Elliptic Curves and Cryptography

Aleksandar Jurišić* Alfred J. Menezes†

March 23, 2005

Elliptic curves have been intensively studied in number theory and algebraic geometry for over 100 years and there is an enormous literature on the subject. To quote the mathematician Serge Lang:

> It is possible to write endlessly on elliptic curves. (This is not a threat.)

Elliptic curves also figured prominently in the recent proof of Fermat's Last Theorem by Andrew Wiles. Originally pursued for purely aesthetic reasons, elliptic curves have recently been utilized in devising algorithms for factoring integers, primality proving, and in public-key cryptography. In this article, we aim to give the reader an introduction to elliptic curve cryptosystems (ECC), and to demonstrate why ECC provides relatively small block sizes, high-speed software and hardware implementations, and offer the most security per key bit of any known public-key scheme.

Introduction

Since the introduction of the concept of public-key cryptography by Whit Diffie and Martin Hellman in 1976, the cryptographic importance of the apparent intractability of the well-studied discrete logarithm problem has been recognized. Taher ElGamal first described how this problem can be utilized in public-key encryption and digital signature schemes. ElGamal's methods have been refined 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).

We begin by introducing some basic mathematical terminology. A *group* is an abstract mathematical object consisting of a set G together with an operation * defined on pairs of elements of G; the *order* of the group is the number of elements in G. The operation

*Aleksandar received his Ph.D. in mathematics from the University of Waterloo (Canada) in 1994. He works for Certicom Corp. (Canada), where he conducts research in cryptography. Aleksandar can be contacted at ajurisic@certicom.ca.

†Alfred is a co-author, together with Paul van Oorschot and Scott Vanstone, of *Handbook of Applied Cryptography* (CRC Press, 1996). He also is the author of *Elliptic Curve Public Key Cryptosystems* (Kluwer Academic Publishers, 1993). Alfred is a professor of mathematics at Auburn University in Alabama, and consults on a regular basis for Certicom Corp. He can be reached at menezal@mail.auburn.edu.

must have certain properties, similar to those we are familiar with from ordinary integer arithmetic. For example, the integers modulo n, namely $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$, forms a group under the operation of addition modulo n. If p is a prime number, then the non-zero elements of $\mathbb{Z}_p$, namely $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$, forms a group under the operation of multiplication modulo p. The *order* of a group element $g \in G$ is the least positive integer n such that $g^n = 1$. For example, in the group $\mathbb{Z}_{11}^*$, the element $g = 3$ has order 5, since
$3^1 \equiv 3 \pmod{11}$,
$3^2 \equiv 9 \pmod{11}$,
$3^3 \equiv 5 \pmod{11}$,
$3^4 \equiv 4 \pmod{11}$, and
$3^5 \equiv 1 \pmod{11}$.

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 $\mathbb{Z}_p^*$: given an element $g \in \mathbb{Z}_p^*$ of order n, and given $h \in \mathbb{Z}_p^*$, find an integer x, $0 \leq x \leq n-1$, such that $g^x \equiv h \pmod{p}$, provided that such an integer exists. The integer x is called the *discrete logarithm* of h to the base g. For example, consider $p = 17$. Then $g = 10$ is an element of order $n = 16$ in $\mathbb{Z}_{17}^*$. If $h = 11$, then the discrete logarithm of h to the base g is 13 because $10^{13} \equiv 11 \pmod{17}$.

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. Firstly, the operation in some groups may be easier to implement in software or in hardware than the operation in other groups. Secondly, the discrete logarithm problem in the group may be harder than the discrete logarithm problem in $\mathbb{Z}_p^*$. Consequently, one could use a group G that is smaller than $\mathbb{Z}_p^*$ while maintaining the same level of security. This results is smaller key sizes, bandwidth savings, and faster implementations, features which are especially attractive for security applications where computational power and integrated circuit space is limited, such as smart cards, PC-cards, and wireless devices. Such is the case with elliptic curve groups, which were first proposed for cryptographic use by Neal Koblitz and Victor Miller in 1985.

