ECDSA: An Enhanced DSA

Abstract

Elliptic Curve Cryptography is an approach to cryptography based on the usage of elliptic curves over finite fields. This approach allows for smaller key sizes when compared to other schemes in cryptography such as the RSA, while keeping the same level of security. The Elliptic Curve Digital Signature Algorithm (ECDSA) is the most widely used standardized elliptic curve-based signature scheme, with applications in diverse fields. One modern application of the ECDSA is found in the Bitcoin protocol, which has seen a surge in popularity as an open source, digital currency.The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm (DSA).These algorithms heighten security against various attacks and the same time improve performance to obtain efficiencies (time, memory, reduced computation complexity, and energy saving) in an environment of constrained source and large systems

Introduction

Since the introduction of the concept of public-key cryptography by whitefield Diffie and Martin Hellman in 1976, the cryptographic importance of well-studied discrete logarithm problem's apparent intractability has been recognized. Taher ElGamal first described how this problem could be utilized in public-key encryption and digital signature schemes. ElGamal's methods have been redefined and incorporated into various protocols to meet a variety of applications, and one of its extensions forms the basis for the U.S. government digital signature algorithm (DSA).

Some Prerequisite Mathematics

We begin by introducing some basic mathematical terminology. A *group* is an abstract mathematical object consisting of a set G together with an operation $\star$ defined on pairs of elements of G. The operation must have certain properties, similar to those with which we are familiar from ordinary arithmetic. More precisely

1. (*closure*) $a \star b \ \in G \ , \forall \ a, b \in G$.

2. (*associativity*) $a \star (b \star c) = (a \star b) \star c \; \forall \; a, b, c \in G$.

3. (*existence of identity*) there exists an element $e \in G$, called the *identity*, such that $e \star a = e \star a = a \; \forall \; a \in G$.

4. (*existence of identity*) For each $a \in G$ there is an element $b \in G$ such that $a \star b = b \star a = e$. The element b is called the inverse of a, and is denoted by a^{-1}.

A group G is called abelian if $a \star b = b \star a$ for all $a, b \in G$. The order of group G is the number of elements in G.

If p is a prime number, then the non-zero elements of Z_p, namely $Z_p^* = \{1, 2, .., p-1\}$, form a group of order $p-1$ under the operation of multiplication modulo p. The (multiplicative) identity of this group 1.

The discrete logarithm problem, as first employed by Diffie and Hellman in their key agreement protocol, was defined explicitly as the problem of finding logarithms in the group Z_p^* : given $g \in Z_p^*$ of order n, and given $h \in Z_p^*$, find an integer x, $0 \leq x \leq n-1$, such that $g^x \equiv h(mod\ p)$, provided that such an integer exists.The integer x is called the *discrete logarithm* of h to the base g.

These concepts can be extended to arbitrary groups. Let G be a group of order n, and let α be an element of G. The *discrete logarithm problem* for G is the following: given elements α and $\beta \in G$, find an integer x, $0 \leq x \leq n-1$, such that $\alpha^x = \beta$, provided that such an integer exists.

A variety of groups have been proposed for cryptographic use. There are two primary reasons for this. First, the operation in some groups may be easier to implement in software or in hardware than the operation in other groups. Second, the discrete logarithm problem in the group may be harder than the discrete logarithm problem in Z_p^* . Consequently, one could use a group G that is smaller than Z_p^* while maintaining the same level of security. Such is the case with elliptic curve groups, which were first proposed for cryptographic use independently by Neal Koblitz and Victor Miller in 1985. The resulting elliptic curve cryptosystems (ECC) have smaller key sizes, smaller bandwidth requirements, less power consumption, and faster implementations. These features are especially attractive for security applications where computational power and integrated circuit space is limited, such as smart cards, PC cards, and wireless devices.

The Digital signature Algorithm (DSA)

The DSA was proposed in August 1991 by the U.S. National Institute of Standards and Technology (NIST) and became a U.S. Federal Information Processing Standard also referred to as the Digital Signature Standard (DSS) The DSA was the first digital signature scheme accepted as legally binding by a government. The algorithm is a variant of the ElGamal signature scheme. It exploits small subgroups in Z p in order to decrease the size of signatures. The key generation, signature generation, and signature verification procedures for DSA are given as follows.

