

III – Signatures

David Pointcheval

Ecole normale supérieure, CNRS & INRIA



MPRI – Paris

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

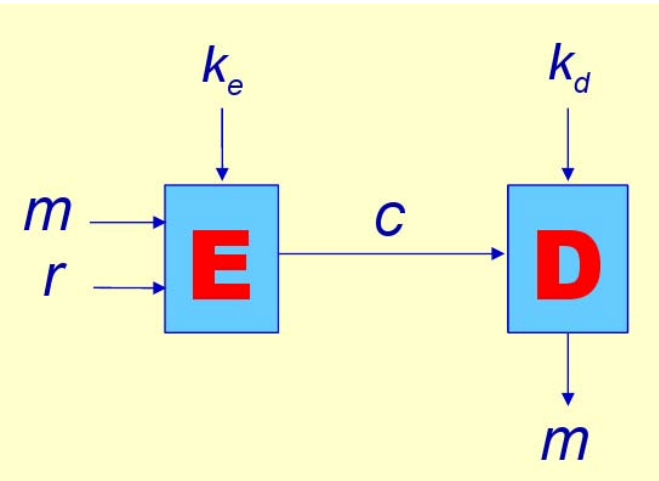
1 Basic Security Notions

- Public-Key Encryption
- Signatures

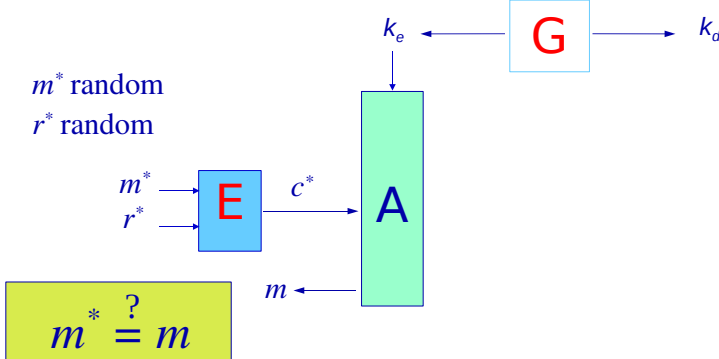
2 Advanced Security for Signature

3 Forking Lemma

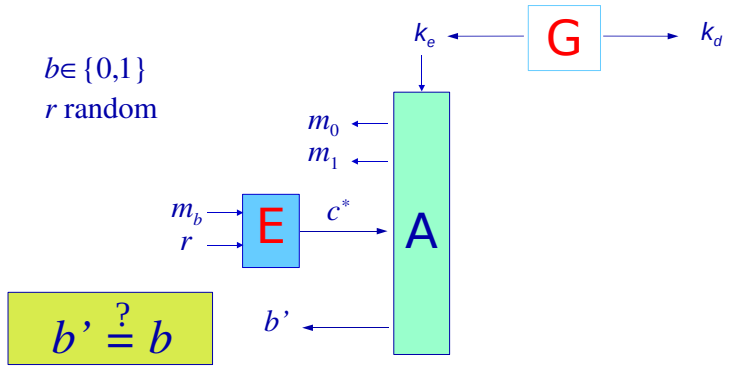
4 Conclusion



Goal: Privacy/Secrecy of the plaintext



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



$$(sk, pk) \leftarrow \mathcal{K}(); (m_0, m_1, \text{state}) \leftarrow \mathcal{A}(pk);$$

$$b \xleftarrow{R} \{0, 1\}; c = \mathcal{E}_{pk}(m_b); b' \leftarrow \mathcal{A}(\text{state}, c)$$