The Digital Signature Algorithm (DSA)

The DSA was proposed in August of 1991 by the U.S. National Institute of Standards and Technology (NIST) and became a U.S. Federal Information Processing Standard (FIPS 186) in 1993. It was the first digital signature scheme to be legally recognized by a government. The algorithm is a variant of the ElGamal signature scheme. It exploits small subgroups in $\mathbb{Z}_p^*$ in order to decrease the size of signatures.

DSA key generation. Each entity A does the following:

1. Select a prime q such that $2^{159} < q < 2^{160}$.
2. Select a 1024-bit prime number p with the property that $q \mid p-1$.
3. Select an element $h \in \mathbb{Z}_p^*$ and compute $g = h^{(p-1)/q} \bmod p$; repeat until $g \neq 1$. (g is a generator of the unique cyclic group of order q in $\mathbb{Z}_p^*$.)
4. Select a random integer x in the interval $[1, q-1]$.
5. Compute $y = g^x \bmod p$.
6. A's public key is (p, q, g, y); A's private key is x.

DSA signature generation. To sign a message m, A does the following:

1. Select a random integer k in the interval $[1, q-1]$.
2. Compute $r = (g^k \bmod p) \bmod q$.
3. Compute $k^{-1} \bmod q$.
4. Compute $s = k^{-1}\{h(m) + xr\} \bmod q$, where h is the Secure Hash Algorithm (SHA-1).
5. If $s = 0$ then go to step 1. (If $s = 0$, then $s^{-1} \bmod n$ does not exist; s^{-1} is required in step 2 of signature verification.)
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, B should do the following:

1. Obtain an authentic copy of A's public key (p, q, g, y).
2. Compute $w = s^{-1} \bmod q$ and $h(m)$.
3. Compute $u_1 = h(m)w \bmod q$ and $u_2 = rw \bmod q$.
4. Compute $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$.
5. 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 length. The security of the DSA relies on two distinct but related discrete logarithm problems. One is the discrete logarithm problem in $\mathbb{Z}_p^*$ where the number field sieve algorithm applies. Since p is a 1024-bit prime, the DSA is currently not vulnerable to this attack. The second discrete logarithm problem works to the base g: given p, q, g, and y, find x such that $y \equiv g^x \pmod{p}$. For large p (e.g. 1024-bits), the best algorithm known for this problem is known as the Pollard rho-method, and takes about $\sqrt{\pi q/2}$ steps. Since $q \approx 2^{160}$, the DSA is not vulnerable to this attack.

Background in elliptic curves

We proceed now to give a quick introduction to the fascinating theory of elliptic curves. For simplicity, we shall restrict our attention to elliptic curves over $\mathbb{Z}_p$, where p is a prime greater than 3. We mention though that elliptic curves can more generally be defined over any finite field. In particular, the so-called *characteristic two finite fields* $\mathbb{F}_{2^m}$ are of special interest since they lead to the most efficient implementation of the elliptic curve arithmetic.

An *elliptic curve* E over $\mathbb{Z}_p$ is defined by an equation of the form

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

where $a, b \in \mathbb{Z}_p$, and $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, together with a special point ∞, called the *point at infinity*. The set $E(\mathbb{Z}_p)$ consists of all points (x, y), $x \in \mathbb{Z}_p$, $y \in \mathbb{Z}_p$, which satisfy the defining equation (**??**), together with ∞.

An example

Let $p = 23$ and consider the elliptic curve $E : y^2 = x^3 + x + 1$ defined over $\mathbb{Z}_{23}$. (In the notation of equation (**??**), we have $a = 1$ and $b = 1$.) Note that $4a^3 + 27b^2 = 4 + 4 = 8 \neq 0$, so E is indeed an elliptic curve. The points in $E(\mathbb{Z}_{23})$ are ∞ and the following:

$(0, 1)$	$(6, 4)$	$(12, 19)$
$(0, 22)$	$(6, 19)$	$(13, 7)$
$(1, 7)$	$(7, 11)$	$(13, 16)$
$(1, 16)$	$(7, 12)$	$(17, 3)$
$(3, 10)$	$(9, 7)$	$(17, 20)$
$(3, 13)$	$(9, 16)$	$(18, 3)$
$(4, 0)$	$(11, 3)$	$(18, 20)$
$(5, 4)$	$(11, 20)$	$(19, 5)$
$(5, 19)$	$(12, 4)$	$(19, 18)$

Addition Formula

There is a rule for adding two points on an elliptic curve $E(\mathbb{Z}_p)$ to give a third elliptic curve point. Together with this addition operation, the set of points $E(\mathbb{Z}_p)$ forms a group. It is this group that is used in the construction of elliptic curve cryptosystems. The addition rule, which can be explained geometrically, is presented below as a sequence of algebraic formulae.

