JASS 2005: Algorithms for IT Security

Digital Signatures

Olga Shishenina

Contents

1 Elliptic curves

Now let's have a brief look at the elliptic curves defined over the real numbers, because some of the basic concepts are easier to explain in this setting.

1.1 Elliptic curves over the reals

Definition 1 *Let a, $b \in \mathbb{R}$ be constants such that $4a^3 + 27b^2 \neq 0$. A non-singular elliptic curve is the set E of solutions $(x, y) \in \mathbb{R} \times \mathbb{R}$ to the equation*

$$y^2 = x^3 + ax + b \tag{1}$$

together with a special point O called the point at infinity. The value $4a^3 + 27b^2$ is called a curve discriminant.

Let us show that the condition $4a^3+27b^2 \neq 0$ is necessary and sufficient to ensure that the equation $x^3 + ax + b = 0$ has three distinct roots(which may be real or complex numbers).

Let us consider a case when the equation $f(x) = x^3 + ax + b = 0$ has a multiple root, i.e.

$$(x - x_1)^2(x - x_2) = 0.$$

It's derivative is defined by the equation

$$f'(x) = 2(x - x_1)(x - x_2) + (x - x_1)^2 = (x - x_1) \cdot (2(x - x_2) + (x - x_1)).$$

One can proove that $\gcd(f(x), f'(x)) \neq 1 \Leftrightarrow f(x)$ has a multiple root.
Let $\exists x$:

$$\left\{ \begin{array}{l} x^3 + ax + b = 0 \\ 3x^2 + a = 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} x^3 + ax + b = 0 \\ x^2 + a = -2x^2 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} -2x^3 + b = 0 \\ x^2 + a = -2x^2 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} x = \sqrt[3]{\frac{b}{2}} \\ 3x^2 = -a \end{array} \right.$$

$$3\left(\sqrt[3]{\frac{b}{2}}\right)^2 = -a \Leftrightarrow 4a^3 + 27b^2 = 0$$

If $4a^3 + 27b^2 = 0$, then the corresponding elliptic curve is called a *singular elliptic curve*

Now suppose that E is a non-singular elliptic curve. To define a group structure over it's points we have to define a binary operation that satisfies group properties. This operation is usually denoted by addition.
Identity element: $\forall P \in E \;\; P + O = O + P = P$.

Now let us define the addition operation over elliptic curves. Suppose $P, Q \in E$, where $P = (x_1, y_1)$, $Q = (x_2, y_2)$. There are three cases:

1. $x_1 \neq x_2$
2. $x_1 = x_2$ and $y_1 = -y_2$
3. $x_1 = x_2$ and $y_1 = y_2$

First case: $x_1 \neq x_2$.
If $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are two distinct points on a curve, then a third point $R = (x_3, y_3)$: $R = P + Q$ is defined as follows. Firstly we draw a chord L between P and Q and find it's third point of intersection with the curve(as the degree of a line equation is equal to one, and degree of elliptic curve is equal to three). The point $R = (x_3, y_3)$ symmetric to this point with respect to the x-axis is the sum $P + Q$.

It was an geometric approach, now let's figure out an algebraetic formulae to compute R.

$$L: \ y = \lambda x + \nu$$

$$\lambda = \tfrac{y_2 - y_1}{x_2 - x_1} \quad \text{and} \quad \nu = y_1 - \lambda x_1 = y_2 - \lambda x_2$$

In order to find points in $E \cap L$ let's substitute $y = \lambda x + \nu$ into equation for E:

$$(\lambda x + \nu)^2 = x^3 + ax + b$$

$$x^3 + \lambda^2 x^2 + (a - 2\lambda\nu)x + b - \nu^2 = 0 \tag{2}$$

As points P and Q lie on L, then their x coordinates are roots of equation 2. Since 2 is a cubic equation over the reals and it has two real roots x_1 and x_2, then it must have a third real root x_3.

$$x_1 + x_2 + x_3 = \lambda^2 \quad \text{or} \quad x_3 = \lambda^2 - x_1 - x_2$$

It is easy to understand that if $R = (x_3, y_3)$, then $R' = (x_3, -y_3)$. Since $P = (x_1, y_1)$ and $R' = (x_3, -y_3)$ lie on L, then

$$\lambda = \frac{y_1 + y_3}{x_1 - x_3} \quad \text{or} \quad y_3 = \lambda(x_1 - x_3) - y_1.$$

