Public-Key Cryptosystem for Mobile Environments: Using Elliptic Curve Cryptography

Hyung-Joon Kim
Dept. of Electrical and Computer Engineering
Stevens Institute of Technology, Hoboken, New Jersey

Abstract

Today, a large portion of Internet applications requires the exchange of sensitive data between servers and clients over the Internet. To securely transmit the sensitive data over the insecure Internet, a combination of symmetric and public-key cryptosystems has been commonly used. In particular, the security of public-key cryptography this paper is focused on is obtained by the difficulty of computing a mathematically hard problem whose computational complexity is mostly very high; in practice, its heavy computation becomes a significant factor in speed performance as key length grows. Remember the inevitable nature of cryptography: a key size granted today becomes no longer secure in the near future. Meanwhile, more and smaller portable devices and embedded systems with limited computing and power resources are being used to access to the Internet and involved in more security-sensitive transactions. In this circumstance, there is a clear need for a novel public-key cryptosystem, like Elliptic Curve Cryptography (ECC), that could offer equivalent security with smaller key sizes, which can be computed with less processing power, memory, and lower power consumption. There have been numerous researches on the ECC algorithm and its performance analysis, and implementations have been already made in some hardware and software by major vendors [1], [3]-[7]. This paper is an overview of public-key cryptography and the benefits of using Elliptic Curve Cryptography for mobile wireless security.

1. Introduction

Along with the success in popularization of PCs, internet technology has been rapidly deployed over the last decade, and a wide range of applications have maximized the usefulness of the Internet. As a result, e-commerce, online banking, stock trading, and remote access over the Internet have been already a part of our everyday tasks. In addition, nowadays, an increasing number of these applications are becoming available on portable devices such as PDAs, cellular phones, and smart phones, which have relatively limited processing power, memory, battery power, and bandwidth. Moreover, looking forward to a ubiquitous computing era, it is not surprising that all the home appliances and electronics would finally feature the functionality of such data communication over the Internet or satellite network. Given these trends, not only is the amount of sensitive data exchanged growing, but also the types of mobile client devices are being diverse in terms of platforms and computing resources. Like all other computing tasks, this could be a major concern in security, too.

To keep the confidentiality and integrity of the sensitive data exchanged between a server and a client, a combination of symmetric and public-key cryptosystems is integratively used for the encryption of the important data. In particular, public-key cryptography is used to encrypt a random session key with which symmetric cryptography encrypts a bulk message. Public-key cryptography is also used for digital signature in order to verify the sender and ensure nonrepudiation of the message. Public-key cryptography that this paper is focused on basically

gets its security from the difficulty of a mathematically hard problem with a high degree of computational complexity. Hence, a public-key algorithm involves intensive mathematical computations. This makes public-key cryptography slow: Some of the algorithms turn out impractical because they are simply too slow. This is why public-key cryptosystems are used only for key management and digital signature.

Now, as regards the future requirement of larger key size, there is a foreseeable challenge to traditional public-key cryptosystems like RSA, Diffie-Hellman, and DSA, which are most commonly used today. According to [1:52], "Doubling the size of the RSA key leads to an approximate eightfold increase in computation time as the computational effort grows proportionally to the cube of the key size." That means, as the key size grows, the computational overhead of the traditional public-key cryptosystems will become soon prohibitive for constrained portable devices that are unlikely to have enough computing resources. Therefore, with the explosive growth in the amount of security-sensitive transactions using mobile wireless devices, there is a clear need for an alternative to the traditional public-key cryptosystems. At the moment, Elliptic Curve Cryptography (ECC) could fulfill the demand for a new efficient public-key algorithm for mobile wireless environments. The reason is that ECC can offer equivalent security with smaller key sizes that can be computed with less processing power, memory, and lower power consumption.

Rather than going deep into the technical details of ECC, this paper will overview the followings: (i) public-key cryptography scheme, (ii) mathematical foundations of ECC, (iii) advantages of ECC, and (iv) current works and future opportunities of ECC.

2. Public-key Cryptography Scheme

Public-key cryptography (a.k.a. asymmetric cryptography) uses two different keys to encrypt and decrypt a message. The encryption key is called the public key, which is publicly distributed to and used by any sender to encrypt a message to a recipient associated with that public key. The decryption key is called the private key, which is kept secret and used by a recipient to decrypt the message encrypted by the recipient's public key. In public-key cryptosystem, therefore, anyone can encrypt and send a message to a recipient using the recipient's public-key whereas the only recipient who has his/her private key can decrypt that message. Another use of public-key cryptography is known as digital signature. Here, the private-key is used to encrypt a message, this is called signing, and the encrypted message can be decrypted by the public key. Digital signatures are used to verify a sender with two main characteristics of a signature: authentic and non-repudiable.

