Digital Signature Schemes and the Random Oracle Model

A. Hülsing
Today‘s goal

Review provable security of “in use” signature schemes. (PKCS #1 v2.x)
Digital Signature

Original text → signing → signed text → verifying → verified text

private key

public key

Source: http://hari-cio-8a.blog.ugm.ac.id/files/2013/03/DCA.jpg
Definition: Digital Signature (formally)

Let $\mathcal{M}$ be the message space. A digital signature scheme $\text{DSig} = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is a triple of PPT algorithms

- **KeyGen($1^k$)**: upon input of a security parameter $1^k$ outputs a private signing key $sk$ and a public verification key $pk$,
- **Sign($sk, M$)**: outputs a signature $\sigma$ under $sk$ for message $M$, if $M \in \mathcal{M}$,
- **Verify($pk, M, \sigma$)**: outputs 1 iff $\sigma$ is a valid signature on $M$ under $pk$,

such that the following correctness condition is fulfilled:

\[
\forall (pk, sk) \leftarrow \text{KeyGen}(1^k), \forall (M \in \mathcal{M}): \text{Verify}(pk, M, \text{Sign}(sk, M)) = 1.
\]
Consider adversary $A$

- **Full break (FB):** $A$ can compute the secret key.
- **Universal forgery (UU):** $A$ can forge a signature for any given message. $A$ can efficiently answer any signing query.
- **Selective forgery (SU):** $A$ can forge a signature for some message of its choice. In this case $A$ commits itself to a message before the attack starts.
- **Existential forgery (EU):** $A$ can forge a signature for one, arbitrary message. $A$ might output a forgery for any message for which it did not learn the signature from an oracle during the attack.
Attack Models

• **Key-only attack (KOA):** $A$ only gets the public key for which it has to forge a signature.

• **Random message attack (RMA):** $A$ learns the public key and the signatures on a set of random messages.

• **Adaptively chosen message attack (CMA):** $A$ learns the public key and is allowed to adaptively ask for the signatures on messages of its choice.
Existential unforgeability under adaptive chosen message attacks

\[ (\sigma^*, M^*) \]

Success if \( M^* \neq M_i, \forall i \in [0, q] \) and \( \text{Verify}(pk, \sigma^*, M^*) = \text{Accept} \)
Reduction

\[ M^A \]

Problem

- Transform Problem
- Implement SIGN

Solution

- Extract Solution

\[ \text{PK, } 1^n \]

\[ q_s \]

\[ M_i \]

\[ (\sigma_i, M_i) \]

\[ A \]

\[ (\sigma^*, M^*) \]
Why security reductions?

- Current RSA signature standards so far unbroken
- Vulnerabilities might exist! (And existed for previous proposals)
- Might be possible to forge RSA signatures without solving RSA problem or factoring!
What could possibly go wrong?
Let \((N, e, d) \leftarrow \text{GenRSA}(1^k)\) be a PPT algorithm that outputs a modulus \(N\) that is the product of two \(k\)-bit primes (except possibly with negligible probability), along with an integer \(e > 0\) with \(\gcd(e, \phi(N)) = 1\) and an integer \(d > 0\) satisfying \(ed = 1 \mod \phi(N)\).

For any \((N, e, d) \leftarrow \text{GenRSA}(1^k)\) and any \(y \in \mathbb{Z}_N^*\) we have
\[
(y^d)^e = y^{de} = y^{de \mod \phi(N)} = y^1 = y \mod N
\]
Definition 1. We say that the RSA problem is hard relative to \texttt{GenRSA} if for all PPT algorithms \texttt{A}, the following is negligible:

\[
Pr[(N, e, d) \leftarrow \text{GenRSA}(1^k); \ y \leftarrow \mathbb{Z}_N^*; \\
\ x \leftarrow A(N, e, y): x^e = y \text{ mod } N].
\]
A simple RSA Signature (a.k.a. textbook RSA)

KeyGen(1^k): Run \((N, e, d) \leftarrow \text{GenRSA}(1^k)\).

Return \((pk, sk)\) with \(pk = (N, e), sk = d\).

