VK-only attack is to prove that that property is equivalent to some "hardness" assumption on a problem X that we believe to be true. On one hand, one shows that the "hard" problem X is such that its solution yields an algorithm to break SIG in polynomial time (for instance, if one can factor n then one can take modular square roots mod$n$ and break the Rabin signature scheme with that modulus). On the other hand, one assumes the existence of a PPT A that on input (VK, m) provides a valid signature σ for any m and shows (to get a contradiction) that that would imply that there exists a PPT (denoted by B) that would use A as a "black box" (i.e. as an oracle) to break the hardness assumption and solve that given instance of X (in the Rabin case, if such an A exists, than one can take square roots, and using that square root algorithm as a black box one can factor the modulus). Thus, the existence of such "universal" B proves that the presumed A does not exist. Notice, however, that B by itself is well-defined: given a good A, B would break X.

But then, if the adversary can perform CMA, he can use this $B = B(VK)$, only that, in place of using "black box calls" to A to get $\sigma' = A(VK, m)$ (for a given m), he uses a CMA instead to get $\sigma = \mathsf{Sign}_{SK}(m)$. Then B can solve X in polynomial time and use that solution to break the signature scheme SIG. Therefore, CMA allows the adversary to substitute attacks to the sender for calls to A. For instance, in Rabin's case, that would mean that with a CMA with some well-chosen messages, one would have enough information to factor the modulus n and then break the signature scheme. □

What is wrong with that "proof"? Well, here is one mistake in it. When B uses A as an oracle in the proof by contradiction of the universal unforgeability of SIG, B can make any query he wants to A. Among other things, he might make a query of the form $A(VK', m)$, where $VK' \neq VK$. That is, he might try different verification keys, maybe because if one knows $\mathsf{Sign}_{VK'}(m)$ for several different VK' one can make a very good guess of what the solution to X is (who knows?). On the other hand, when we try to turn B into a CMA adversary, we must then commit to a *single* verification key: the one that is randomly chosen by the sender. That is, whenever B launches a CMA on the sender, he (the sender) always uses the VK that he himself has chosen; equivalently (in a more formal language), all oracle calls that B can make to Sign_{SK} must use the same SK that is in the output of Gen, and it is quite improbable that a $VK' \neq VK$ will work with SK in the right way. Therefore, the last paragraph of that "proof" is wrong.

3 One-time secure signature schemes

The falsehood of the "signature paradox" in the last section leaves us with some hope that it might be possible to build secure signature schemes after all. But given all the difficulties we have faced so far, we'd better try to do it one step at a time. Our plan, which we begin to put into practice in this section, is: to build a rather simple scheme that is secure as long as the adversary can only make *one* CMA, and to show how to get a secure scheme out of it.

3.1 Lamport's scheme for one bit

One-way functions (OWF) provide a nice way of signing one bit, which we present below.

Definition 8 (Lamport's scheme for 1-bit messages) *Using the notation from the definition of* PKS*, and letting* $f : \{0,1\}^* \to \{0,1\}^*$ *be a fixed* OWF*, we define Lamport's scheme for one-bit messages by:*

a) $SK = (X_0, X_1)$*, where* X_0 *and* X_1 *are drawn randomly and independently from* $\{0,1\}^k$*, and* $VK = (Y_0, Y_1) = (f(X_0), f(X_1))$*.*

b) $\mathsf{Sign}_{SK}(m) = X_m$ *(remember,* $m \in \{0,1\}$*)*

c) $\mathsf{Ver}_{VK}(\sigma, m) = [f(\sigma) = Y_m]$

Lamport's scheme is "one-time secure" in the sense that if the adversary wants to find out what the signature for 1 (say) with only one oracle query, that query must have the form $\mathsf{Sign}_{SK}(0) = X_0$, which is just a random string and doesn't help him at all in finding out what $\mathsf{Sign}_{SK}(1)$ is. The intuition can be easily transformed into a proof.

Remark 9 *In fact, Lamport's scheme is "many-time" secure as well for the trivial reason that there are only two messages. So the only non-trivial attack by the adversary is to forge a signature of* $b \in \{0,1\}$ *given a signature of* $(1-b)$*, i.e. general security is the same as one-time security.*

We next generalize this to many bits.