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

Outline

Signature

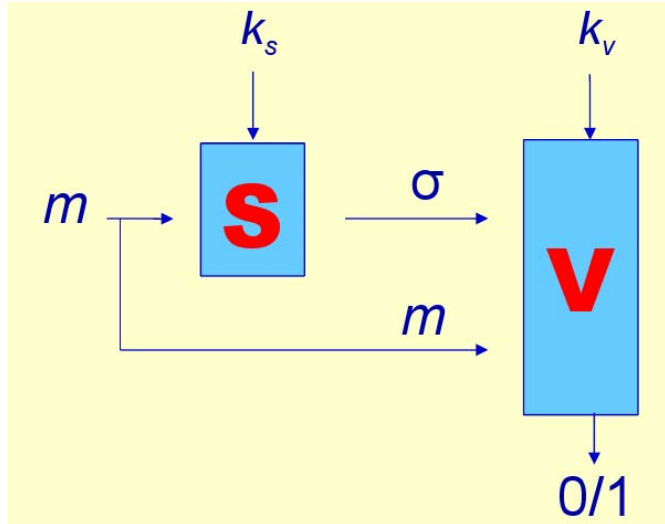
1 Basic Security Notions

- Public-Key Encryption
- Signatures

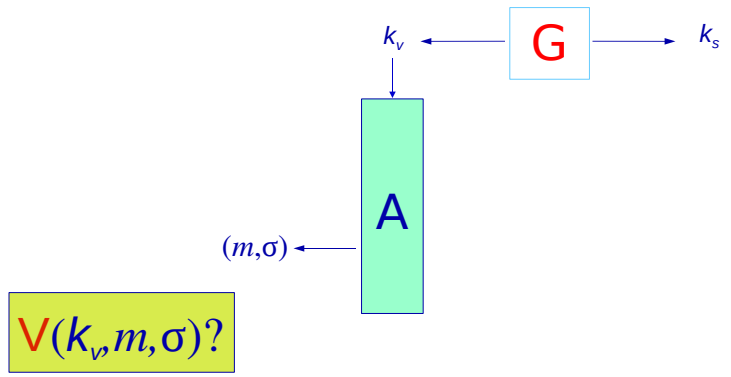
2 Advanced Security for Signature

3 Forking Lemma

4 Conclusion



Goal: Authentication of the sender

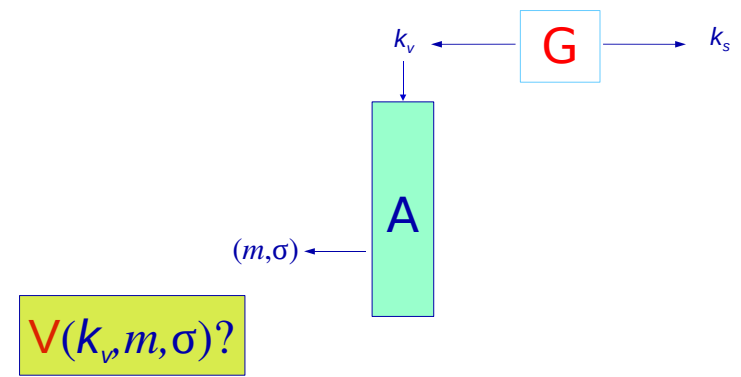
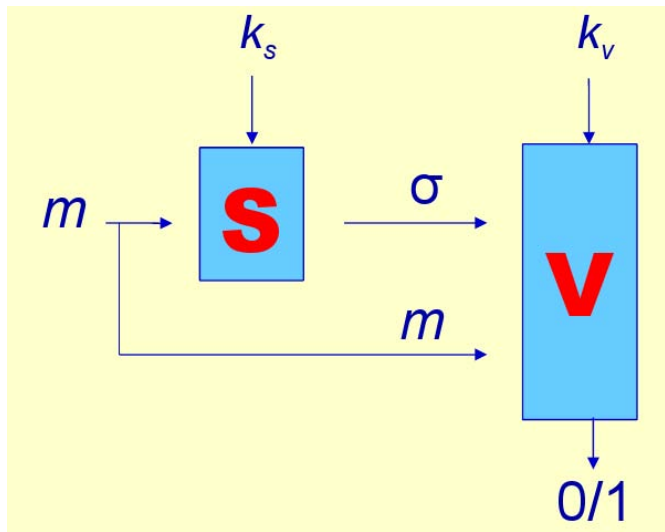


$$\text{Succ}_{SG}^{\text{euf}}(\mathcal{A}) = \Pr[(sk, pk) \leftarrow \mathcal{K}(); (m, \sigma) \leftarrow \mathcal{A}(pk) : \mathcal{V}_{pk}(m, \sigma) = 1]$$

