Lecture 12: Authentication

The Setting

- Alice wants to send a message m to Bob in such a manner that upon receipt, Bob can determine whether the message arrived untampered or not

The Setting

- Alice wants to send a message m to Bob in such a manner that upon receipt, Bob can determine whether the message arrived untampered or not
- Want: Digital analogue of physical signatures

The Setting

- Alice wants to send a message m to Bob in such a manner that upon receipt, Bob can determine whether the message arrived untampered or not
- Want: Digital analogue of physical signatures
- Alice ("signer") signs a message m to produce a signature σ

The Setting

- Alice wants to send a message m to Bob in such a manner that upon receipt, Bob can determine whether the message arrived untampered or not
- Want: Digital analogue of physical signatures
- Alice ("signer") signs a message m to produce a signature σ
- Bob ("verifier") can verify that σ is indeed generated for m

The Setting

- Alice wants to send a message m to Bob in such a manner that upon receipt, Bob can determine whether the message arrived untampered or not
- Want: Digital analogue of physical signatures
- Alice ("signer") signs a message m to produce a signature σ
- Bob ("verifier") can verify that σ is indeed generated for m
- Adversary cannot *forge* a signature

Two Types

1. Private Key: Message Authentication Codes

Two Types

1. Private Key: Message Authentication Codes
2. Public Key: Digital Signatures

Message Authentication Code (MAC)

Message Authentication Code (MAC)

- Signer and Verifier “share a secret”

Message Authentication Code (MAC)

- Signer and Verifier "share a secret"
- **Key Generation**: $\mathsf{Gen}(1^n)$ outputs secret key k

Message Authentication Code (MAC)

- Signer and Verifier "share a secret"
- **Key Generation**: $\mathsf{Gen}(1^n)$ outputs secret key k
- **Sign**: $\mathsf{Tag}_k(m)$ outputs a tag σ

Message Authentication Code (MAC)

- Signer and Verifier “share a secret”
- **Key Generation**: $\mathsf{Gen}(1^n)$ outputs secret key k
- **Sign**: $\mathsf{Tag}_k(m)$ outputs a tag σ
- **Verify**: $\mathsf{Ver}_k(m, \sigma)$ is 1 if and only if σ is a valid tag of m under the secret key k

Message Authentication Code (MAC)

- Signer and Verifier "share a secret"
- **Key Generation**: $\mathsf{Gen}(1^n)$ outputs secret key k
- **Sign**: $\mathsf{Tag}_k(m)$ outputs a tag σ
- **Verify**: $\mathsf{Ver}_k(m, \sigma)$ is 1 if and only if σ is a valid tag of m under the secret key k

Security: An adversary can observe multiple (message,tag) pairs of its choice, but still cannot forge a tag on a new message

MAC: Algorithms

- $k \leftarrow \mathsf{Gen}(1^n)$

MAC: Algorithms

- $k \leftarrow \mathsf{Gen}(1^n)$
- $\sigma \leftarrow \mathsf{Tag}_k(m)$

MAC: Algorithms

- $k \leftarrow \mathsf{Gen}(1^n)$
- $\sigma \leftarrow \mathsf{Tag}_k(m)$
- $\mathsf{Ver}_k\colon \mathcal{M} \times \mathcal{T} \rightarrow \{0,1\}$

MAC: Algorithms

- $k \leftarrow \mathsf{Gen}(1^n)$
- $\sigma \leftarrow \mathsf{Tag}_k(m)$
- $\mathsf{Ver}_k \colon \mathcal{M} \times \mathcal{T} \rightarrow \{0, 1\}$
- Correctness:
  $\Pr[k \leftarrow \mathsf{Gen}(1^n), \sigma \leftarrow \mathsf{Tag}_k(m) \colon \mathsf{Ver}_k(m, \sigma) = 1] = 1$

MAC: Algorithms

- $k \leftarrow \mathsf{Gen}(1^n)$
- $\sigma \leftarrow \mathsf{Tag}_k(m)$
- $\mathsf{Ver}_k : \mathcal{M} \times \mathcal{T} \rightarrow \{0, 1\}$
- Correctness:
  $\Pr[k \leftarrow \mathsf{Gen}(1^n), \sigma \leftarrow \mathsf{Tag}_k(m) : \mathsf{Ver}_k(m, \sigma) = 1] = 1$
- Security (UF-CMA): For all n.u. PPT adversary $\mathcal{A}$ there exists a negligible $\nu(\cdot)$ such that:

$$\Pr\left[\substack{k \leftarrow \mathsf{Gen}(1^n) \\ (m,\sigma) \leftarrow \mathcal{A}^{\mathsf{Tag}_k(\cdot)}(1^n)} : \substack{\mathcal{A} \text{ did not query } m \wedge \\ \mathsf{Ver}_k(m,\sigma)=1}\right] \leqslant \nu(n)$$

