III – Signatures

Basic Security Notions

Advanced Security for Signature

Forking Lemma

Conclusion
Public-Key Encryption

Goal: Privacy/Secrecy of the plaintext

OW – CPA Security Game

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

IND – CPA Security Game

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

Outline

Basic Security Notions
- Public-Key Encryption
- Signatures

Advanced Security for Signature
- Forking Lemma

Conclusion
**Goal:** Authentication of the sender

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

---

**Outline**

- Basic Security Notions
- **Advanced Security for Signature**
  - Advanced Security Notions
  - Hash-then-Invert Paradigm
  - Forking Lemma
- Conclusion
Goal: Authentication of the sender

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):

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

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

\[ \text{Succ}_{\text{Suf-CMA}}(A) = \Pr \left[ (sk, pk) \leftarrow \mathcal{K}(); (m, \sigma) \leftarrow A^S(pk) : \forall i, (m, \sigma) \neq (m_i, \sigma_i) \land V_{pk}(m, \sigma) = 1 \right] \]
Full-Domain Hash Signature

Signature Scheme

- Key generation: the public key $f \leftarrow R \mathcal{P}$ 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 $Y$.
- 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:

\[
\text{Succ}_{\text{euf−cma}}^{\text{FDH}}(\mathcal{A}) = \Pr \left[ \mathcal{H} \leftarrow Y^\infty; (sk, pk) \leftarrow \mathcal{K}(); (m, \sigma) \leftarrow \mathcal{A}^S,\mathcal{H}(pk); \forall i, m \neq m_i \land V_{pk}(m, \sigma) = 1 \right] 
\]

Theorem

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

\[
\text{Succ}_{\text{euf−cma}}^{\text{FDH}}(t) \leq q_H \times \text{Succ}_{\text{cma}}^{\text{OW}}(t + q_H \tau_f)
\]

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

Challenger

$$(pk, sk) \leftarrow K()$$

Checks $$(m, \sigma)$$

- if new and valid: 1
- else 0

Adversary

Game 0

pk, m, σ

Oracles

S HK

Random Oracle

$$(H(m): M \leftarrow Y, output M)$$

Key Generation Oracle

$$K()$$

$$(f, g) \leftarrow \mathcal{P}, sk \leftarrow g, pk \leftarrow f$$

Signing Oracle

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

Simulations

- Game 0: use of the oracles $$K, S$$ and $$H$$
- Game 1: use of the simulation of the Random Oracle

Simulation of $$H$$

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

$$\Rightarrow$$ Hop-D-Perfect: $$Pr_{Game_1}[1] = Pr_{Game_0}[1]$$

- Game 2: use of the simulation of the Signing Oracle

Simulation of $$S$$

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

$$\Rightarrow$$ Hop-S-Perfect: $$Pr_{Game_2}[1] = Pr_{Game_1}[1]$$

OW Instance

- Game 3: random index $$t \leftarrow \{1, \ldots, q_H\}$$

Event Ev

If the $$t$$-th query to $$H$$ is not the output forgery

We terminate the game and output 0 if Ev happens

$$\Rightarrow$$ Hop-S-Non-Negl

Then, clearly

$$Pr_{Game_3}[1] = Pr_{Game_2}[1] \times Pr[\neg Ev]$$

$$Pr[Ev] = 1 - 1/q_H$$

$$Pr_{Game_3}[1] = Pr_{Game_2}[1] \times \frac{1}{q_H}$$

- Game 4: $$P - OW$$ instance $$(f, y)$$ (where $$f \leftarrow \mathcal{P}, x \leftarrow X, y = f(x))$$

Use of the simulation of the Key Generation Oracle

Simulation of $$K$$

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

Modification of the simulation of the Random Oracle

Simulation of $$H$$

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

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:

$$\Rightarrow$$ Hop-S-Perfect: $$Pr_{Game_4}[1] = Pr_{Game_3}[1]$$

ENS/CNRS/INRIA Cascade

David Pointcheval

20/51
In Game$_4$, when the output is 1, $\sigma = g(y) = g(f(x)) = x$
and the simulator computes one exponentiation per hashing:

$$\Pr_{\text{Game}_4}[1] \leq \text{Succ}^{\text{OW}}_{\text{FDH}}(t + q_H T_f)$$

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

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

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

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

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

$$\text{Succ}^{\text{euf−cma}}_{\text{FDH}}(A) \leq q_H \times \text{Succ}^{\text{OW}}_{\text{FDH}}(t + q_H T_f)$$

- 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 $\text{Succ}^{\text{OW}}_{\text{FDH}}(t) \leq \varepsilon$ 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): $f(ab) = f(a)f(b)$

- **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 homomorphic property

Simulation of $H$

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

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

Simulation of $\mathcal{P} − \text{OW}$ instance $(f, y)$ (where $f \xleftarrow{\$} \mathcal{P}, x \xleftarrow{\$} X, y = f(x)$)

**Simulation of S**

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

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

but with probability $p^{qs}$

$\implies$ **Hop-S-Non-Negl**: $\Pr_{\text{Game}_3}[1] = \Pr_{\text{Game}_2}[1] \times p^{qs}$

**Signature Oracle**

- **Game$_3$**: use of the simulation of the Signing Oracle

Simulation of $S$

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

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

but with probability $p^{qs}$

$\implies$ **Hop-S-Non-Negl**: $\Pr_{\text{Game}_3}[1] = \Pr_{\text{Game}_2}[1] \times p^{qs}$
In Game\textsubscript{3}, when the output is 1, with probability $1 - p$:

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

\[
\Pr_{\text{Game}_3} [1] \leq \text{Succ}_{\text{OW}}(t + qH_T f) / (1 - p) \\
\Pr_{\text{Game}_3} [1] = \Pr_{\text{Game}_2} [1] \times p^{q_S} \\
\Pr_{\text{Game}_2} [1] = \Pr_{\text{Game}_1} [1] \\
\Pr_{\text{Game}_1} [1] = \Pr_{\text{Game}_0} [1] \\
\Pr_{\text{Game}_0} [1] = \text{Succ}_{\text{FDH}}^\text{euf-cma}(A)
\]

$$\text{Succ}_{\text{FDH}}^\text{euf-cma}(A) \leq \frac{1}{(1 - p)p^{q_S}} \times \text{Succ}_{\text{OW}}(t + qH_T f)$$

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

$$p = 1 - \frac{1}{q_S + 1} \rightarrow \frac{1}{q_S + 1} \times \left(1 - \frac{1}{q_S + 1}\right)^{q_S} \approx \frac{e^{-1}}{q_S}$$

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

Then one needs $\text{Succ}_{\text{OW}}^\text{OW}(t) \leq \varepsilon$ 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 \( P \)?

- **Prover** \( A \)
  - Communication with **Verifier** \( B \)
  - Secret \( s \)
  - Polynomial Size
  - Polynomial Time

**Verifier** \( B \)

- Communication with Prover \( A \)
- \( \omega_B \)

**Proof of Knowledge: Soundness**

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

- **Prover** \( A \)
  - Communication with **Verifier** \( B \)
  - Communication with **Extractor** \( E \)
  - \( \omega_A \)
  - Secret \( s \)

- **Extractor** \( E \)
  - Communication with **Prover** \( A \)
  - \( \omega_E \)

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

- **Prover** \( A \)
  - Communication with **Verifier** \( B \)
  - \( s \)
  - \( \omega_A \)

- **Verifier** \( B \)
  - \( \omega_B \)

- **Simulator** \( S \)
  - Communication with **Prover** \( A \)
  - \( \omega_S \)

**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:

**Generic Zero-Knowledge Proofs**

- 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 $h \stackrel{R}{\leftarrow} \{0, 1\}^k$ for $\mathcal{P}$
- $\mathcal{P}$ computes and sends the answer $s$
- $\mathcal{V}$ checks $(r, h, s)$

**Signature**

- $\mathcal{P}$ knows $x$, such that $y = g^{-x}$
- $\mathcal{V}$ chooses a challenge $h$ and wants to prove it to $\mathcal{V}$
- $\mathcal{P}$ chooses $K \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$
- $\mathcal{V}$ chooses $r = g^K$
- $\mathcal{P}$ computes and sends $s = K + xh \mod q$
- $\mathcal{V}$ checks whether $r = g^s y^h$

**Schnorr Proofs**

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

**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 $h$ and wants to prove it to $\mathcal{V}$
- $\mathcal{P}$ sends a commitment $r$
- $\mathcal{V}$ sends a challenge $h$
- $\mathcal{P}$ sends the answer $s$
- $\mathcal{V}$ checks $(r, h, s)$

**Signature**

- $\mathcal{V}$ 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 $r = g^s y^h$

**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**

<table>
<thead>
<tr>
<th>Basic Security Notions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Security for Signature</td>
</tr>
<tr>
<td>Forking Lemma</td>
</tr>
<tr>
<td>Zero-Knowledge Proofs</td>
</tr>
<tr>
<td>The Forking Lemma</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
</tbody>
</table>

---

**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

1. $\Pr[B_\alpha] \geq \alpha$
2. $\forall (x,y) \in B_\alpha, \Pr_{y' \in Y}[(x,y') \in A] \geq \varepsilon - \alpha$.
3. $\Pr[B_\alpha \mid A] \geq \alpha/\varepsilon$.

---

**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 $(m,r,h,s)$.

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 $\varepsilon' \geq 1/9$, a replay of this machine outputs two valid signatures $(m,r,h',s')$ such that $h \neq h'$.

---

**Splitting Lemma – Proof**

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

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

(ii) straightforward.

(iii) using Bayes’ law:

$$\Pr[B \mid A] = 1 - \Pr[\overline{B} \mid A]$$

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

- \( \mathcal{A} \) is a PPTM with random tape \( \omega \).
- During the attack, \( \mathcal{A} \) asks a polynomial number of queries to \( \mathcal{H} \).
- 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 \( \mathcal{H} \).

Note: a random choice of \( \mathcal{H} \) is a random choice of \( H \).

- For a random choice of \( (\omega, \mathcal{H}) \), with probability \( \varepsilon \), \( \mathcal{A} \) outputs a valid signature \( (m, r, h, s) \).
- Since \( \mathcal{H} \) is a random oracle, the probability for \( h \) to be equal to \( \mathcal{H}(m, r) \) 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:

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

Forking Lemma – Proof

We then define the sets

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

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

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

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

Forking Lemma – Proof

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

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

Lemma

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

By definition of \( S_i \),

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

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

\[
\sum_{i \notin I} \Pr[S_i \mid S] \leq q_H \times 1/2q_H \leq 1/2.
\]

Forking Lemma – Proof

- 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 \( S \).
- 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 \( \Pr[S_i] \geq \nu/2q_H \), there exists a subset \( \Omega_i \) such that,

\[
\forall (\omega, \mathcal{H}) \in \Omega_i, \quad \Pr[(\omega, \mathcal{H}') \in S_i \mid \mathcal{H}' = \mathcal{H}_i] \geq \frac{\nu}{4q_H},
\]

\[
\Pr[\Omega_i \mid S_i] \geq \frac{1}{2}.
\]
Forking Lemma – Proof

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

Let $\beta$ denote the index $\text{Ind}_H(\omega)$ 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 $\beta = \text{Ind}_H(\omega) \in I$ and $(\omega, H) \in S_\beta$.

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.

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 $H$ or $H'$:

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