0368.4162: Introduction to Cryptography Ran Canetti

Lecture 8: Digital Signatures

28 December 2008 Fall 2008

Scribes: Ido Kasher, Sagi Hed

1 Goal

We would like that a person A would be able to generate message-signature pairs (m, t) such that everyone could verify that the message indeed originated from A, and additionally no verifier will be able to generate pairs (m', t') that would seem as if they had been sent from A.

Also, we would like that a receiver would be able to convince a third party that the message indeed originated from the sender.

MAC does not have the required functionality since it allows anyone who can verify to authenticate ("sign") as well, while here we would like that verifers will not be able to sign (forge) messages.

2 Definitions

2.1 Components

A Digital signature scheme is a tuple of three probabilistic polynomial-time algorithms $(Gen, Sign, Ver)$:

- A random key generation algorithm generates a private signing key, sk, and a public verification key, vk.
$$Gen(1^n) \rightarrow (sk, vk)$$
- A signing algorithm receives the signing key sk and the message m, and outputs a signature s.
$$Sign(sk, m) \rightarrow s$$
- A verification algorithm receives the verification key vk, the message m and the signature s, and outputs the verification outcome.
$$Ver(vk, m, s) \in \{accept, reject\}$$

2.2 Requirements

2.2.1 Completeness

For any message m,

$$Prob_{Gen(1^n),Sign,Ver}[Gen(1^n) \rightarrow (sk, vk), Ver(vk, m, sign(sk, m)) = reject] < \nu(n)$$

where $\nu(n)$ is a negligible function.

2.2.2 Consistency

The verification algorithm is deterministic, that is, for any vk, m, s:

$$Prob[Ver(vk, m, s) = accept] \in \{0, 1\} \quad (1)$$

In other words, it is not possible that the verification algorithm accepts a signature in one run but rejects it in another.

2.2.3 Unforgeability

There are several types of resistance to forgery. In general, we will distinguish between two types of parameters:

I. Interaction type of the adversary with the scheme

(a) Key only attack - The adversary has knowledge of only the public verification key.

(b) Known message attack - The adversary has knowledge of the verification key and a set of message-signature pairs.

(c) Chosen message attack - The adversary has knowledge of the verification key and the ability (a black-box access to the signing algorithm) to generate messages and receive their valid signatures. Note that he has no knowledge of the secret signing key.

II. Forgery types

(a) An adversary can compute the secret signing key.

(b) An adversary can sign any given message.

(c) An adversary can sign a specific message he gets as an input.

(d) An adversary can generate some message of his choice (even if meaningless) with a valid signature. This is called existential forgery.

We would like to have a scheme in which an adversary cannot forge even in the sense of II.d in an attack of type I.c. Such a scheme is called EU-CMA secure (Existential Unforgeability under a Chosen Message Attack).

3 Existential Unforgeability under a Chosen Message Attack

(Goldwasser, Micali, Rivest 83')

Define a game between the adversary and the signature scheme:
$EU-CMA-GAME(Gen, Sign, Ver, A, n)$

1 $Gen(1^n) \rightarrow (sk, vk)$

2 A receives vk

3 A creates a message m

4 A gets $s \leftarrow Sign(sk, m)$

5 A repeats steps 3 and 4 as needed

6 A outputs (m^*, s^*)

7 A wins if $Ver(vk, m^*, s^*) = accept$ and m^* wasn't previously signed by $Sign$

Definition: A signature scheme is EU-CMA secure if for any polynomial-time adversary A, the probability of winning the EU-CMA-GAME is negligible.

Note: There is no digital signature scheme which is secure against unbounded adversaries: Given a verification key, an unbounded adversary can try all possible secret signing keys, and find a key which gives a signature that passes verification. This is true regardless of the size of the keys.

4 Direct Constructions

4.1 Using TDP

Let (G, S, F_k, F_k^{-1}) be an ensemble of trapdoor permutations. Define the following signature scheme:

$$\begin{aligned} Gen(1^n) &: (i,t) \leftarrow G(1^n).\ sk \leftarrow t,\ vk \leftarrow i \\ Sign(t,m) &= F_t^{-1}(m) \\ Ver(i,m,s) &= 1\ iff\ F_i(s) = m \end{aligned}$$

This scheme is known as the "textbook signature" scheme, based on, say, RSA. However, we will show it is not secure.

