1 Identity based signature

As soon as 1984, Adi Shamir proposed a means to obtain identity based signatures. With H a cryptographic hash function, this works as follows:

Key generation	1. Choose two large prime numbers p and q and produce $n = pq$. 2. Choose e prime with $\varphi(n)$.
Public Key	*known of all users*: (n, e).
Private Key	*Known only of the Private Key Generator (PKG)*: (p, q). *Created by the PKG and then only known by the user of the identity i*: g_i such that $g_i^e \equiv i \mod n$.
Signature	for a message m: 1. Randomly choose r; 2. Compute $t \equiv r^e \mod n$; 3. Compute $s \equiv g_i \cdot r^{H(t,m)} \mod n$; 4. The signature of message m is the pair (s, t).
Verification	check that $s^e \stackrel{?}{=} i \cdot t^{H(t,m)} \mod n$.

1. Show that the check is correct.
2. Explain how the signer and the receiver can *in practice* use the signer's identity to generate i?
3. How can the PKG generate g_i from i?
4. How can g_i remain secret and on which hard problem does this scheme relies?

(1) $s^e == i \cdot t^{H(t,m)} \mod n$

$s \equiv g_i r^{H(t,m)} \mod n \Rightarrow \left[g_i r^{H(t,m)} \right]^e \mod n \stackrel{?}{==} i\, t^{H(t,m)} \mod n$

$g_i^e r^{eH(t,m)} \mod n == i\, t^{H(t,m)} \mod n$

$i\, t^{H(t,m)} \mod n == i\, t^{H(t,m)} \mod n$ ✓

(2) They have to obtain $g_i^e \equiv i \mod n$

e and n are public parameters distributed as public key (e, n)

Starting from a common identity text format, for n of size multiple of the size k (the output of H). It is possible, for example, to split the identity into n/k blocks, each hashed by the cryptographic hash function H, then concatenate these blocks to obtain an i modulo n

(3) Knowing the factorization of n, he can calculate $d \equiv e^{-1} \bmod \phi(n)$ then $g_i \equiv i^d \bmod n$.

(4) They can remain secret since in this schema they are transparent to the user. This problem is the factorization of large primes.

2 Hashing into elliptic curves in a constant number of operations

Many elliptic curve cryptosystems require to hash into an elliptic curve. For example in the Boneh-Franklin IBE scheme, the public-key for identity $bob \in \{0,1\}^*$ is a point $P_{bob} = H_1(bob)$ on the curve.

1. We consider a Boneh-Franklin IBE scheme $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with $\mathbb{G}_1$ a group of prime order q and generator G. Public and private keys of users of the OIBE scheme are respectively defined by $PK_{Bob} = H_1(\text{“}Bob\text{”}) \in \mathbb{G}_1$ and $SK_{Bob} = [s]PK_{Bob}$, for a secret s only known by the PKG, with H_1 a cryptographic hash function with values in $\mathbb{G}_1$. Consider then the following proposition : let $h_q : \{0,1\}^* \hookrightarrow (\mathbb{Z}/q\mathbb{Z})^*$ be a collision-resistant cryptographic hash function with values modulo q. Then define $H_1 : \{0,1\}^* \hookrightarrow \mathbb{G}_1$ such that $H(m) = [h(m)]G$. Show now that any user of the system can find the private key of any other user of the system.
2. What happens if $H_1(m) = h_1(m)G_1 + h_2(m)G_2$ for two distinct generators G_1 and G_2, and two distinct hash functions h_1 and h_2 with values q (or for ny other such linear combination)?

This first idea for H_1 is thus a very bad idea in an IBE setting.

Now suppose that $\mathbb{G}_1 = \mathbb{E}(q; a, b) = \{(x, y) \in \mathbb{F}_q^2, y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$, is the group of poiints of an elliptic curve. Hashing by trial and error on the quadratic residuosity of $x^3 + ax + b$ can be subject to timing attacks in a side-channel setting (for instance in a different framework, password-based authentication, where the number of operations required to hash the password would then depend on the password itself).

On the contrary, the following algorithm, $f_{a,b}(u)$, can map elements of $\mathbb{F}_q$ into points of an elliptic curve $\mathbb{E}(q; a, b)$ in constant number of operations over $\mathbb{F}_q$ (that is independently of the bits composing the element of $\mathbb{F}_q$). Thus one can easily hash into elliptic curves via $H_1(bob) = f_{a,b}(H_q(bob))$, using any other hash function H_q that maps from $\{0,1\}^*$ to $\mathbb{F}_q$.

