University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

Lecture 36

1 The Lamport 1-Time Signature Scheme

We briefly review the Lamport 1-time signature scheme (for messages of length ℓ) from last time. Recall that $f : \{0,1\}^m \to \{0,1\}^n$ is a one-way function.

1. Key generation consists of choosing 2ℓ elements at random from $\{0,1\}^m$ (i.e., the domain of f). Thus, we choose $x_{1,0}, x_{1,1}, \ldots, x_{\ell,0}, x_{\ell,1} \leftarrow \{0,1\}^m$. For all i, j (with $1 \le i \le \ell$ and $j \in \{0,1\}$) we then compute $y_{i,j} = f(x_{i,j})$. The public key PK and the secret key SK are as follows:
$$SK = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix} \qquad PK = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix}$$

2. To sign an ℓ-bit message $m = m_1 \cdots m_\ell$, simply "pick out" the corresponding entries from the secret key and send them. Thus, the signature will be $(x_{1,m_1}, x_{2,m_2}, \ldots, x_{\ell,m_\ell})$. To illustrate, if we want to sign a message $m = 01 \cdots 1$, we send the boxed entries:
$$\begin{pmatrix} \boxed{x_{1,0}} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & \boxed{x_{2,1}} & \cdots & \boxed{x_{\ell,1}} \end{pmatrix}$$

3. To verify a signature $(x_1, x_2, \ldots, x_\ell)$ on message $m_1 \cdots m_\ell$, we simply verify that for all i (with $1 \le i \le \ell$) we have $f(x_i) \overset{?}{=} y_{i,m_i}$.

We now prove the security of this scheme as a 1-time signature scheme. Let us recall what this means. We have an adversary who gets the public key PK, can ask for a signature on any message m it chooses, gets the signature, and then tries to forge a valid signature on a new message $m' \neq m$. We want to bound the success of any adversary of this type. We will do this in the standard way: we show that any adversary who can forge signatures with high probability can be used to invert the one-way function f with high probability, a contradiction.

Theorem 1 *If f is a (t, ϵ)-one-way function, then the Lamport signature scheme is a $(t, 2\ell\epsilon)$-secure 1-time signature scheme.*

Proof Assume we have an adversary A who forges signatures with probability δ. We show how to use A to invert the one-way function f. Construct algorithm A' (which gets a value $y \in \{0,1\}^n$ and tries to find an $x \in \{0,1\}^n$ such that $f(x) = y$) as follows:

```
$A'(y)$
$i^* \leftarrow \{1, \ldots, \ell\}$; $j^* \leftarrow \{0,1\}$
$y_{i^*,j^*} = y$
for all $i, j$ with $1 \leq i \leq \ell$ and $j \in \{0,1\}$ and $(i,j) \neq (i^*, j^*)$:
  $x_{i,j} \leftarrow \{0,1\}^m$; $y_{i,j} = f(x_{i,j})$
$PK = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix}$
run $A(PK)$ until it requests a signature on $m = m_1 \cdots m_\ell$
if $m_{i^*} = j^*$, abort; otherwise, return the correct signature to $A$
eventually, $A$ outputs a forged signature $(x_1, \ldots, x_\ell)$ on $m'_1 \cdots m'_\ell$
if $m'_{i^*} \neq j^*$ abort; otherwise, output $x_{i^*}$
```

In words, A' does the following: it first chooses a random index i^* and a random bit j^*. This defines a position in the public key at which A' will place the value y that it wants to invert. (Namely, A' sets $y_{i^*,j^*} = y$.) The remainder of the public key is generated honestly. This means that A' *can sign any message honestly except those with* $m_{i^*} = j^*$. Now, A' runs A (giving A the public key that A' prepared) until A requests a signature on message m. As we noted, A' can generate this signature as long as $m_{i^*} \neq j^*$. Otherwise, A' simply aborts and gives up.

Assuming A' has not aborted, it gives the signature to A and continues running A until it returns a forgery $(x_1, \ldots, x_\ell)$ on message $m'_1 \cdots m'_\ell$. Conditioned on the fact that A' has not aborted, we may note that this is a valid forgery with probability δ. A valid forgery in particular means that $f(x_{i^*}) = y_{i^*, m'_{i^*}}$ and also that $m' \neq m$. So, assuming that this is a valid forgery we have found the inverse for y as long as $m'_{i^*} = j^*$ (since $y_{i^*,j^*} = y$).