Attack 1: It is possible to create valid message-signature pairs by using only the public key i:
Pick some arbitrary r and compute $s \leftarrow F_i(r)$ using the public verification key i. The pair (s, r) is a valid signature on the message s. This is an example of *existential forgery*.

Attack 2: When F_i is homomorphic, using known message-signature pairs, it is possible to generate valid signatures for new messages:

$$\begin{aligned} RSA(m) &= m^e\ (mod\ N) \\ RSA^{-1}(c) &= c^d\ (mod\ N) \end{aligned}$$

Let (m_1, m_1^d) and (m_2, m_2^d) be two message-signature pairs.
Multiplying the two signatures we get $m_1^d \cdot m_2^d = (m_1 \cdot m_2)^d$, which is the signature for the message $m_1 \cdot m_2$.

4.2 Using TDP and Hashing

A potential way to overcome these attacks is by adding a hash function H to the signature scheme and update the signing and verification algorithms as follows:

$$\begin{aligned} Sign(t,H,m) &= F_t^{-1}(H(m)) \\ Ver(i,H,m,s) &= (F_i(s) \stackrel{?}{=} H(m)) \end{aligned}$$

If H is not a homomorphic function the second attack will not work. However, we don't know how to formalize properties of H that will make the scheme secure with any TDP. Specifically, collision resistance is not sufficient. That is, there exist functions H,F such that H is collision resistant and F is a TDP, yet the signature scheme is insecure.

It can be proven that the scheme works if H is a truly random function (then the scheme is called "secure in the Random Oracle Model"). However, such a proof does not say anything about the case of interest, namely when H is a real efficiently computable function.

Important Remark: There is a difference between modeling H as a truly random function and choosing H to be a PRF. PRFs can be computed only with the help of a secret key and seem pseudorandom only to adversaries who don't have the secret key. In our case anyone can compute H (so there can be no secret key).

Despite that, the existing standards for signing messages do use this scheme with specific hash functions (for example, the trapdoor function RSA is used along with MD5 or SHA1 in the PKCS #1 scheme).

5 Construction Based On OWF

As opposed to the direct constructions, we will see that it is possible to construct signature schemes whose security is derived from basic assumptions. Specifically, we will see:

Theorem 1. *if OWFs exist then EU-CMA signature schemes exist*

In class we will prove a weaker theorem:

Theorem 2. *if CRFs exist then EU-CMA signature schemes exist*

We will prove the theorem in stages.

5.1 One-time signature scheme for bounded-length messages

(Lamport '79)

We would like to construct a scheme that is secure for signing a single k-bit message.
Let f be a one-way function.
Define:

$$\begin{aligned} Gen(1^n) = sk &: x_1^0, ..., x_k^0 \\ &\quad x_1^1, ..., x_k^1 \ where \ x_i^j \in \{0,1\}^n \\ vk &: v_1^0 = f(x_1^0), ..., v_k^0 = f(x_k^0) \\ &\quad v_1^1 = f(x_1^1), ..., v_k^1 = f(x_k^1) \end{aligned}$$

$$\begin{aligned} Sig(sk, m) &= x_1^{m_1} ... x_k^{m_k} \ where \ m = m_1 ... m_k \ and \ m_i \in \{0,1\} \\ Ver(vk, m, s) &= accept \ iff \ \forall i : f(x_i^{m_i}) = v_i^{m_i} \end{aligned}$$

Claim 1. *The Lamport scheme is EU-CMA secure for a single signature.*

Proof: We assume a polynomial adversary A that can forge a message with probability ϵ, and construct an adversary A' that inverts f with probability $\frac{\epsilon}{2k}$.

Recall the EU-CMA game.
A receives the verification key $vk = v_1^0 ... v_k^0, v_1^1 .. v_k^1$ as input.
A generates a message $m = m_1 ... m_k$ and gets a valid signature $s = s_1 ... s_k$.
Finally, A generates a new message $m' \neq m$ and a valid signature for it, $s' = s'_1 ... s'_m$.

Intuitively, since $m' \neq m$ there exists an index i for which $m'_i \neq m_i$. Note that s'_i, the signature associated with m'_i, equals $x_i^{m'_i}$, which is the inverse of $f(x_i^{m'_i}) = v_i^{m'_i}$.

