The RSA cryptosystem

Part 3: RSA signatures: attacks and security proofs

Jean-Sébastien Coron

University of Luxembourg

Textbook RSA signature scheme

- Key generation
  - Public modulus: $N = p \cdot q$ where p and q are large primes.
  - Public exponent: e
  - Private exponent: d, such that $d \cdot e = 1 \pmod{\phi(N)}$
- To sign a message m, the signer computes :
  - $s = m^d \pmod{N}$
  - Only the signer can sign the message.
- To verify the signature, one checks that:
  - $m = s^e \pmod{N}$
  - Anybody can verify the signature

Attacks against textbook RSA signature

- Existential forgery
  - $r^e = m \pmod N$
  - r is a valid signature of m, so we can construct a valid message/signature pair without knowing the private key.
- Chosen message attack
  - $(m_1 \cdot m_2)^d = m_1^d \cdot m_2^d \pmod N$
  - Given two signatures, we can construct a 3rd signature without knowing the private key.
- Countermeasure
  - First encapsulate m using an encoding function $\mu(m)$

$$\sigma = \mu(m)^d \pmod N$$

Encoding functions

- Two kinds of encoding functions $\mu(m)$
  - Ad-hoc encodings
    - PKCS#1 v1.5, ISO 9796-1, ISO 9796-2.
    - Designed to prevent specific attacks, but can exhibit some weaknesses
  - Provably secure encodings
    - RSA-FDH, RSA-PSS
    - Proven to be secure under well-defined assumptions.

Ad-hoc encoding functions

- Examples of ad-hoc encoding functions, with signature $\sigma = \mu(m)^d \pmod N$
  - ISO 9796-1:
$$\mu(m) = \bar{s}(m_z)s(m_{z-1})m_z m_{z-1} \ldots s(m_1)s(m_0)m_0 6$$
  - ISO 9796-2:
$$\mu(m) = \mathtt{6A} \| m[1] \| H(m) \| \mathtt{BC}$$
  - PKCS#1 v1.5:
$$\mu(m) = \mathtt{0001\ FF\ldots .FF00} || c_{\mathsf{SHA}} || \mathsf{SHA}(m)$$

The Desmedt-Odlyzko attack [DO85]

Suppose the encoded messages $\mu(m)$ are relatively short.

1. Let $p_1, \ldots, p_\ell$ be the primes smaller than some bound B.
2. Find $\ell + 1$ messages m_i such that the $\mu(m_i)$ are B-smooth:
$$\mu(m_i) = p_1^{v_{i,1}} \cdots p_\ell^{v_{i,\ell}}$$
3. Obtain a linear dependence relation between the exponent vectors $\vec{V}_i = (v_{i,1} \bmod e, \ldots, v_{i,\ell} \bmod e)$ and deduce
$$\mu(m_\tau) = \prod_i \mu(m_i)$$
4. Ask for the signatures of the m_i's and forge the signature of m_τ.
$$\mu(m_\tau)^d = \prod_i \mu(m_i)^d \pmod{N}$$

The Desmedt-Odlyzko attack (1)

- Assume that $\mu(m_i)$ is B-smooth for all $1 \le i \le \tau$:

$$\mu(m_i) = \prod_{j=1}^{\ell} p_j^{v_{i,j}}$$

- To each $\mu(m_i)$ associate the vector exponents modulo e:

$$\vec{V}_i = (v_{i,1} \bmod e, \ldots, v_{i,\ell} \bmod e) \in \mathbb{Z}^{\ell}$$

- Assuming that e is prime, the set of all ℓ-dimensional vectors modulo e forms a linear space of dimension ℓ
  - If $\tau \ge \ell + 1$, one can express one vector, say $\vec{V}_\tau$, as a linear combination of the others modulo e,
    using Gaussian elimination:

$$\vec{V}_\tau = \vec{\Gamma} \cdot e + \sum_{i=1}^{\tau-1} \beta_i \vec{V}_i$$

The Desmedt-Odlyzko attack (2)

