Signing more than one document

Chain-based signature schemes

Foundations of Cryptography
Computer Science Department
Wellesley College

Fall 2016

Table of contents

Introduction

Stateful Signature Schemes

Chain-based Schemes

Tree-based Schemes

Stateless schemes
Good enough by golly

- Collision-resistant hash functions can be used to construct signature schemes that allow the signer to sign arbitrarily many messages.
- We start with a scheme that maintains state that must be updated after each signature.
- Later this scheme is modified to be stateless.

State signature scheme

**Definition 12.18.** A stateful signature scheme is a tuple of probabilistic polynomial-time algorithms \((\text{Gen}, \text{Sign}, \text{Vrfy})\) satisfying the following:

1. The key-generation algorithm \(\text{Gen}\) takes as input a security parameter \(1^n\) and outputs \((pk, sk, s_0)\). These are called the public key, private key, and initial state, respectively. we assume \(pk\) and \(sk\) each has length at least \(n\), and that \(n\) can be determined from \(pk, sk\).

2. The signing algorithm \(\text{Sign}\) takes as input a private key \(sk\), a value \(s_{i-1}\), and a message \(m \in \{0,1\}^*\). It outputs a signature \(\alpha\) and a value \(s_i\).

3. The deterministic verification algorithm \(\text{Vrfy}\) takes as input a public key, \(pk\), a message \(m\), and a signature \(\alpha\). It outputs a bit \(b\).

We require that for every \(n\), every \((pk, sk, s_0)\) output by \(\text{Gen}(1^n)\), and any messages \(m_1, \ldots, m_t \in \{0,1\}^*\), if we iteratively compute \((\alpha_i, s_i) \leftarrow \text{Sign}_{sk, s_{i-1}}(m_i)\) for \(i = 1, \ldots, t\), then for every \(i \in \{1, \ldots, t\}\), it holds that \(\text{Vrfy}_{pk}(m_i, \alpha_i) = 1\).
Security of signature schemes

Let $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ be a stateful signature scheme.

The stateful signature experiment $\text{Sig-forge-stateful}_{\mathcal{A},\Pi}(n)$:

1. $\text{Gen}(1^n)$ is run to obtain keys $(pk, sk)$.
2. Adversary $\mathcal{A}$ is given $pk$ and oracle access to $\text{Sign}_{sk}(\cdot)$ which returns only the signature (not the state)*. The adversary then outputs $(m, \alpha)$. Let $Q$ denote the set of messages whose signatures were requested by $\mathcal{A}$ during its execution.
3. The output of the experiment is defined to be 1 if and only if (1) $\text{Vrfy}_{pk}(m, \alpha) = 1$, and (2) $m \notin Q$.

Definition. A stateful signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ is existentially unforgeable under an adaptive chosen-message attack if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}$ such that

$$\Pr[\text{Sig-forge-stateful}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

*The signing oracle updates the state each time it is invoked.

$t$-time signature schemes

Construction. For any polynomial $t = t(n)$, we can construct a $t$-time-secure signature scheme as follows:

- **Gen**: Set $pk := \langle pk_1, \ldots, pk_t \rangle$ and $sk := \langle sk_1, \ldots, sk_t \rangle$ where each $(pk_i, sk_i)$ is an independently generated key-pair for some one-time signature scheme. The state is a counter $i$ initially set to 1.
- **Sign**: Given a private key, $sk$, a message $m$, and a current state $i \leq t$, compute $\alpha \leftarrow \text{Sign}_{sk_i}(m)$ and output $(\alpha, i)$; set $i := i + 1$.
- **Vrfy**: Given a public key $pk$, a message $m$, and signature $(\alpha, i)$, output bit $\text{Vrfy}_{sk_i}(m, \alpha)$.

Remark. This scheme is secure* if used to sign $t$ message since each private key of the underlying one-time-scheme is used to sign only a single message.

*The definition of security is a generalization of security for a one-time signature scheme given in Definition 12.14.
Improved space efficiency

- In this signature scheme, signatures have constant length, but the public key has length linear in $t$.

- It is possible to trade off the length of the public key and the signature using Merkle trees. The public key is now constant and the signature grows logarithmically with $t$.

- However, the number of message that can be signed is still fixed in advance at the time of key generation. Once the limit is reached, a new public key would have to generated and distributed.

Signing an unbounded number of messages securely

- We start with a one-time signature scheme
  \[ \Pi = (\text{Gen, Sign, Vrfy}) \]
  and build a scheme the signs an unbounded number of messages securely.

- This is done by having the signer generate additional public keys on-the-fly.
Construction of chain-based signatures

Let $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ be a one-time-secure signature scheme.

