CS 276: Cryptography	**Spring'06**

Lecture 21 — April, 04

Lecturer: David Wagner — *Scribe: Assane Gueye*

Signature scheme with the random oracle model

In the previous lectures, we have seen digital signature ($\mathcal{DS}$) schemes based on TOWP and we presented attacks against these schemes.The main problem of the schemes was that "onewayness" of TOWP was not enough to guaranty security of the signature schemes.
We have also seen that for the "hash-then-invert" scheme to be secure, collision-resistant hash functions were not sufficient.
Today we will explore the idea of a "perfect" hash function, where in perfect, we mean that the output of the hash function "looks random". We will call such a hash function "a random oracle".
A hash-then-invert signature scheme that uses a random oracle as hash function is called Full Domain Hash (FDH).

Definition: A FDH is a digital signature scheme $\mathcal{DS}$ =(Key,Sign,Verif) whose signature and verification algorithms are defined as follows:

$$\begin{array}{l|l} Sign_{tk}(M) & Verif_{pk}(M,x) \\ y \leftarrow H_{ro}(M) & y \leftarrow H_{ro}(M) \\ x \leftarrow f^{-1}(y) & y' \leftarrow f(x) \\ return \quad x & if \quad y = y' \quad then \quad return \quad 1 \\ & else \quad return \quad 0 \end{array}$$

where (pk,tk) is uniformly drawn from Key, H_{ro} is a random oracle.

Let A be an adversary against FDH. The advantage of A is define as:

$$adv(A) = Pr|(pk,tk) \stackrel{\$}{\leftarrow} Key, (m,T) \leftarrow A^{H_{ro},S_{tk}}, Verif(m,T) = 1| \leq \epsilon$$

where A sends queries to the oracles H_{ro}, S_{tk} to forge the pair (m,T) such that m has never been queried. H_{ro} is the random oracle and S_{tk} is the signing oracle (that has access to H_{ro}).
$Verif(m,T)$ is the verification algorithm (that all the parties can run).
For RSA, $f(x) = x^e \ modN$ and $f^{-1}(y) = y^d \ modN$. In this case, the efficiency of the scheme is given by the hash function (in the other parts of the signing algorithm and in the verification algorithm, we just take the power a number which can be done efficiently).
Now, how secure is FDH?

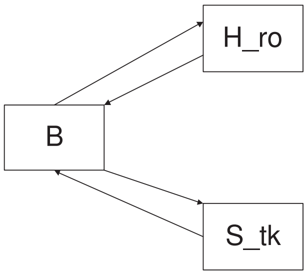

Figure 21.1. Adversary B that have access to two oracles

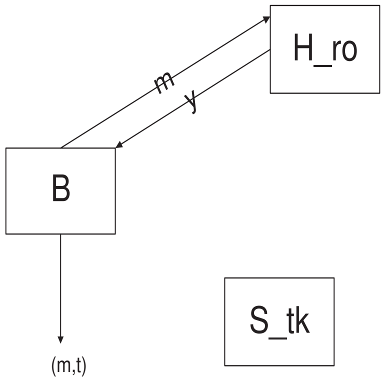

Figure 21.2. Adversary B queries oracle H_{ro} only once and does not use oracle S_{sk}

Theorem 21.1. *If (f, f^{-1}) is a $(t, \epsilon) - TOWP$, then FDH is a $(t - o(q), q, q\epsilon)$-signature scheme (in the random oracle model).*

Proof: We will show that if there exists an attack against FDH, then we can find an adversary that invert f with almost the same amount of resources.
Let B be an adversary that breaks FDH. B has two access to two oracles: the random oracle H_{ro}, and a signing oracle S_{tk} (Figure 21.1).
Note that an adversary B that ignores H_{ro} will perform poorly. In fact if (m, T) is a "forgery" where m has never been queried to H_{ro}, then the chance that the signature is valid (e.g for RSA) is equal to $Pr[H_{ro}(m) = T^e mod N] = \frac{1}{N}$ for $N = 2^{128}$. So we ignore such attacks.
We will assume from now on, that m has already been queried to H_{ro}.
Let A be an adversary whose goal is, given $y \stackrel{\$}{\leftarrow} \mathcal{S}$, to compute $f^{-1}(y)$.
Let's first consider the case where B uses H_{ro} only once and never uses S_{tk} (Figure 21.2).
The adversary A will use a substitute oracle H_s that responds with y when B queries H_{ro}. Since y is uniform on $\mathcal{S}$, H_s will behave (statistically) the same like H_{ro}. If B forges (m, T), then $y = H(m)$, and $T = f^{-1}(H(m)) = f^{-1}(y)$ and A succeeds in inverting f. Further the advantage of A is the same as that of B.
Now suppose that B makes more than one query $(m_1, \ldots, m_k)$ to H_{ro}, and still no query to S_{tk} (Figure 21.3).

