III - Signatures

David Pointcheval
Ecole normale supérieure, CNRS & INRIA

MPRI - Paris

Outline

1. Basic Security Notions
   - Public-Key Encryption
   - Signatures

2. Advanced Security for Signature
   - Advanced Security Notions
   - Hash-then-Invert Paradigm

3. Forking Lemma
   - Zero-Knowledge Proofs
   - The Forking Lemma

4. Conclusion

Outline

Public-Key Encryption

1. Basic Security Notions
   - Public-Key Encryption
   - Signatures

2. Advanced Security for Signature

3. Forking Lemma

4. Conclusion

Goal: Privacy/Secrecy of the plaintext
**OW – CPA Security Game**

\[ m^* \overset{?}{=} m \]

\[ \text{Succ}^{\text{OW}}_S(A) = \Pr[(sk, pk) \leftarrow \mathcal{K}(); m \overset{R}{\leftarrow} \mathcal{M}; c = \mathcal{E}_{pk}(m) : A(pk, c) \rightarrow m] \]

**IND – CPA Security Game**

\[ b' \overset{?}{=} b \]

\[ \text{Adv}^{\text{IND-CPA}}_S(A) = \Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0] = 2 \times \Pr[b' = b] - 1 \]

---

**Outline**

1. **Basic Security Notions**
   - Public-Key Encryption
   - Signatures

2. Advanced Security for Signature

3. Forking Lemma

4. Conclusion

---

**Signature**

**Goal: Authentication of the sender**
**EUF – NMA**

**Outline**

1. Basic Security Notions
2. Advanced Security for Signature
   - Advanced Security Notions
   - Hash-then-Invert Paradigm
3. Forking Lemma
4. Conclusion

\[
\text{Success}_{SG}^{\text{euf}}(A) = \Pr[(sk, pk) \leftarrow \mathcal{K}(); (m, \sigma) \leftarrow A(pk) : V_{pk}(m, \sigma) = 1]
\]

**Signature**

**EUF – NMA**

The adversary knows the public key only, whereas signatures are not private!
The adversary has access to any signature of its choice: Chosen-Message Attacks (oracle access):

\[ \text{Succ}_{S^\text{euf-cma}}^\text{euf-cma}(A) = \Pr\left[ (sk, pk) \leftarrow \mathcal{K}(); (m, \sigma) \leftarrow A^S(pk) : \forall i, m \neq m_i \land \mathcal{V}_{pk}(m, \sigma) = 1 \right] \]

The notion is even stronger (in case of probabilistic signature): also known as non-malleability:

\[ \text{Succ}_{S^\text{cma}}^\text{cma}(A) = \Pr\left[ (sk, pk) \leftarrow \mathcal{K}(); (m, \sigma) \leftarrow A^S(pk) : \forall i, (m, \sigma) \neq (m_i, \sigma_i) \land \mathcal{V}_{pk}(m, \sigma) = 1 \right] \]

Outline

1. Basic Security Notions
2. Advanced Security for Signature
   - Advanced Security Notions
   - Hash-then-Invert Paradigm
3. Forking Lemma
4. Conclusion

Full-Domain Hash Signature

Signature Scheme

- Key generation: the public key \( f \overset{R}{\leftarrow} \mathcal{P} \) is a trapdoor one-way bijection from \( X \) onto \( Y \); the private key is the inverse \( g : Y \rightarrow X \);
- Signature of \( M \in Y \): \( \sigma = g(M) \);
- Verification of \( (M, \sigma) \): check \( f(\sigma) = M \)

Full-Domain Hash (Hash-and-Invert)

\( \mathcal{H} : \{0, 1\}^* \rightarrow Y \)

- in order to sign \( m \), one computes \( M = \mathcal{H}(m) \in Y \), and \( \sigma = g(M) \)
- and the verification consists in checking whether \( f(\sigma) = H(m) \)
Security of the FDH Signature

Theorem

The FDH signature achieves EUF- CMA security, under the One-Wayness of $\mathcal{P}$, in the Random Oracle Model:

$$\text{Succ}^{\text{euf-cma}}_{\text{FDH}}(t) \leq q_H \times \text{Succ}^{\text{OW}}_{\mathcal{P}}(t + q_H \tau_i)$$

Assumptions:

- any signing query has been first asked to $\mathcal{H}$
- the forgery has been asked to $\mathcal{H}$
- $\tau_i$ is the maximal time to evaluate $f \in \mathcal{P}$

Real Attack Game

Simulations

- **Game$_0$**: use of the oracles $\mathcal{K}$, $\mathcal{S}$ and $\mathcal{H}$
- **Game$_1$**: use of the simulation of the Random Oracle