- We write the linear relation on the exponents:

$$v_{\tau,j} = \gamma_j \cdot e + \sum_{i=1}^{\tau-1} \beta_i \cdot v_{i,j}$$

- Multiplicative relation on the $\mu(m_i)$:

$$\begin{aligned}
\mu(m_\tau) &= \prod_{j=1}^{\ell} p_j^{v_{\tau,j}} = \prod_{j=1}^{\ell} p_j^{\gamma_j \cdot e + \sum_{i=1}^{\tau-1} \beta_i \cdot v_{i,j}} = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{j=1}^{\ell} \prod_{i=1}^{\tau-1} p_j^{v_{i,j} \cdot \beta_i} \\
&= \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{i=1}^{\tau-1} \left(\prod_{j=1}^{\ell} p_j^{v_{i,j}} \right)^{\beta_i} \\
&= \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i}
\end{aligned}$$

The Desmedt-Odlyzko attack (3)

- Multiplicative relation on the $\mu(m_i)$

$$\mu(m_\tau) = \delta^e \cdot \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i}, \quad \text{where } \delta := \prod_{j=1}^{\ell} p_j^{\gamma_j}$$

- Signature forgery
  - The attacker asks the signatures σ_i of $m_1, \ldots, m_{\tau-1}$ and forges the signature σ_τ of m_τ:

$$\sigma_\tau = \mu(m_\tau)^d = \delta \cdot \prod_{i=1}^{\tau-1} \left(\mu(m_i)^d\right)^{\beta_i} \pmod N$$

$$\sigma_\tau = \delta \cdot \prod_{i=1}^{\tau-1} \sigma_i^{\beta_i} \pmod N$$

Smoothness probability

Theorem (CEP83)

Let x be an integer and let $L_x[\beta] = \exp\left(\beta \cdot \sqrt{\log x \log \log x}\right)$. Let t be an integer randomly distributed between zero and x. Then for large x, the probability that all the prime factors of t are less than $L_x[\beta]$ is given by $L_x\left[-1/(2\beta) + o(1)\right]$.

- Smoothness probability
  - Let x be a bound on $\mu(m)$ and let $\ell = L_x[\beta]$ be the number of primes, for some parameter β.
  - The smoothness probability is $L_x\left[-1/(2\beta) + o(1)\right]$

Asymptotic complexity of Desmedt-Odlyzko attack

- Asymptotic complexity analysis
  - The smoothness probability is $L_x\left[-1/(2\beta) + o(1)\right]$.
  - $\Rightarrow$ it takes $L_x\left[1/(2\beta) + o(1)\right]$ time to find a smooth $\mu(m_i)$
  - We need $\ell + 1$ smooth $\mu(m_i)$, therefore:

$$T = L_x\left[1/(2\beta) + \circ(1)\right] \cdot L_x[\beta] = L_x\left[1/(2\beta) + \beta + \circ(1)\right]$$

  - The complexity is minimal for $\beta = \sqrt{2}/2$.
  - Asymptotic complexity: $L_x\left[\sqrt{2} + \circ(1)\right]$
- The complexity is sub-exponential in the size of $\mu(m)$
  - The attack is only practical for relatively small $\mu(m)$ (less than 160 bits).

Application of Desmedt-Odlyzko attack

- Cryptanalysis of ISO 9796-1 and ISO 9796-2 signatures [CNS99]
  - Extension of Desmedt-Odlyko attack
  - Following this attack ISO 9796-1 was withdrawn
  - ISO 9796-2 was amended by increasing the message digest to at least 160 bits.
- Cryptanalysis of ISO 9796-2 [CNTW09]
  - Improved detection of smooth numbers using Bernstein's algorithm.
  - Works against the amended ISO 9796-2.
  - Following this attack ISO 9796-2 was amended again in late 2010.

