Sigs. with special properties

Fast one-time signatures and applications
One-time signatures: definition

Suppose signing key is used to sign a single message
Can we give a simple (fast) construction $SS = (Gen, S, V)$?

A wins if $V(vk, m, \sigma) = \text{`accept'}$ and $m \neq m_1$

Security: for all “efficient” $A$, $Adv_{1-SIG}[A, SS] = Pr[ A \text{ wins}] \leq \text{negl}$
Application: authenticating streams

1. Next section: secure one-time sigs $\Rightarrow$ secure many-time sigs

2. Authenticating a video stream:

Too slow: signing every packet with $sk$
Solution using a fast one-time sig

(sk,vk): key-pair for a many-time signature scheme

(Gen_{1T}, S_{1T}, V_{1T}): secure one-time signature (fast)

Packet #0: (sk_1,vk_1) ← Gen_{1T}, \quad \sigma_0 ← S(\text{sk}, (\text{data}_0, \text{vk}_1))

Packet #1: (sk_2,vk_2) ← Gen_{1T}, \quad \sigma_1 ← S_{1T}(sk_1, (\text{data}_1, \text{vk}_2))

Packet #2: (sk_3,vk_3) ← Gen_{1T}, \quad \sigma_2 ← S_{1T}(sk_2, (\text{data}_2, \text{vk}_3))
Recipient accepts packet #2 = (data₂, vk₃, σ₂) once it verifies σ₂

How does the recipient verify the signature σ₂ in packet #2?

Accept if σ₀ and σ₁ were valid and:

- V₁T(vk₃, (data₂, vk₃), σ₂) = “accept”
- V(vk, (data₂, vk₃), σ₂) = “accept”
- V₁T(vk₂, (data₂, vk₃), σ₂) = “accept”
- V(vk₂, (data₂, vk₃), σ₂) = “accept”
Application: authenticating streams

Practical difficulties:

• Packet loss, out of order delivery
• Many solutions: see further reading at end of module

Authenticating streams with a MAC:

• Harder, but can be done: TESLA
End of Segment
Sigs. with special properties

Constructing fast one-time signatures
One-time signatures

Secure when sk only signs a single message

Attacker: gets vk and can ask for sig. on any single \( m_1 \) of her choice. should be unable to forge signature on \( m \neq m_1 \)

This module: one-time sigs from fast one-way functions (OWF)

- \( f: X \rightarrow Y \) is a OWF if
  1. \( f(x) \) is efficiently computable,
  2. hard to invert on random \( f(x) \)

- Examples: \( f(x) = AES(x, 0^{128}) \), \( f(x) = SHA256(x) \)
Lamport one-time signatures (simple)

\[ f: X \to Y \] a one-way function.  

\[ \text{Msg space: } M = \{0,1\}^{256} \]

**Gen:** generate \( 2 \times 256 \) random elements in \( X \)
Lamport one-time signatures (simple)

f: X → Y a one-way function. Msg space: M = \{0,1\}^{256}

Gen: generate 2×256 random elements in X

\[
\begin{align*}
m &= 0 & 1 & 1 & \ldots & 0 & 0 \\
\end{align*}
\]

\[S(sk, m): \sigma = \text{(pre-images corresponding to bits of } m)\]
Lamport one-time signatures (simple)

f: X → Y a one-way function.  
Msg space:  M = \{0,1\}^{256}

Gen: generate 2×256 random elements in X

σ = \begin{bmatrix}
\vdots \\
0 & 1 & 1 & \cdots & 0 & 0
\end{bmatrix} \in X^{256} \text{ (2KB)}

S(sk, m):  \sigma = (\text{pre-images corresponding to bits of } m)
Lamport one-time signatures (simple)

\[ f: X \rightarrow Y \text{ a one-way function.} \]

\[ \text{Msg space: } M = \{0,1\}^{256} \]

Gen: generate 2\times256 random elements in \( X \)

\[ \sigma = \begin{bmatrix} \text{\ddots} \\ m = \begin{bmatrix} 0 \& 1 \& 1 \& \text{\ddots} \\ 0 \& 0 \end{bmatrix} \end{bmatrix} \in X^{256} \quad \text{(2KB)} \]

\[ \text{V}( \text{vk}, m, \sigma ): \text{ accept if all pre-images in } \sigma \text{ match values in } \text{vk} \]
Very fast signature system. Will prove one-time security in a bit.