Simulation of $\mathcal{H}$

$\mathcal{H}(m): \mu \xleftarrow{\$} X$, output $M = f(\mu)$

$\xRightarrow{\text{Hop-D-Perfect}} \Pr_{\text{Game$_0$}}[1] = \Pr_{\text{Game$_1$}}[1]$

Simulation of $\mathcal{S}$

$\mathcal{S}(m): M = \mathcal{H}(m)$, output $\sigma = g(M)$

$\xRightarrow{\text{Hop-S-Perfect}} \Pr_{\text{Game$_2$}}[1] = \Pr_{\text{Game$_1$}}[1]$
**H-Query Selection**

- **Game$_3$:** random index $t \overset{R}{\leftarrow} \{1, \ldots, q_H\}$

**Event Ev**

If the $t$-th query to $\mathcal{H}$ is not the output forgery

$\implies \textbf{Hop-S-Non-Negl}$

Then, clearly

$$\Pr[1] = \Pr[1] \times \Pr[\neg \text{Ev}] \quad \Pr[\text{Ev}] = 1 - 1/q_H$$

$$\Pr[1] = \Pr[1] \times \frac{1}{q_H}$$

**Summary**

In Game$_4$, when the output is 1, $\sigma = g(y) = g(f(x)) = x$ and the simulator computes one exponentiation per hashing:

$$\Pr[1] \leq \text{Succ}^{\text{euf-cma}}_{FD_H}(t + q_H \tau_f)$$

$$\Pr[1] = \Pr[1]$$

$$\Pr[1] = \Pr[1] \times \frac{1}{q_H}$$

$$\Pr[1] = \Pr[1]$$

$$\Pr[1] = \text{Succ}^{\text{euf-cma}}_{FD_H}(A)$$

$$\text{Succ}^{\text{euf-cma}}_{FD_H}(A) \leq q_H \times \text{Succ}^{\text{OW}}_P(t + q_H \tau_f)$$

- **Game$_4$:** $\mathcal{P}$ – OW instance $(f,y)$ (where $f \overset{R}{\leftarrow} \mathcal{P}, x \overset{R}{\leftarrow} X, y = f(x)$)
  - Use of the simulation of the Key Generation Oracle

**Simulation of $\mathcal{K}$**

$\mathcal{K}()$: set $pk \leftarrow f$

- Modification of the simulation of the Random Oracle

**Simulation of $\mathcal{H}$**

If this is the $t$-th query, $\mathcal{H}(m) \leftarrow y$, output $M$

The unique difference is for the $t$-th simulation of the random oracle, for which we cannot compute a signature.

But since it corresponds to the forgery output, it cannot be queried to the signing oracle:

$\implies \textbf{Hop-S-Perfect}$: $\Pr_{\text{Game}_4}[1] = \Pr_{\text{Game}_3}[1]$

**Key Size**

$$\text{Succ}^{\text{euf-cma}}_{FD_H}(A) \leq q_H \times \text{Succ}^{\text{OW}}_P(t + q_H \tau_f)$$

- If one wants $\text{Succ}^{\text{euf-cma}}_{FD_H}(t) \leq \varepsilon$ with $t/\varepsilon \approx 2^{80}$
- If one allows $q_H$ up to $2^{60}$

Then one needs $\text{Succ}^{\text{OW}}_P(t) \leq \varepsilon$ with $t/\varepsilon \geq 2^{140}$.

If one uses FDH-RSA: at least 3072 bit keys are needed.
In the case that \( f \) is homomorphic (as RSA): \( f(ab) = f(a)f(b) \)

- **Game_0**: use of the oracles \( K, S \) and \( \mathcal{H} \)
- **Game_1**: use of the simulation of the Random Oracle

### Simulation of \( \mathcal{H} \)

\( \mathcal{H}(m): \mu \overset{R}{\leftarrow} X, \text{output } M = f(\mu) \)

\[ \implies \text{Hop-D-Perfect: } \Pr_{\text{Game}_1}[1] = \Pr_{\text{Game}_2}[1] \]

- **Game_2**: use of the homomorphic property
  \( \mathcal{P} \) – OW instance \((f, y)\) (where \( f \overset{R}{\leftarrow} \mathcal{P}, x \overset{R}{\leftarrow} X, y = f(x) \))

