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
## 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

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

\[
(\text{sk}, pk) \leftarrow \mathcal{K}(); (m_b, m_1, \text{state}) \leftarrow A(pk);
\]

\[
b \overset{R}{\leftarrow} \{0, 1\}; c = e_{pk}(m_b); b' \leftarrow A(\text{state}, c)
\]

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

---
**Outline**

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

---

**Signature**

**EUF − NMA**

\[ \text{Goal: Authentication of the sender} \]

---

**EUF − NMA**

\[ \text{The adversary knows the public key only, whereas signatures are not private!} \]
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 $f(\sigma) = H(m)$
Random Oracle Model

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

Any security game becomes:

$$\operatorname{Succ}_{S\mathcal{G}}^{\text{euf-cma}}(A) = \Pr_{\mathcal{H} \leftarrow \mathcal{Y}^\infty; (\text{sk}, \text{pk}) \leftarrow \mathcal{K}(); (m, \sigma) \leftarrow \mathcal{A}^S, \mathcal{H}(\text{pk}) : \forall i, m \neq m_i \wedge \forall \text{pk}(m, \sigma) = 1}$$

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:

$$\operatorname{Succ}_{\text{euf}}$$

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 $\mathcal{H}$

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

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

Simulation of $\mathcal{S}$

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

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

We terminate the game and output 0 if Ev happens

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

Then, clearly

\[
\Pr_{\text{Game}_3}[1] = \Pr_{\text{Game}_2}[1] \times \Pr[\neg \text{Ev}] = 1 - 1/q_H
\]

\[
\Pr_{\text{Game}_3}[1] = \Pr_{\text{Game}_2}[1] \times \frac{1}{q_H}
\]

**Summary**

In \( \text{Game}_4 \), when the output is 1, \( \sigma = g(y) = g(f(x)) = x \)

and the simulator computes one exponentiation per hashing:

\[
\Pr_{\text{Game}_4}[1] \leq \text{Succ}_{FDH}^{\text{euf-cma}}(t + q_H \tau_f)
\]

\[
\Pr_{\text{Game}_4}[1] = \Pr_{\text{Game}_3}[1]
\]

\[
\Pr_{\text{Game}_3}[1] = \Pr_{\text{Game}_2}[1] \times \frac{1}{q_H}
\]

\[
\Pr_{\text{Game}_2}[1] = \Pr_{\text{Game}_1}[1]
\]

\[
\Pr_{\text{Game}_1}[1] = \text{Succ}_{FDH}^{\text{euf-cma}}(A)
\]

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


**OW Instance**

- **Game_4**: \( \mathcal{P} - \text{OW} \) instance \((f, y)\) (where \( f \overset{R}{\leftarrow} \mathcal{P}, x \overset{R}{\leftarrow} \mathcal{X}, y = f(x) \))

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, \( \mathcal{H}(m) \): \( M \leftarrow y \), 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:

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


**Key Size**

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

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

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

Then one needs \( \text{Succ}_{\mathcal{P}}^{\text{ow}}(t) \leq \varepsilon \) 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 $\mathcal{K}$, $\mathcal{S}$ and $\mathcal{H}$
- **Game$_1$:** use of the simulation of the Random Oracle

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

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

- $\implies$ **Hop-D-Perfect**: $\Pr_{\text{Game}_1}[1] = \Pr_{\text{Game}_0}[1]$
- **Game$_2$:** use of the homomorphic property
  - $\mathcal{P} - \text{OW instance } (f, y)$ (where $f \leftarrow \mathcal{P}, x \leftarrow X, y = f(x)$)

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

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

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

---

### Summary

In **Game$_3$**, when the output is 1, with probability $1 - p$:

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

$$\Pr_{\text{Game}_3}[1] \leq \frac{\text{Succ}^{\text{P}}_\mathcal{H}(t + q_H \tau_f)}{(1 - p)}$$

$$\Pr_{\text{Game}_3}[1] = \Pr_{\text{Game}_2}[1] \times p^{q_S}$$

$$\Pr_{\text{Game}_2}[1] = \Pr_{\text{Game}_1}[1]$$

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

$$\Pr_{\text{Game}_0}[1] = \text{Succ}^{\text{euf-cma}}_{FDH}(A)$$

$$\text{Succ}^{\text{euf-cma}}_{FDH}(A) \leq \frac{1}{(1 - p)p^{q_S}} \times \text{Succ}^{\text{P}}_\mathcal{H}(t + q_H \tau_f)$$

### Key Size

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

$$p = 1 - \frac{1}{q_S + 1} \implies \frac{1}{q_S + 1} \times 1 - \frac{1}{q_S + 1} \approx e^{-1}$$

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

Then one needs $\text{Succ}^{\text{P}}_\mathcal{H}(t) \leq \varepsilon$ with $t/\varepsilon \geq 2^{110}$.

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

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

---

**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

[Schon – Eurocrypt ’89 - Crypto ’89]

Signature

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

Zero-Knowledge Proof

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

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

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

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

$$\varepsilon \leq \Pr[B] \cdot \Pr[A \mid B] + \Pr[\tilde{B}] \cdot \Pr[A \mid \tilde{B}] < \alpha \cdot 1 + 1 \cdot (\varepsilon - \alpha) = \varepsilon.$$
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 $\varepsilon \geq 7q_H/2^k$, a valid signature $(m, r, h, s)$.

