6.875J/18.425J Cryptography and Cryptanalysis	November 14, 2001
Lecture 19	
Instructor: Anna Lysyanskaya	*Scribed by: Jon Hunt*

1 Secure Signature Definition

A secure singature scheme consists of three algorithms:

- G - key generation
- $SIGN$
- $VERIFY$

that form a secure signature scheme if it satisfies the following two properties:

1. whatever was signed, $VERIFY$ accepts
2. unforgeability.

1.1 Whatever was signed, $VERIFY$ accepts

$$\begin{array}{l} \forall m \forall (PK, SK) \in G \\ \forall \sigma \in SIGN_{SK}(m) \\ VERIFY(PK, m, \sigma) = ACCEPT \end{array}$$

1.2 Unforgeability

The adversary, when given oracle access to $SIGN_{SK}$, cannot forge a valid signature for message that the adversary did not already ask the oracle.

$$\begin{array}{rl} \forall PPTF & A_k \exists \nu(k) \\ Pr[& (PK, SK) \leftarrow G(1^k); \\ & (m, \sigma, Q) \leftarrow A_k^{SIGN_{SK}(\cdot)}(PK) : VERIFY(m, \sigma) = ACCEPT \wedge m \notin Q] < \nu(k) \end{array}$$

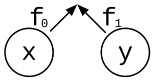
This definition is due to Golwasser, Micali, and Rivest's seminal paper "A Digital Signature Scheme Secure against Adaptive Chosen Message Attacks."

Observation 1 *We can relax the definition of signature a little and still have security, but that makes the math messy so we won't do it in this lecture.*

Example: *The signature key generation based on primes relies on primality tests which are probabilistic and consequently have a small probability of error that we do not worry about above.*

2 GMR signatures

We will now take a look at the first provably secure signature scheme, also due to Goldwasser, Micali, Rivest (GMR).

This construction is based on a primitive "claw-free" permutation pair. A claw is formed when two different permutations on different inputs generate the same output. Claw-free is when it is hard for an adversary to find such a claw. 

Intuition: f_0 and f_1 form a claw-free permutation pair if

1. f_0 and f_1 are TDP over the same domain $\mathcal{D}_f$
2. Hard to find x, y such that $f_0(x) = f_1(y)$

Observation 2 $2 \nrightarrow 1$*: Instead 2 implies OWP and 2 also does not have to have the same domains for* f_0 *and* f_1*.*

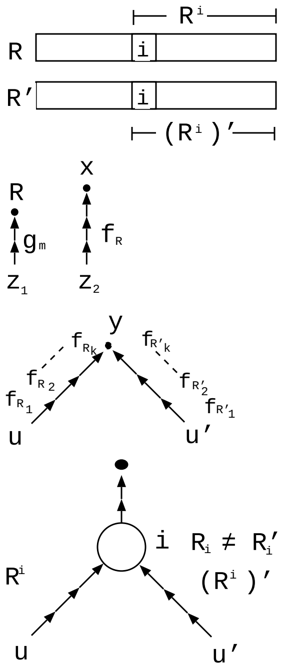
Formalized: Permutations over f_0, f_1 are claw-free permutation pair if

$$\begin{aligned} &\forall A_k \quad \exists \nu(k) \\ &\Pr[\quad f_0, f_1 \leftarrow G(1^k); \\ &\qquad (x,y) \leftarrow A_k(f_0, f_1) : f_0(x) = f_1(y)] < \nu(k) \end{aligned}$$

Claw-free permutations also have the following useful property:

Claim 3 *If* f_0, f_1 *is a claw-free permutation pair then it is hard to find* R, R' *and* u, u' *such that* $|R| = |R'|$ *and* $f_R(u) = f_{R'}(u')$*.*

Sketch of Proof Want to find a claw using R, R', u, u'. Let i be a bit where $R_i \neq R'_i$ and $f_{(R^i)'}(u') = f_{R^i}(u)$. Then we find a claw at bit i:



■

Example: Consider $n = pq$ primes such that $p \equiv 3 \bmod 4$ and $q \equiv 7 \bmod 8$ (a special case of Blum integers. Squaring mod n is a TDP).