MAC: Construction

MAC: Construction

Theorem

$PRF \implies MAC$

MAC: Construction

Theorem

$PRF \implies MAC$

- $\mathsf{Gen}(1^n)$: Output $k \xleftarrow{\$} \{0,1\}^n$

MAC: Construction

Theorem

$PRF \implies MAC$

- $\mathsf{Gen}(1^n)$: Output $k \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{Tag}_k(m)$: Output $f_k(m)$

MAC: Construction

Theorem

$PRF \implies MAC$

- $\mathsf{Gen}(1^n)$: Output $k \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{Tag}_k(m)$: Output $f_k(m)$
- $\mathsf{Ver}_k(m,\sigma)$: Output $f_k(m) \stackrel{?}{=} \sigma$

MAC: Construction

Theorem

$PRF \implies MAC$

- $\mathsf{Gen}(1^n)$: Output $k \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{Tag}_k(m)$: Output $f_k(m)$
- $\mathsf{Ver}_k(m,\sigma)$: Output $f_k(m) \stackrel{?}{=} \sigma$
- Think: Proof?

One-time MAC

- Weaker Security: Adversary is allowed only one query

One-time MAC

- Weaker Security: Adversary is allowed only one query
- Advantage: Unconditional security!

One-time MAC

- Weaker Security: Adversary is allowed only one query
- Advantage: Unconditional security!
- Analogue of OTP for authentication

One-time MAC

- Weaker Security: Adversary is allowed only one query
- Advantage: Unconditional security!
- Analogue of OTP for authentication
- Think & Read

Digital Signature

- Only Signer can sign but everyone can verify

Digital Signature

- Only Signer can sign but everyone can verify
- **Key Generation**: $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$

Digital Signature

- Only Signer can sign but everyone can verify
- **Key Generation**: $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
- **Sign**: $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$

Digital Signature

- Only Signer can sign but everyone can verify
- **Key Generation**: $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
- **Sign**: $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
- **Verify**: $\mathsf{Ver}_{pk}(m, \sigma) \colon \mathcal{M} \times \mathcal{S} \to \{0, 1\}$

Digital Signature

- Only Signer can sign but everyone can verify
- **Key Generation**: $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
- **Sign**: $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
- **Verify**: $\mathsf{Ver}_{pk}(m, \sigma) \colon \mathcal{M} \times \mathcal{S} \rightarrow \{0, 1\}$
- Correctness:

$$\Pr[(sk, pk) \leftarrow \mathsf{Gen}(1^n), \sigma \leftarrow \mathsf{Sign}_{sk}(m) \colon \mathsf{Ver}_{pk}(m, \sigma) = 1] = 1$$

- Only Signer can sign but everyone can verify
- **Key Generation**: $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
- **Sign**: $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
- **Verify**: $\mathsf{Ver}_{pk}(m, \sigma) \colon \mathcal{M} \times \mathcal{S} \rightarrow \{0, 1\}$
- Correctness:

$$\Pr[(sk, pk) \leftarrow \mathsf{Gen}(1^n), \sigma \leftarrow \mathsf{Sign}_{sk}(m) \colon \mathsf{Ver}_{pk}(m, \sigma) = 1] = 1$$

- Security (UF-CMA):

$$\Pr\left[\substack{(sk,pk) \leftarrow \mathsf{Gen}(1^n) \\ (m,\sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^n, pk)} \colon \substack{\mathcal{A} \text{ did not query } m \,\wedge \\ \mathsf{Ver}_{pk}(m,\sigma)=1} \right] \leqslant \nu(n)$$