Is it two-time secure? That is, if \textit{sk} is used to sign \textit{two} messages, can an attacker do an existential forgery?

- No, one-time security implies two-time security
- It depends on the one-way function used
- The attacker can ask for a signature on $0^{128}$ and on $1^{128}$. He gets all of \textit{sk} which he can use to sign new messages.
Abstraction: cover free set systems

Sets: \( S_1, S_2, \ldots, S_{2^{256}} \subseteq \{1, \ldots, n\} \)

Def: \( S = \{S_1, S_2, \ldots, S_{2^{256}}\} \) is **cover-free** if \( S_i \nsubseteq S_j \) for all \( i \neq j \)

Example: if all sets in \( S \) have the same size \( k \) then \( S \) is cover free
Abstract Lamport signatures

\( f: X \rightarrow Y \) a one-way function.  
Msg space:  \( M = \{0,1\}^{256} \)

\( S = \{S_1, S_2, \ldots, S_{2^{256}}\} \) is **cover-free** over \( \{1,\ldots,n\} \)

\( H: \{0,1\}^{256} \rightarrow S \) a bijection (one-to-one)

**Gen:** generate \( n \) random elements in \( X \)

\[ \begin{array}{cccccc}
1 & f & S_1 & f & S_2 & f & \cdots & f & S_{2^{256}} & \vdots \\
1 & S_1 & v_1 & S_2 & v_2 & \cdots & S_{2^{256}} & v_{2^{256}} & \in X^n \\
 & & & & & & & & \in Y^n \\
\end{array} \]
Abstract Lamport signatures

\( f: X \rightarrow Y \) a one-way function. 
\( \text{Msg space: } M = \{0,1\}^{256} \)

\( S = \{S_1, S_2, \ldots, S_{2^{256}}\} \) is cover-free over \( \{1,\ldots,n\} \)

\( H: \{0,1\}^{256} \rightarrow S \) a bijection (one-to-one)

\( \text{Gen: generate } n \text{ random elements in } X \)

\[ \sigma = \left[ \begin{array}{cccc} 1 & \text{blue} & \text{blue} & \text{blue} & \ldots & \text{blue} & n \end{array} \right] \]

\[ \sigma = \left( \text{pre-images corresponding to elements of } H(m) \right) \]
Why cover free?

Suppose $S$ were not cover free

$\Rightarrow$ exists $m_1, m_2$ such that $H(m_1) \subseteq H(m_2)$

$\Rightarrow$ signature on $m_2$ gives signature on $m_1$

$\sigma_{m_1} = \text{pre-images corresponding to elements of } H(m)$

$\sigma_{m_2} = \text{pre-images corresponding to elements of } H(m)$

$S(sk, m): \sigma = \{ \text{pre-images corresponding to elements of } H(m) \}$
Security statement

**Thm:** if $f: X \rightarrow Y$ is one-way and $S$ is cover-free then Lamport signatures (Lam) are one-time secure.

$$\forall A \exists B: \text{Adv}_{1-SIG}[A, Lam] \leq n \cdot \text{Adv}_{OWF}[B, f]$$

**Proving security:**

$y = f(x)$

Proving security:

Signature Forger

Adversary (A)
Proving security

\[ y = f(x) \]

choose: \( i \leftarrow \{1, \ldots, n\} \)

\[ x_1, \ldots, x_n \leftarrow X \]

\[ \text{vk=} \quad f(x_1) \quad \cdots \quad f(x_{i-1}) \quad y \quad f(x_{i+1}) \quad \cdots \quad f(x_n) \]

Dan Boneh
Parameters \quad (f: X \to Y \text{ where } X = Y)

\[ S = \{ S_1, S_2, \ldots, S_{2^{256}} \} \text{ is cover-free over } \{1, \ldots, n\} \]

In particular: \[ S = \{ \text{ all subsets of } \{1, \ldots, n\} \text{ of size } k \} \]

\[ v_k \in Y^n \implies v_k \text{ size } = (n \text{ elements of } Y) \]
\[ \text{sig. size } = (k \text{ elements of } X) \]

Msg-space = \{0,1\}^{256} \implies |S| = (n \text{ choose } k) \geq 2^{256}

• To shrink signature size, choose small \( k \)
  example: \( k=32 \implies n \geq 3290 \)