1. $P + \infty = \infty + P = P$ for all $P \in E(\mathbb{Z}_p)$.
2. If $P = (x, y) \in E(\mathbb{Z}_p)$, then $(x, y) + (x, -y) = \infty$. (The point $(x, -y)$ is denoted by $-P$, and is called the *negative* of P; observe that $-P$ is indeed a point on the curve.)

3. Let $P = (x_1, y_1) \in E(\mathbb{Z}_p)$ and $Q = (x_2, y_2) \in E(\mathbb{Z}_p)$, where $P \neq -Q$. Then $P + Q = (x_3, y_3)$, where

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned}$$

and

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

Example of elliptic curve addition

Consider the elliptic curve defined in the previous example.

1. Let $P = (3, 10)$ and $Q = (9, 7)$. Then $P + Q = (x_3, y_3)$ is computed as follows:

$$\begin{aligned} \lambda &= \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in \mathbb{Z}_{23}, \\ x_3 &= 11^2 - 3 - 9 = 6 - 3 - 9 = -6 \equiv 17 \pmod{17}, \text{ and} \\ y_3 &= 11(3 - (-6)) - 10 = 11(9) - 10 = 89 \equiv 20 \pmod{17}. \end{aligned}$$

   Hence $P + Q = (17, 20)$.

2. Let $P = (3, 10)$. Then $2P = P + P = (x_3, y_3)$ is computed as follows:

$$\begin{aligned} \lambda &= \frac{3(3^2)+1}{20} = \frac{5}{20} = \frac{1}{4} = 6 \in \mathbb{Z}_{23}, \\ x_3 &= 6^2 - 6 = 30 \equiv 7 \pmod{23}, \text{ and} \\ y_3 &= 6(3 - 7) - 10 = -24 - 10 = -11 \equiv 12 \pmod{23}. \end{aligned}$$

   Hence $2P = (7, 12)$.

For historical reasons, the group operation for an elliptic curve $E(\mathbb{Z}_p)$ has been called *addition*. By contrast, the group operation in $\mathbb{Z}_p^*$ 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 $\mathbb{Z}_p^*$ and $E(\mathbb{Z}_p)$.

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 $\mathbb{Z}_p^*$, we work in an elliptic curve group $E(\mathbb{Z}_p)$. The ECDSA is currently begin considered by the ANSI X9F1 and IEEE P1363 standards committees as a digital signature

Group	$\mathbb{Z}_p^*$	$E(\mathbb{Z}_p)$
Group elements	Integers $\{1, 2, \ldots, p-1\}$	Points (x, y) on E plus ∞
Group operation	multiplication modulo p	addition of points
Notation	Elements: g, h Multiplication: $g \cdot h$ Inverse: g^{-1} Division: g/h Exponentiation: g^a	Elements: P, Q Addition: $P + Q$ Negative: $-P$ Subtraction: $P - Q$ 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.

Table 1: Correspondence between $\mathbb{Z}_p^*$ and $E(\mathbb{Z}_p)$ notation.

DSA notation	ECDSA notation
q	n
g	P
x	d
y	Q

Table 2: Correspondence between DSA and ECDSA notation.

standard. Table 1 shows the correspondence between some math notation used in DSA and ECDSA. Using Tables 1 and 2, the analogies between the DSA and ECDSA should be more apparent.

