Provable Security in the Computational Model

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

Public-Key Encryption

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

\[
\begin{align*}
\text{Succ}^{ow}(A) &= \Pr[(sk,pk) \leftarrow \mathcal{K}; m \xleftarrow{\text{R}} \mathcal{M}; c = \mathcal{E}_{pk}(m) : A(pk, c) \rightarrow m]
\end{align*}
\]

**IND – CPA Security Game**

\[
\begin{align*}
\text{Adv}^{ind\text{-cpa}}_S(A) &= \Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0] = 2\times\Pr[b' = b] - 1
\end{align*}
\]

**Outline**

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

2. **Advanced Security for Signature**

3. **Forking Lemma**

4. **Conclusion**
EUF − NMA

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

The adversary knows the public key only, whereas signatures are not private!
EUF – CMA

∀i, m ≠ m ∀i, m ≠ m
V(k_v, m, σ)? V(k_v, m, σ)?

The adversary has access to any signature of its choice:
Chosen-Message Attacks (oracle access):

Succ_{euf-cma}(A) = \Pr \left[ (sk, pk) \leftarrow K(); (m, σ) \leftarrow A^S(pk): \forall i, m \neq m_i \land V_{pk}(m, σ) = 1 \right] 

SUF – CMA

∀i, m ≠ m ∀i, m ≠ m
V(k_v, m, σ)? V(k_v, m, σ)?

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

Succ_{suf-cma}(A) = \Pr \left[ (sk, pk) \leftarrow K(); (m, σ) \leftarrow A^S(pk): \forall i, (m, σ) \neq (m_i, σ_i) \land V_{pk}(m, σ) = 1 \right] 

Outline

1 Basic Security Notions
2 Advanced Security for Signature
   2.1 Advanced Security Notions
   2.2 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 ∈ Y: σ = g(M);
- Verification of (M, σ): check f(σ) = M

Full-Domain Hash (Hash-and-Invert)

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

- in order to sign m, one computes M = \mathcal{H}(m) ∈ Y, and σ = g(M)
- and the verification consists in checking whether f(σ) = H(m)
**Random Oracle Model**

**Random Oracle**
- $\mathcal{H}$ is modelled as a truly random function, from $\{0,1\}^*$ into $Y$.
- Formally, $\mathcal{H}$ is chosen at random at the beginning of the game.
- More concretely, for any new query, a random element in $Y$ is uniformly and independently drawn.

Any security game becomes:

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

**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:

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

**Assumptions:**
- any signing query has been first asked to $\mathcal{H}$
- the forgery has been asked to $\mathcal{H}$
- $\tau_f$ 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
- **Game_2**: use of the simulation of the Signing Oracle

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

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

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

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

$\mathcal{S}(m)$: find $\mu$ such that $M = \mathcal{H}(m) = f(\mu)$, output $\sigma = \mu$

$\Rightarrow$ **Hop-S-Perfect**: $\Pr_{\text{Game}_2}[1] = \Pr_{\text{Game}_1}[1]$
**Game**\textsubscript{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

We terminate the game and output 0 if \(\text{Ev}\) happens

\(\Rightarrow \) \text{Hop-S-Non-Negl}

Then, clearly

\[
\Pr[\text{Ev}] = 1 - 1/q_H
\]

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

**Summary**

In **Game**\textsubscript{4}, when the output is 1, \(\sigma = g(y) = g(f(x)) = x\)

and the simulator computes one exponentiation per hashing:

\[
\Pr[1] \leq \text{Succ}_{\text{FDH}}^{\text{euf-cma}}(A) \leq q_H \times \text{Succ}_{P}^{\text{ow}}(t + q_H \tau_f)
\]

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

If one allows \(q_H\) up to \(2^{60}\)

Then one needs \(\text{Succ}_{P}^{\text{ow}}(t) \leq \varepsilon\) with \(t/\varepsilon \geq 2^{140}\).

If one uses FDH-RSA: at least 3072 bit keys are needed.
Improvement

In the case that \( f \) is homomorphic (as RSA): \( f(ab) = f(a)f(b) \)

- \( \text{Game}_0 \): use of the oracles \( K, S \) and \( H \)
- \( \text{Game}_1 \): use of the simulation of the Random Oracle

Simulation of \( H \)

\( H(m) \): \( \mu \leftarrow X \), output \( M = f(\mu) \)

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

- \( \text{Game}_2 \): use of the homomorphic property

\( P - \text{OW} \) instance \( (f, y) \) (where \( f \leftarrow P, x \leftarrow X, y = f(x) \))

Simulation of \( S \)

\( S(m) \): find \( \mu \) such that \( M = H(m) = f(\mu) \), output \( \sigma = \mu \)

Fails (with output 0) if \( H(m) = M = y \times f(\mu) \):

\[ \Rightarrow \text{Hop-S-Non-Negl} \quad \Pr_{\text{Game}_3[1]} = \Pr_{\text{Game}_2[1]} \times q^S \]

Summary

In \( \text{Game}_3 \), when the output is 1, with probability \( 1 - p \):

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

\[ \Pr_{\text{Game}_3[1]} \leq \text{Succ}^\omega_P(t + q_H T_f) / (1 - p) \]

\[ \Pr_{\text{Game}_2[1]} = \Pr_{\text{Game}_1[1]} \times p^q_S \]

\[ \Pr_{\text{Game}_1[1]} = \Pr_{\text{Game}_0[1]} \]

\[ \Pr_{\text{Game}_0[1]} = \text{Succ}^{\omega-\text{cma}}(A) \]

\[ \text{Succ}^{\omega-\text{cma}}(A) \leq \frac{1}{(1 - p)p^q_S} \times \text{Succ}^\omega_P(t + q_H T_f) \]

