CS276 Cryptography	**Spring 2004**

Lecture 16: 3.17.04

Lecturer: David Wagner — *Scribe: Yitao Duan*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

16.1 RSA-PFDH Signature Scheme

$$S(m) = (r, H(r \parallel m)^d \pmod{n})$$
$$H : \{0,\}^* \to \mathbb{Z}/n\mathbb{Z}, r \in \{0,\}^l$$

Theorem 16.1 *If RSA is (t, ϵ)-trapdoor permutation, then RSA-PFDH is $(t/c, q, \epsilon + tq/2^l)$-secure for some constant c.*

Proof:

We prove this theorem by proving its contrapositive. Suppose A is an algorithm that $(t/c, q, \epsilon+tq/2^l)$-breaks RSA-PFDH as a signature scheme, as illustrated in Figure 16.1, we can construct another algorithm, B, that runs in time t and given any input y, outputs its cubic root with advantage more than ϵ. I.e. B outputs z s.t. $y = z^3 \pmod{n}$ with advantage more than ϵ.

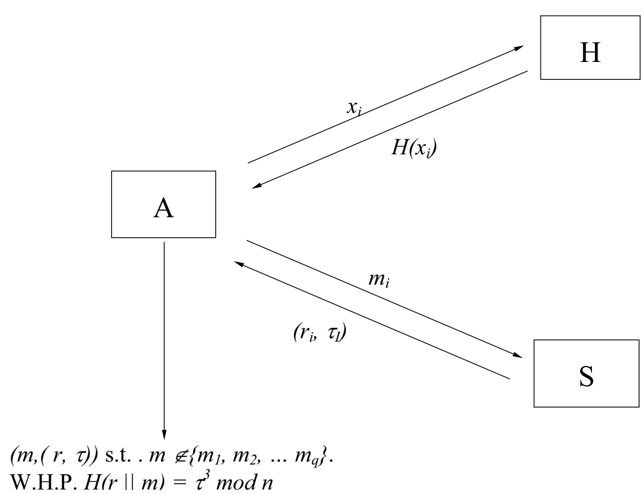

Figure 16.1: *An attack on RSA-PFDH*

Algorithm B is illustrated in Figure 16.2.

In B, S and H are replace by another two oracles S' and H'.

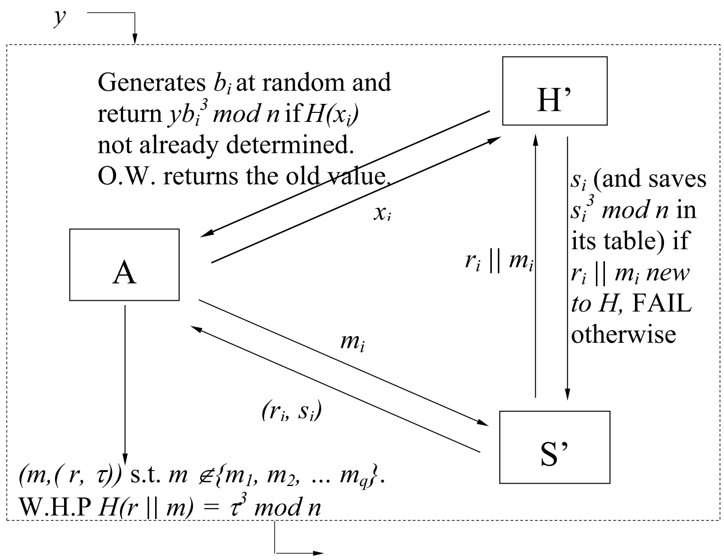

Figure 16.2: *An attack on RSA*

S':

On input m_i, it picks a $r_i \in \{0,\}^l$ at random and sends $(r_i||m_i)$ to H' and obtains s_i back (if H' does not output FAIL. S' then sends (r_i, s_i) to A.

H':

Internally H' maintains a table of all the input/output pairs it has seen/produced so far. On input x_i from A, H' checks the table. If x_i already has an entry in the table, H' returns the corresponding output. Otherwise it generates $b_i \in \mathbb{Z}/n\mathbb{Z}$ in random and returns $yb_i^3 \pmod n$ to A. It also add the entry $(x_i, yb_i^3 \pmod n)$ to its table.

On input $(r_i||m_i)$ from S', H' checks the table. If $(r_i||m_i)$ already has an entry in the table, it returns FAIL. Otherwise it generates $s_i \in \mathbb{Z}/n\mathbb{Z}$ in random and returns s_i to H'. It also add the entry $((r_i||m_i), s_i^3 \pmod n)$ to its table.