DSA Key generation

For the genration of key-pair for an Entity A, the following steps are followed:

1. Select a prime q such that $2^{159} < q < 2^{160}$.

2. Select a 1024-bit prime number p with the property that $q|p-1$.

   (The DSS mandates that p be a prime such that $2^{511+64t} < p < 2^{512+64t}$ where $0 \leq t \leq 8$. If t = 8 then p is a 1024-bit prime.)

3. Select an element $h \in Z_p^*$ and compute $g = h^{(p-1)/q}\ mod\ p$; repeat until $g \neq 1$. (g is a generator of the unique cyclic group of order q in Z_p^*.)

4. Select a random integer x in the interval [1, q - 1].

5. Compute $y = g^x\ mod\ p$.

6. A's Public key is (p, q, g, y);

   A's private key is x.

DSA signature generation

Now the entity A has to sign a message m.It is done as follows:

1. Select a random integer k in the interval [1, q-1]
2. Compute $r = (g^k \ mod\ p)$
3. Compute $K^{-1}\ mod\ p$.
4. Compute $s = k^{-1}\{h(m) + xr\}\ mod\ q$, where h is the Secure Hash Algorithm(SHA).
5. If $s = 0$ then go to step 1 and repeat same.(If s = 0, then $s^{-1}\ mod\ q$ does not exist; s^{-1} is required in step 3 of signature verification so $s\ cant\ be$ 0.)
6. The signature for the message m is the pair of integers (r, s).

DSA Signature Verification

To verify A's signature (r, s) on m, the other entity B should do the following:

1. Obtain an authentic copy of A's public key (p, q, g, y).
2. Verify that r and s are integers in the interval [1, q-1].
3. Compute $w = s^{-1}\ mod\ q$ and $h(m)$. Where h is Secure Hashing algorithm(SHA).
4. Compute $u_1 = h(m)w\ mod\ q$ and $u_2 = rw\ mod\ q$.
5. Compute $v = (g^{u_1}y^{u_2}\ mod\ p)\ modq$.
6. Accept the signature if and only if $v = r$.

Since r and $s$are each integers less than q, DSA signatures are 320 bits in size. The security of the DSA relies on two distinct but related discrete logarithm problems. One is the discrete logarithm problem in $Z_p^*$where the number field sieve algorithm applies; this algorithm has a subexponential running time. More precisely, the expected running time of the algorithm is

$$O(e^{((c+o(1))(ln(p))^{1/3})(ln(ln(p))^{2/3})})$$

where c ≈ 1.923 , and $ln(n)$ denotes the natural logarithm function. if p is a 1024-bit prime, then the above expression represents an infeasible amount of computation hence making DSA unvulnerable to this attack. The second discrete logarithm problem works to the base g: given p, q, g and y, find x such that $y \equiv g^x \ (mod\ p)$. For larger p, the best algorithm known for this is the Pollard rho-method, and takes about

$$\sqrt{\pi q/2}$$

steps. If $q \approx 2^{160}$, then the above expression represents infeasable amount of computation; thus DSA is not vulnerable to attack.

However,there are two primary security parameters for DSA, the size of p and the size of q. Increasing one without a corresponding increase in the other will not result in an effective increase in security.

Elliptic Curves

Here Elliptic curves will be briefly introduced.For simplicity,lets restrict this discussion to elliptic curves over Z_p^*, where p is a prime greater than 3. We mention though that elliptic curves can more generally be defined over any finite field.

An *elliptic curve* E over Z_p^* is defined by an equation of the form

$$y^2 = x^3 + ax + b,$$

where $a, b \in Z_p^*$ and $4a^3 + 27b^2 \not\equiv 0 \ (mod\ p)$, together with a special point O, called the *point at infinity.* The set $E(Z_p^*)$ consists of all points $(x, y),\ x \in Z_p^*,\ y \in Z_p^*$, which satisfy the defining equation, together with O

Addition Formulae

There is a rule for adding two points on an elliptic curve $E(Z_p^*)$ to give a third elliptic curve point. Together with this addition operation, the set of points $E(Z_p^*)$ forms a group with O serving as its identity.It is this group that is used in the construction of elliptic curve cryptosystems.