There is an analogy between a common mailing system and a public-key cryptosystem. Consider a mail box on a building. Anyone knowing the mail address of a person can send a mail to that person but only the intended recipient who possesses the key can open the mail box. In that case, the mail address is the public-key and the mailbox key is the private key. The mail address (public key) may be distributed person-to-person or by a public directory (Public Key Infrastructure). In addition, an official seal or an envelope with a pre-paid stamp printed on it, commonly used by corporations these days, is analogous to a digital signature in the public-key cryptosystem.

The security and performance of public-key cryptography has been debated against its historical competitor – symmetric cryptography (a.k.a. single-key or secret-key cryptography). However, it seems obsolete to make an argument about superiority over one and another. As regards this, there is a well-thought comment by Bruce Schneier:

> Public-key cryptography and symmetric cryptography are different sorts of animals, they solve different sorts of problems. Symmetric cryptography is best for encrypting data. … Public-key cryptography can do things that symmetric cryptography can't; it is best for key management and a myriad of protocols… [2:216].

Indeed, in today's most practical implementations, public-key cryptography is used to encrypt a random session key with which fast symmetric cryptography encrypt a bulk message, and/or used to digitally sign the encrypted message. This combinational use of public-key and symmetric cryptography forms a "hybrid cryptosystem" – adopting the virtues of both cryptosystems. Pubic-key cryptography is computationally more expensive (thus slower) than symmetric cryptography of equivalent security. Yet, public-key cryptography uses two separate keys for encryption and decryption, and the private-key for decryption is literally kept private and secure. Hence, public-key cryptography works well for key management while symmetric cryptography serves for encryption of a message. The reason that public-key cryptography requires intense computations is fundamentally related to its mathematical foundation.

3. Mathematics behind Elliptic Curve Cryptography

In essence, public-key cryptography gets security from a computational difficulty of solving a mathematically hard problem. Here, a "hard" problem should be computationally infeasible, at least in a reasonable amount of time, to any known cracking algorithm running on a set of supercomputers in a given time span. For that, public-key cryptography makes use of a *one-way function* – it's easy to compute one way, but significantly hard to solve it on the other way around. That is, given $f(x)$ it is easy to compute $f(x)$, but given $f(x)$ it is very hard to find its inverse function, $f^{-1}(x)$. Analogously, in public-key cryptography, given the encryption function, $E_k(M)$, where M is plaintext, it is easy to encrypt the message by $E_k(M)$, but given the encrypted message, $E_k(M) = C$, where C is ciphertext, it is extremely hard to derive the decryption function, $D_k(C)$, from $C = E_k(M)$. Thus, finding the inverse function of such one-way function is an analogy for crackers to extract the decryption key from an encrypted message, with the encryption key in hand. The characteristics of public-key cryptography can be summarized as shown below:

$E_k(M) = C$, where M is plaintext and therefore C is ciphertext,

$D_k(C) = M$,

Let $f(x) = E_k(M)$ and $f^{-1}(x) = D_k(C)$. Then, $f^{-1}(f(x)) = D_k(E_k(M)) = M$.

The subscript, k, in E_k and D_k denotes a public-key and a private key, respectively. Note that two keys are different even though they are denoted as the same, by k. The encryption and decryption functions, E_k and D_k, depend on the keys, which are randomly chosen from a large finite field, often called the keyspace. The one-way functions and keyspaces used in public-key cryptography have important implications for their security as well as different algorithms.

Different public-key algorithms choose different one-way functions to seek their security. For example, RSA, the most common public-key algorithm, is based on factoring large numbers; Diffie-Hellman, DSA, and ElGamal use a discrete logarithm problem, and Rabin is based on finding square roots modulo a composite number [2:475]. According to Whitfield Diffie [2:501], most public-key algorithms are based on one of three hard problems:

1. Knapsack: Given a set of unique numbers, find a subset whose sum is N.
2. Discrete logarithm: If p is a prime and g and M are integers, find x such that $g^x \equiv M \pmod{p}$.
3. Factoring: If N is the product of two primes, either
   a) factor N,
   b) given integers M and C, find d such that $M^d \equiv C \pmod{N}$,
   c) given integers e and C, find M such that $M^e \equiv C \pmod{N}$, or
   d) given an integer x, decide whether there exists an integer y such that $x \equiv y^2 \pmod{N}$.