Therefore we have derived a formula for $P + Q$ in the first case:
if $x_1 \neq x_2$, then $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

$$\lambda = \tfrac{y_2 - y_1}{x_2 - x_1}$$

Second case: $x_1 = x_2$ and $y_1 = -y_2$.
We define:

$$\forall\ (x, y) \in E \quad (x, y) + (x, -y) = O$$

It means that $(x, -y)$ is inverse to (x, y) with respect to the elliptic curve addition operation.
Third case: $x_1 = x_2$ and $y_1 = y_2$.
We consider that $y_1 \neq 0$, otherwise we would be in the second case. The third case can be considered as an extreme case of the first one. If $P = Q$, then we draw a the tangent line to the curve at P instead of a chord. A function's derivative at a certain point equals the slope of the tangent measured from the positive direction of the x-axis. Using an implicit differentiation of the equation of E we get:

$$2y\frac{dy}{dx} = 3x^2 + a.$$

Since we draw a tangent line at the point $P = (x_1, y_1)$,then:

$$\lambda = \frac{3x_1^2 + a}{2y_1}.$$

The rest of analysis is the same as in case 1, i.e.:
if $x_1 = x_2$ and $y_1 = y_2$, then $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$x_3 = \lambda^2 - x_1 - x_2$$

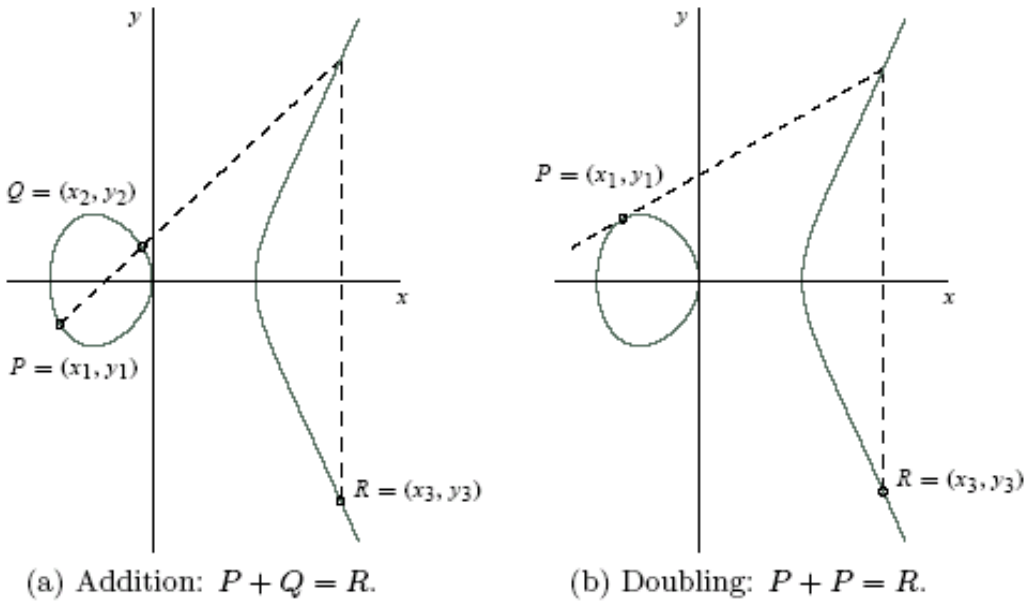
$$y_3 = \lambda(x_1 - x_3) - y_1$$

$$\lambda = \tfrac{3x_1^2+a}{2y_1}.$$

If E is the set of points of the elliptic curve $y^2 = x^3 + ax + b$ then the above defined addition operation satisfies the following properties:

- $\forall\ P,\ Q \in E \quad P + Q \in E$
- $\exists\ O \in E \quad \forall\ P \in E \quad : \quad P + O = O + P = P$
- $\forall\ P \in E \quad \exists\ Q \in E \quad : \quad P + Q = Q + P = O$
- $\forall\ P,\ Q \in E \quad P + Q = Q + P$

So $(E, +)$ is a group under addition.



(a) Addition: $P + Q = R$. (b) Doubling: $P + P = R$.