3.2 Lamport's Scheme for many bits

The generalization for long (say, length $n = p(k)$) messages of Lamport's scheme is:

Definition 10 (Lamport's scheme for n-bit messages) *Using the notation from the definition of* PKS*, and letting* $f : \{0,1\}^* \to \{0,1\}^*$ *be a fixed* OWF*, we define Lamport's scheme for* $\{0,1\}^n$ *by:*

a) Let $SK = (X_0^1, X_1^1, X_0^2, X_1^2, \ldots, X_0^n, X_1^n)$*, where the* X_i^j*'s are drawn randomly and independently from* $\{0,1\}^k$*, and* $VK = (Y_0^1, Y_1^1, Y_0^2, Y_1^2, \ldots, Y_0^n, Y_1^n)$ *with* $Y_i^j = f(X_i^j)$*.*

b) $\sigma = \mathsf{Sign}_{SK}(m_1 \ldots m_n) = X_{m_1}, \ldots, X_{m_n}$*.*

c) $\mathsf{Ver}_{VK}(\sigma_1 \ldots \sigma_n, m_1 \ldots m_n) = [\forall i \in \{1, \ldots, n\}, \quad f(\sigma_i) = Y_{m_i}]$.

What Lamport's scheme does is it builds two tables, one for signing (in which entry (i, j) corresponds to the block that is used at the j^{th} position of σ if $m_j = i$, that is X_i^j) and one for verification (in which entry (i, j) corresponds to the block that is used at the j^{th} position of σ if $m_j = i$, that is $Y_i^j = f(X_i^j)$). See the illustration below for $n = 5$.

bit/position	1	2	3	4	5
0	X_0^1	X_0^2	X_0^3	X_0^4	X_0^5
1	X_1^1	X_1^2	X_1^3	X_1^4	X_1^5

Table for Signing

bit/position	1	2	3	4	5
0	Y_0^1	Y_0^2	Y_0^3	Y_0^4	Y_0^5
1	Y_1^1	Y_1^2	Y_1^3	Y_1^4	Y_1^5

Table for Verification

bit/position	1	2	3	4	5
0	$\mathbf{X_0^1}$	X_0^2	$\mathbf{X_0^3}$	$\mathbf{X_0^4}$	X_0^5
1	X_1^1	$\mathbf{X_1^2}$	X_1^3	X_1^4	$\mathbf{X_1^5}$

The signature of 01001 is shown in bold.

We notice that an adversary can break the scheme with 2 queries, for if it gets $\mathsf{Sign}(0^n) = X_0^1 \ldots X_0^n$ and $\mathsf{Sign}(1^n) = X_1^1 \ldots X_1^n$, then it knows all X_i^j and can forge a signature for any given message. However, the scheme is one-time secure in the following sense.

Definition 11 (One-time security for PKS) *A PKS* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *is said to be one-time secure, that is, existentially unforgeable against* CMA *with one chosen message query only if for all* PPT A *we have*

$$\Pr(Ver(m, \sigma) = accept \mid (SK, VK) \leftarrow \mathsf{Gen}(1^k), (m, \sigma) \leftarrow A^{\mathsf{Sign}_{SK}}(VK)) \leq \mathsf{negl}(k)$$

where A A can use the oracle for on at most one query q *and cannot output forgery* $m = q$.

Theorem 12 (One-time security of Lamport's scheme)
Lamport's scheme is one-time secure provided f *is a* OWF.

Proof: By contradiction. Suppose that there exists a PPT adversary A that violates the definition of one-time security for Lamport's scheme with non-negligible probability ε. We can assume without loss of generality that A makes exactly one oracle call, and that the query and the forgery string both the (expected) length n.

We consider the following "experiment" B. Given input y that B tries to invert, B runs $\mathsf{Gen}(1^k)$ to get SK and VK. It then modifies one of the blocks of the verification key: instead of Y_i^j we now have $Y_i'^j = y$ for some random pair (i, j), and all the other blocks stay the same. Let's call this new verification key VK'; note that VK and VK' have the same distribution. Also, B knows the secret key SK' corresponding to VK' except for the value $x \in f^{-1}(y)$ that B tries to extract from A. The key observation (that is easy to check)

is the following: *If $y = f(x)$ where $x \in \{0,1\}^k$ is random, VK and VK' have the same distribution; moreover, i and j are independent of VK'.* The new verification table looks like this (for $i = 1$, $j = 3$ and $n = 5$):

bit/position	1	2	3	4	5
0	Y_0^1	Y_0^2	Y_0^3	Y_0^4	Y_0^5
1	Y_1^1	Y_1^2	$\mathbf{y}$	Y_1^4	Y_1^5