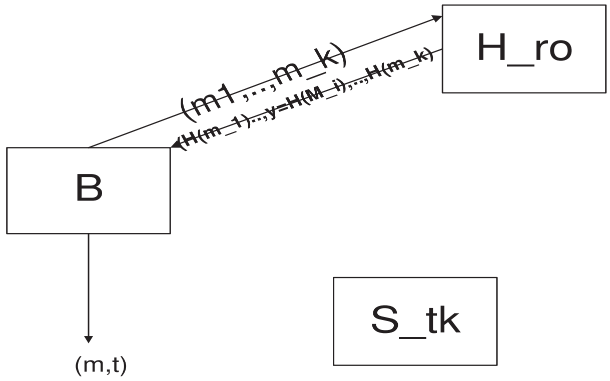

Figure 21.3. Adversary B that have access to two oracles

Now the problem is that A does not now in advance which of the $m_i's$ will be used (by B) to forge a signature. Also it's the substitute oracle cannot always respond with y. So we have to change a bit the algorithm A.
Let's construct A on the following way:

```
A(y)
  init :
  pick i uniformely in  {1,...,t}
  on query m_j to H, do:
    if j=i then return y
    else return H_ro(m_j)
  call algo B on m picked in {m_1,...,m_t}, B outputs (m,t)
  if m_i=m then "success" return T (=f(y))
  else return "fail"
```

Note that the substitute random oracle behaves like the real one because all the outputs are uniformely distributed on $\mathcal{S}$. The chance that the algorithm succeeds (m=mi) is $\frac{1}{t}$. Thus if (f, f^{-1}) is a (t, ϵ)-TOWP, then $\mathcal{DS}$ is $(t - o(q), q, t\epsilon)$-$\mathcal{DS}$.
In the case of RSA the algorithm can be more specialized as follow:

```
A(y)
    init:
    pick i uniformely in {1,...,t}
    draw r_1,...,r_t uniformely from (Z/nZ)*
    on query m_j to H, do:
        return y*(r_j)^e mod N
    call algo B on m_i, B outputs (m,T)
        y*(r_j)^e=H(m_i)=t^e mod N
            ==> y=(T/r_i)^e mod N ==> inv(f,y)=T/r_i
```

In this case the advantage of A is the same as that of B.
Let's now consider the general case where B uses both the oracles H_{ro} and S_{tk}. There are

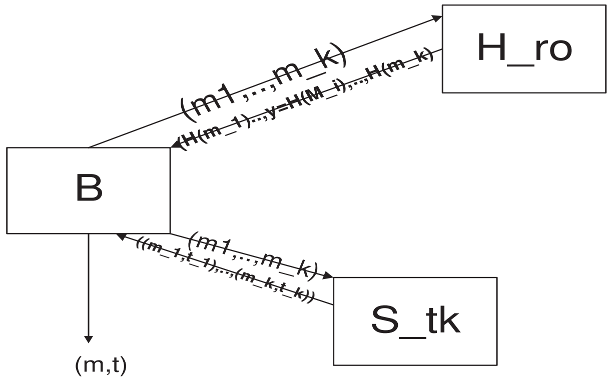

Figure 21.4. Adversary B that have access to two oracles

some issues that are deeply commented in B&R chapter 9 pages 15-16. Please reffer to the book for the details.
The attack will proceed as follow:

```
A(y)
    init:
    pick i uniformely in {1,...,t} /* Golden index */
    draw x_1,...,x_t uniformely from (Z/nZ)* /*Prepare some values */
                                            /*for the signing oracle*/
    on query m_j to H, do:
        if i=j then return y
        else return f(x_j) /*which is (x_j)^e mod N for RSA*/
    call algo B on m_i, B outputs (m,T)
        y*(r_j)^e=H(m_i)=T^e mod N
            ==> y=(T/r_i)^e mod N ==> inv(f,y)=T/r_i
    on query m_l to S, do:
        if l=i then return fail  /* Because we are actually trying to*/
                                 /* invert f on y=m_i*/
        else return x_l
    call algo B in m_n, B outputs (m_n,T)
    if n=i then return T /* same as previously*/
    else return "fail"
```

Note that the algorithm fails if either the signing oracle is queried with the "golden value" or if B forges a signature with a message different from the golden value.
Again for RSA we can improve the algorithm by picking more golden values as in the previous case. □