The addition rule is best explained geometrically. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two distinct points on an elliptic curve E. Then the sum of P and Q, denoted $R = (x_3, y_3)$, is defined as follows. First draw the line through P and Q; this line intersects the elliptic curve in a third point. Then R is the refection of this point in the x-axis. This is depicted in Figure 1. The elliptic curve in the figure consists of two parts, the ellipse-like figure and the infinite curve.

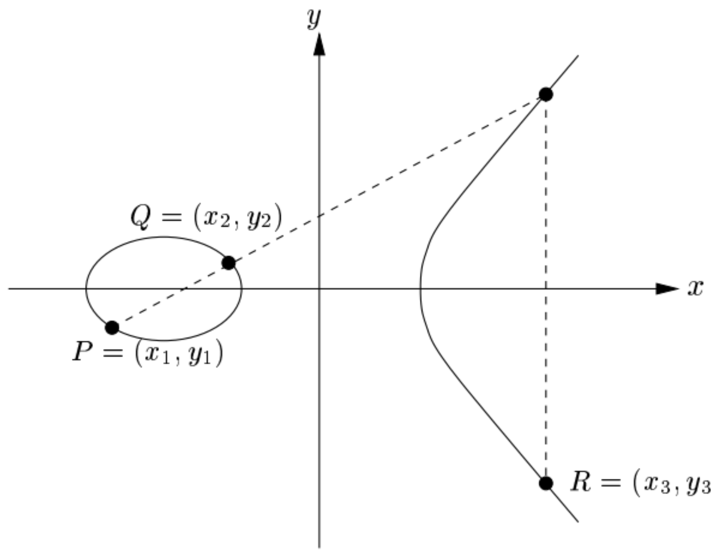

Figure 1: Geometric description of the addition of two distinct elliptic curve points: P + Q = R.

If $P = (x_1, y_1)$, then the double of P , denoted $R = (x_3, y_3)$, is de

ned as follows. First draw the tangent line to the elliptic curve at P . This line intersects the elliptic curve in a second point. Then R is the refection of this point in the x-axis. This is depicted in Figure 2.

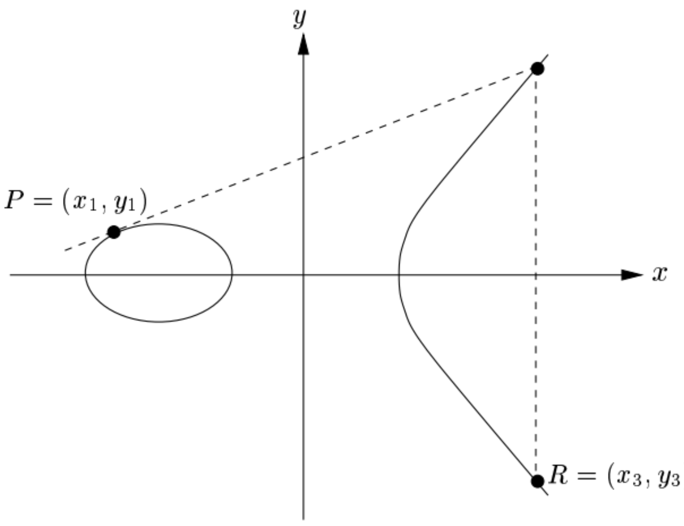

Figure 2: Geometric description of the doubling of an elliptic curve point: P + P = R.

The following algebraic formulae for the sum of two points and the double of a point can now be derived from the geometric description.

1. $P + O = O + P = P$ for all $P \in E(Z_p^*)$.
2. If $P = (x, y) \in E(Z_p^*)$, then $(x, y) + (x, -y) = O$. (The point (x, y) is denoted by P , and is called the negative of P ;P is indeed a point on the curve
3. Let $P = (x_1, y_1) \in E(Z_p^*)$ and $Q = (x_2, y_2) \in E(Z_p^*)$, where $P \neq Q$. Then $P + Q = (x_3, y_3)$, where

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

$$y_3 = \lambda(x_1 - X_2) - y_1,$$

and

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}$$

For historical reasons, the group operation for an elliptic curve $E(Z_p^*)$ has been called addition. In contrast, the group operation in Z_p^* is multiplication. The differences in the resulting additive notation and multiplicative notation can sometimes be confusing. Table 1 shows the correspondence between notation used for the two groups Z_p^* and $E(Z_p^*)$.

