The OSS Signature Scheme

See:

Ong, Schnorr, Shamir,
An Efficient Signature Scheme Based on Quadratic Equations, proceedings of the 16'th symposium on theory of computing, pp. 208–216, 1984. (A similar paper appears in the proceeding of CRYPTO'84).

Drawbacks of Existing Signature Schemes

All existing digital signature schemes are either

- slow in signature generation
- slow in signature verification
- generate a signature of a huge size

Examples: RSA, Rabin's RSA variant, ElGamal, DSS

All these examples require exponentiation to a large power in either signature generation or verification.

Some of them are relatively fast in verification (e.g., RSA with $e = 3$, Rabin's RSA variant).

The Search for Very Fast Schemes

Many fast schemes were proposed in the past.

For public key encryption, the Merkle-Hellman cryptosystem was quite fast, but was found later totally insecure (although it was based on an NP-complete problem).

For signatures, the OSS scheme is given here as a simple very fast example.

Other schemes based on similar ideas were proposed, but in many cases they were broken in the same conference where they were presented for the first time.

Due to its importance, people are still seeking for fast signature schemes.

If such schemes are found, they will be used by many important application within a short time.

The OSS Signature Scheme

The OSS signature scheme is very fast, both in generation and verification:

- only 2 modular multiplications and one inversion are required for signature generation, and
- only 3 modular multiplications for are required for verification.

The OSS Signature Scheme (cont.)

The OSS scheme is based on the difficulty of solving equations of the form

$$x^2 + ky^2 \equiv m \pmod{n},$$

where $n = pq$ is an RSA moduli, m is the document (or its hash value), and k is the public key of the signer.

Any pair x, y which satisfies this equation is considered as a valid signature on m.

Example: Given k and m, we can fix x to any value, but then finding y requires computing a square root, i.e., is equivalent to factoring n.

Similarly fixing y and finding x is equivalent to factoring n.

Ong, Schnorr, and Shamir proposed that the problem of solving this equation is difficult.

The OSS Signature Scheme (cont.)

The public and private keys:

1. Select $n = pq$ as in RSA (unlike RSA, everybody can use the same n, as long as nobody knows the factorization).
2. Select a random u, $0 < u < n$, and compute $u^{-1} \pmod{n}$.
3. Compute $k = -u^2 \bmod n$.
4. The public key is the pair k, n.
5. The secret key is u^{-1}.

The OSS Signature Scheme (cont.)

Signature generation:

Given a document m $(0 \leq m < n)$

1. Select a random r, where $1 < r < n$
2. Compute

$$\begin{aligned} x &\equiv (r + mr^{-1}) \cdot 2^{-1} \pmod{n} \\ y &\equiv (r - mr^{-1}) \cdot 2^{-1} \cdot u^{-1} \pmod{n}. \end{aligned}$$

3. The pair x, y forms the signature.

Signature verification:

Given x, y, test whether

$$x^2 + ky^2 \stackrel{?}{\equiv} m \pmod{n}.$$

Correctness

Claim: The result of the signature generation algorithm forms a valid signature.

Proof: The selection of x and y is based on the following equality

$$x^2 - u^2y^2 = (x + uy)(x - uy)$$

where $x + uy$ is set as the random r, and $x - uy$ is computed by mr^{-1}.

Thus, the verification results in

$$\begin{aligned}
x^2 + ky^2 &\equiv \left(\left(r + mr^{-1}\right)2^{-1}\right)^2 + \left(\left(r - mr^{-1}\right)2^{-1}u^{-1}\right)^2 \equiv \\
&\equiv 2^{-2}\left[\left(r^2 + m^2r^{-2} + 2m\right) + \underbrace{ku^{-2}}_{\equiv -1}\left(r^2 + m^2r^{-2} - 2m\right)\right] \equiv \\
&\equiv 2^{-2}[2m + 2m] \equiv m \pmod{n}
\end{aligned}$$

QED

Security

Claim: Finding the secret key of OSS is equivalent to factoring n.

Proof: Let A be an algorithm that recovers the secret key given the public key (and possibly signatures on random documents).

We now design a factoring Algorithm B based on Algorithm A:

1. Given $n = pq$ for which p, q are unknown,
2. Select a random u, and compute $u^{-1} \pmod{n}$ and $k = -u^2 \bmod n$.
3. (If signatures on random documents are required for A to work: sign random documents using u^{-1} as the secret key).
4. Call Algorithm A with k, n as the public key (and with the signatures).
5. A returns some u' such that $-u'^2 \equiv k \pmod{n}$.

Security (cont.)

6. With probability of $1/2$ $u' \not\equiv \pm u \pmod{n}$. In these cases

$$\gcd(u' \pm u, n) > 1$$

   are the two prime factors of n.

7. If $u' \equiv \pm u \pmod{n}$ in the previous step, select another u and try again.

8. After t iterations, the probability to find the factorization is $1 - 2^{-t}$.

QED

Pollard's Solution of OSS Equations

See: Pollard, Schnorr, *An efficient solution to the congruence* $x^2 + ky^2 \equiv m \pmod{n}$, IEEE IT, 1984.