Figure 1: Geometric addition and doubling of elliptic curve points.

1.2 Elliptic curves over the finite fields

Elliptic curves could be defined over the finite fields exactly as they were defined over the reals provided that all operations in $\mathbb{R}$ are replaced by analogous operations in $GF(q)$. In practice elliptic curves are usually defined over $GF(p)$ or $GF(2^m)$. From now on we will work with finite fields of a prime order. If q is prime, then all operations are made in $\mathbb{Z}_p$.

Definition 2 *Let $p > 3$ be a prime. Let a, $b \in \mathbb{Z}_p$ be constants such that $4a^3 + 27b^2 \neq 0 \bmod p$. A non-singular elliptic curve is the set E of solutions $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ to the equation*

$$y^2 = x^3 + ax + b \bmod p \tag{3}$$

together with a special point O called the point at infinity.

The same formulas can be used to define addition and $(E, +)$ still forms a group under addition. We will denote this group as $E(GF(q))$.

To determine elements of $E(GF(q))$ we have to try all possible $x \in \mathbb{Z}_p$, compute $x^3+ax+b \bmod p$ and then find if the resulted value is a quadratic residue $\bmod p$. A well-known Hasse's theorem states that the cardinality $\#E(GF(q)) = q+1-t$, where $|t| \leq 2q$. The interval $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$ is called a *Hasse's interval*. $\#E(GF(q))$ is called the *order* of E and t is called the *trace* of E. $\#E(GF(q))$ is roughly equal to the size q of the underlying field. The curve E is said to be *supersingular* if $t^2 = 0,\ q,\ 2q,\ 3q,\ 4q$. Otherwise the curve is said to be *non-supersingular*.

$\#E(GF(q))$ is a finite abelian group of rank 1 or 2, in other words, either it is cyclic or else a product of two cyclic groups. If C_n denotes a cyclic group of order n, then $\#E(GF(q))$ is isomorphic to $C_{n_1} \times C_{n_2}$, for unique integers n_1 and n_2 where $n_2|n_1$ and furthermore $n_2|q-1$. If $n_2 = 1$, then $E(GF(q))$ is said to be *cyclic*. In this case $E(GF(q))$ is isomorphic to C_{n_1}, and there exists a *generator* of $E(GF(q))$: a point $P \in E(GF(q))\ :\ E(GF(q)) = \{kP\ :\ 0 \leq k \leq n_1 - 1\}$.

2 Elliptic Curve Digital Signature Algorithm

Discrete log cryptosystems were first described in the setting of the multiplicative group of the integers modulo a prime p. Such systems can be modified to work in the group of points on an elliptic curve. We next describe the elliptic curve digital signature algorithm (ECDSA), which is analogous to the DSA.

This algorithm could be divided into six steps:

1. Domain parameters generation
2. Domain parameters validation
3. Key pair generation
4. Key pair validation
5. Signature generation
6. Signature verification

2.1 ECDSA Domain Parameters

Let's assume that there are the parties involved in a communication. The first step is to create random public abstract groups, which are called domains. For each domain it is necessary to generate domain parameters, which are identical for all users in the domain. This procedure is rather complex, however domain parameters can be taken from the published standards.

The domain parameters for ECDSA are:

- a field size $q : q$ is prime or $q = 2^m$
- suitably chosen elliptic curve E defined over $GF(q)$
- (an optional parameter) a bit string of length ≥ 160 bits
- two elements: $x_G, y_G \in GF(q) : G = (x_G, y_G)$ has a prime order in $E(GF(q))$
- the order n of G
- the cofactor $h = \#E(GF(q))/n$

Some restrictions are placed on the domain parameters.
Elliptic curve requirements:

- against Pohlig-Hellman attack n is prime
- against Pollard's rho attack $n \geq 2^{160}$. In this case it is computationally infeasible to mount this attack.
- against Menezes, Okamoto and Vanstone attack E should be non-supersingular, i.e. $p \nmid q + 1 - \#E(GF(q))$. More generally, one should verify that $n \nmid q^k - 1, \quad \forall\ k\ :\ 1 \leq k \leq 20$. 20 suffices in practice so that it is computationally infeasible to mount this attack.
- finally, to avoid the attack of Semaev, Smart, Satoh and Araki $\#E(GF(q)) \neq q$

A prudent way to guard against these attacks is to generate curve at random or verifiably at random.The probability that a random curve succumbs to these special-purpose attacks is negligible.