Group	$\mathbb{Z}_p^*$	$E(\mathbb{Z}_p)$
Group elements	Integers $\{1, 2, \ldots, p-1\}$	Points (x, y) on E plus $\mathcal{O}$
Group operation	Multiplication modulo p	Addition of points
Notation	Elements: g, h	Elements: P, Q
	Multiplication: $g \cdot h$	Addition: $P + Q$
	Inverse: g^{-1}	Negative: $-P$
	Division: g/h	Subtraction: $P - Q$
	Exponentiation: g^a	Multiple: aP
Discrete Logarithm Problem	Given $g \in \mathbb{Z}_p^*$ and $h = g^a \bmod p$, find a	Given $P \in E(\mathbb{Z}_p)$ and $Q = aP$, find a

Figure 3: Correspondence between Z_p^* and $E(Z_p^*)$ notation

The Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is the elliptic curve analogue of the DSA. That is, instead of working in a subgroup of order q in Z_p^* we work in an elliptic curve group $E(Z_p^*)$

The key generation, signature generation, and signature veri cation procedures for ECDSA are given below.

ECDSA key Generation

Each entity A does the following:

1. Select an elliptic curve E defined over Z_p^*. The number of points in $E(Z_p^*)$ should be divisible by a large prime n.
2. Select a point $P \in E(Z_p^*)$ of order n.
3. Select a statistically unique and unpredictable integer d in the interval $[1, n-1]$
4. Compute $Q = dP$.
5. A's public key is (E, P, n, Q);

   A's Private key is d.

ECDSA signature generation

To sign a message m A does the following:

1. Select a statistically unique and unpredictable integer k in the interval $[1, n-1]$
2. Compute $kP = (x_1, y_1)$ and $r = x_1 mod\ n$. (Here x_1 is regarded as an integer, for example by conversion from its binary representation.) If r = 0, then go to step 1. (This is a security condition: if $r = 0$, then the signing equation $s = k^{-1}\{h(m) + dr\} mod\ n$ does not involve the private key d.)
3. Compute $k^{-1} mod\ n$.

4. Compute $s = k^{-1}\{h(m) + dr\} mod\ n$, where h is the secure Hash Algorithm(SHA)

5. If $s = 0$, then go to step 1.(If $s = 0$, then $s^{-1}\ mod\ n$ does not exist; s^{-1} is required in step 3 of signature verification.)

6. The signature for the message m is the pair of integers(r, s).

ECDSA Signature Verification

To verify A's signature (r, s) on m, B should:

1. Obtain an authentic copy of A's public key (E,P,n,Q).

2. Verify that r and s are integers in the interval $[1, n - 1]$.

3. Compute $w = s^{-1}\ mod\ n$ and $h(m)$.

4. Compute $u_1 = h(m)w\ mod\ n$ and $u_2 = rw\ mod\ n$.

5. Compute $u_1P + u_2Q = (x_0, y_0)$ and $v = x_0 mod\ n$.

6. Accept the signature if and only if $v = r$.

ECDSA has a number of analogies with the DSA.The important ones are:

1. Both algorithms are based on the ElGamal signature scheme and use the same signing equation $s = k^{-1}\{h(m) + dr\} mod\ n$.

2. In both algorithms, the values that are relatively difficult to generate are the system parameters (p, q and g for the DSA; E, P and n for the ECDSA) which are public – their generation can be audited and independently checked

3. Both DSA and ECDSA use the SHA as the sole cryptographic hash function.

Security Considerations

The security objective of ECDSA is to be existentially unforgeable against a chosenmessage attack. The goal of an adversary who launches such an attack against a legitimate entity

is to obtain a valid signature on a single message m, after having obtained A

s signature on a collection of messages (not including m) of the adversarys choice. Some progress has been made on proving the security of ECDSA, albeit in strong theoretical models. Slight variants of DSA and ECDSA (but not ECDSA itself) have been proven to be existentially unforgeable against chosen-message attacks by Pointcheval and Stern under the assumptions that the discrete logarithm problem is hard and that the hash function employed is a random function. ECDSA itself has been proven secure by Brown under the assumption that the underlying group is a generic group and that the hash function employed is collision resistant.

The possible attacks on ECDSA can be classified as follows:

1. Attacks on the elliptic curve discrete logarithm problem
2. Attacks on the hash function employed.
3. Other attacks.