Fact 4 *(-1) has Jacobi symbol 1*

Fact 5 *2 has Jacobi symbol -1*

$$\begin{aligned} f_0(x) &= x^2 \text{ where } x \in QR(n) \\ f_1(x) &= 4x^2 \end{aligned}$$

Why is this claw-free?

Suppose x, y such that $x^2 = 4y^2$. What are the square roots of $4y^2$?

$$x, -x, 2y, -2y$$

$$\begin{array}{rl} \text{let } x, y & \text{be such that} \\ x^2 & \equiv y^2 \bmod n \text{ and } x \neq \pm y \\ x^2 - y^2 & \equiv 0 \bmod n \\ & = kn \\ (x-y)(x+y) & = kn \end{array}$$

$(x-y)$ and $(x+y)$ do not divide n because one of them is a proper divsor of kn. x and $-x$ have Jacobi symbol 1, while $2y$ and $-2y$ have Jacobi symbol -1 and the square roots are distinct because the Jacobi symbols are different.

Why is it hard to find x, y? Because, as we saw a few lectures ago, finding four distinct roots is equivalent to factoring.

Let us now use it in a signature scheme.

2.1 First signature scheme (one time)

PK	SK
$x \in \mathcal{D}_f, f_0, f_1$	f_0^{-1}, f_1^{-1}
g_0, g_1	g_0^{-1}, g_1^{-1}

To sign m:

1. pick random $R \in \mathcal{D}_g$
2. let $z_1 = f_R^{-1}(x)$ and $z_2 = g_m^{-1}(R)$

What are f_R and g_m? 1^{st} bit is 1^{st} for inverses $f_R(u) = f_{R_1}(f_{R_2}(\ldots(f_{R_k}(u))\ldots))$ where $R = R_1 \circ R_2 \circ \ldots \circ R_k$. g_m defined in the same way.

Example: $x = 00101$ then

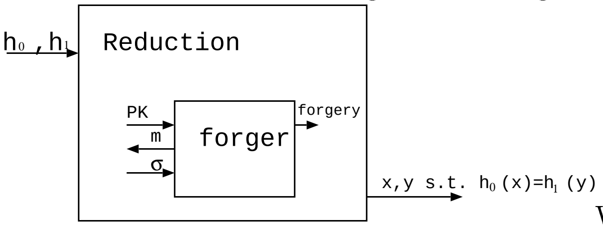
$$f_x(u) = f_0(f_0(f_1(f_0(f_1(u)))))$$

Verification: $(z_1, z_2), R = g_m(z_2)$, check $x = f_R(z_1)$?

What is the message space of σ? It must contain equal length strings otherwise an adversary could apply g additional time to get the secret key.

Claim 6 *This is a secure one-time signature for message space* $\{0,1\}^l$.

Proof Reduction on input h_0, h_1 with the goal of finding a claw to contradict the claw-free statement.



We have two possible types of forgers:

1. Type 1 forger reuses R
2. Type 2 forger does not reuse R

Setup PK for Type 1 forger:

PK	SK
$x \in \mathcal{D}_f, f_0, f_1$	f_0^{-1}, f_1^{-1}
g_0, g_1	?,?

to sign m, pick random z_2 such that $R = g_m(z_2)$ and $z_1 = f_R^{-1}(x)$. The forgery is m', z' such that $g_{m'}(z_2') = R \Rightarrow$ claw for g_0, g_1.

Setup PK for Type 2 forger (we are now looking for a claw for function f):

PK	SK
$x \in \mathcal{D}_f, f_0, f_1$	random R, solve for $x = f_R(z_1)$
g_0, g_1	g_0^{-1}, g_1^{-1}

Signature output z_1, solve for z_2.

$$z_2 = g_m^{-1}(R)$$

The forgery is m', z_2 where $g_m'(z_2') \neq R$. The claw z_1, z_1' is found in f such that $x = f_R(z_1) = f_{R'}(z_1')$.