2.1.1 Generating an Elliptic Curve verifiably at Random

To generate a curve $y^2 = x^3 + ax + b$ we should generate curve coefficients $a,\ b\ \in GF(q)\ :$ $4a^3 + 27b^2 \neq 0 \bmod p$. Method which generates the curve verifiably at random determines these parameters to be outputs of the one way hash function SHA-1. The input seed to SHA-1 then serves as proof (under the assumption that SHA-1 cannot be inverted) that the elliptic curve was indeed generated at random. This provides some assurance to the user of the elliptic curve that the entity who generated the elliptic curve did not intentionally construct a "weak" curve which it could subsequently exploit to recover the user's private keys.

The case q=p The following notation is used: $t = \lceil \log_2 p \rceil$, $s = \lfloor \frac{t-1}{160} \rfloor$ and $v = t - 160s$.

Algorithm 3 *: Generating a Random Elliptic Curve Over $GF(q)$*

Input: A field size p, where p is an odd prime.
Output: A bit string seedE of length at least 160 bits and field elements $a, b \in GF(q)$ which define an elliptic curve E over $GF(q)$.

1. Choose an arbitrary bit string seedE of length $g \geq 160$ bits.
2. Compute H =SHA-1(seedE), and let c_0 denote the bit string of length v bits obtained by taking the v rightmost bits of H.
3. Let W_0 denote the bit string of length v bits obtained by setting the leftmost bit of c_0 to 0. (This ensures that $r < p$.)
4. Let z be the integer whose binary expansion is given by the g-bit string seedE.
5. For i from 1 to s do:
   (a) Let s_i be the g-bit string which is the binary expansion of the integer $(z + i) \bmod 2^g$.
   (b) Compute W_i = SHA-1(s_i).
6. Let W be the g-bit string obtained by the concatenation of $W_0, W_1, \ldots W_s$ as follows: $W = W_0 \| W_1 \| \cdots \| W_s$.
7. Let r be the integer whose binary expansion is given by W.
8. If $r = 0$ or if $4r + 27 \equiv 0 (\bmod p)$ then go to step 1.
9. Choose arbitrary integers $a, b \in GF(q) : a \neq 0$ and $b \neq 0$ and $r \cdot b^2 \equiv a^3 \bmod p$. (For example one may take $a = r$ and $b = r$.)
10. The elliptic curve chosen over $GF(p)$ is $E : y^2 = x^3 + ax + b$.
11. Output (seedE, a, b).

Isomorphism Classes of Elliptic Curves Over GF(q)

Definition 4 *Two elliptic curves $E_1 : y^2 = x^3 + a_1 x + b_1$ and $E_2 : y^2 = x^3 + a_2 x + b_2$ defined over $GF(q)$ are isomorphic over $GF(q)$ if and only if there exists $u \in GF(q)$, $u \neq 0 : a_1 = u^4 a_2$ and $b_1 = u^6 b_2$.*

If E_1 is isomorphic to E_2, then the abelian groups $E_1(GF(q))$ and $E_1(GF(q))$ are isomorphic as abelian groups. If E_1 and E_2 are isomorphic and $b_1 \neq 0$ (so $b_2 \neq 0$), then $\frac{a_1^3}{b_1^2} = \frac{a_2^3}{b_2^2}$.
The singular curves, i.e. a curves for which $4a^3 + 27b^2 = 0 \bmod p$ are precisely those which have $a = 0$ and $b = 0$, or $\frac{a^3}{b^2} = -\frac{27}{4}$. This means that at step 9 of algorithm 3 we exclude the singular elliptic curves from further consideration.
Let us prove that there are precisely two choices for (a, b) in step 9.