The maximal for \( p \mapsto (1 - p)p^q_S \) is reached for

\[ p = 1 - \frac{1}{q_S + 1} \Rightarrow \frac{1}{q_S + 1} \times \left( 1 - \frac{1}{q_S + 1} \right)^q_S \approx e^{-1} \]

- If one wants \( \text{Succ}^{\omega-\text{cma}}(t) \leq \varepsilon \) with \( t/\varepsilon \approx 2^{80} \)
- If one allows \( q_S \) up to \( 2^{30} \)

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

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

Signature Oracle

- \( \text{Game}_3 \): use of the simulation of the Signing Oracle

Key Size

\[ \text{Succ}^{\omega-\text{cma}}(A) \leq \frac{1}{(1 - p)p^q_S} \times \text{Succ}^\omega_P(t + q_H T_f) \]
Outline

1 Basic Security Notions
2 Advanced Security for Signature
3 Forking Lemma
   - Zero-Knowledge Proofs
   - The Forking Lemma
4 Conclusion

Proof of Knowledge

How do I prove that I know a solution $s$ to a problem $P$?

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 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.

Schnorr Proofs

Zero-Knowledge Proof

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

Signature

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

Generic Zero-Knowledge Proofs

Secure Multiple Proofs of Knowledge: Authentication

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

- Cheater
  - \(C_1\)
  - \(\omega_1\)
- Simulator
  - \(S\)
  - \(\omega_S\)
- Extractor
  - \(E\)
  - \(\omega_E\)
- Public Data \(P\)
- Communication
- History \(Hist\)
- Cheater \(C_2\)
  - \(\omega_2\)
- Signature \(H\) viewed as a random oracle
- Key Generation \(\rightarrow (y, x)\)
  - private: \(x\)  public: \(y\)
- Signature of \(m \rightarrow (r, h, s)\)
  - Commitment \(r\)
  - Challenge \(h = H(m, r)\)
  - Answer \(s\)
- Verification of \((m, r, s)\)
  - compute \(h = H(m, r)\)
  - and check \((r, h, s)\)
Protocols

### Zero-Knowledge Proof
- Proof of knowledge of $x$
- $P$ sends a commitment $r$
- $V$ sends a challenge $h$
- $P$ sends the answer $s$
- $V$ 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 $x$

## Outline

1. Basic Security Notions
2. Advanced Security for Signature
3. Forking Lemma
   - Zero-Knowledge Proofs
   - The Forking Lemma
4. Conclusion

### 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 $A \subset X \times Y$ 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 $\bar{B}$ for the complement of $B$ in $X \times Y$. Assume that $\Pr[B_\alpha] < \alpha$. Then,

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

(ii) straightforward.

(iii) using Bayes’ law:

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

**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 $\varepsilon \geq 7q_H/2^k$, a valid signature $(m, r, h, s)$. Then, within time $T' \leq 16q_HT/\varepsilon$, and with probability $\varepsilon' \geq 1/9$, a replay of this machine outputs two valid signatures $(m, r, h, s)$ and $(m, r, h', s')$ such that $h \neq h'$.

**Forking Lemma - Proof**

We then define the sets

\[
S = \{(ω, H) | AH(ω) succeeds & Ind_H(ω) \neq \infty\},
\]

\[
S_i = \{(ω, H) | AH(ω) succeeds & Ind_H(ω) = i\} \quad i \in \{1, \ldots, q_H\}.
\]

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

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

Since $\varepsilon \geq 7q_H/2^k \geq 7/2^k$, then

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

Let $I$ be the set consisting of the most likely indices $i$,

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

**Lemma**

\[
\Pr[\text{Ind}_H(ω) \in I | S] \geq \frac{1}{2}.
\]

By definition of $S_i$,

\[
\Pr[\text{Ind}_H(ω) \in I | S] = \sum_{i \in I} \Pr[S_i | S] = 1 - \sum_{i \notin I} \Pr[S_i | S].
\]

Since the complement of $I$ contains fewer than $q_H$ elements,

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

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

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

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

$$\forall (\omega, \mathcal{H}) \in \Omega_i, \quad \Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in \mathcal{S} | \mathcal{H}'_i = \mathcal{H}_i] \geq \frac{\nu}{4q_H}$$

$$\Pr[\Omega_i | \mathcal{S}] \geq \frac{1}{2}.$$

We let $\beta$ denote the index $\text{Ind}_H(\omega)$ of to 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 provided a successful pair $(\omega, \mathcal{H})$, with $\beta = \text{Ind}_H(\omega) \in I$ and $(\omega, \mathcal{H}) \in S_\beta$.

Forking Lemma - Proof

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

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

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

Using the assumption that $\varepsilon \geq 7q_H/2^k$, 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 $\mathcal{H}'_\beta = \mathcal{H}_\beta$, and get another success with probability greater than

$$1 - (1 - \varepsilon/14q_H)^{14q_H/\varepsilon} \geq 3/5.$$

Finally, after less than $2/\varepsilon + 14q_H/\varepsilon$ repetitions of the attack, with probability greater than $1/5 \times 3/5 \geq 1/9$, we have obtained two signatures $(m, r, h, s)$ and $(m, r, h', s')$, both valid w.r.t. their specific random oracle $\mathcal{H}$ or $\mathcal{H}'$:

$$Q_\beta = (m, r) \text{ and } h = H(Q_\beta) \neq H'(Q_\beta) = h'.$$
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