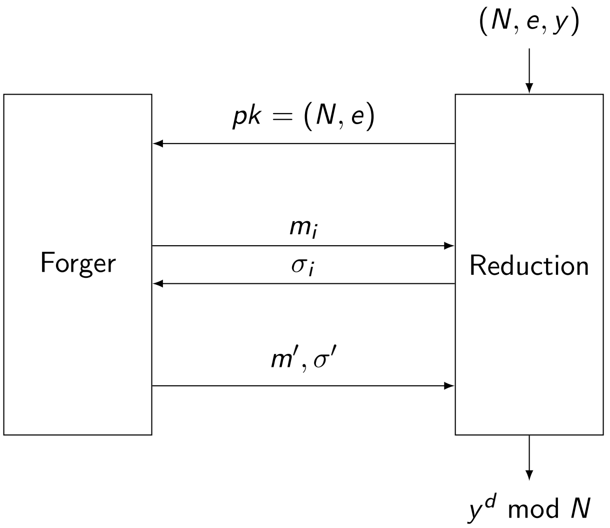
Security proofs in cryptography

- Since the invention of public-key cryptography
  - Many schemes have been proposed...
  - And many of them have been broken.
- How can we justify security rigorously ?
  - Prove that if an adversary can break the scheme, he can solve a hard problem such as:
    - Factoring large integers.
    - RSA problem: given y, compute $y^d \bmod N$.
  - This shows that the scheme is secure, assuming that the underlying problem is hard to solve.
- To be rigorous, one must first specify what it means to break a scheme.
  - Security definition

Provable security for signatures

- Strongest security notion for signatures (Goldwasser, Micali and Rivest, 1988):
  - It must be infeasible for an adversary to forge the signature of a message, even if he can obtain the signature of messages of his choice.
- Security proof:
  - Show that from an adversary who is able to forge signature, you can solve a difficult problem, such as inverting RSA.
- Examples of provably secure signature schemes for RSA:
  - Full Domain Hash (FDH)
  - Probabilistic Signature Scheme (PSS)

Security model



The FDH scheme

- The FDH signature scheme:
  - was designed in 1993 by Bellare and Rogaway.

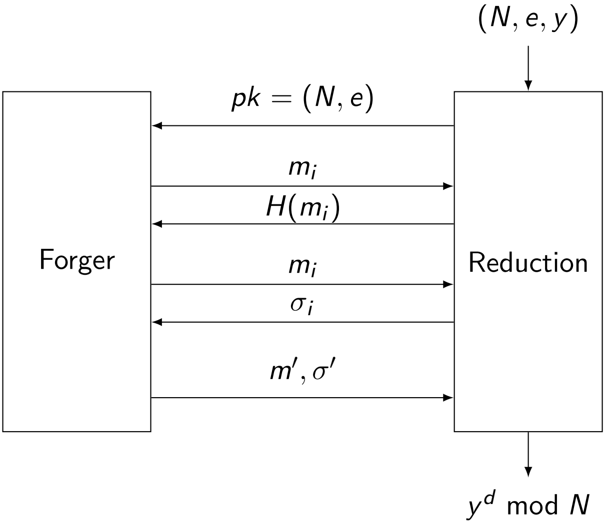
$$m \longrightarrow H(m) \longrightarrow s = H(m)^d \bmod N$$

  - The hash function $H(m)$ has the same output size as the modulus.
- Security of FDH
  - FDH is provably secure in the random oracle model, assuming that inverting RSA is hard.
  - In the random oracle model, the hash function is replaced by an oracle which outputs a random value
    for each new query.

Security proof for FDH

- Proof in the random oracle model
  - The adversary cannot compute the hash-function by himself.
  - He must make a request to the random oracle, which answers a random, independantly distributed answer for each new query.
    - Randomly distributed in $\mathbb{Z}_N$.
- Idealized model of computation
  - A proof in the random oracle model does not imply that the scheme is secure when a concrete hash-function like SHA-1 is used.
  - Still a good guarantee.