Since both b_i and s_i are selected uniformly randomly independent of other selections, the distribution of H''s outputs as seen by A is the same as the random oracle. And the queries to S' always produce signatures that look "valid" to A. So A's capability of breaking the signature is not affected.

B's output is determined as follows:

Clearly in H''s internal table, there are two types of entries: those created in response to S' queries and those entered in response to queries from A, as shown in Table 16.1. A will forge signature using only the second type since it cannot use messages with which it has queried S'. Therefore if no FAIL happens, for any A's output $(m,(r,\tau))$ satisfying $H(r||m) = \tau^3 \pmod n$, $\exists\, i$ s.t. $x_i = r||m$. I.e. $H(x_i) = \tau^3 \pmod n$. Since $H(x_i) = yb_i^3$, B can find the entry in H''s table and output $z = \tau/b_i$. Since $y = (\tau/b_i)^3 \pmod n = z^3 \pmod n$, B inverts RSA.

The running time and advantage of B can be calculated as follows. B only needs to respond to up to $O(t)$ A's queries. Since A runs in time t/c, for some constant c, B runs in time t.

Table 16.1: H''s Internal Table

x	$H(x)$
$\ldots$	$\ldots$
$r_i \| m_i$	s_i^3
$\ldots$	$\ldots$
$\ldots$	$\ldots$
$\ldots$	$\ldots$
x_i	yb_i^3

Define the event $FAILURE$ as: H outputs FAIL to S' when $(r_i||m_i)$ has already been determined. This happens when both of the following two events happen (1) A queries H with $x_i = (r_i||m_i)$ and (2) S' picks r_i as response to query m_i from A.

$$\begin{aligned}
Pr(FAILURE) &\leq Pr(A \text{ and } S' \text{ pick the same } r_i) \\
&= \sum_{r_i} Pr(A \text{ picks } r_i) Pr(S' \text{ picks } r_i) \\
&= 2^l (t/2^l)(q/2^l) \\
&= tq/2^l
\end{aligned}$$

Noticing that $Pr(FORGE|\overline{FAILURE}) = \epsilon + tq/2^l$, then

$$\begin{aligned}
Pr(INVERSION) &= Pr(INVERSION, FAILURE) + Pr(INVERSION, \overline{FAILURE}) \\
&= Pr(INVERSION|FAILURE)Pr(FAILURE) \\
&\quad + Pr(INVERSION|\overline{FAILURE})Pr(\overline{FAILURE}) \\
&\geq Pr(INVERSION|\overline{FAILURE})Pr(\overline{FAILURE}) \\
&= Pr(FORGE|\overline{FAILURE})(1 - Pr(FAILURE)) \\
&= Pr(FORGE|\overline{FAILURE}) - Pr(FORGE|\overline{FAILURE})Pr(FAILURE)) \\
&\geq Pr(FORGE|\overline{FAILURE}) - Pr(FAILURE)) \\
&= \epsilon + tq/2^l - tq/2^l \\
&= \epsilon
\end{aligned}$$

I.e. B inverts RSA with advantage more than ϵ. ■

16.2 Random Oracle Model Controversy

Random oracle model avoids the asymptotic assumptions that reduction-based proofs must rely on (such as intractability of factoring). However, a proof of security in the random oracle model only tells you that the protocol is secure against generic attacks. There might be attacks that exploit the structure of the particular hash function used in an instantiation of the protocol. As a result, some schemes are secure in the random oracle model but broken in practice. We give the following two examples.

Example 1.

$Sig^H(m)$:

1. If $H(0) = SHA1(0)$, then output d.
2. Else output $H(m)^d \pmod{n}$.

In the random oracle model where H returns random bit strings, the probability of $H(0) = SHA1(0)$ is $1/2^l$. This only subtracts a very small constant to the advantage of any algorithm that attacks the signature as $H(m)^d \pmod{n}$. This signature scheme is secure in the random oracle model but completely broken if you plug in SHA1.

Example 2.

$Sig^H(m)$:

1. Parse m as a circuit C, if you can.
2. If $C(r) = H(r)$ for some $r \in \{0,1\}^{160}$, then output d.
3. Else output $H(m)^d \pmod{n}$.

In the random oracle model where H returns random bit strings, for any given C the probability of $H(r) = C(r)$ is $1/2^l$. This scheme is secure in the random oracle model. However, in practice, with any hash function H that is chosen for the scheme, an attacker can always find a message m such that the circuit C determined by m produces the same output as H on any input. This shows that this scheme is completely broken with *any* hash function.