In order to implement one of these hard problems, public-key cryptosystems extensively use modular arithmetic operations whose characteristics are communicative, associative, and distributive. For example, the fundamental operations in RSA and Diffie-Hellman are based on modular exponentiation, which can be efficiently done by recurring successive multiplications. For example,

$$a^{16} \bmod n = (((a^2 \bmod n)^2 \bmod n)^2 \bmod n)^2 \bmod n.$$

The implementations of such modular multiplications are straightforward despite the intensity of computations, and various cryptography libraries are widely available in these days. On the other hand, Elliptic Curve Cryptography (ECC) needs another arithmetic operation, so-called scalar point multiplication. Scalar point multiplication is not performed by monotonous operations like those for modular exponentiation in RSA. Instead, scalar point multiplication, the core arithmetic operation in ECC, can be efficiently computed by decomposing it into a series of point additions and point doublings. For example,

$$15P = (2 * ((2 * ((2 * P) + P)) + P)) + P.$$

ECC uses this scalar point multiplication to compute $Q = kP$. That is, a point, P, on an elliptic curve, multiplied by a large integer number, k, results in another point, Q, on the curve. In the ECC scheme, k is the private key, and Q is the public key, and P is an elliptic curve parameter called the base point. More details of ECC's arithmetic operations are introduced along with acceleration technique and "modular reduction" technique for ECC [1], [5]. Having all said that, the security of ECC is established by the difficulty of a discrete logarithm problem – considered equivalent to reverse modular exponentiation – over the points on the elliptic curve.

All the hard problems herein mentioned are assumed very hard for a large range of numbers. For this reason, a pair of public-key and private-key each of which is an arbitrary large number is chosen from sufficiently large finite field, depending on the algorithms. For the keyspace, public-key algorithms uses the finite number groups defined in abstract algebra: the multiplicative group of prime fields, denoted by $GF(p)$ or Φ_p, and the multiplicative group of binary polynomial fields, denoted by $GF(2^m)$ or $\Phi_2{}^m$. The ECC algorithm is particularly interested in the *points* on elliptic curves defined over *both* finite fields, and the set of such points is called the elliptic curve group over finite fields, distinctly denoted by $EC(\Phi)$.

The details of underlying mathematics that employs one-way function and number theory are very complex and therefore beyond the scope of this paper. Instead, the above discussions aim at appreciating the essentials of public-key cryptography and the ECC scheme.

4. Advantages of Elliptic Curve Cryptography

The core idea of the ECC algorithm lies in the keyspace of elliptic curve groups over finite fields, $EC(\Phi)$. By mathematicians and cryptologists, a discrete logarithm problem on elliptic curve group over finite fields, $EC(\Phi)$, is assumed to be more difficult than a corresponding problem in the underlying finite fields, such as the multiplicative group of prime fields, $GF(p)$, or that of binary polynomial fields, $GF(2^m)$, which is used by most traditional public-key cryptosystems. The assumption implies that a smaller key can be sufficient for ECC to offer an equivalent level of security provided with the keys chosen from $GF(p)$ or $GF(2^m)$. This idea gives ECC a prospective advantage over its predecessors such as RSA, Diffie-Hellman, and DSA. In fact, the comparison of key sizes for equivalent security levels, Table 1, is found in [3], [4].

Symmetric (AES)	ECC	RSA/DH/DSA
80	163	1024
128	283	3072
192	409	7680
256	571	15360

Table 1. Computationally equivalent key sizes (in bits).

Remember the inevitable nature of cryptography: a key size granted today becomes no longer secure in the near future. As computing power has been doubling every 18 months by Moore's law [2:153], the accordingly different lengths of both symmetric and public-key are required for equivalent security during different years:

> The security of a system is only as good as that of its weakest component; for this reason, the work factor needed to break a symmetric [cryptosystem] must match that needed to break the public-key cryptosystem used for [the random session key]. Due to expected advances in cryptanalysis and increases in computing power available to an adversary, both symmetric and public-key sizes must grow over time to offer acceptable security for a fixed protection life span… [3:87].

The Advanced Encryption Standard (AES), today's most common symmetric cryptosystem, requires the minimum of 128-bit, and 192-bit and 256-bit keys for enhanced security. According to Table 1, the traditional public-key cryptosystems such as RSA, Diffie-Hellman, and DSA need the key size of 3072-bit, 7680-bit, and 15360-bit keys, respectively, to balance the security levels of the corresponding symmetric cryptosystem. The increase of the symmetric key length is quite affordable, but that of the public-key key length makes the intense computations of the public-key cryptosystem too costly. Meanwhile, more and smaller portable devices and embedded computers with limited computing and power resources are being used to access to the Internet and involved in security-sensitive transactions. In this circumstance, ECC provides an alternative solution to the need of a cheaper and more efficient public-key algorithm; for example, ECC needs only 283-bit, 409-bit, 571-bit keys for equivalent security levels using RSA's 3072-bit, 7680-bit, and 15360-bit keys, respectively. A major advantage of ECC comes from those smaller key sizes because they can be computed with less processing power, memory, and lower power consumption.