Sign\((sk, M)\): Return \(\sigma = (M^d \mod N)\)

Verify\((pk, M, \sigma)\): Return 1 iff \(\sigma^e \mod N = M\)
Some RSA properties

- Public function:
  \[ P(x) = x^e \mod N \]

- Secret function:
  \[ S(x) = x^d \mod N \]

- Reciprocal property (RP):
  \[ P \circ S = S \circ P = Id \]

- Multiplicative property (MP):
  \[ \forall x, y \in \mathbb{Z}_N: S(xy) = S(x)S(y) \]
Existential forgery under KOA

Given public key $pk = (N, e)$

- Choose random $\sigma \in \mathbb{Z}_N$.
- Apply public function:

$$P(\sigma) = \sigma^e \mod N = M$$

- Return signature-message pair $(\sigma, M)$. 
Universal forgery under CMA

Given public key $pk = (N, e)$
To create a forgery on a given message $M$:

1. Choose two messages $x, y \in \mathbb{Z}_N$ such that $xy = M \mod N$
2. Ask for signatures $\sigma_x$ of $x$ and $\sigma_y$ of $y$
3. Output forgery $(\sigma_x \sigma_y, M)$
Universal forgery under CMA: The Blinding Attack

Given public key $pk = (N, e)$
To create a forgery on any given message $M$:

1. Sample random $r \in \mathbb{Z}_N^*$
2. Ask for signature $\sigma$ on $r^e M \mod N$
3. Output forgery $\left(\frac{\sigma}{r} \mod N, M\right)$

Recall $\sigma = (r^e M)^d = r^{ed} M^d = rM^d \mod N$
Hence $\frac{\sigma}{r} = M^d \mod N$
A slightly better RSA Signature

Assume Hashfunction $H: \{0, 1\}^* \rightarrow \{0, 1\}^n$ for e.g. $n = 160$ (like with SHA1)

**KeyGen($1^k$):** Run $(N, e, d) \leftarrow \text{GenRSA}(1^k)$.

Return $(pk, sk)$ with $pk = (N, e), sk = d$.

**Sign($sk, M$):** Pad with suff. zeros that

$\mu(M) = (0 \ldots 0||H(M)) \in \mathbb{Z}_N^*$

Return $\sigma = \mu(M)^d \mod N$

**Verify($pk, M, \sigma$):** Return 1 iff

$\sigma^e \mod N = \mu(M) = (0 \ldots 0||H(M))$
**B-smooth**: An integer is B-smooth if all its prime factors are smaller than B.
Remember Index Calculus?

Given public key $\mathbf{pk} = (N, e)$

1. Select a bound $y$ and let $S = (p_1, \ldots, p_l)$ be the list of primes smaller than $y$.

2. Find at least $l + 1$ messages $M_i$ such that each $\mu(M_i) = (0 \ldots 0 || H(M_i))$ is a product of primes in $S$ (i.e. $y$–smooth).

3. Express one $\mu(M_j)$ as a multiplicative combination of the other $\mu(M_i)$ by solving a linear system given by the exponent vectors of the $\mu(M_i)$ with respect to the primes in $S$.

4. Ask for the signatures on all $M_i, i \neq j$ and forge signature on $M_j$. 

Step 3

Recall $\mu(M_i) = (0 \ldots 0||H(M_i))$

1. We can write $\forall M_i, 1 \leq i \leq \tau$: $\mu(M_i) = \prod_{j=1}^{l} p_j^{v_{i,j}}$

2. Associate with $\mu(M_i)$ length $l$ vector $V_i(v_{i,1} \mod e, \ldots, v_{i,l} \mod e)$

3. $\tau \geq l + 1$ and there are only $l$ linearly independent length $l$ vectors: We can express one vector as combination of others mod $e$. Let this be $V_\tau = \sum_{i=1}^{\tau-1} \beta_i V_i + e\Gamma$; for some $\Gamma = (\gamma_1, \ldots, \gamma_l)$

4. Hence,

$$\mu(M_\tau) = \left( \prod_{j=1}^{l} p_j^{\gamma_j} \right)^e \prod_{i=1}^{\tau-1} \mu(M_i)^{\beta_i}$$
Step 4

1. Ask for signatures $\sigma_i = \mu(M_i)^d \mod N$ on $M_i$ for $1 \leq i < \tau$

2. Compute:

$$\sigma^* = \mu(M_\tau)^d = \left( \prod_{j=1}^{l} p_j^{y_j} \right)^{\tau-1} \prod_{i=1}^{\tau-1} (\mu(M_i)^d)^{\beta_i} \mod N$$

3. Output forgery $(\sigma^*, M_\tau)$
Summing up

• Original attack (Misarsky, PKC’98) works even for more complicated paddings (ISO/IEC 9796-2)

• Attack only works for small $n$! (Complexity depends on $l$ and probability that an $n$-bit number is $y$-smooth).

• But using SHA-1 ($n = 160$) the attack takes much less than $2^{50}$ operations!

There are many ways to make mistakes...
(Similar attacks apply to encryption!)
That‘s why we want security reductions
The Random Oracle Model
Standard model vs. idealized model

Standard model:
Assume building block has property P (e.g., collision resistance). Use property in reduction.