We now construct an adversary A':
A' receives $y = f(x)$ as an input, and randomly chooses $i \in [1, n], b \in \{0, 1\}$.
A' runs Gen to generate a signing key and a verification key for A but substitutes v_i^b with y .
A generates a message $m = m_1 ... m_k$ and requests A' for its signature (recall that A' has the signing key).
If $m_i \neq b$, A' gives a valid signature to A, otherwise it aborts (since it cannot produce a signature that can be verified versus y).
A generates a new message m' and a valid (forged) signature s' for it.
$m' \neq m$, so there exists an index j for which $m_j \neq m'_j$.

If $j = i$, and therefore $m'_j = m'_i = b$, then $s'_j = s'_i = x_i^{m'_i}$ is the inverse of $f(x_i^{m'_i}) = v_i^b = y$, and A' outputs s'_i, otherwise it aborts.

It's easy to see that A' inverts correctly if $j = i$ (probability $\frac{1}{k}$) and $m_i \neq b$ (probability $\frac{1}{2}$) and A succeeds in the forgery (probability ϵ). In addition, A's view is independent of A's random choices, i.e. (i, b). So A' succeeds in probability at least $\frac{\epsilon}{2k}$.

5.2 One-time signature scheme for arbitrary-length messages

We will use a collision-resistant function $h_k \in H_n$ where $H = \{H_n\}_{n \in N}$ (H is a CRF family ensemble). k will be included in the signing and verification keys. The scheme remains as before, with the exception that both $Sign$ and Ver algorithms are applied on $h_k(m)$.

Claim 2. *The scheme is EU-CMA secure for a single signature of unrestricted length.*

Proof: We assume a polynomial adversary A that can forge a message with probability ϵ.
Let COL be the following event:
A creates a message m.
A receives a valid signature s on m. A creates a forgery (m^*, s^*), where $s^* = s$, that is, A creates a new message with a signature identical to the signature produced by the oracle. In other words, $h(m^*) = h(m)$.

Claim: $Prob[COL] < \nu(n)$ where $\nu(n)$ is a negligible function.
Proof: Assume by contradiction that COL occurs in A with a non-negligible probability. Then we construct a collision-finder algorithm C for H:
On input h, C runs Gen to generate the signature and verification key. It then runs A on an EU-CMA game. With non-negligible probability COL will occur and then we receive a collision of m, m^* in h. So:

$$Prob[Collision\ of\ m\ and\ m^*\ in\ h] = Prob[COL] \geq \nu(n)$$

In contradiction to h being from a CRF family.

Now we separate into two distinct cases:

1 Assume COL occurs. Here we make no assumptions on the forgery probability i.e. it can be 1.

2 Assume COL does not occur, then:
Claim: $Prob[A\ forges\ succesfully \mid \overline{COL}] < \nu(n)$
Proof: The security results from the security of the underlying 1-time signature scheme in the previous section. That is, let A be an adversary that forges a signature with high probability. We can construct an adversary A' that forges a signature on messages of fixed length.

So overall,

$$\begin{aligned} Prob[A\ forges\ successfully] &= Prob[A\ forges\ succesfully \mid COL] \cdot Prob[COL] \\ &+ Prob[A\ forges\ succesfully \mid \overline{COL}] \cdot Prob[\overline{COL}] \\ &< 1 \cdot \nu(n) + \nu(n) \cdot 1 < \nu(n) \end{aligned}$$

5.3 A multi-time signature scheme

For every new message m_i, create a new one-time signing and verification key pair (sk_i, vk_i). Prepare this pair in advance when receiving the previous message and sign for it with the previous secret key

sk_{i-1}. That is, when signing the $(i-1)$'th message - m_{i-1} sign both the message and the verification key for the next message $m_{i-1}.vk_i$ using sk_{i-1}. This way no signing key is used more than once (on its subsequent message), and in addition all new verification keys are verifiable via the original vk.
Note: We can sign the message $m_{i-1}.vk_i$ since we have an arbitrary length signature scheme.

Formally, let (Gen, Sig, Ver) be a 1-time EU-CMA arbitrary length signature scheme. Our multi-time signature scheme (Gen', Sig', Ver') is as follows:

$$
\begin{aligned}
Gen'(1^n): \quad & (sk, vk) \leftarrow Gen(1^n) \\
& sk_1 = sk \\
& state = (sk_1,\ M = \emptyset) \\
Sig'(sk, m_i, state) = \quad & (sk_{i+1}, vk_{i+1}) \leftarrow Gen(1^n) \\
& output\ s_i = Sig(sk_i,\ m_i.vk_{i+1},\ M) \\
& state = (sk_{i+1},\ M \cup m_i.vk_{i+1}.s_i) \\
Ver'(vk, m, s, M) = \quad & vk_1 = vk, s_{|M|} = s \\
& accept\ if\ \ \forall i = 1..|M|:\ \ Ver(vk_i,\ m_i.vk_{i+1},\ s_i)\ accepts
\end{aligned}
$$

where $|M|$ denotes the number of records in M.

