Elliptic Curve Cryptography – How it Works

Sheueling Chang, Hans Eberle, Vipul Gupta, Nils Gura, *Sun Microsystems Laboratories*

{sheueling.chang, hans.eberle, vipul.gupta, nils.gura}@sun.com

http://research.sun.com/projects/crypto

1. Introduction

While conventional public-key cryptosystems (RSA, Diffie-Hellman and DSA) operate directly on large integers, an Elliptic Curve Cryptosystem (ECC) operates over points on an elliptic curve. The following sections illustrate the fundamental operations underlying ECC and describe the Elliptic Curve Discrete Logarithm Problem which makes ECC more efficient relative to traditional counterparts.

2. Elliptic Curve Point Addition & Doubling

A cryptosystem often requires the use of an algebraic group – a set of elements with custom-defined arithmetic operations. Elliptic curve groups used in cryptography are defined over two kinds of fields: GF(p), where p is a prime, and GF(2^m) where each element is a binary polynomial of degree m (that can be represented as an m-bit string since each coefficient is either 0 or 1); but it is easier to illustrate group operations by first examining curves over real numbers.

Figure 1 shows point addition on an elliptic curve. The curve equation is $y^2 = x^3 + ax + b$ with a = -4, b = 4. To add two points, draw a line through them and reflect the third point, where this line intersects the curve, in the x-axis.

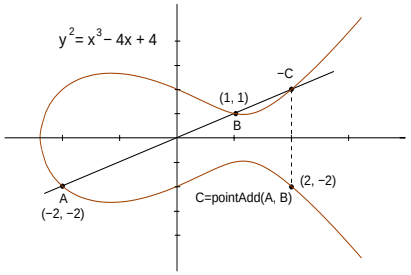


Fig 1. Elliptic curve *point addition*.

Algebraically, the result of adding points A(x_A, y_A) and B(x_B, y_B) is C(x_C, y_C) such that

$$x_C = s^2 - x_A - x_B, \quad y_C = -y_A + s(x_A - x_C)$$

where $s = (y_A - y_B)/(x_A - x_B)$ is the slope of the line through A and B. When A equals B, the line through A and B degenerates to the tangent at A (see Figure 2) and $s = (3x_A^2 + a)/2y_A$.

The result of adding A and -A is defined to be a special point called the *point at infinity*.

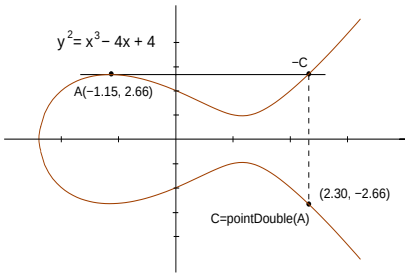


Fig 2. Elliptic curve *point doubling*.

Elliptic curves used for cryptography contain a finite set of points. Figure 3 shows an elliptic curve consisting of all points that satisfy $y^2 \bmod p = x^3 + ax + b \bmod p$ (for a = 4, b = 3, p = 23). While the algebraic equations for point addition and doubling still apply, one loses the nice geometrical visualization of these operations.

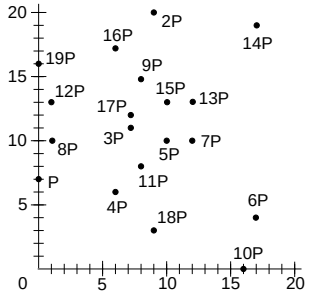


Fig 3. An elliptic curve over GF(p).

3. Scalar Point Multiplication

The main cryptographic operation in ECC is *scalar point multiplication* which computes Q = kP, a point P is multiplied by an integer k resulting in another point Q on the curve. Scalar multiplication is performed through a combination of *point additions* and *point doublings, e.g.* 11P = 2((2(2P)) + P) + P*.

Each curve has a specially designated point G called the *base point* chosen such that a large fraction of the elliptic curve points are multiples of it. To generate a key pair, one selects a random integer k which serves as the private key, and computes kG which serves as the corresponding public key.

** Here + is used to denote point addition*

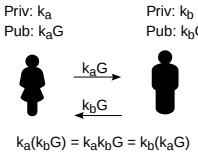
4. What makes ECC hard to crack?

The security of ECC relies on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP), *i.e.* finding k, given P and Q = kP. The problem is computationally intractable for large values of k.

Public-key system	Mathematical Problem	Best known method for solving (running time)
Integer factorization *e.g.* RSA	Given a number *n*, find its prime factors	Number field sieve: $\exp[1.923(\log n)^{1/3} (\log \log n)^{2/3}]$ (Sub-exponential)
Discrete logarithm *e.g.* DH, DSA	Given a prime *n*, and numbers *g* and *h*, find *x* such that $h = g^x \bmod n$	Number field sieve: $\exp[1.923(\log n)^{1/3} (\log \log n)^{2/3}]$ (Sub-exponential)
Elliptic curve Discrete logarithm *e.g.* ECDH,ECDSA	Given an elliptic curve and points P and Q find *k* such that Q = *k*P	Pollard-rho algorithm: sqrt(n) (Fully exponential)

Table 1. A comparison of public-key cryptosystems.

Among other things, this makes it possible for two entities to agree on a shared secret across an insecure communication channel without revealing that secret to an eavesdropper. This secret can then be used as a key to encrypt/decrypt sensitive information.



Each entity generates a key pair and sends its public key.

Each entity multiplies its private key with the other's public key to compute a shared secret.

5. Summary

Since the best known algorithm to attack ECC runs more slowly than the best known algorithm to attack other cryptosystems (see Table 1), ECC can offer equivalent security with substantially smaller key sizes. For example, a 160-bit ECC key provides the same level of security as a 1024-bit RSA key and 224-bit ECC is equivalent to 2048-bit RSA.

Smaller keys result in faster computations, lower power consumption, as well as memory and bandwidth savings. While these characteristics make ECC especially appealing for small embedded devices [1], they can also alleviate the computational burden on secure web servers [2].

References

[1] N. Gura *et al.*, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs", *CHES 2004*, Aug. 2004.

[2] V. Gupta *et al.*, "Speeding up Secure Web Transactions Using Elliptic Curve Cryptography", *NDSS 2004*, Feb. 2004.