ECDSA key generation. Each entity A does the following:

1. Select an elliptic curve E defined over $\mathbb{Z}_p$. The number of points in $E(\mathbb{Z}_p)$ should be divisible by a large prime n.
2. Select a point $P \in E(\mathbb{Z}_p)$ of order n.
3. Select a random integer d such that $[2, n-2]$.
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 random integer k in the interval $[2, n-2]$.
2. Compute $kP = (x_1, y_1)$ and $r = x_1 \bmod n$. (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 $s = k^{-1}\{h(m) + dr\} \bmod n$ does not involve the private key d!)
3. Compute $k^{-1} \bmod n$.
4. Compute $s = k^{-1}\{h(m) + dr\} \bmod n$, where h is the Secure Hash Algorithm (SHA-1).
5. If $s = 0$ then go to step 1. (If $s = 0$, then $s^{-1} \bmod n$ does not exist; s^{-1} is required in step 2 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 do the following:

1. Obtain an authentic copy of A's public key (E, P, n, Q). Verify that r and s are integers in the interval $[1, n-1]$.
2. Compute $w = s^{-1} \bmod n$ and $h(m)$.
3. Compute $u_1 = h(m)w \bmod n$ and $u_2 = rw \bmod n$.
4. Compute $u_1P + u_2Q = (x_0, y_0)$ and $v = x_0 \bmod n$.
5. Accept the signature if and only if $v = r$.

The only significant difference between ECDSA and DSA is in the generation of r. The DSA does this by taking the random element $(g^k \bmod p)$ and reducing it modulo q, thus obtaining an integer in the interval $[1, q-1]$. The ECDSA generates the integer r in the interval $[1, n-1]$ by taking the x-coordinate of the random point kP and reducing it modulo n.

To obtain a security level similar to that of the DSA, the parameter n should have about 160 bits. If this is the case, then DSA and ECDSA signatures have the same bitlength (320 bits).

Instead of each entity generating its own elliptic curve, the entities may elect to use the same curve E and point P of order n. In this case, an entity's public key consists only of the point Q. This results in public keys of smaller sizes. Additionally, there are techniques whereby the point $Q = (x_Q, y_Q)$ can be efficiently constructed from its x-coordinate x_Q and a specific bit of the y-coordinate y_Q. Thus, for example, if $p \approx 2^{160}$ (so elements in $\mathbb{Z}_p$ are 160-bit strings), then public keys can be represented as 161-bit strings.

Security issues

The basis for the security of elliptic curve cryptosystems such as the ECDSA is the apparent intractability of the following *elliptic curve discrete logarithm problem* (ECDLP): given an elliptic curve E defined over $\mathbb{Z}_p$, a point $P \in E(\mathbb{Z}_p)$ of order n, and a point $Q \in E(\mathbb{Z}_p)$, determine the integer l, $0 \leq l \leq n-1$, such that $Q = lP$, provided that such an integer exists.

Over the past eleven years, the ECDLP has been received considerable attention from leading mathematicians around the world, and no significant weaknesses have been reported. An algorithm due to Pohlig and Hellman reduces the determination of l to the determination of l modulo each of the prime factors of n. Hence, in order to achieve the maximum possible security level, n should be prime. The best algorithm known to date for the ECDLP in general is the Pollard rho-method which takes about $\sqrt{\pi n/2}$ steps, where a *step* here is an elliptic curve addition. In 1993, Paul van Oorschot and Michael Wiener showed how the Pollard rho-method can be parallelized so that if r processors are used, then the expected number of steps by each processor before a single discrete logarithm is obtained is $(\sqrt{\pi n/2})/r$.

Software attacks

We assume that a MIPS (Million Instructions Per Second) machine can perform 4×10^4 elliptic curve additions per second. (This estimate is indeed conservative – an application-specific integrated circuit (ASIC) for performing elliptic curve operations over the field $\mathbb{F}_{2^{155}}$ has a 40 MHz clock-rate and can perform roughly 40,000 elliptic operations per second.) Then the number of elliptic curve additions that can be performed by a 1 MIPS machine

in one year is

$$(4 \times 10^4) \cdot (60 \times 60 \times 24 \times 365) \approx 2^{40}.$$

Table **??** shows, for various values of n, the computing power required to compute a single discrete logarithm using the Pollard ρ-method.

Field size (in bits)	Size of n (in bits)	$\sqrt{\pi n/2}$	MIPS years
155	150	2^{75}	3.8×10^{10}
210	205	2^{103}	7.1×10^{18}
239	234	2^{117}	1.6×10^{23}

Table 3: Computing power to compute elliptic curve logarithms with the Pollard ρ-method.

