Cryptography

Course no. 10

Jean-Sébastien Coron

University of Luxembourg

May 9, 2014

Summary

- Introduction to security proofs
  - The RSA signature scheme
  - Attacks against RSA signatures
  - Security proofs for signature schemes
- The Full Domain Hash scheme
  - Original and improved security proof
- The PSS scheme
  - Original security proof
  - Improved security proof and practical consequences

- What is cryptography ?
    - Cryptography's aim is to contruct schemes that achieve some goal despite the presence of an adversary.
    - Example: encryption, key-exchange, signature, electronic voting...
- Scientific approach:
    - To be rigorous, one must specify what it means to be secure.
    - Then one tries to construct schemes that achieve the desired goal, in a provable way.
    - Plain RSA encryption and signature cannot be used !

The 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 $d \cdot e = 1 \mod \phi(N)$
- To sign a message m, the signer computes :
  - $s = m^d \mod N$
  - Only the signer can sign the message.
- To verify the signature, one checks that:
  - $m = s^e \mod N$
  - Anybody can verify the signature

Hash-and-sign paradigm

- There are many attacks on basic RSA signatures:
  - Existential forgery: $r^e = m \mod N$
  - Chosen-message attack: $(m_1 \cdot m_2)^d = m_1^d \cdot m_2^d \mod N$
- To prevent from these attacks, one usually uses a hash function. The message is first hashed, then padded.
  - $m \longrightarrow H(m) \longrightarrow \texttt{1001...0101} \| H(m)$
  - Example: PKCS#1 v1.5:
    $\mu(m) = \texttt{0001 FF....FF00} \| c_{\text{SHA}} \| \text{SHA}(m)$
  - ISO 9796-2: $\mu(m) = \texttt{6A} \| m[1] \| H(m) \| \texttt{BC}$

Security proofs

- Since the invention of public-key cryptography
  - Many schemes have been proposed...
  - And many of them have been broken.
  - Until recently, a scheme was considered as secure if no one was able to break it.
- 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 $y^d \mod N$.
  - This shows that the scheme is secure, assuming that the underlying problem is hard to solve.

Proofs for signature schemes

- Strongest security notion (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:
  - Full Domain Hash (FDH)
  - Probabilistic Signature Scheme (PSS)

The FDH scheme

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

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

  - 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.

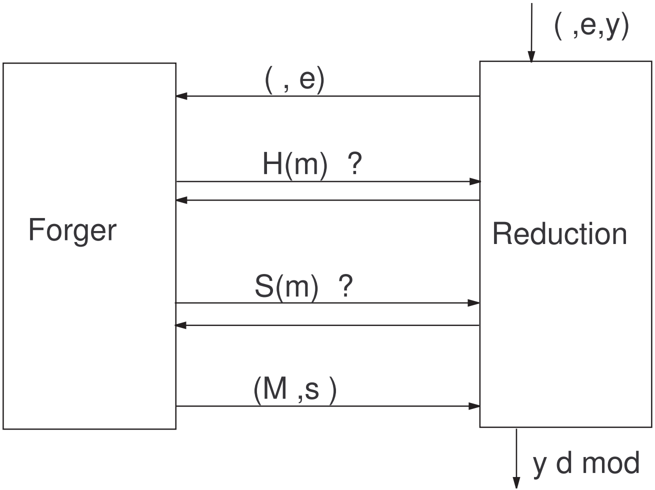
Security proof for FDH

- We want to show that FDH is a secure signature scheme:
  - Even if the adversary requests signatures of messages of his choice, he is still unable to produce a forgery.
  - Forgery: a couple (m', s') such that s is a valid signature of m but the signature of m was never requested by the adversary.

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 proof



Proof of security

- We assume that there exists a succesfull adversary.
  - This adversary is an 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 $y^d \mod N$.
  - The adversary's forgery will be used to compute $y^d \mod N$, without knowing d.
  - If solving such RSA challenge is assumed to be hard, then producing a forgery must be hard.

Security proof for FDH

- 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 $H(m_i) = r_i^e \mod N$ for random r_i.
  - If $i = j$, answer $H(m_j) = y$.
- Answering a signature query for m_i:
  - If $i \neq j$, answer $r_i = H(m_i)^d \mod N$, otherwise ($i = j$) abort.
  - We can answer all signature queries, except for message m_j

Using the forgery

- Let (m', s') 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 $s' = H(m_j)^d = y^d \mod N$.
  - We return s' as the solution to the RSA challenge (N, e, y).