We just need to analyze the probability that A' returns the correct answer. This happens as long as the following three things happen: (1) A' is able to return a correct signature to A (i.e., $m_{i^*} \neq j^*$), (2) A outputs a valid forgery, and (3) the forgery has $m'_{i^*} = j^*$. The probability of event (1) is simply $1/2$ (since j^* was chosen at random, and is independent of the view of A). Conditioned on the fact that event (1) occurs, the probability of event (2) is exactly δ, the assumed probability of forgery for A. Finally, conditioned on the fact that events (1) and (2) both occur, we know that we must have $m' \neq m$. So there exists at least one i such that $m_i \neq m'_i$. If this $i = i^*$ then we are done (since event (1) occurred we know that $m_{i^*} \neq j^*$ so $m'_{i^*} = j^*$). Since m' is ℓ bits long, there is probability at least $1/\ell$ that $i = i^*$.

Putting everything together, we see that A' inverts f with probability $\frac{1}{2} \cdot \delta \cdot \frac{1}{\ell} = \delta/2\ell$. Since we know that f is (t, ϵ)-one-way, we must have $\delta/2\ell \leq \epsilon$ and therefore $\delta \leq 2\ell\epsilon$, completing the proof. ■

2 Improving the Lamport Scheme

We may note (at least) two limitations of the Lamport scheme, and we suggest here some ways to avoid them. In subsequent lectures, we will see other, better ways to avoid these problems.

Signing multiple messages. The Lamport 1-time signature scheme is exactly that: it only allows secure signing of a single message. (It is easy to find attacks which allow forgery if two or more messages are signed.) Can we improve this? One thing we can note immediately is that if we need to sign M messages (each of length ℓ) we can just generate M instances of the Lamport scheme (or, in fact, any 1-time signature scheme). This gives a new signature scheme defined as follows:

1. Key generation proceeds by running *any* key generation algorithm for a 1-time signature scheme M times to give $(PK_1, SK_1), (PK_2, SK_2), \ldots, (PK_M, SK_M)$. The public key PK is simple $(PK_1, \ldots, PK_M)$ and the secret key is $(SK_1, \ldots, SK_M)$.

2. To sign securely we need to maintain a counter indicating which key we should use. The counter is initialized to 1 and is incremented each time we sign. If c is the current value of the counter, then to sign message m_c (which is the c^{th} message we are signing) we simply sign using key SK_c to give signature σ_c and output (c, σ_c). We include c in the signature just to make verification easier. Note that we must refuse to sign any more messages once $c > M$.

3. To verify a signature (c, σ) on message m with respect to known public key PK, simply run the verification algorithm for the 1-time signature scheme on m using public key PK_c.

It should not be difficult to convince yourself that this is indeed secure for signing M messages. Even so, there are still a number of limitations (we will remove some of these in subsequent lectures): (1) the number of messages M to be signed needs to be known in advance, at the time of key generation; (2) the secret key and public key have size $O(M)$ which will be unwieldy as M gets large; (3) the signer needs to maintain state (the counter).

SIGNING LONGER MESSAGES. Another limitation of the Lamport scheme is that it is only defined for messages of a particular length (which must be known in advance); furthermore, the public/secret keys as well as the signature itself have size $O(\ell)$ so signing very long messages is extremely inefficient. We may note that it is easy to improve this, however, by hashing the message before signing it. I.e., to sign message m first compute $\tilde{m} = H(m)$ and then sign $\tilde{m}$ as usual. If $H : \{0,1\}^* \to \{0,1\}^\ell$ then this allows the signing of arbitrarily-long messages without growing the public/secret key or the signature length. In particular (and this will become important later), *this allows signing messages which are longer than the public key.*

But what properties do we need from H in order for this to be *secure*? It should not be hard to convince yourself that the scheme is secure if and only if H is a collision-resistant hash function (a definition was given in a previous lecture). In fact, when H is collision resistant this construction looks exactly like a construction we gave previously in the context of message authentication codes.