1. There are at least two choices for (a, b).
   We choose u as a quadratic non-residue in $GF(p)$, we chose an arbitrary non-zero pair (a_1, b_1). Then we can define the second pair (a_2, b_2) : $a_1 \neq u^4 a_2$ and $b_1 \neq u^6 b_2$ as:

$$\left.\begin{array}{l} \nexists k : u \equiv k^2 \bmod p \\ a_2 \equiv u^2 a_1 \bmod p \\ b_2 \equiv u^3 b_1 \bmod p \end{array}\right| \Rightarrow \frac{a_1^3}{b_1^2} \equiv \frac{a_2^3}{b_2^2} \equiv r \bmod p, \text{i.e. curves are not isomorphic.}$$

2. $\nexists E_3$: $E_3 \neq E_1$ and $E_3 \neq E_2$ and E_3 is not isomorphic to E_1 and E_3 is not isomorphic to E_2.
   Let's $\frac{a_1^3}{b_1^2} \equiv \frac{a_2^3}{b_2^2} \equiv \frac{a_3^3}{b_3^2} \equiv r \bmod p$.
   Let's $\exists\ u_3$: $\frac{a_3^3}{a_1^3} \equiv \frac{b_3^2}{b_1^2} \equiv u_3^6 \bmod p \Rightarrow a_3 \equiv u_3^2 a_1 \bmod p, \quad b_3 \equiv u_3^3 b_1 \bmod p$.
   If $\exists\ l$: $l^2 \equiv u_3 \bmod p \Rightarrow a_3 \equiv u_3^2 a_1 \bmod p, \quad b_3 \equiv u_3^3 b_1 (\bmod p)$, i.e. isomorphic to E_1.
   Else

$$\left.\begin{array}{l} a_3 \equiv u_3^2 a_1 \bmod p \\ a_2 \equiv u^2 a_1 \bmod p \end{array}\right| \Rightarrow a_3 \equiv \left(\frac{u_1}{u}\right)^2 a_2 \bmod p, \text{ i.e. isomorphic to } E_2.$$

It means that if at step 9 $r \in GF(p),\ r \neq 0,\ r \neq -\frac{27}{4}$, then there are precisely 2 isomorphism classes of curves E : $y^2 = x^3 + ax + b$ with $\frac{a^3}{b^2} \equiv r \bmod p$.

2.1.2 Domain parameters generation

The following is one way to generate cryptographically secure domain parameters:

1. Generate coefficients $(a, b) \in GF(q)$ verifiably at random using 3.
2. Compute $N = \#E(GF(q))$(using for example Schoof's polynomial time algorithm).
3. Verify that $\exists\ n$: $n > 2^{160},\ n > 4\sqrt{q}$ and $n|N$. If not, then go to step 1.
4. Verify that $n \nmid q^k - 1,\ \forall k$: $1 \leq k \leq 20$. If not, then go to step 1.
5. Verify that $n \neq q$. If not, then go to step 1.
6. Select an arbitrary point $G' \in E(GF(q))$ and set $G = (N/n)G'$. Repeat until $G \neq O$.

2.1.3 Domain parameters validation

Domain parameters validation prevent from malicious insertion of invalid domain parameters which may enable some attacks. To validate domain parameters one should check that they have the requisite arithmetical properties.

2.2 ECDSA key pairs

A key pair is assocoated with particular domain parameters. Each entity must have the assurance that the domain parameters are valid prior to key generation.

2.2.1 ECDSA key pair generation

Each entity A does the following:

1. Select a random or pseudorandom integer d in the interval $[1, n-1]$.
2. Compute $Q = dG$.
3. A's public key is Q; A's private key is d.

Public key validation ensures that a public key has a requisite arithmetical properties and that the corresponding private key logically exists. However it does not demonstrate that someone actually has computed the private key nor that the claimed owner actually possesses the private key.

Methods for validating public key are:

- A performs explicit key validation procedure using algorithm shown below.
- A generates Q itself using a trusted system.
- A receives assurance from a trustd party T (e.g. a Certification Authority) that T has performed explicit key validation procedure.
- A receives assurance from a trusted party T that T was generated using a trusted system.

2.2.2 Explicit validation of an ECDSA Public Key

Input: A public key $Q = (x_Q, y_Q)$ associated with valid domain parameters.
Output: Acceptance or rejection of the validity of Q.

1. Check that $Q \neq O$.
2. Check that $x_Q,\ y_Q \in GF(q)$.
3. Check that $y_Q^2 = x_Q^2 + ax_Q + b$.
4. Check that $nQ = O$.
5. If any check fails, then Q is invalid; otherwise Q is valid.

2.3 ECDSA Signature Generation and Verification

This section describes the procedures for generating and verifying signatures using the ECDSA.