Note: Why did we bother with R and not jst use x, f_0, f_1 to sign $m, f_m^{-1}(x)$? There has to be some information good enough to sign. In other words there has to be two pieces to the secret key; the reduction will "know" one of them, and hope to obtain the other one from the adversary. ■

3 Generic multiple-signatures constructions

Now let us extend this to a signature scheme that will work many times. **Idea**: Use the previous signature scheme to validate public information for the next signature.

If the PK is the public key and H is the list of all messages signed so far by the signer, then to sign a new message:

1. Choose a one-time public key P_i and a secret key S_i
2. $\sigma_i = SIGN(S_{i-1}, m_i, P_i)$

3. Output H, σ_i

4. $H = H \circ \sigma_i$

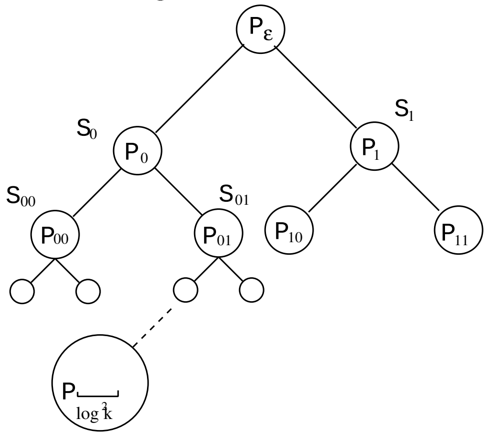
Example: Start with P_0.

$$\begin{array}{r|l} H = \emptyset & \text{generate } S_1, P_1 \\ & m_1\sigma_1 = SIGN(S_0, m_1, P_1) \\ H = \sigma_1 & \text{generate } S_2, P_2 \\ & m_2\sigma_2 = SIGN(S_1, m_2, P_2) \end{array}$$

But this is very inefficient. It includes every message ever signed in the signature of a new message.

4 GMR tree-like construction

Let us look at extending this in a tree like fashion instead. In the last lecuture we used a hash function, but for that we had to precompute everything.

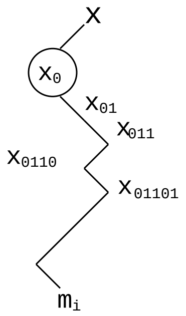


Use P_ϵ to sign string $P_0 \circ P_1$
Use P_0 to sign string $P_{00} \circ P_{01}$
$\vdots$
Use $P_{0...0}$ to sign string $P_{00...00} \circ P_{00...01}$
$(log^2 k) - 1$
$\vdots$
Use $P_{000...000}$ to sign m_1

Observation 7 *We only compute new value when a node has no children and we need to remember only 2 things from each level. The time to compute this goes from linear to polylogorithmic.*

We can optimize this a little bit further. The only one time information that is needed is the $x_i's$, we can keep the rest.

PK	SK
x, f_0, f_1	f_0^{-1}, f_1^{-1}
g_0, g_1	g_0^{-1}, g_1^{-1}



To sign m_i use x:

$$\begin{array}{lrcl}
 & \text{pick } x_0, x_1 & & \\
 & \text{compute } z_\epsilon & = & f^{-1}_{x_0 \circ x_1}(x) \\
\text{To sign } m_1 & z_0 & = & f^{-1}_{x_{00} \circ x_{01}}(x_0) \\
 & & \vdots & \\
 & z_{00\ldots0} & = & f^{-1}_{x_{0\ldots00} \circ x_{0\ldots01}}(x_{000\ldots0})
\end{array}$$

Observation 8 *This construction does not use the fact that the signature scheme is claw-free, rather it is just turning any one-time signature scheme into a many time signature scheme.*

The proof for the one-time signature scheme applies almost directly for the Type 1 forgers who reuse R. The adversary computes the entire (or some significant portion) of the tree and uses g at the leaves. We did not get to the proof for the Type 2 forgers in this lecture.