• For optimal (sig-size + vk-size) choose \( n = 261, k = 123 \)
  \( (\text{sig-size} + \text{vk-size}) \approx 1.5 \times 256 \text{ elements of } X \)
Further improvement: Winternitz

**Gen:** generate $n$ random elements in $X$: \((f: X \rightarrow X)\)

- Depth $d$
- \(sk \in X^n\)
- \(vk \in X^n\)
Further improvement: Winternitz

$H: \{0,1\}^{256} \rightarrow \{0,1,\ldots,d\}^n$

$S(sk, m): \sigma = \left(\text{pre-images indicated by } H(m)\right)$
Further improvement: Winternitz

\[ H: \{0,1\}^{256} \rightarrow \{0,1,\ldots,d\}^n \]

\[ \text{ex: } H(0^{256}) = (2, 1, 3, 0, \ldots, 0, 1) \]

\[ S(sk, m): \quad \sigma = \left( \text{pre-images indicated by } H(m) \right) \]
For what $H$ is this a secure one-time signature?

Suppose

- $H(0^{256}) = (2, 1, 3, 0, 0, 1)$
- $H(1^{256}) = (2, 2, 3, 1, 1, 2)$

Is the signature one-time secure?

- No, from a sig. on $0^{256}$ one can construct a sig. on $1^{256}$
- No, from a sig. on $1^{256}$ one can construct a sig. on $0^{256}$
- Yes, the signature is one-time secure
- It depends on how $H$ behaves at other points
Optimized parameters

For one-time security need that:
for all $m_0 \neq m_1$ we have $H(m_0)$ does not “cover” $H(m_1)$

Parameters:

• Time(sign) = Time(verify) = $O(n \cdot d)$
• $vk$ size = sig. size = (n elements in X)
• $msg$-space = $\{0,1\}^{256} \Rightarrow n > 256 / \log_2(d)$ (approx.)

\[(vk \text{ size})+(sig. \text{ size}) \approx 256 \times (2/\log_2(d)) \text{ elems. of } X\]

For Lamport: \[(vk \text{ size})+(sig. \text{ size}) \approx 256 \times (1.5) \text{ elems. of } X\]
End of Segment
Sigs. with special properties

One-time signatures $\Rightarrow$ many-time signatures
Review

Recall: one-time signatures need not be 2-time secure
example: Lamport signatures

Goal: convert any one-time signature into a many-time signature

Main tool: collision resistant hash functions
Construction

\( \text{(Gen}_{1T}, S_{1T}, V_{1T}) \): secure one-time signature (fast)

Four-time signature: (stateful version)

• Gen:

\[
\text{Gen}_{1T} \rightarrow (vk_{0123}, sk_{0123})
\]

\[
(vk_{01}, sk_{01}) \quad (vk_{23}, sk_{23})
\]

\[
(vk_{0}, sk_{0}) \quad (vk_{1}, sk_{1}) \quad (vk_{2}, sk_{2}) \quad (vk_{3}, sk_{3})
\]
Construction

(Gen\textsubscript{1T}, S\textsubscript{1T}, V\textsubscript{1T}): secure one-time signature (fast)

Four-time signature: (stateful version)

• Gen:

\begin{align*}
\text{Gen:} & \quad (\text{vk}_{0123}, \sigma_{0123}) \\
& \quad S_{1T}(sk, (\text{vk}_{01}, \text{vk}_{23})) \\
& \quad (\text{vk}_{01}, sk_{01}) \quad (\text{vk}_{23}, sk_{23}) \\
& \quad (\text{vk}_0, sk_0) \quad (\text{vk}_1, sk_1) \quad (\text{vk}_2, sk_2) \quad (\text{vk}_3, sk_3)
\end{align*}
Construction

(Gen₁T, S₁T, V₁T): secure one-time signature (fast)

Four-time signature: (stateful version)

- Gen:

\[ \text{vk}_{0123}, \text{σ}_{0123} \]

\[ (\text{vk}_{01}, \text{σ}_{01}) \]

\[ (\text{vk}_{0}, \text{sk}_{0}) \]

\[ (\text{vk}_{1}, \text{sk}_{1}) \]

\[ (\text{vk}_{23}, \text{σ}_{23}) \]

\[ (\text{vk}_{2}, \text{sk}_{2}) \]

\[ (\text{vk}_{3}, \text{sk}_{3}) \]
Construction

$(Gen_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)

Four-time signature: (stateful version)