Success probability

- Our reduction succeeds if $i = j$
  - This happens with probability $1/(q_{hash} + q_{sig} + 1)$
- 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'.

Improving the security bound

- Instead of letting $H(m_i) = r_i^e \mod N$ for all $i \neq j$ and $H(m_j) = y$, one lets
  - $H(m_i) = r_i^e \mod N$ with probability α
  - $H(m_i) = r_i^e \cdot y \mod N$ with probabiliy $1 - \alpha$
- Idea (published at CRYPTO 2000 by me).
  - When $H(m_i) = r_i^e \mod N$ one can answer the signature query but not use a forgery for m_i.
  - When $H(m_i) = r_i^e \cdot y \mod N$ one cannot answer the signature query but can use the forgery to compute $y^d \mod N$.
  - Optimize for α.

Improved security proof

- Answering a hash query for the i-th message m_i:
  - With probability α, answer $H(m_i) = r_i^e \mod N$ for a random r_i. Otherwise answer $H(m_i) = y \cdot r_i^e \mod N$.
- 2 kinds of messages:
  - Messages for which we know the signature, but the forgery can not be used.
  - Messages for which we can use the forgery, but we can not answer the signature query.
- Answering a signature query for m_i:
  - $H(m_i) = r_i^e \mod N$, answer r_i, otherwise abort.
- Using the forgery (m_i, s_i):
  - If $H(m_i) = y \cdot r_i^e \mod N$, then $s_i = H(m_i)^d = y^d \cdot r_i \mod N$ and return s_i / r_i.

Improving the bound

- 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, s') is useful :
  - Useful if $H(m_i) = r_i^e \cdot y \mod N$
    - $s' = H(m_i)^d = r_i \cdot y^d \mod N \Rightarrow y^d = s'/r_i \mod N$
- Global success probability :
  - $f(\alpha) = \alpha^{q_{sig}} \cdot (1 - \alpha)$
  - $f(\alpha)$ is maximum for $\alpha_m = 1 - 1/(q + 1)$
  - $f(\alpha_m) \simeq 1/(e \cdot q_{sig})$ for large q_{sig}

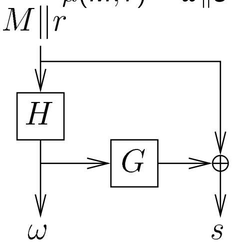
Success probability

- 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 $\varepsilon = 2^{60} \cdot \varepsilon'$
  - More secure for a given modulus size k.
  - A smaller modulus can be used for the same level of security: improved efficiency.

The PSS signature cheme

- 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
  - PSS-R:

$$\mu(M, r) = \omega \| s$$



- Parameters
  - k_0 is the size of r.
  - k_1 is the size of ω.

Original security proof

- If is impossible to invert RSA with probability greater than ε' in time t', it is impossible to break PSS in time $t \simeq t'$ with probability greater than

$$\varepsilon = \varepsilon' + 3 \cdot (q_{sig} + q_{hash})^2 \cdot \left(2^{-k_0} + 2^{-k_1}\right)$$

- Tight security proof ($\varepsilon' \simeq \varepsilon$), provided that $k_0 \geq k_{min}$ and $k_1 \geq k_{min}$, with:

$$k_{min} = 2 \cdot \log_2(q_{hash} + q_{sig}) + \log_2 \frac{1}{\varepsilon'}$$

- With $q_{hash} = 2^{60}$, $q_{sig} = 2^{30}$ and $\varepsilon' = 2^{-60}$, k_0 and k_1 must be greater than $k_{min} = 180$ bits.
- The value of k_1 is optimal.

Improved security proof

- If is impossible to invert RSA with probability greater than ε' in time t', it is impossible to break PSS in time $t \simeq t'$ with probability greater than