For instance, if 10,000 computers each rated at 1,000 MIPS are available, and $n \approx 2^{150}$, then an elliptic curve discrete logarithm can be computed in 3,800 years. Andrew Odlyzko has estimated that if 0.1% of the world's computing power were available for one year to work on a collaborative effort to break some challenge cipher, then the computing power available would be 10^8 MIPS years in 2004 and $10^{10} - 10^{11}$ MIPS years in 2014.

To put the numbers in Table **??** in some perspective, Table **??** (due to Odlyzko shows the estimated computing power required to factor integers with current versions of the general number field sieve.

Size of n (in bits)	MIPS years
512	3×10^4
768	2×10^8
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

Table 4: Computing power required to factor integers using the general number field sieve.

Hardware attacks

A more promising attack (for well-funded attackers) on elliptic curve systems would be to build special-purpose hardware for a parallel search using the Pollard rho-method. Van Oorschot and Wiener provide a detailed study of such a possibility. They estimated that if $n \approx 10^{36} \approx 2^{120}$, then a machine with $m = 325,000$ processors that could be built for about $10 million would compute a single discrete logarithm in about 35 days.

Discussion

It should be pointed out that in the software and hardware attacks described above, computation of a single elliptic curve discrete logarithm has the effect of revealing a *single* user's private key. The same effort must be repeated in order to determine another user's private key.

Blaze et al. reported on the minimum key lengths required for secure symmetric-key encryption schemes (such as DES and IDEA). Their report comes to the following conclusion:

> To provide adequate protection against the most serious threats – well-funded commercial enterprises or government intelligence agencies – keys used to protect data today should be at least 75 bits long. To protect information adequately for the next 20 years in the face of expected advances in computing power, keys in newly-deployed systems should be at least 90 bits long.

Extrapolating these conclusions to the case of elliptic curves, we see that n should be at least 150 bits for short-term security and at least 180 bits for medium-term security. This extrapolation is justified by the following considerations:

1. Exhaustive search through a k-bit symmetric key cipher takes about the same time as the Pollard rho-algorithm applied to an elliptic curve having a $2k$-bit parameter n.
2. Both exhaustive search with a symmetric-key cipher and the Pollard rho-algorithm can be parallelized with a linear speedup.
3. A basic operation with elliptic curves (addition of two points) is computationally more expensive than a basic operation in a symmetric key cipher (encryption of one block).
4. In both symmetric-key ciphers and elliptic curve systems, a "break" has the same effect: it recovers a single private key.

Implementation issues

Since the elliptic curve discrete logarithm problem appears to be harder that the discrete logarithm problem in $\mathbb{Z}_p^*$ (or the problem of factoring a composite integer n), one can use an elliptic curve group that is significantly smaller that $\mathbb{Z}_p^*$ (respectively, n). For example, an elliptic curve $E(\mathbb{Z}_p)$ with a point $P \in E(\mathbb{Z}_p)$ whose order is a 160-bit prime offers approximately the same level of security as DSA with a 1024-bit modulus p and RSA with a 1024-bit modulus n.

In order to get a rough idea of the computational efficiency of elliptic curve systems, let us compare the times to compute

(i) kP where $P \in E(\mathbb{Z}_p)$, E is a non-supersingular curve, $p \approx 2^{160}$, and k is a random 160-bit integer (this is an operation in ECDSA); and

(ii) $g^k \bmod p$, where p is a 1024-bit prime and k is a random 160-bit integer (this is an operation in DSA).

Let us assume that a field multiplication in $\mathbb{Z}_p$, where $\log_2 p = l$, takes l^2 bit operations; then a modular multiplication in (ii) takes $(1024/160)^2 \approx 41$ times longer than a field multiplication in (i). Now, computing kP by repeated doubling and adding requires on average 160 elliptic curve additions and 80 elliptic curve doublings. From the addition formula we see that an elliptic curve addition or doubling requires 1 field inversion and 2 field multiplications. (The cost of field addition is negligible, as is the cost of a field squaring if the field $\mathbb{F}_{2^m}$ is used instead of $\mathbb{Z}_p$.) Assume also that the time to perform a field inversion is equivalent to that of 3 field multiplications (this is what has been reported in practice for the case of $\mathbb{F}_{2^m}$). Then, computing kP requires the equivalent of 1200 field multiplications, or $1200/41 \approx 29$ 1024-bit modular multiplications. On the other hand, computing $g^k \bmod p$ by repeated squaring and multiplying requires on average 240 1024-bit modular multiplications. Thus, the operation in (i) can be expected to be about 8 times faster than the operation in (ii). It must be emphasized that such a comparison is indeed very rough, as it does not take into account the various enhancements that are possible for each system. Since multiplication in $\mathbb{F}_{2^m}$ is in fact substantially faster than modular multiplication in $\mathbb{Z}_p^*$, even more impressive speedups can be realized in practice.

