III - Signatures

David Pointcheval
Ecole normale supérieure, CNRS & INRIA

Public-Key Encryption

Goal: Privacy/Secrecy of the plaintext

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

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

\[
m^* \text{ random}
\]
\[
r^* \text{ random}
\]
\[
m^* \xrightarrow{?} m
\]
\[
m \rightarrow E
\]
\[
A
\]
\[
b' \xrightarrow{?} b
\]

**IND – CPA Security Game**

\[
(k_s, k_d) \xleftarrow{\mathcal{G}} k_s \rightarrow k_d
\]
\[
b \in \{0,1\}
\]
\[
r \text{ random}
\]
\[
m_0, m_1 \xrightarrow{\mathcal{R}} M
\]
\[
c = E_{pk}(m)
\]
\[
A \rightarrow m
\]
\[
\text{Succ}^w_S(A) = \Pr[(sk, pk) \leftarrow \mathcal{K}(); m \xrightarrow{R} M; c = E_{pk}(m) : A(pk, c) \rightarrow m]
\]
\[
(.sk, pk) \leftarrow \mathcal{K}(); (m_0, m_1, \text{state}) \leftarrow A(pk);
\]
\[
b \xrightarrow{R} \{0,1\}; c = E_{pk}(m_b); b' \leftarrow A(\text{state}, c)
\]

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

---

**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**
Signature

Goal: Authentication of the sender

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

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):**

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

The notion is even stronger (in case of probabilistic signature):

also known as **non-malleability**:

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

---

**Outline**

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

---

**Signature Scheme**

- Key generation: the public key \( f^{-1} \) is a trapdoor one-way bijection from \( X \) onto \( Y \); the private key is the inverse \( g : Y \to X \);
- Signature of \( M \in Y \): \( \sigma = g(M) \);
- Verification of \( (M, \sigma) \): check \( f(\sigma) = M \)

**Full-Domain Hash (Hash-and-Invert)**

\[ H : \{0, 1\}^2 \to Y \]

- in order to sign \( m \), one computes \( M = H(m) \in Y \), 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 $P$, in the Random Oracle Model:

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

Assumptions:

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

Real Attack Game

Game 0

Oracles

K S H

Challenger

• (pk, sk) ← K()
• Checks ($m$, $\sigma$)
• if new and valid: 1
• else 0

Adversary

0 / 1

Game 1

Simulation of $H$

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

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

Game 2

Simulation of $S$

$S(m): \mu \overset{R}{\leftarrow} X$, output $\sigma = g(M)$

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

Simulations

- Game$_0$: use of the oracles K, S and H
- Game$_1$: use of the simulation of the Random Oracle
- Game$_2$: use of the simulation of the Signing Oracle

Random Oracle

$H(m): M \overset{R}{\leftarrow} Y$, output $M$

Key Generation Oracle

$K(): (f, g) \overset{R}{\leftarrow} P$, $sk \leftarrow g$, $pk \leftarrow f$

Signing Oracle

$S(m): M = H(m)$, output $\sigma = g(M)$
H-Query Selection

- **Game\(_3\)**: random index \( t \leftarrow \{1, \ldots, q_H\} \)

**Event Ev**

If the \( t \)-th query to \( H \) is not the output forgery

⇒ **Hop-S-Non-Negl**

Then, clearly

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

**Summary**

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

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

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

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

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

### Simulation of $H$

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

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

### Simulation of $H$

$H(m)$: flip a biased coin $b$ (with $\Pr[b = 0] = p$), $\mu \xleftarrow{\$} X$.

If $b = 0$, output $M = f(\mu)$, otherwise output $M = y \times f(\mu)$

- $\Rightarrow$ **Hop-D-Perfect**: $\Pr_{\text{Game}_2}[1] = \Pr_{\text{Game}_0}[1]$

### Summary

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

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

### Key Size

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

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

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

- 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 $\text{Succ}^P(t) \leq \varepsilon$ 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 $P$?

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

Zero-Knowledge Proof
- Setting: \(G = \mathbb{G}_1\) of order \(q\)
- \(P\) knows \(x\), such that \(y = g^{-x}\) and wants to prove it to \(V\)
- \(P\) chooses \(K \xleftarrow{\$} \mathbb{Z}_q^*\)
- \(V\) chooses \(h \xleftarrow{\$} \{0, 1\}^k\)
- \(P\) computes and sends \(r = g^K\)
- \(V\) checks whether \(r \equiv g^s y^h\)

Signature
- \((G = \mathbb{G}_1)\) 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 \(m \rightarrow (r, h, s)\)
  - \(K \xleftarrow{\$} \mathbb{Z}_q^*\)
  - \(r = g^K\)
  - \(h = H(m, r)\)
  - \(s = K + xh \mod q\)
- Verification of \((m, r, s)\)
  - compute \(h = H(m, r)\)
  - and check \(r \equiv g^s y^h\)

Generic Zero-Knowledge Proofs

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

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 \((r, h, s)\)
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 $h = H(m, r)$
  - Answer $s$
- Verification of $(m, r, s)$
  - compute $h = H(m, r)$
  - 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 $x$
**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 \(\varepsilon' \geq 1/9\), 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

\[
S = \{(\omega, H) | A^H(\omega) \text{ succeeds } \text{ and } \text{Ind}^H(\omega) \neq \infty\},
\]

\[
S_i = \{(\omega, H) | A^H(\omega) \text{ succeeds } \text{ and } \text{Ind}^H(\omega) = i \quad i \in \{1, \ldots, q_H\}\}.
\]

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**

\[
\Pr[\text{Ind}^H(\omega) \in I | S] \geq \frac{1}{2}.
\]

By definition of \(S_i\),

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

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

\[
\Pr[S_j | S] \leq q_H \times 1/2q_H \leq 1/2.
\]
We run $2/\varepsilon$ times $A$, with independent random $\omega$ and random $H$. 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 $(\omega, H)$ in $S$.

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

Since $\Pr[S_i] \geq \nu/2q_h$, there exists a subset $\Omega_i$ such that:

$$\forall (\omega, H) \in \Omega_i, \quad \Pr_{H'}[(\omega, H') \in S_i | H_{|i} = H_{|i}] \geq \frac{\nu}{4q_h}.$$ 

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

Since all the subsets $S_i$ are disjoint,

$$\Pr_{(\omega, H)} \left[ (\exists i \in I)(\omega, H) \in \Omega_i \cap S_i | S \right] \\
= \Pr_{(\omega, H)} \left[ \bigcap_{i \in I} (\Omega_i \cap S_i) | S \right] \\
= \prod_{i \in I} \Pr[\Omega_i | S_i] \cdot \Pr[S_i | S] \geq \prod_{i \in I} \frac{1}{2} \geq \frac{1}{4}.$$

We let $\beta$ denote the index $Ind_H(\omega)$ of the successful pair. With prob. at least $1/4$, $\beta \in I$ and $(\omega, H) \in S_\beta \cap \Omega_\beta$.

With prob. greater than $4/5 \times 1/4 = 1/5$, the $2/\varepsilon$ attacks provided a successful pair $(\omega, H)$, with $\beta = Ind_H(\omega) \in I$ and $(\omega, H) \in S_\beta$.

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

$$\Pr_{H'}[(\omega, H') \in S_\beta \text{ and } h_\beta \notin h'_\beta | H_{|\beta} = H_{|\beta}]$$

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

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 $H'$ such that $H_{|\beta} = H_{|\beta}$, and get another success with probability greater than

$$1 - (1 - \varepsilon/14q_h)^{14q_h/\varepsilon} \geq 3/5.$$
Chosen-Message Attacks

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.

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

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