Idealized model:
Assume a building block behaves perfectly (e.g. hash function behaves like truly random function). Replace building block by an oracle in reduction.
Problem

Implement SIGN

Transform Problem

Solution

Extract Solution

$M^A$

$\mathcal{M}^A$

$\Sigma_i, M_i$

$q_s$

$M_i$

$(\sigma_i, M_i)$

$\mathcal{A}$

$(\sigma^*, M^*)$

PK, $1^n$
Random Oracle Model (ROM)

- Idealized Model
- Perfectly Random Function
"Lazy Sampling":

- Keep list of \((x_i, y_i)\)
- Given \(M_j\), search for \(x_i = M_j\)
- If \(x_i = M_j\) exists, return \(y_i\)
- Else sample new \(y\) from Domain, using uniform distribution
- Add \((M_j, y)\) to table
- Return \(y\)
ROM security

- Take scheme that uses cryptographic hash
- For proof, replace hash by RO
  - Different flavors:
    Random function vs. Programmable RO

- Heuristic security argument
- Allows to verify construction
- Worked for ”Natural schemes” so far

- However: Artificial counter examples exist!
Full Domain Hash Signature Scheme
Trapdoor (One-way) Permutation

\[ F(pk, x) = \pi(x) \]

\[ F(sk, y) = \pi^{-1}(y) \]

Computing \( \pi^{-1}(y) \) without knowledge of \( sk \) computationally hard
RSA Trapdoor (One-way) Permutation

\[(N, e, d) \leftarrow \text{GenRSA}(1^k); \quad pk = (N, e); \quad sk = d\]

\[F(pk, x) = x^e \mod N\]

\[F(sk, y) = y^d \mod N\]

Computing \( \pi^{-1}(y) \) without knowledge of \( sk \) computationally hard if RSA Assumption holds
Generic FDH: Sign

\[ \sigma = F(sk, y) = \pi^{-1}(y) \]

\[ y = H(M) \]

\[ \sigma = \text{Sign}(sk, M) = \pi^{-1}(H(M)) = F(sk, H(M)) \]
Generic FDH: Verify

Verify\((pk, M, \sigma)\):
check \(y = H(M) == \pi(\sigma) = F(pk, \sigma) = y'\)
• Randomized FDH

• Simplified RSA-PSS
  • Standardized in PKCS #1 v2
    (slightly different randomization)

• Tight Reduction from RSA Assumption in ROM
RSA-PFDH

Assume Hashfunction $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$

$\textbf{KeyGen}(1^k)$: Run $(N, e, d) \leftarrow \text{GenRSA}(1^k)$.

Return $(pk, sk)$ with $pk = (N, e), sk = d$.

$\textbf{Sign}(sk, M)$: Sample $r \leftarrow U_k$; Compute $y = H(r||M)$

Return $\sigma = (r, y^d \mod N)$

$\textbf{Verify}(pk, M, \sigma)$: Return 1 iff $\sigma^e \mod N = H(r||M)$
If the RSA Assumption holds, RSA-PFDH is existentially unforgeable under adaptive chosen message attacks in the ROM.
TODO:
Show that any forger $A$ against RSA-PFDH can be used to break the RSA Assumption with approx. the same time and success probability.

"Given a forger $A$ against RSA-PFDH with success probability $\varepsilon$, we construct an oracle Machine $M^A$ that succeeds with probability $\varepsilon/4$."

Reduction

\[
\begin{align*}
\mathcal{M}^A & \quad \text{Implement SIGN} \\
& \quad \text{Transform Problem} \\
& \quad \text{Extract Solution}
\end{align*}
\]

\[x^e = y \mod N\]

\[N, e, y\]

\[pk, 1^n\]

\[\mathcal{A}\]

\[(\sigma^*, M^*)\]

\[M_j\]

\[H(M_j)\]

\[\mathcal{N}, \epsilon, y\]

\[\mathcal{M}_j\]
Reduction: Transform Problem

\[ M^A \]

Implement SIGN

\[ N, e, y \]

\[ x^e = y \mod N \]

Transform Problem

\[ (\sigma, M_i) \]

Extract Solution

\[ \sigma^*, M^* \]

\[ pk = (N, e) \]

\[ pk, 1^n \]

RO

\[ \mathcal{A} \]

\[ H(M_j) \]

\[ M_j \]
Simulate RO such that you can answer SIGN-queries.
Reduction: Implement SIGN

\[ M^A \]