Denote by $q = q_1 \dots q_n$ the string whose signature $A(VK')$ asks the signing oracle (simulated by B). If it happens that $q_j = i$, then we fail in our attempt to recover x. However, since i is random and VK' is independent of i, we have that $q_j \neq i$ with probability $1/2$. In this latter case, B can easily "sign" q for A since it does not need $x \in f^{-1}(y)$ for the signature. Thus, with probability at least $\varepsilon/2$ we get that A outputs a valid message/signature pair (m, σ), where $m \neq q$, i.e. $m_1 \dots m_n \neq q_1 \dots q_n$. Hence, there must be at least one index ℓ such that $m_\ell \neq q_\ell$. Since j is chosen at random at the view of A so far was independent from j, we get that $\ell = j$ with probability $1/n$. Thus, with overall non-negligible probability $\varepsilon/2n$ we have $m_j = i$, and thus $\sigma_j \in f^{-1}(y)$. Therefore, with non-negligible probability B can output this σ_j and invert the OWF f. □

4 From One-time to Full-fledged Security

4.1 First Attempt

Let OT-SIG = (Gen, Sign, Ver) denote *any* one-time secure signature scheme. Our question now is: is there a general way of building a secure PKS from OT-SIG? It is *not* enough to divide the message into blocks and sign each block separately: Lamport's scheme did that and yet didn't meet the security standards that we set for ourselves.

We try a different idea here. The idea is to use "fresh information" for each message sent so as to circumvent the "one-timeness" of OT-SIG. We will then have to keep track of past activity; this means that our scheme will be **stateful**. It will also be quite inefficient. In particular, the length of the signature will grow. But we shall take care of these problems later on.

What we do is the following. Suppose one wants to sign the messages $m_0, m_1 \dots$ (of appropriate length) in that given order. The scheme SIG that we propose proceeds as follows.

a) First, it gets $(SK_0, VK_0) \leftarrow \mathsf{Gen}(1^k)$. VK_0 is the (public) verification key of our new scheme SIG.

b) To sign m_0, SIG gets $(SK_1, VK_1) \leftarrow \mathsf{Gen}(1^k)$, then computes $\sigma_1 = \mathsf{Sign}_{SK_0}(m_0, VK_1)$ (namely, it signs a tuple $[m_0, VK_1]$, where , is some special character that doesn't show up in other places so that we can separate m_0 from VK_1) and outputs (σ_1, VK_1, m_0) as the signature of m_0 (for notational convenience, we include the message inside the signature). It also remembers (σ_1, VK_1, m_0) for future use.

c) To check whether (σ_1, VK_1) is a valid signature for m_0 under SIG, the receiver checks if $\mathsf{Ver}_{VK_0}([m_0, VK_1], \sigma_1) = accept$ (i.e. whether σ_1 is a valid signature for $[m_0, VK_1]$ in the OT-SIG scheme.

d) Inductively, to sign m_i (for $i \geq 2$), SIG gets $(SK_{i+1}, VK_{i+1}) \leftarrow \mathsf{Gen}(1^k)$, computes $\sigma_{i+1} = \mathsf{Sign}_{SK_i}(m_i, VK_{i+1})$ and outputs $(\sigma_{i+1}, VK_{i+1}, m_i \ldots, \sigma_1, VK_1, m_0)$ (i.e. the entire history so far!) as the signature of m_i.

e) Finally, to check whether $(\sigma_{i+1}, VK_{i+1}, m_i \ldots, \sigma_1, VK_1, m_0)$ is a good signature of m_i, one successively checks all the signatures by $\mathsf{Ver}_{VK_j}([\sigma_{j+1}, VK_{j+1}], m_j)$, and accepts only if the entire chain is valid $(0 \leq j \leq i)$, where the last key VK_0 is taken from the public file.

On an intuitive level, this scheme is secure because *no key is used more than once*, and therefore the "one-timeness" of OT-SIG is enough. Indeed, if an adversary B attacks the sender with successive chosen messages, each answer it will get will correspond to a different secret key and that will circumvent the original limitations of OT-SIG.