Construction. For any polynomial $t = t(n)$, we can construct a $t$-time-secure signature scheme as follows:

- **Gen**: Run Gen to obtain a single public/private key pair $(pk_1, sk_1)$.
- **Sign**: Given a private key, $sk_i$, a message $m_i$ do the following:
  1. Generate a new public/private key pair $(pk_{i+1}, sk_{i+1})$ using Gen.
  2. Use Sign and $sk_i$ to obtain a signature
     \( \alpha_i \leftarrow \text{Sign}_{sk_i}(m_i \parallel pk_{i+1}) \).
     Output signature
     \( (pk_{i+1}, \alpha_i, \{m_j, pk_{j+1}, \alpha_j\}_{j=1}^{i-1}) \).
  3. The signer then adds $(m_i, pk_{i+1}, sk_{i+1}, \alpha_i)$ to its state.
- **Vrfy**: Given a signature $(pk_{i+1}, \alpha_i, \{m_j, pk_{j+1}, \alpha_j\}_{j=1}^{i-1})$ on message $m = m_i$, Output 1 if and only if $\text{Vrfy}_{pk_j}(m_j \parallel pk_{j+1}, \alpha_j) = 1$ for all $j \in \{1, \ldots, i\}$.

Intuitively, this scheme is existentially unforgeable under an adaptive chosen-message attack ...

- ... because each key-pair $(pk_i, sk_i)$ is used to sign only one “message” pair $m_i \parallel pk_{i+1}$.
- The hash-and-sign paradigm applied to Lamport produces a one-time-secure scheme that can sign message of arbitrary length.
- **Lemma 12.19**. If collision-resistant hash functions exist, then there exists a one-time-secure scheme (for messages of arbitrary length).
Intuitively, this scheme is existentially unforgeable under an adaptive chosen-message attack ...

- There is no immediate way to eliminate state from our chain-based signature scheme and it isn’t very efficient. Signature size, state, and verification time are linear in the number of messages signed.

- Furthermore, each signature reveals all previously signed messages; this may not always be a good idea.

- Tree-based signature schemes address these issues.

Tree-based signatures

- The chain-based scheme may be viewed as maintaining a tree of degree 1, rooted at the public key \( pk_1 \). Verification is linear in the number of signed messages.

- If this were CS230, we would try to improve efficiency by using a binary tree.

- A signature would still correspond to a signed path from root to leaf, but as long as the depth of the tree is polynomial, verification can be done in same even if there are exponentially many nodes.
Tree-based signatures

- Each node $w$ is associated with a pair of keys $pk_w, sk_w$ for a one-time secure signature scheme $\Pi$. The root public key, is the public key of the signer.

- To sign a message $m \in \{0, 1\}^n$:
  1. Generate keys for nodes on a path from root to the leaf $m$.
  2. Certify the path by computing a signature on $pk_0 \parallel pk_1$, using private key $sk_w$, for each string $w$ that is a prefix of $m$.
  3. Certify $m$ by computing a signature using private key $sk_m$.

- The final signature consists of the signature on $m$ w.r.t. $pk_m$, plus all that is needed to verify the path from leaf $m$ to root.

Construction. 12.20.

Let $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ be a signature scheme. For a binary string $m$, let $m_i \stackrel{\text{def}}{=} m_1 \ldots m_i$. Define scheme $\Pi^* = (\text{Gen}^*, \text{Sign}^*, \text{Vrfy}^*)$ as follows:

- Gen*: On input $1^n$, compute $(pk_e, sk_e) \leftarrow \text{Gen}(1^n)$, and output $pk_e$.
  Private key and initial state are $sk_e$.

- Sign*: on input $m \in \{0, 1\}^n$ do:
  1. For $i = 0$ to $n - 1$:
     If $pk_{m_i,0}, pk_{m_i,1}$, and $\alpha_{m_i}$ are not in state, compute $pk_{m_i,0}, sk_{m_i,0} \leftarrow \text{Gen}(1^n)$, $pk_{m_i,1}, sk_{m_i,1} \leftarrow \text{Gen}(1^n)$, and $\alpha_{m_i} \leftarrow \text{Sign}_{sk_m}(pk_{m_i,0} \parallel pk_{m_i,1})$. Add these to state.
  2. If $\alpha_m$ is not in state, compute $\alpha_m \leftarrow \text{Sign}_{m}(m)$. Store in state.
  3. Output $(\{\alpha_{m_i}, pk_{m_i,0}, pk_{m_i,1}\}_{i=0}^{n-1}, \alpha_m)$

- Vrfy*: on input $pk_e, m$, and $(\{\alpha_{m_i}, pk_{m_i,0}, pk_{m_i,1}\}_{i=0}^{n-1}, \alpha_m)$ output 1 iff
  1. $Vrfy_{pk_{m_i}}(pk_{m_i,0} \parallel pk_{m_i,1}, \alpha_{m_i}) \equiv 1$ for $i \in \{0, \ldots, n - 1\}$.
  2. $Vrfy_{pk_m}(m, \alpha_m) \equiv 1$. 
Some observations

- The tree is not constructed in its entirety, but built up by the signer as needed. There are potentially $2^n$ leaves and the messages in the message space.