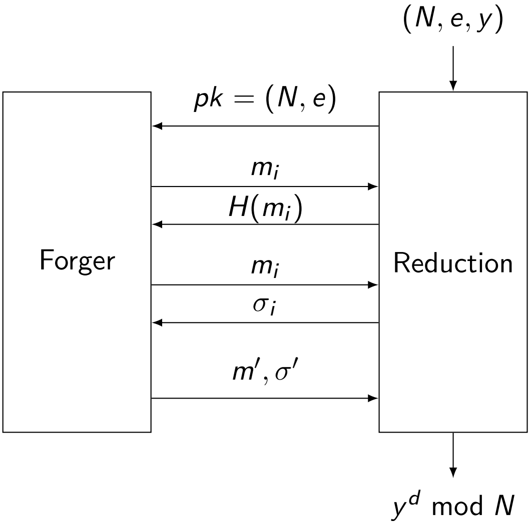
Security model with hash queries



Proof of security

- We assume that there exists a successful adversary.
  - This adversary is a forger algorithm that given the public-key (N, e), after at most q_{hash} hash queries and q_{sig} signature queries, outputs a forgery (m', s').
- We will use this adversary to solve a RSA challenge: given (N, e, y), output $y^d \bmod N$.
  - The adversary's forgery will be used to compute $y^d \bmod N$, without knowing d.
  - If solving such RSA challenge is assumed to be hard, then producing a forgery must be hard.

Security proof for FDH



$j \leftarrow [1, q_{hash} + q_{sig} + 1]$

$i \neq j$: $H(m_i) = r_i^e \bmod N$

$i = j$: $H(m_j) = y \bmod N$

$i \neq j$: $\sigma_i = r_i \bmod N$

$i = j$: $\perp$

$m' = m_j$: $\sigma' = y^d \bmod N$

Security proof for FDH

- Let q_{hash} be the number of hash queries and q_{sig} be the number of signature queries.
  - Select a random $j \in [1, q_{hash} + q_{sig} + 1]$.
- Answering a hash query for the i-th message m_i:
  - If $i \neq j$, answer $H(m_i) = r_i^e \bmod N$ for random r_i.
  - If $i = j$, answer $H(m_j) = y$ where y is the challenge.
- Answering a signature query for m_i:
  - If $i \neq j$, answer $\sigma_i = H(m_i)^d = r_i \bmod N$, otherwise (if $i = j$) abort.
  - We can answer all signature queries, except for message m_j

Using the forgery

- Let (m', σ') be the forgery
  - We assume that the adversary has already made a hash query for m', *i.e.* , $m' = m_i$ for some i.
    - Otherwise we can simulate this query.
  - Then if $i = j$, then $\sigma' = H(m_j)^d = y^d \bmod N$.
  - We return σ' as the solution to the RSA challenge (N, e, y).
- Our reduction succeeds if $i = j$:
  - Since j was selected at random in $[1, q_{hash} + q_{sig} + 1]$
  - this happens with probability $1/(q_{hash} + q_{sig} + 1)$

Success probability

- From a forger that breaks FDH with probability ε in time t, we can invert RSA with probability $\varepsilon' = \varepsilon/(q_{hash} + q_{sig} + 1)$ in time t' close to t.
- Conversely, if we assume that it is impossible to invert RSA with probability greater than ε' in time t', it is impossible to break FDH with probability greater than

  $$\varepsilon = (q_{hash} + q_{sig} + 1) \cdot \varepsilon'$$

  in time t close to t'.
- This gives us a security guarantee the FDH signature scheme is secure if inverting RSA is hard.