There is a problem, however, with this construction: *at each call of* Sign_{SK} *the input is larger than what we know how to handle with current one0time signature schemes!* For instance, for the construction of one-time secure PKS that we saw in the last section, if the message length it can handle is n, the size of the verification key is $n\alpha \ll n$ (where α is the size of the output of our OWF). Thus, we cannot "fit" the verification key inside the signature.

Seems like everything we tried was in vein. But not all is lost. Namely, *if we can construct a one-time signature scheme where* $|VK|$ *could be sufficiently less than the length* n *of the messages it can handle*, then the above construction can be used to sign the messages of $(n - |VK|)$. In fact, if one fixed key could be used to sign messages of *arbitrary unbounded length*, then we can also sign messages of any unbounded length. This is stated below:

Theorem 13 *Provided* OT-SIG *can sign messages longer that the length of its verification key, the above (stateful and inefficient) construction is existentially unforgeable against chosen message attack (for messages of corresponding length as explained above).*

Proof: The proof is simple, but a bit tedious, so we just sketch the idea (the sketch below can be easily transofrmed into a formal proof).

Say some A asks to sign messages $m_0 \ldots m_t$, gets a chain $(\sigma_{t+1}, VK_{t+1}, m_t \ldots, \sigma_1, VK_1, m_0)$ from the oracle, and forges the signature $(\sigma'_{i+1}, VK'_{i+1}, m'_i \ldots, \sigma'_1, VK'_1, m'_0)$ of some $m'_i \notin \{m_1 \ldots m_t\}$. We claim that there exists and index $j \leq \max(i, t)$ such that "along the way", A produced a forgery σ'_j of a "new message" $[m'_j, VK_{j+1}]$ under the key SK_j, which contradicts one-time security of the j-th one-time signture. The proof is a bit boring:

1. If $[m'_0, VK'_1] \neq [m_0, VK_1]$, then $[m'_0, VK'_1]$ is a new message w.r.t. VK_0, and σ_1 is the forgery.

2. Otherwise (equality so far), if $[m'_1, VK'_2] \neq [m_1, VK_2]$, then $[m'_1, VK'_2]$ is a new message w.r.t. $VK_1 = VK'_1$, and σ_2 is the forgery.

3. Otherwise (equality so far), if $[m'_2, VK'_3] \neq [m_2, VK_3]$, then $[m'_2, VK'_3]$ is a new message w.r.t. $VK_2 = VK'_2$, and σ_3 is the forgery.

4. And so on. The point is that since we have $m'_i \notin \{m_1 \dots m_t\}$, at some point j we *must* have inequality: at the worst case, if $i > t$, $[m'_{t+1}, VK_{t+2}]$ is a new message w.r.t. VK_{t+1}, since no signatures w.r.t. VK_{t+1} were given to A by its oracle.

Of course, how do we find this j, and how do we simulate the run of A with this j. Well, we pick j at random from $\{0 \dots T\}$ (where T is the upper bound of A's running time). We generate all the keys on our own, except for the j-th key, where we use the given verification key VK whose one-time seurity we want to compromise. The formal proof follows quite easily from the above. □

Remark 14 *Notice, however, that this construction (known as the* **Naor-Yung** *construction) shows that our explanation of the flaws of the "proof of the signature paradox" is not at all artificial. The scheme* SIG *above uses* Sign *(from* OT-SIG*) as a black box, but each time it does so the secret key is different.*

Now, of course, the question is how to construct one-time signatures capable to sign long messages? The answer is similar to what we did for the case of MAC's: use an approproate hash function family. Namely, rather than one-time signing a long m (which might not "fit"), we first hash it down to a short string $h(m)$, and one-time sign $h(m)$. What properties are needed from h. Intuitively, we have problems if it is easy to find m_1 and m_2 which collide: $h(m_1) = h(m_2)$, since the single one-time signature of $h(m_1) = h(m_2)$ corresponds to both m_1 and m_2 now. This motivates the next section — a detaour to "appropriate" hash function families.