Claim: If (Gen, Sig, Ver) is a 1-time EU-CMA signature scheme then (Gen', Sig', Ver') is a multi-time EU-CMA signature scheme.

Proof: We assume a polynomial adversary A' that forges a signature in (Gen', Sig', Ver') scheme after requesting multiple signatures of his choice with probability ϵ. We construct a polynomial adversary A that forges a signature in (Gen, Sig, Ver) scheme after requesting just one signature of his choice with probability $\frac{\epsilon}{q(n)}$ where $q(n)$ is the (polynomial) bound on the number of signatures A' requests.
A chooses a random $i \in [1, q(n)]$. Note that A is given a verification key v and a 1-time oracle access to sign a message in the (Gen, Sig, Ver) scheme.
A runs A' in the (Gen', Sig', Ver') scheme, generating all the (sk_j, vk_j) pairs along the way, except for (sk_i, vk_i) instead of which A uses the signing oracle and v.
So A' sees the same interaction as in a normal (Gen', Sig', Ver') scheme, and forges a signature with probability ϵ. This forgery must include a forgery of one of the underlying one-time signatures. So with probability at least $\frac{\epsilon}{q(n)}$ A' can forge the i'th scheme, in which case A can output a forgery for the original (Gen, Sig, Ver) scheme.

5.4 A multi-time stateless signature scheme

In the previous scheme, the state is linear in size to the number of messages signed so far. We will generate a stateless scheme in two steps.

5.4.1 A multi-time scheme using a tree-structure

Again use an underlying 1-time arbitrary length signature scheme. Let $m|i$ be the i-bit prefix of message m, such that $m|i \in \{0,1\}^i$. For every new message prefix encountered $x = m|i \in \{0,1\}^i$ we use a different one-time signing and verification key pair (sk_x, vk_x). In addition, when encountering prefix x we create 2 new pairs of keys, one for each continuation of x by one bit (i.e. one for $x0$ and one for $x1$). We sign x and the 2 new verification keys using x's signature key:
$s_x = Sig(sk_x,\ x.vk_{x0}.vk_{x1})$.
For every signed prefix, we will maintain in our internal state $(s_x,\ sk_{x0},\ vk_{x0},\ sk_{x1},\ vk_{x1})$. If we

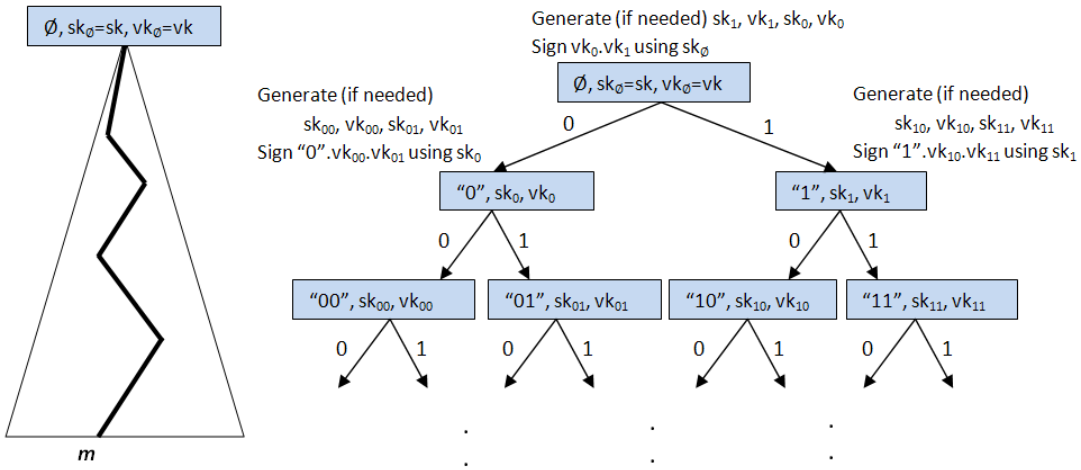

Figure 1: A tree-structure signature scheme. We descend the tree according to message m. For every prefix x we sign $x.vk_{x0}.vk_{x1}$ using key sk_x

encounter a prefix x and sk_{x0}, sk_{x1} were already generated, we retrieve them from the state and do not regenerate them.
For a message m, we perform such a signature for every prefix $x = m|i$.