This algorithm is another example for the fact that the difficulty of finding the secret key does not necessarily ensures that the scheme is secure.

We will now show how to forge signatures without knowing the secret key $u^{-1} \pmod{n}$.

Pollard's Solution of OSS Equations (cont.)

Properties of equations of the form $x^2 + ky^2 \equiv m \pmod{n}$:

1. The algorithm is based on the equality

$$\left(x_1^2 + ky_1^2\right)\left(x_2^2 + ky_2^2\right) = X^2 + kY^2$$

where

$$\begin{aligned} X &= x_1x_2 \pm ky_1y_2 \\ Y &= x_1y_2 \mp y_1x_2. \end{aligned}$$

This equation suggests that it is possible to write two equations of the form

$$x^2 + ky^2 \equiv m \pmod{n}$$

for each equation of the same form.

Pollard's Solution of OSS Equations (cont.)

2. It is possible to exchange the roles of k and m in the equation by the following change of variables

$$\begin{aligned} x' &\equiv x/y \pmod{n} \\ y' &\equiv 1/y \pmod{n} \end{aligned}$$

By dividing the equation

$$x^2 + ky^2 \equiv m \pmod{n}$$

by y^2 we obtain

$$\begin{aligned} \frac{x^2}{y^2} + k &\equiv \frac{m}{y^2} \pmod{n} && \Rightarrow \\ x'^2 + k &\equiv my'^2 \pmod{n} && \Rightarrow \\ x'^2 - my'^2 &\equiv -k \pmod{n} \end{aligned}$$

Pollard's Solution of OSS Equations (cont.)

3. Pollard observed that it is possible to replace m by a smaller m' of the order of $O(\sqrt{k})$, such that the solution for the equation with m' provides the solution for the original equation (with m).

Pollard's Solution of OSS Equations (cont.)

Pollard's algorithm:

Without loss of generality we can assume that $-k$ is not a perfect square (otherwise it is easy to find the secret key).

1. We first replace m by a smaller prime number m_0 for which $-k$ is a quadratic residue $\left(\frac{-k}{m_0}\right) = 1$, and look for x_0 for which

$$x_0^2 \equiv -k \pmod{m_0} :$$

- Select random $u, v \in Z_n^*$ such that $u^2 + kv^2 \in Z_n^*$, and compute

$$m_0 = m(u^2 + kv^2) \bmod n.$$

- Using a probabilistic algorithm for computing modular square roots modulo prime numbers we find x_0 such that $x_0^2 \equiv -k \pmod{m_0}$.

Pollard's Solution of OSS Equations (cont.)

- Remark: it is a waste of time to test whether m_0 is prime. If the algorithm for computing square roots fail, we just select another u, v and try again.

Pollard's Solution of OSS Equations (cont.)

2. Solve the equation

$$x'^2 + ky'^2 \equiv m_0 \pmod{n}:$$

Define the series

$$m_1, x_1, m_2, x_2, \ldots, m_{I-1}, x_{I-1}, m_I = m'$$

by

(a) m_0 satisfies $m_0 | x_0^2 + k$, and thus there exists m_1 such that

$$x_0^2 + k = m_0 m_1.$$

Let $x_1 = \min(x_0 \bmod m_1, m_1 - x_0 \bmod m_1)$.

(b) m_1 satisfies $m_1 | x_1^2 + k$, and thus there exists m_2 such that

$$x_1^2 + k = m_1 m_2.$$

Let $x_2 = \min(x_1 \bmod m_2, m_2 - x_1 \bmod m_2)$.

Pollard's Solution of OSS Equations (cont.)

(c) Continue with

$$x_i^2 + k = m_i m_{i+1}$$

$$x_{i+1} = \min(x_i \bmod m_{i+1}, m_{i+1} - x_i \bmod m_{i+1}) < m_{i+1}/2$$

$$\vdots$$

$$x_{I-1}^2 + k = m_{I-1} m_I$$

until for some $i = I$

$$\begin{cases} x_{I-1} \le m_I \le m_{I-1} & \text{if } k > 0 \\ |m_I| \le \sqrt{|k|} & \text{if } k < 0 \end{cases}$$

Pollard's Solution of OSS Equations (cont.)

3. Once we found m_I, we multiply all the equations of the form

$$x_i^2 + k = m_i m_{i+1}$$

for $i = 0, 1, \ldots, I-1$, and obtain the non-modular equality

$$(x_0^2 + k)(x_1^2 + k)(x_2^2 + k) \cdots (x_{I-1}^2 + k) = m_0 m_1^2 m_2^2 \cdots m_{I-1}^2 m_I$$

The left side can be replaced by some $s_0^2 + kt_0^2$, and the equality remains valid modulo n:

$$s_0^2 + kt_0^2 \equiv m_0 \left(m_1 m_2 \cdots m_{I-1}\right)^2 m_I \pmod{n}.$$

Change variables $s_1 \equiv s_0/M \pmod{n}$, $t_1 \equiv t_0/M \pmod{n}$, where $M = m_1 m_2 \cdots m_{I-1} m_I \pmod{n}$, and obtain

$$s_1^2 + kt_1^2 \equiv m_0 m_I^{-1} \pmod{n}.$$

Pollard's Solution of OSS Equations (cont.)