Sig. on msg $m_0$:

\[(\sigma_{0123}, \sigma_{01}, \sigma_0,\]
\[vk_{01}, vk_{23}, vk_0, vk_1)\]
Construction

(Gen₁₁, S₁₁, V₁₁): secure one-time signature (fast)

Four-time signature: (stateful version)

Sig. on msg $m_1$:

\[
(\sigma_{0123}, \sigma_{01}, \sigma_1, \text{vk}_0, \text{vk}_1, \text{vk}_01, \text{vk}_2, \text{vk}_3, \text{vk}_{01}, \text{vk}_{23})
\]
Construction

(\text{Gen}_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)

Four-time signature: (stateful version)

Sig. on msg $m_2$:

$\left( \sigma_{0123}, \sigma_{01}, \sigma_2, \right.
\left. \text{vk}_{01}, \text{vk}_{23}, \text{vk}_2, \text{vk}_3 \right)$
Construction

\((\text{Gen}_{1T}, S_{1T}, V_{1T})\): secure one-time signature (fast)

Four-time signature: (stateful version)

Sig. on msg \(m_3\):

\(\left(\sigma_{0123}, \sigma_{01}, \sigma_3, vk_{01}, vk_{23}, vk_2, vk_3\right)\)
More generally: $2^d$-time signature

Tree of depth $d$:
- Every signature contains $d+1$ one-time signatures along with associated $vk$’s

Tree is generated on-the fly:
- Signer stores only $d$ secret keys at a time

Stateful signature:
- Signer maintains a counter indicating which leaf to use for signature
- Every leaf must only be used once!
Optimized $2^d$-time signatures

Combined with Lamport signatures:
• collision resistant hash funs $\Rightarrow$ many-time signature

With further optimizations:
• For $2^{40}$ signatures: signature size is $\approx 5$KB
  ... signing time is about the same as RSA signatures

• Recall: RSA sig size is 256 bytes  (2048 bit RSA modulus)
End of Segment
Sigs. with special properties

Super-fast online signatures
Goals

Problem: generating RSA, ECDSA, BLS signatures can be slow
- On low power devices

Goal:
- Do heavy signature computation before message is known
- Quickly output signature once user supplies message
Method 1: using one-time sigs

(Gen, S, V): secure many-time signature (slow)

(Gen_{1T}, S_{1T}, V_{1T}): secure one-time signature (fast)

- Gen → (sk, vk)
- PreSign(sk): (sk_{1T}, vk_{1T}) ← Gen_{1T}, σ ← S(sk, vk_{1T})
- S_{online}( (σ, sk_{1T}, vk_{1T}), m): σ_{1T} ← S_{1T}(sk_{1T}, m)
  output σ^{*} ← (vk_{1T}, σ, σ_{1T})
- V_{online}(vk, m, σ^{*}=(vk_{1T}, σ, σ_{1T})): accept if V(vk, vk_{1T}, σ) = V_{1T}(vk_{1T}, m, σ_{1T}) = “accept”
Method 1: using one-time sigs

One-time sigs. \( \Rightarrow \) fast-online sigs.

Problem: Lamport results in very long signatures

A more suitable one-time signature:

• Hard Dlog in group \( G \) \( \Rightarrow \) secure one-time sigs. with fast signing
  
  – Signature size: if \( |G|=p \) then signature is \( (r,s) \in (\mathbb{Z}_p)^2 \)
  
  – How: see homework problem

\[ 64 \text{ bytes} \]
Better method: chameleon hash

G: finite cyclic group of order p. \( g, h=g^\alpha \in G \) generators.

Define: \( H(m,r) = g^r \cdot h^m \in G \)

Properties:

- \( H(m,r) \) can be efficiently evaluated
- \( H \) is collision resistant if Dlog in \( G \) is hard \( (\text{collision} \rightarrow \alpha = \text{Dlog}_g(h)) \)
- If \( \alpha \) is known: given \( m \) and \( t \) can find \( r \) s.t. \( H(m,r) = h^t \)
  \[ r = (t-m) \cdot \alpha. \]
  Indeed: \( H(m,r) = g^r \cdot h^m = \)
Fast online signatures

$(\text{Gen}, S, V)$: secure many-time signature (slow)

$G$: finite cyclic group of order $p$. \quad g, h = g^\alpha \in G \quad \text{rand. generators}$