- Implement SIGN
- Transform Problem
- Extract Solution

\[ N, e, y \]

\[ x^e = y \mod N \]

\[ pk = (N, e) \]

\[ pk, 1^n \]

\[ \sigma_i, M_i \]

\[ \sigma^*, M^* \]

\[ H(M_j) \]

\[ \mathcal{A} \]

\[ N, e, y \]
Implement SIGN – Simulate RO

- Keep table of tripples (*,*,*,*)
- When A asks for $H(r||M)$:
  1. If there is an entry $((r||M), x, z)$ in table, return z
  2. If list $L_M$ already exists, go to 3. Otherwise, choose $q_s$ values $r_{M,1}, ..., r_{M,q_s} \leftarrow \{0, 1\}^k$ and store them in a list $L_M$.
  3. If $r \in L_M$ then let $i$ be such that $r = r_{M,i}$. Choose random $x_{M,i} \in \mathbb{Z}_N^*$ and return the answer $z = x_{M,i}^e \mod N$. Store $(r||M, x_{M,i}, z)$ in the table. (RP)
  4. If $r \notin L_M$, choose random $x \in \mathbb{Z}_N^*$ and return the answer $z = y x^e \mod N$. Store $(r||M, x, z)$ in the table.
Implement SIGN

- When A requests some message $M$ to be signed for the $i$th time:
  - let $r_{M,i}$ be the $i$th value in $L_M$ and
  - compute $z = H(r_{M,i}||M)$ using RO.
  - Let $(r||M, x_{M,i}, z)$ be the corresponding entry in the RO table.
  - Output signature $(r_{M,i}, x_{M,i})$. 
Observation

- All **SIGN** queries can be answered!
- **SIGN** queries are answered using hash
  \[ H(r_{M,i}||M) = z = x_{M,i}^e \text{mod } N \]
  - Signature \((r_{M,i}, x_{M,i})\) known by programming RO
- All other hash queries are answered with
  \[ H(r||M) = z = yx^e \text{mod } N \]
  (with high probability).
  - Signature not known!
  - **BUT**: Allows to extract solution from forgery!
- Note: Any **A** with non-negligible success probability has to query RO for digest of forgery message!
Reduction: Extract Solution

\[ M^A \]

\[ N, e, y \]

Implement SIGN

Transform Problem

\[ pk = (N, e) \]

\[ pk, 1^n \]

\[ x^e = y \mod N \]

\[ x: \]

Extract Solution

\[ \sigma^*, M^* \]

\[ (\sigma_i, M_i) \]

\[ q_s \]

\[ q_H \]

\[ A \]

\[ M_j \]

\[ H(M_j) \]

\[ \mathcal{RO} \]

\[ \mathcal{M} \]
Reduction: Extract Solution

• If $A$ outputs a forgery $(M^*, (r^*, \sigma^*))$:
  • If $r^* \in L_{M^*}$ abort.
  • Else, let $(r^* || M^*, x, z)$ be the corresponding entry of the table.
  • Output $\frac{\sigma^*}{x} \mod N$.

• Note:

\[
\left(\frac{\sigma^*}{x}\right)^e = \frac{\sigma^*^e}{x^e} = \frac{H(r^* || M^*)}{x^e} = \frac{yx^e}{x^e} = y \mod N
\]

\[
\Rightarrow \frac{\sigma^*}{x} = e\sqrt{y} \mod N
\]
Analysis

• **Transform Problem:**
  - Succeeds always
  - Generates exactly matching distribution

• **Implement SIGN / RO:**
  - Succeeds always (we choose $r$)
  - Generates exactly matching distribution:
    - RO: Outputs are uniform in $\mathbb{Z}_N^*$
    - SIGN: Follows from RO

• **Extract Solution:**
  - Succeeds iff $A$ succeeds AND
  - $r^* \not\in L_{M^*} \Rightarrow p = \Pr[r^* \not\in L_{M^*}] = (1 - 2^{-\kappa})q_s$
  - Setting $\kappa = \log_2 q_s$: $p \geq \frac{1}{4}$ assuming $q_s \geq 2$
What have we shown?

- We can turn any forger $A$ against RSA-PFDH with success probability $\epsilon$ into an algorithm $M^A$ that solves the RSA problem with probability $\epsilon/4$.

- In reverse:
  If there exists no algorithm to solve the RSA problem with probability $\geq \epsilon$ then there exists no forger against RSA-PFDH that succeeds with probability $\geq 4\epsilon$.

- As proof is in ROM we have to add ”... As long as the used hash function behaves like a RO.”
How to implement RO? (In practice)

Formerly:
- Use hash function + PRG
- E.g. SHA2 in counter mode /
- SHA2-HMAC in counter mode keyed with SHA2(M)

Today:
- Use XOF
- E.g. SHAKE128
Conclusion

• Ad Hoc constructions problematic
  • Blinding / Index Calculus

• Proofs (even in ROM) allow to check construction

• There is one standardized RSA Sig with proof

• Similar situation for DSA (ROM proof)