The performance advantage of ECC will spectacularly increase as higher security levels are demanded, and the computational benefit from ECC is highlighted on the narrow data path in microprocessors [1:58]. An ultimate question to any cryptosystem is probably how immune it is against potential attacks. So far, little or no successful attacks are known for ECC with carefully selected curves; in compassion, there exist today much more efficient algorithms to attack RSA and others [1:53], [4:63].

5. Current and Future Works

Although ECC is first proposed about 20 years ago, independently by Victor Miller and Neal Koblitz, ECC was once considered as fairly new cryptography in regard of its standardization and application. In recent years, however, the high demand for more efficient public-key cryptography for mobile wireless networks has spurred the adoption of ECC as a next-generation public-key cryptosystem. As a result, a number of groups have put tremendous efforts into implementing ECC efficiently in both hardware and software and deploying the ECC technology in practical internet applications. For instance, a research group at Sun Microsystems introduced acceleration techniques using "dual-field multiplier" and "multiply-accumulate instruction" in order to gain speedup of ECC, and found the significant performance results in microprocessors [1]. A subgroup of the above group proposed an elegant algorithm using "partial reduction" that allows generic implementations in hardware and software for arbitrary curves over $GF(2^m)$ with variable field degrees m [5]. In the Internet industry, ECC technology has been integrated into many notable Internet applications including OpenSSL, Apache, Netscape Security Services (NSS) and Mozilla [3], [6]. Besides, mobile wireless industry leaders such as Qualcomm, Motorola, Docomo, and RIM employed ECC as well as major computer companies such as IBM, Sun Microsystems, Microsoft, and Hewlett-Packard put their efforts into ECC [4]. Recently, Microsoft's Windows Vista equipped ECC capability, as a key security feature, within "Suite B", which is a complete coordinated suite of encryption, message digest, key agreement, and digital signatures, announced by NSA for future US government use [7].

Since the security of ECC heavily relies on the carefully-chosen curves that would raise the adversary's work exponentially, a poor selection of the curves could possibly compromise its security. Thus, ongoing researches and standards involve the recommendation of selecting the curves. For compatibility and support for newly-defined curves, generic curves as well as reconfigurable hardware should be also efficiently implemented in the future. Lastly, as always, ECC leaves room for improvement in implementation reducing even more computational complexity.

6. Conclusion

The public-key cryptosystem has been efficient to settle both key management and digital signatures problem for a variety of cryptography schemes. With the striking development of wireless technology, however, an increasing number of smaller portable devices are connecting to the Internet network, and the amount of exchanging sensitive data are also remarkably growing. With these trends, the expensive computational costs of traditional public-key cryptosystems are unlikely affordable by the constrained portables, especially when the key length is required to double in every few years in order to offer equivalent security. Opportunely, elliptic curve cryptography has adopted as an attractive alternative for the next-generation public-key cryptography. The reason is that ECC benefits from its low computational complexity as well as

trustworthy strength of security. Still, many parties are actively working on accelerating techniques for ECC, recommendatory sets of the curves as well as support for generic curves, and implementing elliptic curve variants of existing public-key cryptography.

References

[1] Hans Eberle, Sheueling Shantz, Vipul Gupta, Nils Gura, Leonard Rarick, and Lawrence Spracklen, "Accelerating Next-Generation Public-Key Cryptosystems on General-Purpose CPUs," *IEEE Micro*, IEEE Computer Society Press, 2005, pp. 52-59.

[2] Bruce Schneier, *Applied Cryptography*, Second Edition, John Wiley & Sons, 1996.

[3] Vipul Gupta, Sumit Gupta, Sheueling Chang, and Douglas Stebila, "Performance Analysis of Elliptic Curve Cryptography for SSL," *Proceedings of the 3rd ACM Workshop on Wireless Security WiSE `02*, ACM Press, 2002, pp. 87-94.

[4] Kristin Lauter, "The Advantages of Elliptic Curve Cryptography for Wireless Security," *IEEE Wireless Communications*, IEEE Communications Society Press, 2004, pp. 62-67.

[5] Nils Gura, Hans Eberle, and Sheueling Chang Shantz, "Generic Implementations of Elliptic Curve Cryptography using Partial Reduction," *Proceedings of the 9th ACM Conference on Computer and Communications Security CCS '02*, ACM Press, 2002, pp. 108-116.

[6] Vipul Gupta, Douglas Stebila, and Sheueling Chang Shantz, "Integrating Elliptic Curve Cryptography into the Web's Security Infrastructure," *Proceedings of the 13th International World Wide Web Conference*, ACM Press, 2004, pp. 402-403.

[7] Brian A. LaMacchia, John L. Manferdelli, "New Vistas in Elliptic Curve Cryptography," *Information Security Technical Report*, Vol. 11, No. 4, Elsevier Ltd., 2006, pp. 186-192.