2.3.1 ECDSA Signature Generation

To sign a message m, an entity A with domain parameters $D = (q, a, b, G, n, h)$ and associated key pair (d, Q) does the following:

1. Select a random or pseudorandom integer $k,\ 1 \leq k \leq n-1$.
2. Compute $kG = (x_1, y_1)$ and $r = x_1 \bmod n$. If $r = 0$ then go to step 1.
3. Compute $k^{-1} \bmod n$.

4. Compute e =SHA-1(m).
5. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$ then go to step 1.
6. A's signature for the message m is (r, s).

2.3.2 ECDSA Signature Verification

To verify A's signature (r, s) on m, B does the following:

1. Obtains an authentic copy of A's domain parameters $D = (q, a, b, G, n, h)$.
2. Obtains an authentic copy of A's associated public key Q.
3. (Optional) B validates D and Q.
4. Verify that r and s are integers in the interval $[1, n - 1]$.
5. Compute e =SHA-1(m).
6. Compute $w = s^{-1} \bmod n$.
7. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
8. Compute $X = u_1G + u_2Q$. If $X = O$, then reject the signature. Otherwise, compute $v = x_1 \bmod n$ where $X = (x_1, y_1)$.
9. Accept the signature if and only if $v = r$.

Proof that Signature Verification Works. Notice that if a signature (r, s) on a message m was indeed generated by A, then:

$$u_1G + u_2Q = (u_1 + du_2)G = kG \text{ , because}$$

$$k \equiv s^{-1}(e + dr) \equiv s^{-1}e + s^{-1}rd \equiv we + wrd \equiv u_1 + u_2d \bmod n.$$

Thus $v = r$ as required.

2.4 Security

The basis for the security of elliptic curve cryptosystems such as ECDSA is the apparent intractability of the following *elliptic curve discrete logarithm problem (ECDLP):*Given an elliptic curve E defined over $GF(q)$, a point $P \in E(GF(q))$ of order n, and a point $Q \in E(GF(q))$, determine the integer $x \ : \ 1 \leq x \leq n - 1$,such that $Q = xP$, provided that such an integer exists.

The Pohlig-Hellman algorithm reduces the determination of x to the determination of x modulo each of the prime factors of n. Hence, in order to achieve the maximum possible security level, n should be prime. The best general-purpose algorithm known to date for the ECDLP is the Pollard-ρ method which takes fewer than $n^{1/2+\epsilon} = 2^{(1/2+\epsilon)l}$ steps if n is an l-bit prime. We now describe this method.

Given P and Q in a cyclic order-n subgroup $G \subset E(GF(q))$, we want to find x such that $Q = xP$. First, partition $G = S_1 \cup S_2 \cup S_3$ randomly into three sets of roughly equal size. Select $X_0 = a_0P + b_0Q$ with random a_0, b_0. Construct a recursive sequence of points

$$X_{i+1} = \begin{cases} Q + X_i & \text{if } X_i \in S_1; \\ 2X_i & \text{if } X_i \in S_2; \\ P + X_i & \text{if } X_i \in S_2; \end{cases}$$

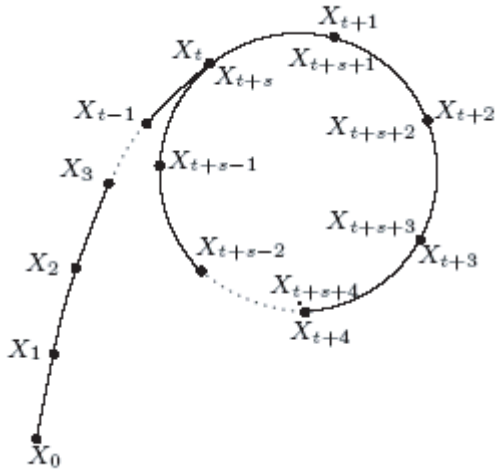


Figure 2: ρ-like shape of the sequence $\{X_i\}$ in the Pollard ρ-method, where t = tail length and s = cycle length.

and recursive sequences of integers

$$a_{i+1} = \begin{cases} a_i & \text{if } X_i \in S_1; \\ 2a_i & \text{if } X_i \in S_2; \\ 1 + a_i & \text{if } X_i \in S_2; \end{cases}$$

and

$$b_{i+1} = \begin{cases} b_i + 1 & \text{if } X_i \in S_1; \\ 2b_i & \text{if } X_i \in S_2; \\ b_i & \text{if } X_i \in S_2; \end{cases}$$

Then $X_i = a_i P + b_i Q$ for all i. The idea is that this sequence eventually becomes periodic. Figure 2 shows how the ρ-method got its name.