4.2 The "Fix": New Hash Families and the Hash-then-Sign Paradigm

As it turns out, in many applications of signatures we would like to be able to "shrink" the input so as to make our signares smaller as well. That is, we apply a hash function to the message and then sign the hashed version. That is the (extremely useful) *hash-and-sign* paradigm for which our problem is but an example. We now present the "right" definitions that we need for *hash-and-sign*, along with the older definition of weakly universal hash families, so as to make the differences between them stand out (we give a bit general definition below where the seed, input and output lengths are all some polynomials in the security parameter k).

Definition 15 (Hash families) *Let $\mathcal{H} = \{h_s : \{0,1\}^{L(k)} \to \{0,1\}^{\ell(k)}\}_{s \in \{0,1\}^{p(k)}}$ be a family of functions that are easy to evaluate (i.e. the joint map $(s, x) \to h_s(x)$ can be computed in polynomial time), with $L(k) > \ell(k)$ for all k, together with a* PPT *algorithm I that outputs some $s \in \{0,1\}^{p(k)}$ on input 1^k. Below, we let $p = p(k)$, $L = L(k)$ and $\ell = \ell(k)$ where k is the security parameter. Such a family is called:*

a) **Weakly universal hash family** *if for all distinct $x_0, x_1 \in \{0,1\}^L$ we have*

$$\Pr_{s \leftarrow I(1^k)}(h_s(x_0) = h_s(x_1)) \leq \mathsf{negl}(k)$$

b) **Universal one-way hash family** *(*UOWHF*) if for all* PPT *adversaries A, and all $x_0 \in \{0,1\}^L$ we have*

$$\Pr_{s \leftarrow I(1^k)}(h_s(x_0) = h_s(x_1) \mid x_1 \leftarrow A(x_0, s)) \leq \mathsf{negl}(k)$$

where A must output $x_1 \neq x_0$.

c) **Collision-resistant hash family** *(*CRHF*) if for all* PPT *adversaries A we have*

$$\Pr_{s \leftarrow I(1^k)}(h_s(x_0) = h_s(x_1) \mid (x_0, x_1) \leftarrow A(s)) \leq \mathsf{negl}(k)$$

where A must output $x_1 \neq x_0$.

We will often abbreviate $h \leftarrow \mathcal{H}$ to indicate that $s \leftarrow I(1^k)$ and $h = h_s$.

Recall that weakly universal hash families were sufficient for the "hash-then-mac" paradigm in the secret-key authentication. As we will see below, UOWHF's and CRHF's will play a similar role in the public-key authentication. We will not prove the existence of UOWHF's and CRHF's in this lecture, but the following lemmas should make it clear that (1) UOWHF and CRHF are the only missing elements in our construction of secure signature schemes and (2) the new — "hash-and-sign" method — really works with these functions. The lemmas below work for both regular ("many-time") and one-time signature scheme: the former case being extensively used in practice, and the latter will be used to finish the Naor-Yung construction. We prove both statements for regular (multi-time) signature scheme, the one-time case will be a special case.

Lemma 16 (Secure schemes with CRHF**)** *If $\mathcal{H} = \{h_s\}$ is a* CRHF *and* $\mathsf{SIG}' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ *is a (one-time) secure signature scheme for ℓ-bit messages, then the signature scheme* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *defined below is (one-time) secure for L-bit messages:*

a) $SK = SK'$, $VK = (VK', h)$, *where* $(SK', VK') \leftarrow \mathsf{Gen}'(1^k)$ *and* $h \leftarrow \mathcal{H}$.

b) $\mathsf{Sign}_{SK}(m) = \mathsf{Sign}'_{SK'}(h(m))$.

c) $\mathsf{Ver}_{VK}(m, \sigma) = \mathsf{Ver}'_{VK'}(h(m), \sigma)$.

Proof: Assume that Lemma 16 is false for some SIG', that is, there exists an adversary A such that

$$\Pr(\mathsf{Ver}'_{SK'}(h(m), \sigma) = 1 \mid (SK', VK') \leftarrow G(1^k), h \leftarrow \mathcal{H}, (m, \sigma) \leftarrow A^{\mathsf{Sign}_{SK'}}(VK', h)) = \varepsilon$$

where A queried the oracle on messages $m_1 \ldots m_t$, and, when successful, outputs the signature σ of $m \notin \{m_1, \ldots, m_t\}$. Then at least one of the following events happens with non-negligible probability $\varepsilon/2$:

(a) $h(m) \in \{h(m_1), \ldots, h(m_t)\}$.