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

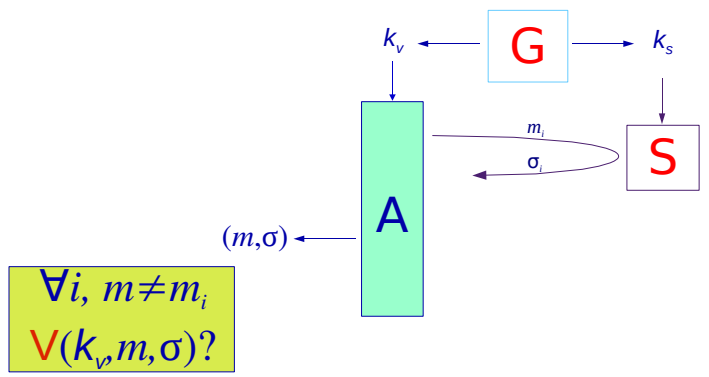
Signature

EU – NMA



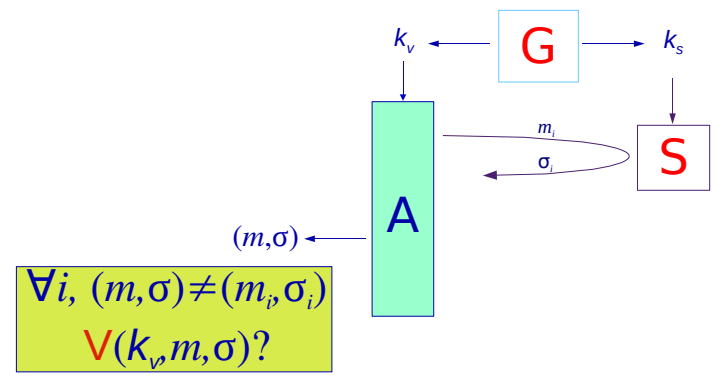
The adversary knows the public key only, whereas signatures are not private!

Goal: Authentication of the sender



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

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



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

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

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

[Bellare-Rogaway – Eurocrypt '94]

Signature Scheme

- Key generation: the public key $f \xleftarrow{R} \mathcal{P}$ 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, σ) : 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

- \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:

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

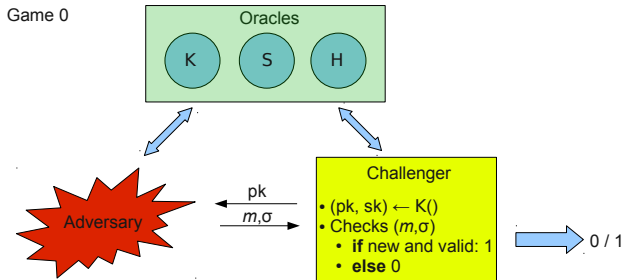
Theorem

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}
 - τ_f is the maximal time to evaluate $f \in \mathcal{P}$

Real Attack Game



Random Oracle

$\mathcal{H}(m): M \xleftarrow{R} Y$, output M

Key Generation Oracle

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

Signing Oracle

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

Simulations

- **Game₀**: use of the oracles \mathcal{K} , S and \mathcal{H}
- **Game₁**: use of the *simulation of the Random Oracle*

Simulation of \mathcal{H}

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

- \implies **Hop-D-Perfect**: $\Pr_{\text{Game}_1}[1] = \Pr_{\text{Game}_0}[1]$
- **Game₂**: use of the *simulation of the Signing Oracle*

Simulation of S

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

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

- **Game₃**: random index $t \xleftarrow{R} \{1, \dots, q_H\}$

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 **Hop-S-Non-Negl**
 Then, clearly

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

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

- **Game₄**: \mathcal{P} – OW instance (f, y) (where $f \xleftarrow{R} \mathcal{P}, x \xleftarrow{R} X, y = f(x)$)
 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 **Hop-S-Perfect**: $\Pr_{\text{Game}_4}[1] = \Pr_{\text{Game}_3}[1]$

Summary

Key Size

In **Game₄**, when the output is 1, $\sigma = g(y) = g(f(x)) = x$
 and the simulator computes one exponentiation per hashing:

$$\Pr_{\text{Game}_4}[1] \leq \text{Succ}_{\mathcal{P}}^{\text{ow}}(t + q_{HTf})$$