Digital Signature

- Only Signer can sign but everyone can verify
- **Key Generation**: $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
- **Sign**: $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
- **Verify**: $\mathsf{Ver}_{pk}(m, \sigma) \colon \mathcal{M} \times \mathcal{S} \to \{0, 1\}$
- Correctness:

$$\Pr[(sk, pk) \leftarrow \mathsf{Gen}(1^n), \sigma \leftarrow \mathsf{Sign}_{sk}(m) \colon \mathsf{Ver}_{pk}(m, \sigma) = 1] = 1$$

- Security (UF-CMA):

$$\Pr\left[\substack{(sk,pk)\leftarrow\mathsf{Gen}(1^n) \\ (m,\sigma)\leftarrow\mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^n,pk)} \colon \substack{\mathcal{A} \text{ did not query } m \,\wedge \\ \mathsf{Ver}_{pk}(m,\sigma)=1} \right] \leqslant \nu(n)$$

- <u>One-time Signatures</u>: Adversary is allowed only one query

One-time Signature: Construction [Lamport]

Let f be a one-way function

One-time Signature: Construction [Lamport]

Let f be a one-way function

- $sk := \begin{pmatrix} x_1^0 & x_2^0 & \ldots & x_n^0 \\ x_1^1 & x_2^1 & \ldots & x_n^1 \end{pmatrix}$, where $x_i^b \xleftarrow{\$} \{0,1\}^n$ for all $i \in [n]$ and $b \in \{0,1\}$

One-time Signature: Construction [Lamport]

Let f be a one-way function

- $sk := \begin{pmatrix} x_1^0 & x_2^0 & \dots & x_n^0 \\ x_1^1 & x_2^1 & \dots & x_n^1 \end{pmatrix}$, where $x_i^b \xleftarrow{\$} \{0,1\}^n$ for all $i \in [n]$ and $b \in \{0,1\}$
- $pk := \begin{pmatrix} y_1^0 & y_2^0 & \dots & y_n^0 \\ y_1^1 & y_2^1 & \dots & y_n^1 \end{pmatrix}$, where $y_i^b = f(x_i^b)$ for all $i \in [n]$ and $b \in \{0,1\}$

One-time Signature: Construction [Lamport]

Let f be a one-way function

- $sk := \begin{pmatrix} x_1^0 & x_2^0 & \cdots & x_n^0 \\ x_1^1 & x_2^1 & \cdots & x_n^1 \end{pmatrix}$, where $x_i^b \xleftarrow{\$} \{0,1\}^n$ for all $i \in [n]$ and $b \in \{0,1\}$
- $pk := \begin{pmatrix} y_1^0 & y_2^0 & \cdots & y_n^0 \\ y_1^1 & y_2^1 & \cdots & y_n^1 \end{pmatrix}$, where $y_i^b = f(x_i^b)$ for all $i \in [n]$ and $b \in \{0,1\}$
- $\mathsf{Sign}_{sk}(m)$: $\sigma := (x_1^{m_1}, x_2^{m_2}, \ldots, x_n^{m_n})$

One-time Signature: Construction [Lamport]

Let f be a one-way function

- $sk := \begin{pmatrix} x_1^0 & x_2^0 & \dots & x_n^0 \\ x_1^1 & x_2^1 & \dots & x_n^1 \end{pmatrix}$, where $x_i^b \xleftarrow{\$} \{0,1\}^n$ for all $i \in [n]$ and $b \in \{0,1\}$
- $pk := \begin{pmatrix} y_1^0 & y_2^0 & \dots & y_n^0 \\ y_1^1 & y_2^1 & \dots & y_n^1 \end{pmatrix}$, where $y_i^b = f(x_i^b)$ for all $i \in [n]$ and $b \in \{0,1\}$
- $\mathsf{Sign}_{sk}(m)\colon \sigma := (x_1^{m_1}, x_2^{m_2}, \dots, x_n^{m_n})$
- $\mathsf{Ver}_{pk}(m,\sigma) : \quad \wedge_{i\in[n]} f(\sigma_i) \stackrel{?}{=} y_i^{m_i}$

One-time Signature: Construction [Lamport]

Let f be a one-way function

- $sk := \begin{pmatrix} x_1^0 & x_2^0 & \dots & x_n^0 \\ x_1^1 & x_2^1 & \dots & x_n^1 \end{pmatrix}$, where $x_i^b \xleftarrow{\$} \{0,1\}^n$ for all $i \in [n]$ and $b \in \{0,1\}$
- $pk := \begin{pmatrix} y_1^0 & y_2^0 & \dots & y_n^0 \\ y_1^1 & y_2^1 & \dots & y_n^1 \end{pmatrix}$, where $y_i^b = f(x_i^b)$ for all $i \in [n]$ and $b \in \{0,1\}$
- $\mathsf{Sign}_{sk}(m)\colon \sigma := (x_1^{m_1}, x_2^{m_2}, \dots, x_n^{m_n})$
- $\mathsf{Ver}_{pk}(m,\sigma)$: $\wedge_{i\in[n]} f(\sigma_i) \stackrel{?}{=} y_i^{m_i}$
- <u>Think</u>: Proof?