(b) $h(m) \notin \{h(m_1), \ldots, h(m_t)\}$.

In case (a), we can break the collision-resistance property of $\mathcal{H}$. Indeed, it implies that for some i, $h(m) = h(m_i)$, while by assumption $m \neq m_i$, so m and m_i form a collision. In case (b), we break the security of our original signature SIG'. Indeed, $h(m)$ is a "new message" w.r.t. SIG', since A only saw $\mathsf{Sign}(m_i) = \mathsf{Sign}'(h(m_i))$, so A managed to forge a new signature. Translating this into a formal proof (i.e., building the actual B breaking SIG') is straightforward. Indeed, B picks its own $h \leftarrow \mathcal{H}$, and simulates oracle calls to $\mathsf{Sign}(m_i)$ by oracle calls to $\mathsf{Sign}'(h(m_i))$. When A forges (m, σ), B outputs its own forgery $(h(m), \sigma)$. □

Lemma 17 (Secure schemes with UOWHF) *If $\mathcal{H} = \{h_s\}$ is a* UOWHF *and $\mathsf{SIG}' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ is a (one-time) secure signature scheme for $(\ell + p)$-bit messages, then the signature scheme $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ defined below is (one-time) secure for L-bit messages:*

a) $SK = SK'$, $VK = VK'$, *where* $(SK', VK') \leftarrow \mathsf{Gen}'(1^k)$.

b) $\mathsf{Sign}_{SK}(m) = (h, \mathsf{Sign}'_{SK'}(h \circ h(m))$, *where* $h \leftarrow \mathcal{H}$ *and* $\circ$ *is concatenation.*

c) $\mathsf{Ver}_{VK}(m, (h, \sigma)) = \mathsf{Ver}'_{VK'}(h \circ h(m), \sigma)$.

Proof: Assume that Lemma 17 is false for some SIG', that is, there exists an adversary A such that

$$\Pr(\mathsf{Ver}'_{SK'}(h \circ h(m), \sigma) = 1 \mid (SK', VK') \leftarrow G(1^k), (m, (h, \sigma)) \leftarrow A^{\mathsf{Sign}_{SK'}}(VK')) = \varepsilon$$

where A queried the oracle on messages $m_1 \ldots m_t$, and, when successful, outputs the signature (h, σ) of $m \notin \{m_1, \ldots, m_t\}$. Let also (h_i, σ_i) denotes the signature of m_i returned by the oracle (i.e., each h_i is truly random, even though h used in the forgery could be chosen by A arbitrarily). Then at least one of the following events happens with non-negligible probability $\varepsilon/2$:

(a) $h \circ h(m) \in \{h_1 \circ h_1(m_1), \ldots, h_t \circ h_t(m_t)\}$.

(b) $h \circ h(m) \notin \{h_1 \circ h_1(m_1), \ldots, h_t \circ h_t(m_t)\}$.

In case (a), we can break the universal one-wayness property of $\mathcal{H}$. Indeed, it implies that for some i, $h = h_i$ and thus $h_i(m) = h_i(m_i)$, while by assumption $m \neq m_i$. Thus, if we set $x_0 = m_i$, we get that $h = h_i$ was chosen at random *after* x_0 was chosen by A, and we can set $x_1 = m$, thus creating a collision $x_0 \neq x_1$ for this randomly selected h_i. Translating this into a formal proof is easy and is omitted.

In case (b), we break the security of our original signature SIG'. Indeed, $h \circ h(m)$ is a "new message" w.r.t. SIG', since A only saw $\mathsf{Sign}'(h_i \circ h_i(m_i))$, so A managed to forge a new signature. Translating this into a formal proof (i.e., building the actual B breaking SIG') is straightforward. Indeed, B simulates oracle calls to $\mathsf{Sign}(m_i)$ by picking a random $h_i \leftarrow \mathcal{H}$ and, getting $\sigma_i = \mathsf{Sign}'(h_i \circ h_i(m_i))$ from its own oracle, and returning (h_i, σ_i). When A forges $(m, (h, \sigma))$, B outputs its own forgery $(h \circ h(m), \sigma)$. □

We conclude this lecture by noticing that if "good" (i.e., well shrinking) CRHF's or UOWHF's exist, the Naor-Yung construction will be complete (albeit inefficient for now).