Then, within time $T' \leq 16q_H T/\varepsilon$, 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 $h \neq h'$.

A is a PPTM with random tape $\omega$.

During the attack, $A$ asks a polynomial number of queries to $H$.

We may assume that these questions are distinct:

- $Q_1, \ldots, Q_{q_H}$ are the $q_H$ distinct questions
- and let $H = (h_1, \ldots, h_{q_H})$ be the list of the $q_H$ answers of $H$.

Note: a random choice of $H = \text{a random choice of } H$.

For a random choice of $(\omega, H)$, with probability $\varepsilon$, $A$ outputs a valid signature $(m, r, h, s)$.

Since $H$ is a random oracle, the probability for $h$ to be equal to $H(m, r)$ is less than $1/2^k$, unless it has been asked during the attack.

Accordingly, we define $\text{Ind}_H(\omega)$ to be the index of this question:

$(m, r) = Q_{\text{Ind}_H(\omega)}$ \quad $(\text{Ind}_H(\omega) = \infty$ if the question is never asked).

We then define the sets

\[ S = (\omega, H) \mid A^H(\omega) \text{ succeeds & } \text{Ind}_H(\omega) \neq \infty, \]
\[ S_i = (\omega, H) \mid A^H(\omega) \text{ succeeds & } \text{Ind}_H(\omega) = i \quad i \in \{1, \ldots, q_H\}. \]

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

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

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

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

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

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

Lemma

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

By definition of $S_i$,

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

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

\[ \text{Pr}[S_i \mid S] \leq q_H \times 1/2q_H \leq 1/2. \]
Forking Lemma – Proof

We run $2/\varepsilon$ times $A$, with independent random $\omega$ and random $\mathcal{H}$. Since $\nu = \Pr[S] \geq 6\varepsilon/7$, 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 $\varepsilon = \nu/2q_{\mathcal{H}}$ and $\alpha = \varepsilon/2$, for $i \in I$. We denote by $\mathcal{H}_{ij}$ the restriction of $\mathcal{H}$ to queries of index $< i$.

Since $\Pr[S_i] \geq \nu/2q_{\mathcal{H}}$, there exists a subset $\Omega_i$ such that
\[
\forall (\omega, \mathcal{H}) \in \Omega_i, \quad \Pr[(\omega, \mathcal{H}') \in S_i | \mathcal{H}'_{ij} = \mathcal{H}_{ij}] \geq \frac{\nu}{4q_{\mathcal{H}}},
\]
\[
\Pr[\Omega_i | S_i] \geq \frac{1}{2}.
\]

Since all the subsets $S_i$ are disjoint,
\[
\Pr[(\exists i \in I) (\omega, \mathcal{H}) \in \Omega_i \cap S_i | S] = \left[\text{\# \Omega_i} \cap S_i \right] \Pr[\Omega_i \cap S_i | S] = \prod_{i \in I} \Pr[\Omega_i] \cdot \Pr[S_i | S] \geq \prod_{i \in I} \Pr[S_i | S] / 2 \geq \frac{1}{4}.
\]

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 provided a successful pair $(\omega, \mathcal{H})$, with $\beta = \text{Ind}_\mathcal{H}(\omega) \in I$ and $(\omega, \mathcal{H}) \in S_\beta$.

Forking Lemma – Proof

We know that $\Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in S_\beta | \mathcal{H}'_{\beta} = \mathcal{H}_{\beta}] \geq \nu/4q_{\mathcal{H}}$. Then
\[
\Pr[(\omega, \mathcal{H}') \in S_\beta \text{ and } h_\beta \neq h'_\beta] \leq \Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in S_\beta | \mathcal{H}'_{\beta} = \mathcal{H}_{\beta}] - \Pr_{\mathcal{H}'}[h'_\beta = h_\beta] \geq \nu/4q_{\mathcal{H}} - 1/2^k,
\]
where $h_\beta = \mathcal{H}(Q_\beta)$ and $h'_\beta = \mathcal{H}'(Q_\beta)$.

Using the assumption that $\varepsilon \geq 7q_{\mathcal{H}}/2^k$, the above prob. is $\geq \varepsilon/14q_{\mathcal{H}}$.

We replay the attack $14q_{\mathcal{H}}/\varepsilon$ times with a new random oracle $\mathcal{H}'$ such that $\mathcal{H}'_{\beta} = \mathcal{H}_{\beta}$, and get another success with probability greater than $1 - (1 - \varepsilon/14q_{\mathcal{H}})^{14q_{\mathcal{H}}/\varepsilon} \geq 3/5$.

Finally, after less than $2/\varepsilon + 14q_{\mathcal{H}}/\varepsilon$ 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 $\mathcal{H}'$:
\[
Q_\beta = (m, r) \text{ and } h = \mathcal{H}(Q_\beta) \neq \mathcal{H}'(Q_\beta) = h'.
\]
In order to answer signing queries, one simply uses the simulator of the zero-knowledge proof: \( (r, h, s) \), and we set \( ?(m, r) \leftarrow h \). 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