### Simulation of \( \mathcal{H} \)

\( \mathcal{H}(m): \text{flip a biased coin } b \text{ (with } \Pr[b = 0] = p), \mu \overset{R}{\leftarrow} X. \)
If \( b = 0 \), output \( M = f(\mu) \), otherwise output \( M = y \times f(\mu) \)

\[ \implies \text{Hop-D-Perfect: } \Pr_{\text{Game}_2}[1] = \Pr_{\text{Game}_1}[1] \]

### Summary

In **Game_3**, when the output is 1, with probability \( 1 - p \):

\[ \sigma = g(M) = g(y \times f(\mu)) = g(y) \times g(f(\mu)) = g(f(x)) \times \mu = x \times \mu \]

\[ \Pr_{\text{Game}_3}[1] \leq \frac{1}{(1 - p)p^{qs}} \times \text{Succ}^{\text{OW}}_P(t + q_{H\tau_I}) \]

The maximal for \( p \mapsto (1 - p)p^{qs} \) is reached for

\[ p = 1 - \frac{1}{q_s + 1} \rightarrow \frac{1}{q_s + 1} \times \left(1 - \frac{1}{q_s + 1}\right)^{qs} \approx e^{-1} \frac{1}{q_s} \]

- If one wants \( \text{Succ}^{\text{euf-cma}}_{FDH}(A) \leq \varepsilon \) with \( t/\varepsilon \approx 2^{80} \)
- If one allows \( q_s \) up to \( 2^{30} \)

Then one needs \( \text{Succ}^{\text{OW}}_P(t) \leq \varepsilon \) with \( t/\varepsilon \geq 2^{110} \).

If one uses FDH-RSA: 2048 bit keys are enough.
Proof of Knowledge

How do I prove that I know a solution \( s \) to a problem \( P \)?

Forking Lemma
- Zero-Knowledge Proofs
- The Forking Lemma

Proof of Knowledge: Soundness

If I can be accepted, I really know a solution: extractor

Proof of Knowledge: Zero-Knowledge

How do I prove that I know a solution \( s \) to a problem \( P \)?
I reveal the solution. . .
How can I do it without revealing any information?
Zero-knowledge: simulator
Proof of Knowledge

How do I prove that I know a 3-color covering, without revealing any information?

I choose a random permutation on the colors and I apply it to the vertices I mask the vertices and send it to the verifier. The verifier chooses an edge I open it. The verifier checks the validity: 2 different colors.

Secure Multiple Proofs of Knowledge: Authentication

If there exists an efficient adversary, then one can solve the underlying problem:

Schnorr Proofs

Generic Zero-Knowledge Proofs

Zero-Knowledge Proof
- Setting: \( G = \langle g \rangle \) of order \( q \)
  - \( \mathcal{P} \) knows \( x \), such that \( y = g^x \)
  - \( \mathcal{P} \) wants to prove it to \( \mathcal{V} \)
- \( \mathcal{P} \) chooses \( k \overset{R}{\leftarrow} \mathbb{Z}_q^* \)
  - sets and sends \( r = g^k \)
- \( \mathcal{V} \) chooses \( h \overset{R}{\leftarrow} \{0,1\}^k \)
  - and sends it to \( \mathcal{P} \)
- \( \mathcal{P} \) computes and sends
  - \( s = K + xh \mod q \)
- \( \mathcal{V} \) checks whether \( r \overset{?}{=} g^s y^h \)

Signature
- \( (G = \langle g \rangle) \) of order \( q \)
  - \( \mathcal{H}: \{0,1\}^* \rightarrow \mathbb{Z}_q \)
- Key Generation \( \rightarrow (y, x) \)
  - private key \( x \in \mathbb{Z}_q^* \)
  - public key \( y = g^x \)
- Signature of \( m \rightarrow (r, h, s) \)
  - \( k \overset{R}{\leftarrow} \mathbb{Z}_q^* \)
  - \( r = g^k \)
  - \( h = \mathcal{H}(m, r) \) and
  - \( s = K + xh \mod q \)
- Verification of \( (m, r, s) \)
  - compute \( h = \mathcal{H}(m, r) \)
  - and check \( r \overset{?}{=} g^s y^h \)

Zero-Knowledge Proof
- Proof of knowledge of \( x \), such that \( \mathcal{R}(x, y) \)
- \( \mathcal{P} \) builds a commitment \( r \) and sends it to \( \mathcal{V} \)
- \( \mathcal{V} \) chooses a challenge \( h \overset{R}{\leftarrow} \{0,1\}^k \) for \( \mathcal{P} \)
- \( \mathcal{P} \) computes and sends the answer \( s \)
- \( \mathcal{V} \) checks \( (r, h, s) \)

