COS433/Math 473: Cryptography

Mark Zhandry
Princeton University
Spring 2017
Last Time

Digital Signatures
Digital Signatures

Algorithms:
• $\text{Gen()} \rightarrow (sk, pk)$
• $\text{Sign}(sk, m) \rightarrow \sigma$
• $\text{Ver}(pk, m, \sigma) \rightarrow 0/1$

Correctness:
$\Pr[\text{Ver}(pk, m, \text{Sign}(sk, m)) = 1: (sk, pk) \leftarrow \text{Gen}()] = 1$
1-time Security For Signatures

\[(sk, pk) \leftarrow \text{Gen}()\]
\[\sigma \leftarrow \text{Sign}(sk, m)\]

Output 1 iff:

- \(m^* \neq m\)
- \(\text{Ver}(pk, m^*, \sigma^*) = 1\)

\[1\text{CMA-Adv}(\cdot) = \Pr[\text{outputs 1}]\]
Unbounded Use Signatures

\[ (sk, pk) \leftarrow \text{Gen}() \]
\[ \sigma \leftarrow \text{Sign}(sk, m) \]

Output 1 iff:
- \( m^* \notin \{m_1, \ldots \} \)
- \( \text{Ver}(pk, m^*, \sigma^*) = 1 \)

\[ \text{CMA-Adv}(\text{robot}) = \Pr[ \text{Charlie outputs 1} ] \]
Strong Security

\[(m^*, \sigma^*) \in \{(m_1, \sigma_1) \ldots\}\]

Output 1 iff:
- \((m^*, \sigma^*) \notin \{(m_1, \sigma_1) \ldots\}\)
- \(\text{Ver}(pk, m^*, \sigma^*) = 1\)

\[CMA-Adv(\epsilon) = \Pr[\epsilon \text{ outputs 1}]\]
Certificates

Facilitate public key infrastructure

Basically a signature by a CA on your public key
• Certifies that you own the public key, not some adversary
Certificate Chaining

Once Bob’s public key is certified, Bob can sign Charlie’s public key

Charlie can then sign Donald’s public key

Donald is therefor the certified owner of his public key
So Far

Signature constructions from RSA, Factoring
• We will later also see constructions from DDH

Unfortunately, all in the random oracle model

Ideally: construction without random oracles
• Also from general assumptions
Today

One-way functions are sufficient to build signature schemes

Therefore, can build signatures from:
• RSA, DDH, Block Ciphers, CRHF, etc.

Limitation:
• Poor performance in practice
Lamport Signatures

Let $F: X \rightarrow Y$ be a one-way function

Let $M = \{0,1\}^n$ be message space

Gen():

$x_1,0$ $x_2,0$ $x_3,0$ $x_4,0$ $x_5,0$

$x_1,1$ $x_2,1$ $x_3,1$ $x_4,1$ $x_5,1$

$y_{i,b} = F(x_{i,b})$

$y_{1,0}$ $y_{2,0}$ $y_{3,0}$ $y_{4,0}$ $y_{5,0}$

$y_{1,1}$ $y_{2,1}$ $y_{3,1}$ $y_{4,1}$ $y_{5,1}$

sk

pk
Lamport Signatures

Sign(sk, m): \((x_{i,m_i})_{i=1,\ldots,n}\)

Ver(pk, m, σ): \(F(x_{i,m_i}) = y_{i,m_i}\)
Theorem: If $F$ is a secure OWF, then $(Gen, Sign, Ver)$ is a (weakly) secure one-time signature scheme
Proof

\[ y_{1,0} \quad y_{2,0} \quad y_{3,0} \quad y_{4,0} \quad y_{5,0} \]

\[ y_{1,1} \quad y_{2,1} \quad y_{3,1} \quad y_{4,1} \quad y_{5,1} \]

\[ x_{1,0} \quad x_{2,0} \quad x_{3,0} \quad x_{4,0} \quad x_{5,0} \]

\[ x_{1,1} \quad x_{2,1} \quad x_{3,1} \quad x_{4,1} \quad x_{5,1} \]
Proof

Since \( m^* \neq m \), \( \exists \ i \ s.t. \ m^*_i \neq m_i \)

Suppose we know \( i, m_i = 1-b, m^*_i = b \)

Construct adversary that inverts OWF
Proof

View of exactly as in 1-time CMA experiment, assuming

- $i$th bit of $m = b$
- $i$th bit of $m^* = 1-b$

If always chooses $m, m^*$ with these properties, and forges with probability $\varepsilon$, then inverts with probability $\varepsilon$
Proof

In general, may choose $m,m^*$ to differ at arbitrary places
• May be randomly chosen, may depend on $pk$, may even depend on $\sigma$
• May never be at certain places

How do we make still succeed?
Proof

\[ y^* \]

\[ i, b \leftarrow [n] \times \{0, 1\} \]