4. Now solve
$$s_2^2 + kt_2^2 \equiv m_I \pmod{n}$$
   (a) If m_I is a perfect square, solve by selecting $s_2 = \sqrt{m_I}$, and $t_2 = 0$.
   (b) if $m_I = k$, solve it by selecting $s_2 = 0$ and $t_2 = 1$.
   (c) Otherwise, change variables and solve
$$s_3^2 - m_I t_3^2 \equiv -k \pmod{n}$$
   by a recursive call to the algorithm. On return, change variables back, and get a solution to
$$s_2^2 + kt_2^2 \equiv m_I \pmod{n}.$$

Pollard's Solution of OSS Equations (cont.)

5. We selected m_0 such that

$$m_0 \equiv m(u^2 + kv^2) \pmod{n},$$

i.e.,

$$u^2 + kv^2 \equiv m_0 m^{-1} \pmod{n}.$$

and found solutions for

$$s_1^2 + kt_1^2 \equiv m_0 m_I^{-1} \pmod{n},$$

and

$$s_2^2 + kt_2^2 \equiv m_I \pmod{n}.$$

Thus, we can deduce a solution to

$$s_4^2 + kt_4^2 \equiv m_0^2 m^{-1} \pmod{n},$$

where the right side is the product of the right sides of the three solutions.

Pollard's Solution of OSS Equations (cont.)

6. It is now sufficient to change variables

$$\begin{aligned} x &\equiv s_4 m m_0^{-1} \pmod{n} \\ y &\equiv t_4 m m_0^{-1} \pmod{n} \end{aligned}$$

and obtain the solution for

$$x^2 + ky^2 \equiv m \pmod{n}.$$

Pollard's Solution of OSS Equations (cont.)

Complexity of Pollard's algorithm:

In each recursive call to the algorithm $m_I = O(\sqrt{|k|})$, thus the number of bits of $|k|$ is reduced by a factor of 2. Thus, only about $O(\log\log|k|)$ recursive calls are required.

In each recursive call the complexities of the steps are

1. On average $O(\log n)$ trials of u, v are done.

   The square roots cost $O(\log n)$ for each trial.

   Thus, the total complexity is $O((\log n)^2)$.

Pollard's Solution of OSS Equations (cont.)

2,3. $O(\log n)$:

(without loss of generality assume that all the m_i's are coprime to n):

- If $k > 0$: $0 \leq x_i < m_i/2$ and thus $x_i^2 + k < \frac{1}{4}m_i^2 + k$.
  On the other hand, $x_i^2 + k = m_i m_{i+1}$, and thus

$$m_{i+1} < \frac{1}{4}m_i + k/m_i.$$

  Therefore, as long as $m_i \gg \sqrt{k}$, the size of each m_i is smaller by two bits from its predecessor, and $m_i \leq x_i \leq m_{i+1}$ (since $x_{i-1}^2 + k = m_{i-1}m_i$; $k \ll x_{i-1}^2$)
  If $k \approx x_{i-1}^2$ then we obtain $x_{i-1} \leq m_i \leq m_{i-1}$, set $I = i$, and stop.
  In this case $x_{i-1}^2 + k = m_{i-1}m_i$ where $m_{i-1} \geq m_i$ and $x_{i-1}^2 \approx k$.
  Therefore, $2k \approx m_{i-1}m_i \geq m_i^2 \Rightarrow m_I = m_i \leq O(\sqrt{2k})$ (actually $m_I \approx O(\sqrt{4k/3})$).
  In total the complexity of steps 2 and 3 is $I = O(\log n)$.

Pollard's Solution of OSS Equations (cont.)

- If $k < 0$: In this case some m_i's may be negative.

  For two consecutive $m_i > 0$ and $m_{i+1} > 0$ (e.g., always $m_0 > 0$ and most times $m_1 > 0$)

  $$m_{i+1} = \frac{1}{m_i}\left(x_i^2 - |k|\right) < \frac{1}{4} m_i$$

  and thus after at most $j = O(\log n)$ iterations we obtain $m_j < 0$.

  Then,

  $$|x_j|^2 < |k|$$

  and thus after at most $O(\log n)$ iterations, either

  – $|m_j| < \sqrt{k}$, or

  – $|m_j| > \sqrt{k}$ and $m_{j+1} \leq |k|/|m_j| \leq \sqrt{|k|}$

Pollard's Solution of OSS Equations (cont.)

4. $O(1)$ plus the recursive calls.

5,6. $O(1)$.

The total complexity of each recursive call is $O((\log n)^2)$, and thus the total complexity of Pollard's algorithm is $O((\log n)^2 \log\log |k|)$.