Signature
- \( \mathcal{H} \) viewed as a random oracle
- Key Generation \( \rightarrow (y, x) \)
  - private: \( x \) public: \( y \)
- Signature of \( m \rightarrow (r, h, s) \)
- Commitment \( r \)
  - Challenge \( h = \mathcal{H}(m, r) \)
  - Answer \( s \)
- Verification of \( (m, r, s) \)
  - compute \( h = \mathcal{H}(m, r) \)
  - and check \( (r, h, s) \)
Zero-Knowledge Proof

- Proof of knowledge of $x$
- $\mathcal{P}$ sends a commitment $r$
- $\mathcal{V}$ sends a challenge $h$
- $\mathcal{P}$ sends the answer $s$
- $\mathcal{V}$ checks $(r, h, s)$

Signature

- Key Generation $\rightarrow (y, x)$
- Signature of $m \rightarrow (r, h, s)$
  - Commitment $r$
  - Challenge $h = \mathcal{H}(m, r)$
  - Answer $s$
- Verification of $(m, r, s)$
  - compute $h = \mathcal{H}(m, r)$
  - and check $(r, h, s)$

Special soundness

If one can answer to two different challenges $h \neq h'$: $s$ and $s'$ for a unique commitment $r$, one can extract $x$

Splitting Lemma

Idea

When a subset $A$ is “large” in a product space $X \times Y$, it has many “large” sections.

The Splitting Lemma

Let $A \subseteq X \times Y$ such that $\Pr[(x, y) \in A] \geq \varepsilon$. For any $\alpha < \varepsilon$, define

$$B_\alpha = \left\{(x, y) \in X \times Y \mid \Pr_{y' \in Y}[(x, y') \in A] \geq \varepsilon - \alpha\right\},$$

then

(i) $\Pr[B_\alpha] \geq \alpha$

(ii) $\forall (x, y) \in B_\alpha, \Pr_{y' \in Y}[(x, y') \in A] \geq \varepsilon - \alpha$

(iii) $\Pr[B_\alpha \mid A] \geq \alpha / \varepsilon$

Splitting Lemma - Proof

(i) we argue by contradiction, using the notation $\widetilde{B}$ for the complement of $B$ in $X \times Y$. Assume that $\Pr[B_\alpha] < \alpha$. Then,

$$\varepsilon \leq \Pr[B] \cdot \Pr[A \mid B] + \Pr[\widetilde{B}] \cdot \Pr[A \mid \widetilde{B}] < \alpha \cdot 1 + 1 \cdot (\varepsilon - \alpha) = \varepsilon.$$

(ii) straightforward.

(iii) using Bayes’ law:

$$\Pr[B \mid A] = 1 - \Pr[\widetilde{B} \mid A] = 1 - \Pr[A \mid \widetilde{B}] \cdot \Pr[\widetilde{B}] / \Pr[A] \geq 1 - (\varepsilon - \alpha) / \varepsilon = \alpha / \varepsilon.$$
Theorem (The Forking Lemma)

Let \((K,S,V)\) be a digital signature scheme with security parameter \(k\), with a signature as above, of the form \((m,r,h,s)\), where \(h = H(m,r)\) and \(s\) depends on \(r\) and \(h\) only.

Let \(A\) be a probabilistic polynomial time Turing machine whose input only consists of public data and which can ask \(q_H\) queries to the random oracle, with \(q_H > 0\).

We assume that, within the time bound \(T\), \(A\) produces, with probability \(\epsilon \geq 7q_H/2^k\), a valid signature \((m,r,h,s)\).

Then, within time \(T' \leq 16q_HT/\epsilon\), and with probability \(\epsilon' \geq 1/9\), a replay of this machine outputs two valid signatures \((m,r,h,s)\) and \((m,r,h',s')\) such that \(h \neq h'\).

\(\Box\)

Forking Lemma - Proof

We then define the sets

\[
S = \left\{(\omega,H) \mid A_H^H(\omega) \text{ succeeds} \land \text{Ind}_H(\omega) \neq \infty \right\},
\]

\[
S_i = \left\{(\omega,H) \mid A_H^H(\omega) \text{ succeeds} \land \text{Ind}_H(\omega) = i \right\} \quad i \in \{1, \ldots, q_H\}.
\]

Note: the set \(\{S_i\}\) is a partition of \(S\).

\[
\nu = \Pr[S] \geq \epsilon - 1/2^k.
\]

Since \(\epsilon \geq 7q_H/2^k \geq 7/2^k\), then

\[
\nu \geq 6\epsilon/7.
\]

\(\Box\)