Another important consequence of using a smaller group in elliptic curve systems is that low-cost implementations are feasible in restricted computing environments, such as smart cards and cellular telephones. For example, an ASIC built for performing elliptic curve operations over the field $\mathbb{F}_{2^{155}}$ (see Agnew, Mullin, and Vanstone [**??**]) has only 12,000 gates and would occupy less that 5% of the area typically designated for a smart card processor. By comparison, a chip designed to do modular multiplication of 512-bit numbers (see Ivey et al. [**??**]) has about 50,000 gates, while the chip designed to do field multiplications in $\mathbb{F}_{2^{593}}$ (see Agnew et al. [**??**]) has about 90,000 gates.

Another advantage of elliptic curve systems is that the underlying field ($\mathbb{Z}_p$ or $\mathbb{F}_{2^m}$) and a representation for its elements can be selected so that the field arithmetic (addition, multiplication, and inversion) can be optimized. This is not the case for systems based on discrete log (respectively, integer factorization), where the prime modulus p (respectively, the composite modulus n) should not be chosen to have a special form because this might render the underlying problem easy.

Standards activities

The two primary objectives of industry standards are to promote interoperability and to facilitate widespread use of well-accepted techniques. Standards for elliptic curve systems are currently being drafted by various accredited standards bodies around the world; some of this work is summarized below. As these drafts become officially adopted by the appropriate

standards bodies, one can expect elliptic curve systems to be widely used by providers of information security.

1. Elliptic curve systems are being drafted in two work items by the American National Standards Institute (ANSI) X9 (Financial Services) working group: ANSI X9.62, The Elliptic Curve Digital Signature Algorithm (ECDSA); and ANSI X9.63, Elliptic Curve Key Agreement Protocols.

2. Elliptic curves are in the draft IEEE P1363 standard (Standard for Public Key Cryptography), which includes encryption, signature, and key agreement mechanisms. Elliptic curves over $\mathbb{Z}_p$ and over $\mathbb{F}_{2^m}$ are both supported. For the case of $\mathbb{F}_{2^m}$, polynomial bases and normal bases of $\mathbb{F}_{2^m}$ over an arbitrary subfield $\mathbb{F}_{2^l}$ are supported. P1363 also specifies discrete log systems (in subgroups of the multiplicative group of the integers modulo a prime) and RSA encryption and signatures. The latest drafts are available from the web site `http://stdsbbs.ieee.org/`.

3. The OAKLEY Key Determination Protocol of the Internet Engineering Task Force (IETF) describes a key agreement protocol that is a variant of the Diffie-Hellman protocol. It allows for a variety of groups to be used, including elliptic curves over $\mathbb{Z}_p$ and $\mathbb{F}_{2^m}$. The document makes specific mention of elliptic curve groups over the fields $\mathbb{F}_{2^{155}}$ and $\mathbb{F}_{2^{210}}$. A draft is available from the web site `http://www.ietf.cnri.reston.va.us/`.

4. The draft document ISO/IEC 14888: Digital signature with appendix – Part 3: Certificate-based mechanisms specifies elliptic curve analogues of some ElGamal-like signature algorithms.

5. The ATM Forum Technical Committee's Phase I ATM Security Specification draft document aims to provide security mechanisms for ATM (Asynchronous Transfer Mode) networks. Security services provided include confidentiality, authentication, data integrity, and access control. A variety of systems are supported, including RSA, DSA, and elliptic curve systems.

Conclusions

Elliptic curve cryptosystems offer the most security per key-bit of any known public-key system. With a 160-bit modulus, an elliptic curve systems offers the same level of cryptographic security as DSA or RSA with 1024-bit moduli. The smaller key sizes result in smaller system parameters, smaller public-key certificates, bandwidth savings, faster implementations, lower power requirements, and smaller hardware processors.