If need \( x_{i, b} \), abort

If no \( x_{i, b} \), abort
Proof

- \( pk \) independent of \((i,b)\)
- \( m \) independent of \((i,b)\)
- Therefore, \( \Pr[m_i=1-b]=\frac{1}{2} \)

Conditioned on \( m_i=1-b \),
- Signing succeeds
- \( \sigma \) independent of \( i \)
- \( \text{🤖 forgives with probability } \varepsilon \), independent of \( i \)
Proof

We know if fly forgives, then $m^* \neq m$

Since $m^*$ independent of $i$, have prob at least $1/n$
that $m^*_i = 1 - m_i = b$

In this case, succeeds in inverting $y^*$
• $\text{Prob} = \frac{1}{2} \times \epsilon \times \frac{1}{n} = \epsilon/2n$
Limitations of Lamport Signatures

Only weakly secure
• Why?
• How to fix?

$|pk|, |\sigma| >> |m|$
• How to fix?
Theorem: Given a secure OWF, it is possible to construct a strongly secure 1-time signature scheme where $|m| \gg |pk|, |\sigma|$
Signing Multiple Messages

Once adversary sees two signed messages, security is lost (why?)

How do we sign multiple messages?
Signature Chaining

$m_1, \sigma_1 \leftarrow \text{Sign}(sk_1, m_1)$

$\text{Ver}(pk_1, m_1, \sigma_1)$
Signature Chaining

$m_1$

$m_1, \sigma_1 = (pk_2, \sigma_1')$

$\sigma_1' \leftarrow \text{Sign}(sk_1, (m_1, pk_2))$

$(sk_2, pk_2) \leftarrow \text{Gen}()$

$\text{Ver}(pk_1, (m_1, pk_2), \sigma_1')$
Signature Chaining

\[
\begin{align*}
    m_2, \sigma_2 \\
    \sigma' \leftarrow \text{Sign}(sk_2, m_2) \\
    (sk_2, pk_2) \leftarrow \text{Gen}() \\
    \text{Ver}(pk_2, m_2, \sigma_2)
\end{align*}
\]
Signature Chaining

Idea: Bob can be assured that $pk_2$ was in fact generated by Alice

• If Eve tampered with $pk_2$, then signature on first message would have been invalid

Therefore, Alice can sign $m_2$ using $sk_2$, and Eve cannot produce a forgery $m_2'$ with valid signature

Can repeat process to sign arbitrarily many messages
Signature Chaining

\[ \text{Ver} (pk_2, (m_2, pk_3), \sigma_2') \]

\[ (sk_3, pk_3) \leftarrow \text{Gen}() \]

\[ (sk_2, pk_2) \leftarrow \text{Gen}() \]

\[ m_2, \sigma_2 = (pk_3, \sigma_2') \]

\[ \sigma_1' \leftarrow \text{Sign}(sk_2, (m_2, pk_3)) \]

\[ \text{ß} \leftarrow \text{Gen}() \]
Limitations

Alice and Bob must stay synchronized
• Else, Bob won’t be using correct public key to verify

If many users, every pair needs to be synchronized
• What if Alice is sending messages to Bob and Charlie?
(Almost) Stateless Signature Chaining

\[ m_2, \sigma_2 = (m_1, pk_2, \sigma_1', pk_3, \sigma_2') \]

\[ \sigma_1' \leftarrow \text{Sign}(sk_2, (m_2, pk_3)) \]

\[ (sk_2, pk_2) \leftarrow \text{Gen()} \]

\[ (sk_3, pk_3) \leftarrow \text{Gen()} \]

\[ \text{Ver}(pk_1, (m_1, pk_2), \sigma_1') \]

\[ \text{Ver}(pk_2, (m_2, pk_3), \sigma_2') \]
Still Limitations

Now Bob (and Charlie, etc) are stateless

However, Alice is still stateful
• Needs to remember all messages sent
• Signature length grows with number of messages signed
Signature Trees

\[
\begin{align*}
\sigma_0 & \leftarrow \text{Sign}(sk_0, (pk_0, pk_1)) \\
\sigma_1 & \leftarrow \text{Sign}(sk_1, (pk_0, pk_1)) \\
\sigma_{00}, \sigma_{01}, \sigma_{10}, \sigma_{11} & \leftarrow 
\end{align*}
\]
Signature Trees

To sign $m_i$,
- Compute $\sigma_i \leftarrow \text{Sign}(sk_i, m_i)$, where $sk_i$ is the $i$th leaf
- Must include $pk_i$ in signature so Bob can verify $\sigma_i$
- Must authenticate $pk_i$, so include $\sigma_{P(i)}$ (and $pk_{S(i)}$)
- Must include $pk_{P(i)}$ so Bob can verify $\sigma_{P(i)}$
- Must auth $pk_{P(i)}$, so include $\sigma_{P(P(i))}$ (and $pk_{S(P(i))}$)
- ...
Comparison to Chaining

Limitations:
• Bounded number of messages ($2^d$)
• Still requires Alice to keep state (all the $sk$’s, $pk$’s).
  Size of state $\approx 2^d$

Advantages:
• Signature size $\approx d$, logarithmic in number of messages signed
Avoid Large State?