Forking Lemma - Proof

Let \(l\) be the set consisting of the most likely indices \(i\),

\[
l = \{i \mid \Pr[S_i \mid S] \geq 1/2q_H\}.
\]

Lemma

\[
\Pr[\text{Ind}_H(\omega) \in l \mid S] \geq \frac{1}{2}.
\]

By definition of \(S_i\),

\[
\Pr[\text{Ind}_H(\omega) \in l \mid S] = \sum_{i \in l} \Pr[S_i \mid S] = 1 - \sum_{i \notin l} \Pr[S_i \mid S].
\]

Since the complement of \(l\) contains fewer than \(q_H\) elements,

\[
\sum_{i \notin l} \Pr[S_i \mid S] \leq q_H \times 1/2q_H \leq 1/2.
\]
Forking Lemma - Proof

We run $2/\varepsilon$ times $\mathcal{A}$, with independent random $\omega$ and random $\mathcal{H}$. Since $\nu = \Pr[S] \geq 6\varepsilon / 7$, with probability greater than $1 - (1 - \nu)^{2/\varepsilon} \geq 4/5$, we get at least one pair $(\omega, \mathcal{H})$ in $S$.

We apply the Splitting Lemma, with $\varepsilon = \nu / 2q_H$ and $\alpha = \varepsilon / 2$, for $i \in I$. We denote by $\mathcal{H}_{ij}$ the restriction of $\mathcal{H}$ to queries of index $i < j$.

Since $\Pr[S] \geq \nu / 2q_H$, there exists a subset $\Omega_i$ such that,

$$\forall (\omega, \mathcal{H}) \in \Omega_i, \quad \Pr_{\mathcal{H}'}(\omega, \mathcal{H}') \in S_i | \mathcal{H}_{ij} = \mathcal{H}_{ij}] \geq \frac{\nu}{4q_H} \quad \Pr[\Omega_i | S] \geq \frac{1}{2}.$$ 

Since all the subsets $S_i$ are disjoint,

$$\Pr[(\exists i \in I) (\omega, \mathcal{H}) \in \Omega_i \cap S_i | S]$$

$$= \sum_{i \in I} \Pr[\Omega_i \cap S_i | S] = \frac{\sum_{i \in I} \Pr[S_i | S]}{2} \geq \frac{1}{4}.$$

We let $\beta$ denote the index $\text{Ind}_\mathcal{H}(\omega)$ of the successful pair.

With prob. at least $1/4$, $\beta \in I$ and $(\omega, \mathcal{H}) \in S_\beta \cap \Omega_\beta$.

With prob. greater than $4/5 \times 1/4 = 1/5$, the $2/\varepsilon$ attacks provide a successful pair $(\omega, \mathcal{H})$, with $\beta = \text{Ind}_\mathcal{H}(\omega) \in I$ and $(\omega, \mathcal{H}) \in S_\beta$.

We know that $\Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in S_\beta | \mathcal{H}'_{ij} = \mathcal{H}_{ij}] \geq \nu / 4q_H$. Then

$$\Pr_{\mathcal{H}'}(\omega, \mathcal{H}') \in S_\beta \text{ and } h_\beta \neq h'_\beta | \mathcal{H}'_{ij} = \mathcal{H}_{ij}]$$

$$\geq \Pr_{\mathcal{H}'}(\omega, \mathcal{H}') \in S_\beta | \mathcal{H}'_{ij} = \mathcal{H}_{ij}] - \Pr_{\mathcal{H}'}[h'_\beta = h_\beta] \geq \nu / 4q_H - 1/2^\kappa,$$

where $h_\beta = \mathcal{H}(Q_\beta)$ and $h'_\beta = \mathcal{H}'(Q_\beta)$.

Using the assumption that $\varepsilon \geq 7q_H / 2^\kappa$, the above prob. is $\geq \varepsilon / 14q_H$.

We replay the attack $14q_H / \varepsilon$ times with a new random oracle $\mathcal{H}'$ such that $\mathcal{H}'_{ij} = \mathcal{H}_{ij}$, and get another success with probability greater than

$$1 - (1 - \varepsilon / 14q_H)^{14q_H / \varepsilon} \geq 3/5.$$
In order to answer signing queries, one simply uses the simulator of the zero-knowledge proof: \((r, h, s)\), and we set \(h(m, r) \leftarrow h\). The random oracle programming may fail, but with negligible probability.

### Conclusion

Two generic methodologies for signatures
- hash and invert
- the Forking Lemma

Both in the random-oracle model
- Cramer-Shoup: based on the flexible RSA problem
- Based on Pairings
- etc