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

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

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

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

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

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

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

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

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

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

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

- **Game₀**: use of the oracles \mathcal{K} , \mathcal{S} and \mathcal{H}
- **Game₁**: use of the *simulation of the Random Oracle*

Simulation of \mathcal{H}

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

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

- **Game₂**: use of the *homomorphic property*
 \mathcal{P} – **OW** instance (f, y) (where $f \xleftarrow{R} \mathcal{P}$, $x \xleftarrow{R} X$, $y = f(x)$)

Simulation of \mathcal{H}

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

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

- **Game₃**: use of the *simulation of the Signing Oracle*

Simulation of \mathcal{S}

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

Fails (with output 0) if $\mathcal{H}(m) = M = y \times f(\mu)$:
 but with probability p^{q_S}

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

Summary

In **Game₃**, when the output is 1, with probability $1 - p$:

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

$$\Pr_{\text{Game}_3}[1] \leq \text{Succ}_{\mathcal{P}}^{\text{ow}}(t + q_{HT}t)/(1 - p)$$

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

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

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

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

$$\text{Succ}_{\mathcal{F}DH}^{\text{euf-cma}}(\mathcal{A}) \leq \frac{1}{(1 - p)p^{q_S}} \times \text{Succ}_{\mathcal{P}}^{\text{ow}}(t + q_{HT}t)$$

Key Size

$$\text{Succ}_{\mathcal{F}DH}^{\text{euf-cma}}(\mathcal{A}) \leq \frac{1}{(1 - p)p^{q_S}} \times \text{Succ}_{\mathcal{P}}^{\text{ow}}(t + q_{HT}t)$$

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

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

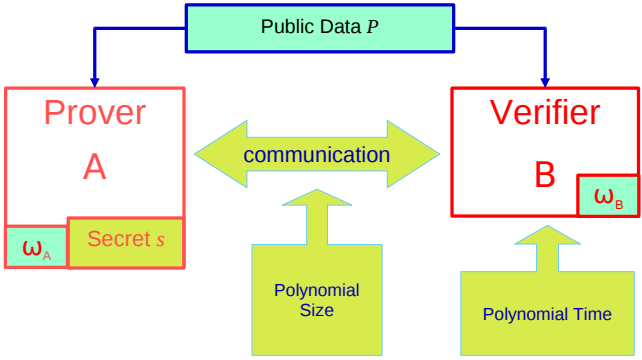
- If one wants $\text{Succ}_{\mathcal{F}DH}^{\text{euf-cma}}(t) \leq \varepsilon$ with $t/\varepsilon \approx 2^{80}$
- If one allows q_S up to 2^{30}

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

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

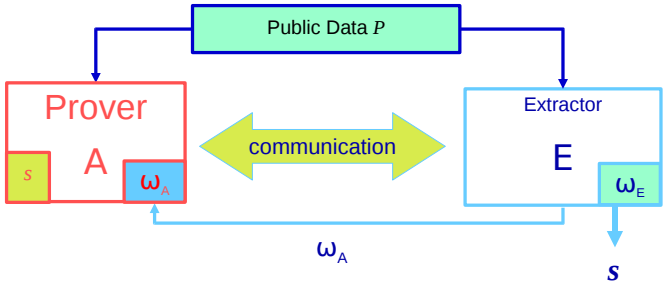
- 1 Basic Security Notions
- 2 Advanced Security for Signature
- 3 **Forking Lemma**
 - Zero-Knowledge Proofs
 - The Forking Lemma
- 4 Conclusion

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



Proof of Knowledge: Soundness

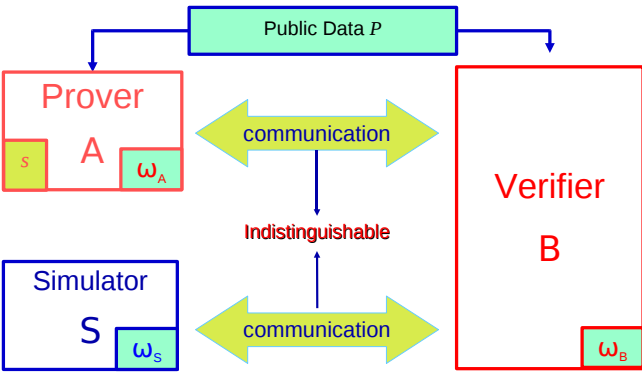
If I can be accepted, I really know a solution: **extractor**