Alice keeps PRF key $k$ as part of secret key

- For all internal nodes or leaves $i$,

\[(sk_i, pk_i) \leftarrow \text{Gen}(; \ PRF(k, i))\]

- Alice never stores signatures or public keys
- Instead, she computes needed signatures/public keys on the fly
Unbounded Messages

Set $d = \lambda$

- Can now sign up to $2^\lambda$ messages (exponential)
- Signature size $\approx d = \lambda$, so short signatures
- Size of state independent of $d$, so short
- Time to compute signature?
  - Only need $pk$’s, $\sigma$’s on path from root to leaf, plus neighbors
  - Only $O(d) = O(\lambda)$ terms
  - Can efficiently compute from PRF key $k$
Fully Stateless?

So far, still need to keep state to remember which leaf we should use next

However, now we can do something different:
• Instead of choosing leafs sequentially, just choose leaf at random
• Except with probability $O(|\text{messages}|^2/2^d)$, never use the same leaf twice
Putting it Together

\[ \text{pk}_\emptyset \quad \text{sk} = (\text{sk}_\emptyset, \ k) \]

\[ i \leftarrow \{0, \ldots, 2^d - 1\} \]
Putting it Together

\[ \text{sk} = (\text{sk}_\emptyset, k) \]

- \( (\text{sk}_0, \text{pk}_0) \leftarrow \text{Gen}(; \text{PRF}(k, 0)) \)
- \( (\text{sk}_1, \text{pk}_1) \leftarrow \text{Gen}(; \text{PRF}(k, 1)) \)
- \( (\text{sk}_{00}, \text{pk}_{00}) \leftarrow \text{Gen}(; \text{PRF}(k, 00)) \)
- \( (\text{sk}_{01}, \text{pk}_{01}) \leftarrow \text{Gen}(; \text{PRF}(k, 01)) \)
- \( \ldots \)
- \( \sigma_\emptyset \leftarrow \text{Sign}(\text{sk}_\emptyset, (\text{pk}_0, \text{pk}_1)) \)
- \( \sigma_0 \leftarrow \text{Sign}(\text{sk}_0, (\text{pk}_{00}, \text{pk}_{01})) \)
- \( \ldots \)
- \( \sigma \leftarrow \text{Sign}(\text{sk}_i, m) \)

Output all \( \text{pk}_j \)'s and all \( \sigma \)'s as signature
Putting it Together

OWF to get 1-time signatures (with large pk’s, σ’s)

Hash first with target collision resistances
• 1-time signatures with small pk’s, σ’s

Create tree of signatures (stateful scheme)

Make stateless by using a PRF
What’s Known

OWP
CRH
OWF
PRG
Com
TCR
PRF
MAC
Auth
SKE
PRP
Enc
CPA
- PKE
CCAPKE
Sig
Theorem: Given a secure OWF, it is possible to construct a strongly CMA-secure signature scheme.
Practical Use?

Lamport signatures are fast:
• Signing is just revealing part of your secret key
• Verifying is just a few OWF evaluations

Tree-based signatures are a bit slower
• Need to generate many signatures
• Need to generate many public keys
• Need many PRF evals
Practical Use?

Main limitation: Signature size
• Basic Lamport: 128 bits per message bit
• With hashing, need to sign 256 bit messages
• For signature trees, signature consists of $d$ Lamport signatures (plus public keys)
  • $d$ must be big enough to prevent collisions
  • E.g. $d = 100$

Overall signature size: around a megarbit
What’s the Smallest Signature?

Signature Trees: 1megabits

RSA Hash-and-Sign: 2 kilobits

ECDSA: around 512 bits

BLS: 256 bits

Are 128-bit signatures possible?
Obfuscation-Based Signatures

Let \((\text{MAC, Ver})\) be a message authentication code

\[
\text{Gen()}: \ k \leftarrow K
\]
\[
\begin{align*}
&\text{sk} = k \\
&\text{pk} = \text{Obf( Ver(k, . , . ) )}
\end{align*}
\]

\[
\begin{align*}
\text{Sign}(sk, m) &= \text{MAC}(k, m) \\
\text{Ver}(pk, m, \sigma) &= \text{pk}(m, \sigma)
\end{align*}
\]

Signature size: 128 bits!
- But running time, public key size is horrible
Next Time

Identification protocols