Require: $q \equiv 2 \mod 3$ a prime power, $a, b, u \in \mathbb{F}_q$, $u \neq 0$.
Ensure: $P = (x, y) \in \mathbb{E}_{a,b}$, s.t. $y^2 = x^3 + ax + b \in \mathbb{F}_q$.
$v = (3a - u^4)(6u)^{-1}$;
$x = 3^{-1}u^2 + (v^2 - b - 27^{-1}u^6)^{(2q-1)/3}$;
$y = ux + v$;
return (x, y).

---

1. If $y = ux + v$, express the curve equation at the output point of the algorithm, $x^3 + ax + b - y^2 = 0$ into an equation of the form $(x + \alpha)^3 + x\beta + \gamma = 0$ with α, β, γ depending on a, b, u, v.
2. Deduce that the algorithm is correct.
3. Why do we need $q \equiv 2 \mod 3$?
4. Reciprocally, show that for any point of the curve, the inverse image of that point is the set of solutions u of the equation $u^4 - 6u^2x + 6uy - 3a = 0$.
5. Deduce that at least $(q - 1)/4$ points of the curve are attained by the algorithm.
6. Using Hasse theorem, show that about one fourth at least of the curve points are attained by the algorithm.

3 Legendre Symbol

Let p be a prime number and a any integer, then Legendre symbol $\left(\frac{a}{p}\right)$ satisfies:

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } p \text{ divides } a \\ 1 & \text{if } a \text{ is a } \textbf{quadratic residue} \text{ modulo } p \text{ (i.e. there exists an integer } k \text{ such that } k^2 \equiv a \mod \\ -1 & \text{if } a \text{ is not a quadratic residue modulo } p \text{ (a } \textbf{quadratic nonresidue}\text{).} \end{cases}$$

1. Show that $\left(\frac{a}{p}\right) = \left(\frac{a \bmod p}{p}\right)$.
2. Give the values of $\left(\frac{a}{2}\right)$, $\left(\frac{0}{p}\right)$, $\left(\frac{1}{p}\right)$.
3. We want to show that for an odd prime number p, we have that $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \mod p$:
   (a) What are the square roots of 1 modulo a prime number p? Deduce the possible values of $a^{\frac{p-1}{2}} \mod p$ for p an odd prime.
   (b) If p is an odd prime, show that if a is a quadratic residue modulo p then $a^{\frac{p-1}{2}} \equiv 1 \mod p$.
   (c) Conversely, let g be a primitive root modulo a prime $p \geq 3$ (i.e. g is a generator of $\mathbb{Z}/p\mathbb{Z}^* = \{g^i, i = 1 \,..\, p-1 = \varphi(p)\}$). Show that if $1 \equiv a^{\frac{p-1}{2}} \equiv g^{i\frac{p-1}{2}}$ then $a = g^i$ is a quadratic residue.
4. Show that $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.
5. Number of quadratic residues:
   (a) Let g be a primitive root modulo a prime $p \geq 3$, what is $\left(\frac{g^{2j}}{p}\right)$?
   (b) Show that g^{2j+1} is not a quadratic residue.
   (c) Concludes on the number of quadratic residues modulo p.

4 Jacobi's Symbol

Jacobi's symbole is a generalization of the Legendre symbol, for all integers: for a prime number factorization $n = \prod p_i^{\alpha_i}$, we define $\left(\frac{a}{n}\right) = \prod \left(\frac{a}{p_i}\right)^{\alpha_i}$. The **quadratic reciprocity law** relates quadratic residues of two different numbers via the following properties:

i $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$

ii $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$

iii For $m > 1$ odd and coprime with n, we have $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)(-1)^{\frac{(m-1)(n-1)}{4}}$

1. What is $\left(\frac{m}{n}\right)$ for m and n non coprime?
2. Show that $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$.
3. If m is even and/or non positive, show how to reduce the computation to the case where $m' > 1$ is odd.
4. Show that $\left(\frac{a}{n}\right) = \left(\frac{a \bmod n}{n}\right)$.
5. Show that a gcd-like algorithm can compute the Jacobi symbol without factoring n.

5 Square root modulo a prime number, Cipolla's method

Modulo 2 square roots are easy to compute, $0^2 \equiv 0 \mod 2$ and $1^2 \equiv 1 \mod 2$. Let then p be an odd prime number and a a quadratic residue modulo p (i.e. $\left(\frac{a}{p}\right) = 1$).

We start by looking for an element z such that $\alpha \equiv z^2 - a \mod p$ is *not* a quadratic residue (i.e. α satisfies $\left(\frac{\alpha}{p}\right) = -1$, or else $\alpha^{\frac{p-1}{2}} \mod p = -1$). Then let's consider the polynomial $P(X) = X^2 - \alpha$.