Once we find i and j such that $X_i = X_j$ we have

$$X_i = a_i P + b_i Q = (a_i + x b_i)P = X_j = (a_j + x b_j)P$$

and hence

$$a_i + x b_i \equiv a_j + x b_j \bmod n$$

from which x can be determined except in the very unlikely event that $b_i \equiv b_j \bmod n$

$$x \equiv \frac{a_i - a_j}{b_j - b_i} \bmod n$$

In order to greatly reduce storage, in practice one looks for a match between X_i and X_{2i}. This slightly increases the running time, but reduces the storage almost to zero. It was a crucial observation (due to Pollard) that the search for a match between X_i and X_j - which would require storage of order $O(\sqrt{n})$ - can be replaced for a search for a match between X_i and X_{2i}. Otherwise, the ρ-method would have been no better than an earlier deterministic matching method of D. Shanks called "baby step - giant step" that takes roughly the same amount of time and requires $O(\sqrt{n})$ storage.

Assuming that the above map from X_i to X_{i+1} behaves like a random mapping, a match can be found by the time i reaches $O(\sqrt{n})$. Much research has been devoted to improving the Pollard-ρ method . The general form of the estimate for the number of steps remains $O(\sqrt{n})$ even after all the modifications. Thus, the aim is to reduce the constant in $O(\sqrt{n})$.

Sometimes ECDLP can be replaced by the DLP in $GF(q^k)$. A necessary condition for a cyclic subgroup of $E(GF(q))$ of order n to be embedded in $GF(q^k)$ is that $n|q^k-1$. In this case the Index Calculus method with subexponential running time $2^{l^{1/3+\epsilon}}$ can be applied, where $l = \log_2(q^k)$. If $k > \log^2(q)$ then the Index Calculus algorithm for $GF(q^k)$ takes fully exponential time in $\log q$.

For the very special class of supersingular curves, it is known that $k < 6$. For these curves a subexponential-time algorithm for the ECDLP is known. However, a randomly generated elliptic curve has an exponentially small probability of being supersingular; and for most randomly generated elliptic curves we have $k > \log^2 q$.

Also if $\#E(GF(q)) = q$ then Satoh-Araki , Semaev, and Smart showed how to imbed the elliptic curve group into the additive group of integers $\bmod p$ and thereby solve the ECDLP very quickly.

No subexponential-time algorithm is known for the ECDLP except for the special classes discussed above.

3 RSA signature algorithm

This chapter describes the RSA signature algorithm.The sequrity of this scheme lies on the intractability of the integer factorization poblem.

Used notation:

- M is a set of elements called the *message space*.
- M_S is a set of elements called the *signing space*.
- R is a 1-1 mapping from M to M_S called the *redundancy function*.
- M_R is the image of R.
- R^{-1} is the inverse to R, i.e. $R^{-1} \ : \ M_R \rightarrow M$

For RSA signature scheme $M = M_S = S = \mathbb{Z}_n$, where $n = pq$. A redundancy function R is a public knowledge.

3.1 Key generation in RSA signature scheme

Each entity creates an RSA public key and a corresponding private key. Each entity A should do the following:

1. Generate two large distinct primes p and q, each roughly the same size.
2. Compute $n = pq$ and $\varphi = (n-1)(q-1)$
3. Select a random integer $e \ : \ 1 < e < \varphi$ and $\gcd(e, \varphi) = 1$.
4. Compute unique integer $d \ : \ 1 < e < \varphi$ and $ed \equiv 1 \bmod \varphi$.
5. A's public key is (n, e). A's private key is d.

3.2 RSA signature generation and verification

Entity A signs message $m \in M$. Any entity B can verify A's signature and recover the message m from the signature.