Proof of Knowledge: Zero-Knowledge

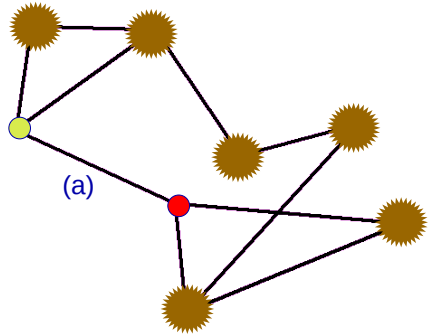
How do I prove that I know a solution s to a problem P ?
I reveal the solution...

How can do it without revealing any information?
Zero-knowledge: **simulator**



Proof of Knowledge

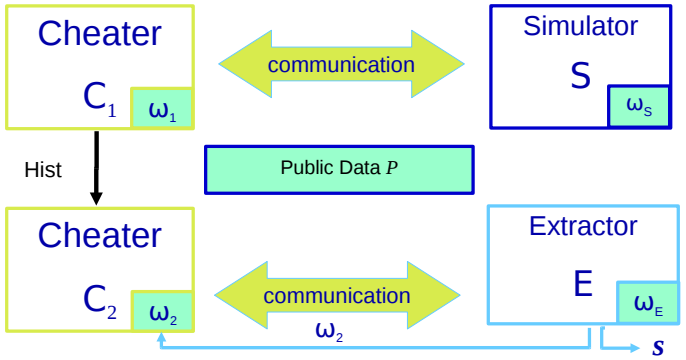
How do I prove that I know a 3-color covering, without revealing any information?



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

Secure Multiple Proofs of Knowledge: Authentication

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



Schnorr Proofs

[Schnorr – Eurocrypt '89 - Crypto '89]

Generic Zero-Knowledge Proofs

Zero-Knowledge Proof

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

Signature

- $(\mathbb{G} = \langle g \rangle)$ 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 $m \rightarrow (r, h, s)$
 $K \xleftarrow{R} \mathbb{Z}_q^*$ $r = g^K$
 $h = \mathcal{H}(m, r)$ and
 $s = K + xh \pmod q$
- Verification of (m, r, s)
compute $h = \mathcal{H}(m, r)$
and check $r \stackrel{?}{=} g^s y^h$

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
 $h \xleftarrow{R} \{0, 1\}^k$ 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

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 $\Pr[(x, y) \in A] \geq \epsilon$. For any $\alpha < \epsilon$, define

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

- (i) $\Pr[B_\alpha] \geq \alpha$
- (ii) $\forall (x, y) \in B_\alpha, \Pr_{y' \in Y} [(x, y') \in A] \geq \epsilon - \alpha$.
- (iii) $\Pr[B_\alpha | A] \geq \alpha/\epsilon$.

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,

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

(ii) straightforward.

(iii) using Bayes’ law:

$$\begin{aligned} \Pr[B | A] &= 1 - \Pr[\bar{B} | A] \\ &= 1 - \Pr[A | \bar{B}] \cdot \Pr[\bar{B}] / \Pr[A] \geq 1 - (\epsilon - \alpha) / \epsilon = \alpha / \epsilon. \end{aligned}$$

Theorem (The Forking Lemma)

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

Let \mathcal{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 , \mathcal{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 $h \neq h'$.

- \mathcal{A} is a PPTM with random tape ω .
 - During the attack, \mathcal{A} asks a polynomial number of queries to \mathcal{H} .
 - We may assume that these questions are distinct:
 - Q_1, \dots, Q_{q_H} are the q_H distinct questions
 - and let $H = (h_1, \dots, h_{q_H})$ be the list of the q_H answers of \mathcal{H} .
- Note: a random choice of \mathcal{H} = a random choice of H .
- For a random choice of (ω, \mathcal{H}) , with probability ε , \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 $Ind_{\mathcal{H}}(\omega)$ to be the index of this question:
 $(m, r) = Q_{Ind_{\mathcal{H}}(\omega)}$ ($Ind_{\mathcal{H}}(\omega) = \infty$ if the question is never asked).