One-time Signature: Construction [Lamport]

Let f be a one-way function

- $sk := \begin{pmatrix} x_1^0 & x_2^0 & \cdots & x_n^0 \\ x_1^1 & x_2^1 & \cdots & x_n^1 \end{pmatrix}$, where $x_i^b \xleftarrow{\$} \{0,1\}^n$ for all $i \in [n]$ and $b \in \{0,1\}$
- $pk := \begin{pmatrix} y_1^0 & y_2^0 & \cdots & y_n^0 \\ y_1^1 & y_2^1 & \cdots & y_n^1 \end{pmatrix}$, where $y_i^b = f(x_i^b)$ for all $i \in [n]$ and $b \in \{0,1\}$
- $\mathsf{Sign}_{sk}(m)\colon \sigma := (x_1^{m_1}, x_2^{m_2}, \ldots, x_n^{m_n})$
- $\mathsf{Ver}_{pk}(m,\sigma) : \bigwedge_{i\in[n]} f(\sigma_i) \stackrel{?}{=} y_i^{m_i}$
- Think: Proof?

Think: How to sign long messages?

Collision-resistant Hash Functions

- Intuition: A compressing function h for which it is hard to find x, x' s.t. $x \neq x'$ but $h(x) = h(x')$

Collision-resistant Hash Functions

- Intuition: A compressing function h for which it is hard to find x, x' s.t. $x \neq x'$ but $h(x) = h(x')$
- Impossible for non-uniform adversary notion

Collision-resistant Hash Functions

- Intuition: A compressing function h for which it is hard to find x, x' s.t. $x \neq x'$ but $h(x) = h(x')$
- Impossible for non-uniform adversary notion
  - Think: Why?

Collision-resistant Hash Functions

- Intuition: A compressing function h for which it is hard to find x, x' s.t. $x \neq x'$ but $h(x) = h(x')$
- Impossible for non-uniform adversary notion
  - Think: Why?
- Need to consider a family of hash functions

Collision-resistant Hash Function Family

Definition (Collision-resistant Hash Function Family)

A family of functions $H = \{h_i : D_i \to R_i\}_{i \in I}$ is a collision-resistant hash function family (CRHF) if:

Collision-resistant Hash Function Family

Definition (Collision-resistant Hash Function Family)

A family of functions $H = \{h_i : D_i \to R_i\}_{i \in I}$ is a collision-resistant hash function family (CRHF) if:

- **Easy to Sample**: There exists a PPT Gen s.t.: $i \leftarrow \mathsf{Gen}(1^n)$, $i \in I$

Collision-resistant Hash Function Family

Definition (Collision-resistant Hash Function Family)

A family of functions $H = \{h_i : D_i \to R_i\}_{i \in I}$ is a collision-resistant hash function family (CRHF) if:

- **Easy to Sample**: There exists a PPT Gen s.t.: $i \leftarrow \mathsf{Gen}(1^n)$, $i \in I$
- **Compression**: $|R_i| < |D_i|$

Collision-resistant Hash Function Family

Definition (Collision-resistant Hash Function Family)

A family of functions $H = \{h_i : D_i \to R_i\}_{i \in I}$ is a collision-resistant hash function family (CRHF) if:

- **Easy to Sample**: There exists a PPT Gen s.t.: $i \leftarrow \mathsf{Gen}(1^n)$, $i \in I$
- **Compression**: $|R_i| < |D_i|$
- **Easy to Evaluate**: There exists a poly-time algorithm Eval s.t. given $x \in D_i$, $i \in I$, $\mathsf{Eval}(x, i) = h_i(x)$

Collision-resistant Hash Function Family

Definition (Collision-resistant Hash Function Family)

A family of functions $H = \{h_i : D_i \rightarrow R_i\}_{i \in I}$ is a collision-resistant hash function family (CRHF) if:

- **Easy to Sample**: There exists a PPT Gen s.t.: $i \leftarrow \mathsf{Gen}(1^n)$, $i \in I$
- **Compression**: $|R_i| < |D_i|$
- **Easy to Evaluate**: There exists a poly-time algorithm Eval s.t. given $x \in D_i$, $i \in I$, $\mathsf{Eval}(x, i) = h_i(x)$
- **Collision Resistance**: For all n.u. PPT $\mathcal{A}$, $\exists$ negligible function $\mu(\cdot)$ s.t.

$$\Pr\left[\begin{array}{c} i \xleftarrow{\$} \mathsf{Gen}(1^n), \\ (x, x') \leftarrow \mathcal{A}(1^n, i) \end{array} : \begin{array}{l} x \neq x' \ \wedge \\ h_i(x) = h_i(x') \end{array}\right] \leqslant \mu(n)$$

Remarks

- One-bit compression implies arbitrary bit compression

Remarks

- One-bit compression implies arbitrary bit compression
  - Think: Proof?

Remarks

- One-bit compression implies arbitrary bit compression
  - Think: Proof?
  - Read: Merkle Trees

Remarks

- One-bit compression implies arbitrary bit compression
  - <u>Think</u>: Proof?
  - <u>Read</u>: Merkle Trees
- Range cannot be too small

Remarks

- One-bit compression implies arbitrary bit compression
  - <u>Think</u>: Proof?
  - <u>Read</u>: Merkle Trees
- Range cannot be too small
  - Enumeration Attacks

Remarks

- One-bit compression implies arbitrary bit compression
  - Think: Proof?
  - Read: Merkle Trees
- Range cannot be too small
  - Enumeration Attacks
  - Birthday Attack

Remarks

- One-bit compression implies arbitrary bit compression
  - Think: Proof?
  - Read: Merkle Trees
- Range cannot be too small
  - Enumeration Attacks
  - Birthday Attack
- **Existence**:

Remarks

- One-bit compression implies arbitrary bit compression
  - Think: Proof?
  - Read: Merkle Trees
- Range cannot be too small
  - Enumeration Attacks
  - Birthday Attack
- **Existence**:
  - Unlikely to be constructed from OWF or OWP [Simon98]

Remarks

- One-bit compression implies arbitrary bit compression
  - Think: Proof?
  - Read: Merkle Trees
- Range cannot be too small
  - Enumeration Attacks
  - Birthday Attack
- **Existence**:
  - Unlikely to be constructed from OWF or OWP [Simon98]
  - Can be constructed from number-theoretic assumptions such as factoring, discrete log

Remarks (contd.)

- **Weaker notion**: Universal One-way Hash Functions (UOWHF)

Remarks (contd.)

- **Weaker notion**: Universal One-way Hash Functions (UOWHF)
  - $$\Pr\left[\begin{array}{r} (x, \mathsf{state}) \leftarrow \mathcal{A}(1^n), \\ i \xleftarrow{\$} \mathsf{Gen}(1^n), \\ x' \leftarrow \mathcal{A}(i, \mathsf{state}) \end{array} : \begin{array}{l} x \neq x' \ \wedge \\ h_i(x) = h_i(x') \end{array}\right] \leqslant \mu(n)$$

- **Weaker notion**: Universal One-way Hash Functions (UOWHF)
  - $$\Pr\left[\begin{array}{r} (x, \mathsf{state}) \leftarrow \mathcal{A}(1^n), \\ i \xleftarrow{\$} \mathsf{Gen}(1^n), \\ x' \leftarrow \mathcal{A}(i, \mathsf{state}) \end{array} : \begin{array}{l} x \neq x' \ \wedge \\ h_i(x) = h_i(x') \end{array}\right] \leqslant \mu(n)$$
  - Can be constructed from OWF [Rompel90]

Remarks (contd.)

- **Weaker notion**: Universal One-way Hash Functions (UOWHF)
  - $$\Pr\left[\begin{array}{r} (x, \mathsf{state}) \leftarrow \mathcal{A}(1^n), \\ i \xleftarrow{\$} \mathsf{Gen}(1^n), \\ x' \leftarrow \mathcal{A}(i, \mathsf{state}) \end{array} : \begin{array}{l} x \neq x' \;\; \wedge \\ h_i(x) = h_i(x') \end{array}\right] \leqslant \mu(n)$$
  - Can be constructed from OWF [Rompel90]
  - Suffices for Digital Signatures [Naor-Yung89]

Remarks (contd.)