- Signature length and verification time are now proportional to the message length, but independent of the number of messages signed.

- The scheme is still stateful, but we’ll fix that shortly.

Proof of security

**Theorem 12.21.** Let $\Pi$ be a one-time-secure signature scheme. Then Construction 12.20 is a secure signature scheme.

**Proof.** Let $\Pi^*$ denote Construction 12.20. Let $A^*$ be a ppt adversary, let $\ell^* = \ell^*(n)$ be a polynomial upper bound on the number signing queries made by $A^*$, and set $\ell = \ell(n) \overset{\text{def}}{=} 2n\ell^*(n) + 1$.

We construct a PPT adversary $A$ attacking the one-time-secure signature scheme $\Pi$.

*Note that $\ell$ is an upper bound on the number of public keys from $\Pi$ needed to generate $\ell^*$ signatures from $\Pi$. 
An adversary attacking the one-time-secure scheme

Adversary $A$. $A$ is given public key $pk$.

- Choose a uniform index $i^* \in \{1, \ldots, \ell\}$. Construct a list $pk^1, \ldots, pk^\ell$ of keys as follows:
  - set $pk^{i^*} := pk$.
  - For $i \neq i^*$, compute $(pk^i, sk^i) \leftarrow \text{Gen}(1^n)$.
- Run $A^*$ on input public key $pk_c = pk^1$. When $A^*$ requests a signature on message $m$ do:
  1. For $i = 0$ to $n - 1$:
     If values $pk_{m|0}, pk_{m|1}$ and $\alpha_{m|i}$ have not yet been defined, set $pk_{m|0}$ and $pk_{m|1}$ equal to next two unused public keys $pk^{i^*}$ and $pk^{i^*+1}$, and compute a signature $\alpha_{m|i}$ on $pk_{m|0} \parallel pk_{m|1}$ with respect to $pk_{m|i}$.
  2. If $\alpha_m$ is not yet defined, compute a signature $\alpha_m$ on $m$ with respect to $pk_m$.
  3. Given $(\{\alpha_{m|i}, pk_{m|0}, pk_{m|1}\}_{i=1}^{n-1})$ to $A^*$.

To be continued ...

continued ...

- Say $A^*$ outputs a message $m$ (not previously requested for a signature) and a signature $(\{\alpha'_{m|i}, pk'_{m|0}, pk'_{m|1}\}_{i=0}^{n-1}, \alpha'_m)$. If this is a valid signature on $m$, then:
  **Case 1:** If there exists $j \in \{0, \ldots, n - 1\}$ for which $pk'_{m|0} \neq pk_{m|0}$ or $pk'_{m|1} \neq pk_{m|1}$ then take the minimal such $j$, and let $i$ be such that $pk^i = pk_{m|j} = pk'_{m|j}$. If $i = i^*$, output $(pk'_{m|0} \parallel pk'_{m|1}, \alpha'_{m|j})$.
  **Case 2:** If case 1 does not hold, then $pk'_m = pk_m$. Let $i$ be such that $pk^i = pk_m$. If $i = i^*$, output $(m, \alpha^m)$.

In experiment $\text{Sig-forge}_{A, \Pi}^{1\text{-time}}(n)$, the view of $A^*$ begin run as a subroutine of $A$ is distributed identically to the view of $A^*$ in experiment $\text{Sig-forge}_{A^*, \Pi^*}(n)$. 
For reference: $\text{Sig-forge}_{A,\Pi}^{1-\text{time}}(n)$

The one-time signature experiment $\text{Sig-forge}_{A,\Pi}^{1-\text{time}}(n)$.

1. Gen($1^n$) is run to obtain keys ($pk, sk$).
2. Adversary $A$ is given $pk$ and asks a single query $m'$ to oracle $\text{Sign}_{pk}(\cdot)$. $A$ then outputs $(m, \alpha)$ where $m \neq m'$.
3. The output of the experiment is defined to be 1 if and only if $\text{Vrfy}_{pk}(m, \alpha) = 1$.