1. Show that P is irreducible modulo p
2. Show that $X^p \equiv -X \mod P(X) \mod p$.
3. Deduce that $(z+X)^p \equiv (z-X) \mod P(X) \mod p$.
4. Deduce that $s \equiv (z+X)^{\frac{p+1}{2}} \mod P(X) \mod p$ is a square root of a.
5. Give a complexity bound for this algorithm.

6 Is computing square roots as hard as factoring?

We want to show that for RSA modulus it is not easier to compute square roots than to factor. For this we consider the following algorithm.

1. First we need to show that a quadratic residue has exactly 4 distinct roots modulo an odd RSA modulus if they are coprime (if there exists k_1 such that $a \equiv k_1^2 \mod n$ then there exists also k_2, k_3 and k_4, all distincts, such that $a \equiv k_i^2 \mod n$, and those are the only ones).
   (a) Give four roots as a function of k thanks to the Chinese Remaindering Theorem.
   (b) Show that all the roots are distincts.
   (c) Show that those are the only roots.
2. If $n = pq$, what are the possible values of $k+t \mod p$, and $k+t \mod q$ as a function of k_p and k_q, for $k_p \equiv k \mod p$ and $k_q \equiv k \mod q$)?
3. Conclude on the average number of times the algorithm is required to go through the **repeat** loop?

Algorithm 1 Factorization by square root computations

Require: n, an odd RSA modulus.
Require: A_n, an algorithm computing a modular square root modulo n, uinformly among the 4 roots of a quadratic residue modulo n.
Ensure: A factor of n

```
repeat
    Choose k randomly coprime with n;
    Compute s = k^2 mod n;
    Launch t = A_n(s);
    Compute p = pgcd(k + t, n);
until p ≠ 1 and p ≠ n
return p.
```

7 Identity based encryption: Cocks' protocol

Cock's protocol uses the fact that Jacobi's symbol can be computed without requiring a factorization with a gcd-like algorithm. The difficult problem it relies on is computing a square root of a composite number. The protocol follows:

Key generation (by the server)

1. Choose two distinct primes $p \equiv 3 \mod 4$ and $q \equiv 3 \mod 4$ and compute $n = pq$.
2. Compute $a \in \mathbb{Z}/n\mathbb{Z}$ in a public and deterministic manner from the identity of the client until $\gcd(a, n) = 1$ and $\left(\frac{a}{n}\right) = 1$ (for instance via successive applications of a hash function on the identity data).
3. Compute $r \equiv a^{\frac{n-p-q+5}{8}} \mod n$, and $\delta = 1$ if $r^2 \equiv a \mod n$ or $\delta = -1$ if $r^2 \equiv -a \mod n$.

General public key (of the server): n.

Client private key: (δ, r).

Encryption

1. Let $m = (m_i)$ be the binary message to be encrypted, with $m_i \in \{0, 1\}$;
2. For each bit m_i, randomly choose a pair $< y_{+i}, y_{-i} > \in (\mathbb{Z}/n\mathbb{Z})^2$, distinct and prime with n, such that $\left(\frac{y_{\pm i}}{n}\right) = -1$ if $m_i = 0$ or such that $\left(\frac{y_{\pm i}}{n}\right) = 1$ if $m_i = 1$;
3. Send the ciphertext $c = (< c_{+i}, c_{-i} >)$ with $c_{\pm i} = y_{\pm i} \pm a \cdot y_{\pm i}^{-1} \mod n$.

Decryption

1. For a ciphertext $c = (< c_{+i}, c_{-i} >)$;
2. for each $< c_{+i}, c_{-i} >$, if $\left(\frac{c_{\delta i} + 2r}{n}\right) = 1$ then $m_i = 1$, otherwise $m_i = 0$.

1. What can be said of $\left(\frac{a}{p}\right)$ and $\left(\frac{a}{q}\right)$?
2. Using the parity of $\frac{q-1}{2}$, show that $a^{\frac{(p-1)(q-1)}{4}} \equiv \pm 1 \mod p$.
3. Deduce that $a^{\frac{(p-1)(q-1)}{4}} \equiv \pm 1 \mod n$.
4. Show that r is a square root of δa modulo n.
5. factor $c_{\delta i} + 2r$ by one of the $y_{\pm i}$ and deduce that decryption is correct.
6. If it were easy to factor n, knowing that computing a square root modulo a prime number is relatively easy (polynomial algorithm by Tonelli-Shanks), how could we break this code?