- **Weaker notion**: Universal One-way Hash Functions (UOWHF)
  - $$\Pr\left[\begin{array}{r} (x, \mathsf{state}) \leftarrow \mathcal{A}(1^n), \\ i \xleftarrow{\$} \mathsf{Gen}(1^n), \\ x' \leftarrow \mathcal{A}(i, \mathsf{state}) \end{array} : \begin{array}{l} x \neq x' \ \wedge \\ h_i(x) = h_i(x') \end{array}\right] \leqslant \mu(n)$$
  - Can be constructed from OWF [Rompel90]
  - Suffices for Digital Signatures [Naor-Yung89]
  - More efficient construction [Haitner-Holenstein-Reingold-Vadhan-Wee10]

One-time Signatures for Long Messages

- Let $H = \{h_i : \{0,1\}^* \to \{0,1\}^n\}_{i \in I}$ be a CRHF

One-time Signatures for Long Messages

- Let $H = \{h_i : \{0,1\}^* \rightarrow \{0,1\}^n\}_{i \in I}$ be a CRHF
- Idea: Sign $h_i(m)$ instead of m using Lamport signature

One-time Signatures for Long Messages

- Let $H = \{h_i : \{0,1\}^* \to \{0,1\}^n\}_{i \in I}$ be a CRHF
- Idea: Sign $h_i(m)$ instead of m using Lamport signature
- Think: Proof?

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:
  - $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}(1^n)$

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:
  - $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}(1^n)$
  - $\tilde{\sigma}_i \leftarrow \mathsf{Sign}_{sk_{i-1}}(m_i \| pk_i)$

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:
  - $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}(1^n)$
  - $\tilde{\sigma}_i \leftarrow \mathsf{Sign}_{sk_{i-1}}(m_i \| pk_i)$
  - Output: $\sigma_i = (i, \tilde{\sigma}_i, m_i, pk_i, \sigma_{i-1})$

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:
  - $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}(1^n)$
  - $\tilde{\sigma}_i \leftarrow \mathsf{Sign}_{sk_{i-1}}(m_i \| pk_i)$
  - Output: $\sigma_i = (i, \tilde{\sigma}_i, m_i, pk_i, \sigma_{i-1})$
  - Increment i

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- <u>Initialize</u>: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:
  - $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}(1^n)$
  - $\tilde{\sigma}_i \leftarrow \mathsf{Sign}_{sk_{i-1}}(m_i \| pk_i)$
  - Output: $\sigma_i = (i, \tilde{\sigma}_i, m_i, pk_i, \sigma_{i-1})$
  - Increment i
- <u>Think</u>: Proof?

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:
  - $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}(1^n)$
  - $\tilde{\sigma}_i \leftarrow \mathsf{Sign}_{sk_{i-1}}(m_i \| pk_i)$
  - Output: $\sigma_i = (i, \tilde{\sigma}_i, m_i, pk_i, \sigma_{i-1})$
  - Increment i
- Think: Proof?
- Think: How to reduce signature size?

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:
  - $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}(1^n)$
  - $\tilde{\sigma}_i \leftarrow \mathsf{Sign}_{sk_{i-1}}(m_i \| pk_i)$
  - Output: $\sigma_i = (i, \tilde{\sigma}_i, m_i, pk_i, \sigma_{i-1})$
  - Increment i
- Think: Proof?
- Think: How to reduce signature size?
- Read: Tree-based signatures

Multi-message Signatures (via chain)

- $(sk_0, pk_0) \xleftarrow{\$} \mathsf{Gen}(1^n)$
- Initialize: $\tilde{\sigma}_i = \emptyset$, $i = 1$
- To sign m_i:
  - $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}(1^n)$
  - $\tilde{\sigma}_i \leftarrow \mathsf{Sign}_{sk_{i-1}}(m_i \| pk_i)$
  - Output: $\sigma_i = (i, \tilde{\sigma}_i, m_i, pk_i, \sigma_{i-1})$
  - Increment i
- Think: Proof?
- Think: How to reduce signature size?
- Read: Tree-based signatures
- Read: Efficient Signatures from Trapdoor Permutations in the Random Oracle Model