Definition 12.14 A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ is existentially unforgeable under a single-message attack, or is a one-time signature scheme, if for all PPT adversaries $A$, there exists a negligible function $\text{negl}$ such that:

$$\text{Pr}[\text{Sig-forge}_{A,\Pi}^{1-\text{time}}(n) = 1] \leq \text{negl}(n).$$

Also for reference: $\text{Sig-forge}_{A,\Pi}(n)$

Let $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ be a signature scheme.

The signature experiment $\text{Sig-forge}_{A,\Pi}(n)$:

1. Gen($1^n$) is run to obtain keys ($pk, sk$).
2. Adversary $A$ is given $pk$ and oracle access to $\text{Sign}_{sk}(\cdot)$. The adversary then outputs $(m, \alpha)$. Let $Q$ denote the set of messages whose signatures were requested by $A$ during its execution.
3. The output of the experiment is defined to be 1 if and only if (1) $\text{Vrfy}_{pk}(m, \alpha) = 1$, and (2) $m \notin Q$

Definition 12.2. A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ is existentially unforgeable under an adaptive chosen-message attack if for all PPT adversaries $A$, there exists a negligible function $\text{negl}$ such that

$$\text{Pr}[\text{Sig-forge}_{A,\Pi}(n) = 1] \leq \text{negl}(n).$$
Case analysis

Case 1. Since $i^*$ is uniform and independent of the view of $A^*$, the probability that $i = i^*$ is exactly $1/\ell$. If $i = i^*$, then $A$ requested a signature on public key $pk = pk^{i^*} = pk_{m|j}$. Moreover,

$$pk'_{m|j,0} \| pk'_{m|j,0} \neq pk_{m|j,0}$$

and yet $\alpha_{m|j}$ is a valid signature on $pk'_{m|j,0} \| pk'_{m|j,0}$ with respect to $pk$. Thus, $A$ outputs a forgery.

Case 2. As before the probability that $i = i^*$ is exactly $1/\ell$. If $i = i^*$, then $A$ did not request a signature w.r.t. public key $pk = pk^i = pk_m$ and yet $\alpha'_m$ is a valid signature on $m$ with respect to $pk$.

We see that if $A^*$ outputs a forgery, then $A$ outputs a forgery with probability $1/\ell$. Thus,

$$\Pr[\text{Sig-forge}_{A,\Pi}^{1\text{-time}}(n) = 1] = \Pr[\text{Sig-forge}_{A^*,\Pi^*}(n) = 1]/\ell(n).$$

and since $\Pi$ is one-time-secure, and $\ell(n)$ is polynomial, $\Pr[\text{Sig-forge}_{A^*,\Pi^*}(n) = 1]$ is negligible.

A stateless scheme: First attempt

- As described, the signer generates state on-the-fly as needed.
- This could be avoided if the signer generated the entire tree in advance at the time of key generation.
- However, this would require exponential time and space.
A stateless scheme: Second attempt

- Rather than store the actual keys, we could store some randomness that is used to generate the values \( \{ pk_w, sk_w \} \) and \( \{ \alpha_w \} \), as needed.

- That is, the signer stores a random string \( r_w \) for each \( w \) and whenever \( pk_w, sk_w \) are needed the signer computes \( (pk_w, sk_w) := \text{Gen}(1^n, r_w) \).

- Similarly, the signer can store \( r'_w \) and then set \( \alpha_w = \text{Sign}_{sk_w}(pk_{w0} \parallel pk_{w1}; r'_w) \).

A stateless scheme: Third attempt

- Instead of storing random \( r_w \) and \( r'_w \), the signer stores two keys \( k, k' \) for a pseudorandom function \( F \).

- When needed the signer computes \( pk_w, sk_w \) as follows:
  1. Compute \( r_w := F_k(w) \).
  2. Compute \( pk_w, sk_w := \text{Gen}(1^n; r_w) \) as before.

- Similarly, the key \( k' \) is used to generate the value \( r'_w \) that is used to compute the signature \( \alpha_w \).
Putting this all together gives a provably secure signature scheme (sort of, kind of)

**Theorem 12.22.** If collision-resistant hash functions exist, then there exists a (stateless) secure signature scheme.

**Proof idea.** The existence of collision-resistant hash functions implies the existence of one-way functions which in turn implies the existence of pseudorandom function. The former is needed to prove the Lamport scheme is one-time secure for messages of length \(\ell\) (Corollary 12.17).

This is then turned into a one-time-secure scheme that can sign messages of arbitrary length using the hash-and-sign paradigm for which we need a collision-resistant hash (Theorem 12.4). Finally, a pseudorandom function is needed to achieve statelessness.