Improving the security bound [C00]

- Instead of letting $H(m_i) = r_i^e \bmod N$ for all $i \neq j$ and $H(m_j) = y$, one lets
  - $H(m_i) = r_i^e \bmod N$ with probability α
  - $H(m_i) = r_i^e \cdot y \bmod N$ with probabiliy $1 - \alpha$
- 2 kinds of messages m_i:
  - When $H(m_i) = r_i^e \bmod N$ one can answer the signature query but not use a forgery for m_i.
    - $\sigma_i = H(m_i)^d = r_i \bmod N$.
  - When $H(m_i) = r_i^e \cdot y \bmod N$ one cannot answer the signature query but we can use a forgery to compute $y^d \bmod N$.
    - If $H(m_i) = y \cdot r_i^e \bmod N$, then $\sigma_i = H(m_i)^d = y^d \cdot r_i \bmod N$ and return $y^d = \sigma_i / r_i \bmod N$.
  - Optimize for α.

Improving the security bound of FDH

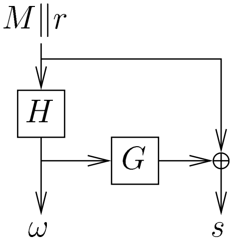
- Probability that all signature queries are answered:
  - A signature query is answered with probability α
  - At most q_{sig} signature queries $\Rightarrow P \geq \alpha^{q_{sig}}$
- Probability that the forgery (m_i, σ') is useful :
  - Useful if $H(m_i) = r_i^e \cdot y \bmod N$, so with probability $1 - \alpha$
- Global success probability :
  - $f(\alpha) = \alpha^{q_{sig}} \cdot (1 - \alpha)$
  - $f(\alpha)$ is maximum for $\alpha_m = 1 - 1/(q_{sig} + 1)$
  - $f(\alpha_m) \simeq 1/(\exp(1) \cdot q_{sig})$ for large q_{sig}

Improved security bound for FDH

- From a forger that breaks FDH with probability ε in time t, we can invert RSA with probability $\varepsilon' = \varepsilon/(4 \cdot q_{sig})$ in time t' close to t.
- Conversely, if we assume that it is impossible to invert RSA with probability greater than ε' in time t', it is impossible to break FDH with probability greater than $\varepsilon = 4 \cdot q_{sig} \cdot \varepsilon'$ in time t close to t'.
- Concrete values
    - With $q_{hash} = 2^{60}$ and $q_{sig} = 2^{30}$, we obtain $\varepsilon = 2^{32}\varepsilon'$ instead of $\varepsilon = 2^{60} \cdot \varepsilon'$ $\Rightarrow$ more secure for a given modulus size k.
    - A smaller RSA modulus can be used for the same level of security: improved efficiency.

The PSS signature scheme

- PSS (Bellare and Rogaway, Eurocrypt'96)
  - IEEE P1363a and PKCS#1 v2.1.
  - 2 variants: PSS and PSS-R (message recovery)
  - Provably secure against chosen-message attacks, in the random oracle model.
  - PSS-R: $\mu(M, r) = \omega \| s$, $\sigma = \mu(M, r)^d \bmod N$



- Tight security proof
  - $\varepsilon' \simeq \varepsilon$, so no security loss.

Implementation attacks

- The implementation of a cryptographic algorithm can reveal more information
- Passive attacks :
  - Timing attacks (Kocher, 1996): measure the execution time
  - Power attacks (Kocher et al., 1999): measure the power consumption
- Active attacks :
  - Fault attacks [BDL97]: induce a fault during computation
  - Invasive attacks: probing.

Fault attack against RSA-CRT

- Induce a fault during computation
  - By modifying the input voltage
- RSA with CRT: to compute $s = m^d \bmod N$, compute :
  - $s_p = m^{d_p} \pmod{p}$ where $d_p = d \pmod{p-1}$
  - $s_q = m^{d_q} \pmod{q}$ where $d_q = d \pmod{q-1}$
  - and recombine s_p and s_q using CRT to get $s = m^d \pmod{N}$
- Fault attack against RSA with CRT [BDL97]
  - If s_p is incorrect, then $s^e \neq m \pmod{N}$ while $s^e = m \pmod{q}$
  - Therefore, $\gcd(N, s^e - m \bmod N)$ gives the prime factor q.