Forking Lemma – Proof

We then define the sets

$$\begin{aligned} \mathcal{S} &= \{(\omega, \mathcal{H}) \mid \mathcal{A}^{\mathcal{H}}(\omega) \text{ succeeds} \ \& \ Ind_{\mathcal{H}}(\omega) \neq \infty\}, \\ \mathcal{S}_i &= \{(\omega, \mathcal{H}) \mid \mathcal{A}^{\mathcal{H}}(\omega) \text{ succeeds} \ \& \ Ind_{\mathcal{H}}(\omega) = i\} \quad i \in \{1, \dots, q_H\}. \end{aligned}$$

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

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

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 ,

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

Lemma

$$\Pr[Ind_{\mathcal{H}}(\omega) \in I \mid \mathcal{S}] \geq \frac{1}{2}.$$

By definition of \mathcal{S}_i ,

$$\Pr[Ind_{\mathcal{H}}(\omega) \in I \mid \mathcal{S}] = \sum_{i \in I} \Pr[\mathcal{S}_i \mid \mathcal{S}] = 1 - \sum_{i \notin I} \Pr[\mathcal{S}_i \mid \mathcal{S}].$$

Since the complement of I contains fewer than q_H elements,

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

Forking Lemma – Proof

- We run $2/\varepsilon$ times \mathcal{A} , with independent random ω 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 (ω, \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 $\Pr[\mathcal{S}_i] \geq \nu/2q_H$, there exists a subset Ω_i such that,

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

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

Forking Lemma – Proof

Since all the subsets \mathcal{S}_i are disjoint,

$$\begin{aligned} & \Pr_{\omega, \mathcal{H}}[(\exists i \in I) (\omega, \mathcal{H}) \in \Omega_i \cap \mathcal{S}_i \mid \mathcal{S}] \\ &= \Pr \left[\bigcup_{i \in I} (\Omega_i \cap \mathcal{S}_i) \mid \mathcal{S} \right] = \sum_{i \in I} \Pr[\Omega_i \cap \mathcal{S}_i \mid \mathcal{S}] \\ &= \sum_{i \in I} \Pr[\Omega_i \mid \mathcal{S}_i] \cdot \Pr[\mathcal{S}_i \mid \mathcal{S}] \geq \left(\sum_{i \in I} \Pr[\mathcal{S}_i \mid \mathcal{S}] \right) / 2 \geq \frac{1}{4}. \end{aligned}$$

We let β denote the index $Ind_{\mathcal{H}}(\omega)$ of to the successful pair. With prob. at least $1/4$, $\beta \in I$ and $(\omega, \mathcal{H}) \in \mathcal{S}_\beta \cap \Omega_\beta$. With prob. greater than $4/5 \times 1/4 = 1/5$, the $2/\varepsilon$ attacks provided a successful pair (ω, \mathcal{H}) , with $\beta = Ind_{\mathcal{H}}(\omega) \in I$ and $(\omega, \mathcal{H}) \in \mathcal{S}_\beta$.

Forking Lemma – Proof

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

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

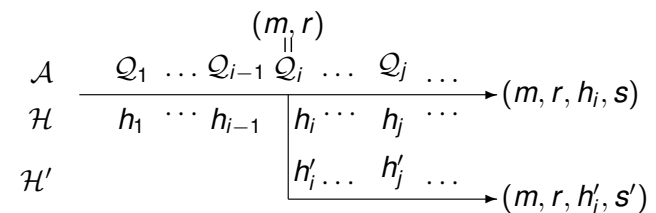
where $h_\beta = \mathcal{H}(Q_\beta)$ and $h'_\beta = \mathcal{H}'(Q_\beta)$.

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

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

Forking Lemma – Proof



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}' :

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

In order to answer signing queries, one simply uses the simulator of the zero-knowledge proof: (r, h, s) , and we set $\mathcal{H}(m, r) \leftarrow h$. The random oracle programming may fail, but with negligible probability.

- 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