- $\text{Gen} \rightarrow (sk, vk) \quad , \quad sk^* = (sk, \alpha)$
- $\text{PreSign}(sk^*)$: random $t \leftarrow Z_p \quad , \quad \sigma \leftarrow S(sk, h^t)$
- $S_{\text{online}} \left( (\sigma, \alpha, t, h^t), m \right)$: $r \leftarrow (t-m) \cdot \alpha \quad , \quad \text{output} \quad \sigma^* \leftarrow (\sigma, h^t, r)$
- $V_{\text{online}} \left( vk, m, \sigma^*=(\sigma, h^t, r) \right)$:
  
  accept if \quad $V(vk, h^t)$ = “accept” \quad and \quad $H(m, r) = h^t$
Fast online signatures

Shorter signatures than one-time sigs. method:
• Total overhead is only 64 bytes
• Signature time: one multiplication in $Z_p$

Security:
• A forger can be used to either
  (1) forge signatures for $(Gen, S, V)$, or
  (2) find collisions on $H(m,r)$
Fast online signatures have a fast online signing time.

If we count the entire signing time (i.e. PreSign + Sign), would the time be better or worse than a standard signature like RSA?

- Online signatures are always faster than regular signatures
- The PreSign step uses a regular signatures, so overall they cannot be faster than a regular signature
- It depends on which online signature is used

Note: signature verification time is always worse than regular sigs.
End of Segment
Sigs. with special properties

Blind signatures
Problem: digital cash (centralized system)

I am Alice: withdraw 1$

coin_{ID} \leftarrow \{0,1\}^{256}

\sigma \leftarrow S(sk_{bank}, \text{coin}_{ID})

(coin_{ID}, \sigma)

\text{anonymous channel (Tor)}

Is \text{coin}_{ID} spent?

no

\text{shop}

\text{vk}_{bank}

Who did I talk to?

For simplicity, assume only one bank and all coins worth 1$.

It's Alice!
Solution: blind signatures

Goal: we want Bank to sign coin_{ID}, but without knowing coin_{ID}

Where:
(1) $\sigma$ is a valid signature on $m$: $V(vk, m, \sigma) = \text{"accept"}$
(2) $m' \leftarrow \text{Blind}(m)$ is independent of $m$
   • That is, $m'$ reveals no "information" about $m$
Blind signatures: security

New definition of existential forgery:
adversary asks for q blind signatures, and
outputs (q+1) message/signature pairs

\[\text{Chal.} (sk, vk) \leftarrow \text{Gen} \]

\[vk \]

\[m_i' \quad i=1,\ldots,q \]

\[\sigma_i' \leftarrow \text{SignBlind}(sk, m_i') \]

\[\text{Adv. A} \]

\[\{(m_i, \sigma_i)\}_{i=1}^{q} \]

A wins if \(V(vk, m_i, \sigma_i) = \text{`accept'}\) for all \(i=1,\ldots,q+1\)

Security: for all “efficient” A, \(\text{Adv}_{\text{Blind}}[A, SS] = \Pr[\text{A wins}] \leq \text{negl}\)
Blind signatures: applications

- Anonymous digital cash
- Anonymous voting systems
  - Election results are known, but not who voted how
- Adaptive oblivious transfer (week 4)
Simple Constructions: RSA and BLS

BLS review: $G$ finite group of order $p$ with a pairing

$$sk = \alpha \in \mathbb{Z}_p, \quad vk = (g, g^\alpha), \quad H: M \rightarrow G$$

$$S(sk, m) = H(m)^\alpha \in G$$

Indeed:
$$\sigma = (m')^\alpha / (g^\alpha)^r = $$

Independent of $m$

Same method also works for RSA. Problem: security under strong assumption.
Suppose the signature scheme is changed so that the random \( r \) is chosen as \( r \leftarrow \{0,1,...,16\} \).

Would the resulting scheme be a secure blind signature?

- No, an attacker can ask one query and generate two signatures
- Yes, this has no impact on security and blindness
- No, the sig. scheme is not blind: \( m' \) is not independent of \( m \)
- It depends on the hash function \( H \)
Further Reading

• Hash Based Digital Signature Schemes. C. Dods, N. Smart, M. Stam, 2005


• Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. L. Reyzin, N. Reyzin, 2002

• Improved Online/Offline Signature Schemes. A. Shamir, Y. Tauman, 2001


• Compact E-Cash. J. Camenisch, S. Hohenberger, A. Lysyanskaya, 2005
End of Segment