Further reading

For an accessible introduction to all aspects of cryptography, check out Schneier's book [**??**]. Stinson's book [**??**] is an excellent textbook. The recent handbook by Menezes, van Oorschot, and Vanstone [**??**] is an extensive source book on cryptography for practitioners.

Elliptic curve cryptosystems were introduced in the papers of Koblitz [**??**] and Miller[**??**]. Chapter 6 of Koblitz's book [**??**] provides an introduction to elliptic curve and elliptic curve systems. Koblitz's book also covers the relevant number theory algorithm including the Pollard rho-method. The parallelization of the Pollard rho-method is described in [**??**]. For a more detailed account on various implementation and security issues, consult Menezes' book [**??**].

Finally, we mention the Information Security Classroom at Certicom's web site `http://www.certicom.ca`. The information presented there was designed to instruct people of various mathematical backgrounds, and include some nifty Java applets which illustrate the theory of elliptic curves.

References

1. G. Agnew, R. Mullin, I. Onyszchuk and S. Vanstone. "An implementation for a fast public-key cryptosystem", *Journal of Cryptology*, **3** (1991), 63-79.

2. G. Agnew, R. Mullin and S. Vanstone, "An implementation of elliptic curve cryptosystems over $F_{2^{155}}$", *IEEE Journal on Selected Areas in Communications*, **11** (1993), 804-813.

3. P. Ivey, S. Walker, J. Stern and S. Davidson, "An ultra-high speed public key encryption processor", *Proceedings of IEEE Custom Integrated Circuits Conference*, Boston, 1992, 19.6.1 – 19.6.4.

4. N. Koblitz, "Elliptic curve cryptosystems", *Mathematics of Computation*, **48** (1987), 203-209.

5. N. Koblitz, *A Course in Number Theory and Cryptography*, 2nd edition, Springer-Verlag, 1994.

6. A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

7. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

8. V. Miller, "Uses of elliptic curves in cryptography", *Advances in Cryptology – CRYPTO '85*, Lecture Notes in Computer Science, **218** (1986), Springer-Verlag, 417-426.

9. National Institute for Standards and Technology, "Digital signature standard", FIPS Publication 186, 1993. Available from `http://csrc.ncsl.nist.gov/fips/`.

10. A. Odlyzko, "The future of integer factorization", *CryptoBytes – The technical newsletter of RSA Laboratories*, volume 1, number 2, Summer 1995, 5-12. Also available from `http://www.rsa.com/`.

11. M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener, "Minimal key lengths for symmetric ciphers to provide adequate commercial security", January 1996, available from `http://theory.lcs.mit.edu/ rivest/publications.html`.

12. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd edition, Wiley, 1996.

13. D. Stinson, *Cryptography – Theory and Practice*, CRC Press, 1995.

14. P. van Oorschot and M. Wiener, "Parallel collision search with cryptanalytic applications", to appear in *Journal of Cryptology*.

Today, we can find elliptic curves cryptosystems in TLS, PGP and SSH, which are just three of the main technologies on which the modern web and IT world are based. Not to mention Bitcoin and other cryptocurrencies.Â With a series of blog posts I'm going to give you a gentle introduction to the world of elliptic curve cryptography. My aim is not to provide a complete and detailed guide to ECC (the web is full of information on the subject), but to provide a simple overview of what ECC is and why it is considered secure, without losing time on long mathematical proofs or boring implementation details. I will also give helpful examples together with visual interactive tools and scripts to play with. Specifically, here are the topics I'll touch Elliptic-Curve Cryptography. The Curves That Keep The Bitcoin Secure. Grayblock.Â 2 Elliptical curve with points defined as yÂ² = xÂ³ + ax + b. In the cryptographic usage, the same ideas of finding two unique numbers (points in a two dimensional curve) which are related and a max ceiling to wrap around apply. How do we go about getting two numbers that are related but in a way that no one can tell? Well we use the curveâ€™s projective property and draw a line tangent to the starting point P, finding where it intersects the curve at a second point PÊ¹. Then flip the axis and draw a line from that new point (2â€¢P) through the starting point and find the new intersection point PÊ¹Ê¹.