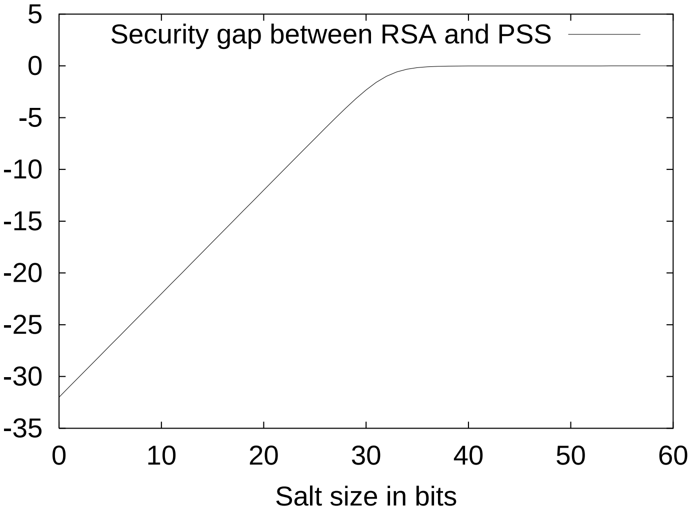
$$\varepsilon = \varepsilon' \cdot \left(1 + 6 \cdot q_{sig} \cdot 2^{-k_0}\right) + 2 \cdot \left(q_{hash} + q_{sig}\right)^2 \cdot 2^{-k_1}$$

- Tight security proof ($\varepsilon' \simeq \varepsilon$), provided that $k_1 \geq k_{min}$ and

$$k_0 \geq \log_2 q_{sig}$$

- With $q_{sig} = 2^{30}$, we can take $k_0 = 30$ bits and using a larger seed does not further improve security.
- When PSS is used with message recovery, with a 1024-bits RSA modulus, 813 bits of message can now be recovered when verifying the signature, instead of 663 bits.

Security as a function of k_0



A variant of PSS: PFDH

- Probabilistic Full-Domain Hash (PFDH)
  - Similar to FDH except that a random seed r of k_0 bits is concatenated to M before hashing it.

$$m \longrightarrow m \| r \longrightarrow H(m \| r) \longrightarrow s = H(m \| r)^d \mod N$$

  - The signature of m is (s, r).
- Security proof very similar to PSS.
  - If is impossible to invert RSA with probability greater than ε' in time t', it is impossible to break PFDH in time $t \simeq t'$ with probability greater than

$$\varepsilon = \varepsilon' \cdot \left(1 + 6 \cdot q_{sig} \cdot 2^{-k_0}\right)$$

Proof for PFDH: old technique

- Answering a hash query for $m\|r$:
  - Answer $H(m\|r) = y \cdot x^e \mod N$ for a random x.
- Answering a signature query for m:
  - Generate a random r of k_0 bits.
  - If r never appeared before, set $H(m\|r) = x^e$ and return x.
  - Otherwise abort.
- Using the forgery (m, s, r):
  - We have $s = H(m\|r)^d = y^d \cdot x \mod N$, so return s/x.
- Success probability:
  - A signature query fails with probability lesser than $(q_{hash} + q_{sig}) \cdot 2^{-k_0}$.

$$\varepsilon = \varepsilon' + q_{sig} \cdot (q_{sig} + q_{hash}) \cdot 2^{-k_0}$$

Proof for PFDH: new technique

- For each new message m_i, we generate a list L_i of q_{sig} random integers of k_0 bits.
- Answering a hash query for $m_i \| r$:
  - If r belongs to L_i, answer $H(m_i \| r) = x^e \mod N$ for a random x.
  - Otherwise answer $H(m_i \| r) = y \cdot x^e \mod N$.
- Answering a signature query for m_i:
  - Take the next random r in the list L_i.
  - Then $H(m_i \| r) = x^e \mod N$ and return x.
- Using the forgery (m_i, s, r):
  - If r does not belong to L_i, then $H(m_i \| r) = y \cdot x^e \mod N$.
  - Then $s = H(m_i \| r)^d = y^d \cdot x \mod N$, so return s/x.

Proof for PFDH: new technique

- Success probability:
  - The reduction answers all the signature queries.
  - The probability that r does not belong to L_i is $(1-2^{-k_0})^{q_{sig}}$
  - If $k_0 \geq \log_2 q_{sig}$, this is greater than $1/4$.

$$\varepsilon = 4 \cdot \varepsilon'$$

- General case:
  - We generate some lists L_i with less than q_{sig} integers.
  - We can fail answering signature queries but we can use the forgery with better probability.

$$\varepsilon = \varepsilon' \cdot \left(1 + 6 \cdot q_{sig} \cdot 2^{-k_0}\right)$$

Conclusion

- What is cryptography ?
  - Cryptography's aim is to contruct schemes that achieve some goal despite the presence of an adversary.
- Scientific approach:
  - To be rigorous, one must specify what it means to be secure.
  - Then one tries to construct schemes that achieve the desired goal, in a provable way.
  - Plain RSA encryption and signature cannot be used !