- **Signature generation.** Entity A should do the following:
  1. Compute $\widetilde{m} = R(m)$, an integer in the range $[0, n-1]$.
  2. Compute $s = \widetilde{m}^d \bmod n$.
  3. A's signature for m is s.
- **Signature verification.** To verify A's signature and recover the message m, B should do the following:
  1. Obtain A's authentic public key (n, e).
  2. Compute $\widetilde{m} = s^e \bmod n$.
  3. Verify that $\widetilde{m} \in M_R$; if not then reject the signature.
  4. Recover $m = R^{-1}(\widetilde{m})$.

Proof that signature verification works:

$$s \equiv \widetilde{m}^d \bmod n, \text{ where } \widetilde{m} = R(m)$$
$$ed \equiv 1 \bmod \varphi \Rightarrow s^e \equiv \widetilde{m}^{ed} \equiv \widetilde{m} \bmod n$$
$$\text{Finally, } R^{-1}(\widetilde{m}) = R^{-1}(R(m)) = m.$$

In practice numbers with the right bit length are chosen randomly and tested for primality using statistical tests, i.e. Strassen–Test or Miller – Rabin – Test, so there always exists a chance that p and q are not prime.

RSA is based on integer factorisation problem, so anyone who succeeds in factoring $n = pq$ can immediately break RSA by finding inverse of e modulo $\varphi(n)$.

3.3 Multiplicative propery of RSA

The RSA signature scheme has the following multiplicative property. If $s_1 = \widetilde{m_1}^d \bmod n$ and $s_2 = \widetilde{m_2}^d \bmod n$ and if $\widetilde{m} = \widetilde{m_1}\widetilde{m_2}$, then $s = (\widetilde{m_1}\widetilde{m_2})^d = s_1 s_2$. If $\widetilde{m} \in M_R$ then s is valid signature for m : $\widetilde{m} = R(m)$. Hence, to avoid this attack the redundancy function R must not be multiplicative, i.e. $\forall\ a,\ b \in m\ R(ab) \neq R(a)R(b)$.

3.4 Performance characteristics

Let $n = pq$ is a $2k$-bit number, where p and q are each k-bit primes. Computing a signature $s = m^d \bmod n$ requires $O(k^3)$ bit operations. One can compute a signature using a Chinese remainder theorem: calculate $s_1 = m^d \bmod p$ and $s_2 = m^d \bmod q$ and then determine s. The complexity of this operation still remains $O(k^3)$, however it is considerably more efficient in some situations.

If one will choose a public exponent e to be a special number (e.g. 3 or $2^{16}+1$; the choice is based on the fact that e is a prime number and $\widetilde{m}^e \bmod n$ can be computed with only 16 modular squarings and one modular multiplication), then verification requires $O(n^2)$ bit operations. The RSA signature is well suited when signature verification is the predominant operation being performed.

It is not recommended to restrict the size of d in order to improve efficiency of signature generation.

3.5 Bandwidth efficiency

For RSA the redundancy function specified by ISO/IEC 9796 takes k-bit messages and encodes them to $2k$-bit elements in M_S from which a $2k$-bit signature is formed. The bandwidth efficiency in this case is $\frac{1}{2}$. So if an entity wants to sign a kt-bit message she should divide it into t blocks each k-bit long such that $m = m_1 \| m_2 \| \cdots \| m_t$ and sign each block individually. The bandwidth requirement for this case is $2kt$ bits. Another variant is to hash message m to a bitstring of length $l \leq k$ and then sign the hash value.The bandwidth requirement for this case is $kt + 2k$ bits (since we have to transmit extra the kt-bit message). If $t \geq 2$ then $kt + 2k \leq 2kt$, so it follows that the most bandwidth efficient is to use RSA schemes with appendix. For message of at most k-bits scheme with message recovery is preferred.

References

[1] D. Johnson, A. Menezes, *The Elliptic Curve Digital Signature Algorithm (ECDSA),* Technical Report CORR 99–34, Dept. of C&O, University of Waterloo, Canada, 2000. Also available from http://www.cacr.math.uwaterloo.ca

[2] D. Stinson, *Cryptography: Theory and Practice,* 2nd edition, CRC Press, 2002.

[3] A. Menezes, P. Oorschot, S. Vanstone, *Handbook of Applied cryptography,* CRC Press, 1996.

[4] A. Menezes, N. Koblitz, *A Survey of Public-Key Cryptosystems,* 2004.