Note: A specific prefix x can be signed multiple times when appearing as a prefix of different messages, but it will always be the exact same string - $x.vk_{x0}.vk_{x1}$ being signed with the same key sk_x (rather than different messages with the same key) and so there is no breach to the 1-time signing limit.

Note: Whenever we encounter a new prefix $x = m|i$, its signing and verification keys (sk_x, vk_x)are already allocated since at least one such shorter prefix x' had been encountered - the prefix $x' = m|(i-1)$ for that same message m.

This process can viewed as a walk through a binary tree of all possible prefixes, as seen in figure 1. A signing of a message m, matches a walk down the tree to our specific leaf m. Along the way for every node x (representing a prefix of m), we generate new keys (if they haven't been already generated) for both of x's right and left children and also use the previously generated signing and verification key pair (sk_x, vk_x) to sign for prefix x and the newly generated keys for x's children.

Formally, let (Gen, Sig, Ver) be a 1-time EU-CMA arbitrary length signature scheme. Our multi-

time signature scheme (Gen', Sig', Ver') is as follows:

$$
\begin{aligned}
Gen'(1^n): \quad & (sk, vk) \leftarrow Gen(1^n). \\
& sk_\emptyset = sk. \\
& vk_\emptyset = vk. \\
& \forall\, x \in \{0,1\}^i,\ i \le n,\ state_x = null \\
Sig'(sk, m, state): \quad & s \leftarrow \emptyset \\
& for\ \ i = 0...n: \\
& \quad (1)\ x = m|i,\ x' = m|(i+1) \\
& \quad (2)\ If\ \ state_x = null \\
& \quad\quad (2a)\ (sk_{x0}, vk_{x0}) \leftarrow Gen(1^n) \\
& \quad\quad (2b)\ (sk_{x1}, vk_{x1}) \leftarrow Gen(1^n) \\
& \quad\quad (2c)\ s_x \leftarrow Sig(sk_x,\ x.vk_{x0}.vk_{x1}) \\
& \quad\quad (2d)\ state_x = (s_x,\ sk_{x0},\ sk_{x1},\ x.vk_{x0}.vkx1) \\
& \quad (3)\ s = s \cup (s_x,\ vk_{x0},\ vk_{x1}) \\
& output\ \ s
\end{aligned}
$$

$$
\begin{aligned}
Ver'(vk, m, s): \quad & vk_\emptyset = vk,\ v = \ accept \\
& for\ \ i = 0...n: \\
& \quad (1)\ x = m|i,\ \ find\ \ (s_x,\ vk_{x0},\ vk_{x1}) \in s \\
& \quad (2)\ v = v \wedge Ver(vk_x,\ x.vk_{x0}.vk_{x1},\ s_x) \\
& accept\ if\ v = accept
\end{aligned}
$$

Claim: If (Gen, Sig, Ver) is a 1-time EU-CMA signature scheme then (Gen', Sig', Ver') is a multi-time EU-CMA signature scheme.

Proof: The proof is a variant of the proof in the previous section. We omit the details.

5.4.2 A stateless scheme

In the tree-structure scheme, we use the state whenever we come across a prefix $x = m|i$ whose associated signing and verification keys (sk_x, vk_x) were already allocated sometime before. Instead of remembering these choices, we will regenerate them whenever we need, but always feed the random tape with the same string so we will repeat the same choices consistently. For this purpose we will use a PRF output as the random, so it will be indistinguishable from ordinary random.

- A secret key k for the PRF ensemble F is added to the signing key sk.
- To generate a signature on a message m, we start with an empty $state$ and follow almost the same process as before. The difference is that to generate a signature on a prefix $m|i$, we use $r = F_k(m|i)$ as the random tape.

Completeness and consistency are clear.
We can show unforgeability: Assume that the scheme can be forged with adversary A, and construct a distinguisher A' between $f \in F$ and a random function. A' will run A with randomness created from the oracle function O, and if A performs a successful forgery then A' outputs that O is a pseudo-random function, otherwise A' outputs that O is a random function.