Using Elliptic Curve Cryptography to Secure Online Data & Content

Kefa Rabah
Center for Advance Research in Cryptography & Cyberscurity (CAREC)
Global Open Versity, Vancouver, BC Canada
URL: www.globalopenversity.org Email: krabah@globalopenversity.org

Abstract: The rapid growth in data communication across the cyberspace has given rise to increase in data vulnerability and possibilities to many types of attack by the man-in-the-middle, especially with mobile and wireless network devices powered by 3G and soon 4G. That said, data security has become a challenging aspect of communication today that touches many areas including network access, memory space and constraint environment, processing speed, code development and maintenance issues. When it comes to dealing with lightweight computing devices, each of these problems is amplified in an attempt to address some of these problems. In this paper we will study Elliptic Curve ElGamal (ECEG) cryptosystems and signature scheme. The ElGamal signature algorithm is similar to the encryption algorithm in that the public-key and private-key have the same form; however, encryption is not the same as signature verification, nor is decryption the same as signature creation as in RSA. The DSA is based in part on the ElGamal signature algorithm. As an example we will implement ECC defined over F_p applied to ElGamal cryptosystems.

Keywords: Elliptic curve cryptosystems, ElGamal cryptosystems, Digital signature, online content management, RSA, AES, Digital Certificate, 3DES, Wireless devices, Internet security.

1.0 INTRODUCTION

In the computing age of today, we have witnessed the growing popularity of the Internet and networks in our society. With these tools at our fingertips, we are able to communicate and do business even more quickly and efficiently than ever before – ala anytime anywhere access to data. For example, businesses can market their products online so customers do not have to leave their homes, and banks can conduct transfers and manage accounts with more ease, speed, and functionality than with the paperwork of the past. Also, what is probably the most popular means of communication, email, is used by just about everyone each and every day – anytime anywhere access. The coming of age of the powerful 3G and the 4G hotting up on its heals – the headache and challenges of securing online digital contents have become extremely complex and will get worse as technology continues to expand beyond the 4G powered mobile devices and their counterpart wireless devices . It is clear that these modern conveniences have made our lives much smoother. However, as we continue to add these conveniences to our lives, we open the door to more numerous, possibly even more dangerous, outlets for attacks. With the prominence of identity theft on the rise, we must all be weary of the security of online communication. One solution is to “hide” our data by using encryption algorithms that only allow those that we trust and/or exchanged cryptographic session keys with, to view the information. We can also verify the integrity of our data through electronic signatures and electronic certificates – this is the heart of cryptography **[1-5]**.

Cryptography offers a set of sophisticated security tools for a variety of problems, from protecting data secrecy, through authenticating information and parties, to more complex multi-party security implementation. Cryptography blends several areas of mathematics: number theory, complexity theory, information theory, probability theory, abstract algebra, and formal analysis, among others **[3]**. With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, which is widely known, but on a number called a *key* that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Currently, there are two popular kinds of cryptographic protocols: *symmetric-key* and *asymmetric-key* protocols **[1,4]**. In the symmetric-key protocols, a common key (the secret-key) is used by both communicating partners to encrypt and decrypt messages. Among these are DES, IDEA and in recent years the Advance Encryption

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

Standard (AES) **[5]**. These symmetric-key cryptosystems provide high-speed key and communication but have the drawback that a common (or session) key must be established for each pair of participants. The process of exchanging the cryptographic key is referred to as *key distribution* and can be very difficult!

2.0 Asymmetric Cryptography – *A Historical Background*

Prior to 1970, symmetric-key had been the crypto-mode in existence. By early 1970s, however, a new concept of encryption known as public-key cryptosystems began to emerge. It was from entirely unexpected source; back in the fall of 1974, Ralph Merkle then a senior undergraduate student settled to pursuer his project on cryptography, which he not only successfully completed, but also invented the idea of public-key crypto-algorithm. Today, Dr. Merkle is a Distinguished Professor of Computing at Georgia Tech's Information Security Center. Merkle's work – which originated around questions of how to recover security in a compromised system, a la secure communication over insecure network – led him to develop the Puzzles Method, a technique that showed it was possible to create problems of controllable difficulty using puzzles. Contained within this realization were the seeds of public-key cryptography: the idea that puzzles – or keys – could be publicly known while the contents of the information exchanged they relate to would remain essentially secure **[1]**. Merkle at the beginning found little support for his ideas and their application to cryptography. However, at around the same time Diffie and Hellman, whom he later collaborated with, were separately also toying with the idea of public-key cryptosystems. It was not until 1976, after the publication of Diffie-Hellman paper, New Direction in Cryptography **[6]**, that the concept of public-key cryptosystems became a reality. In the public-key protocols we have two associated keys; one is kept private by the owner and used either for decryption (*confidentiality*) or encryption (*signature*) of messages. The other key is published in the public domain (public-key server) to be used for the reverse operation or decryption. Different public-key cryptographic systems are used to provide public-key security.

Two basic types of public-key schemes emerged in the mid 1970s; Diffie-Hellman (DH) for key agreement protocol proposed in 1975 which relies on the hardness of the discrete logarithm problem (DLP): given p, g and g^a, find a **[7]**. Two years later Rivest, Shamir and Alderman proposed the key transport and digital signature schemes known by their initials as RSA, which takes it security from the hardness of the integer factorization problem (IFP): given a composite number $n(= p \cdot q)$, find the primes p and q **[8,9]**. Despite the power of the "hardness" of mathematics behind the public-key crypto-algorithms that are capable of withstanding "brute-force" attacks it, however, downed on the cryptographers and researchers at the time that securing data in the real world using these crypto-algorithms required clever design, implementations and a clear knowledge of the kind of threat models to be expected from a determined and powerful adversary possible. That is, a protocol could be deemed secure if an ultra-potent adversary could not achieve his or her ultra-weak goals – a concept known in the crypto-jargon as provable security – which involves specific computational assumptions that can be used to prove that a proposed protocol meets the requirements of its security definition **[1]**. It wasn't until the mid-90s that provable secure protocols actually began to be efficient enough to warrant real world applications – a prospect reinforced by the work of Taher ElGamal and his ultimate proposal of the ElGamal public-key crypto-algorithm for encryption and signature schemes that began to emerge in 1984 to rival those of RSA. Unlike the RSA which was based on IFP, the ElGamal crypto-algorithms were based on the DLP **[10]**. So too was the elliptic curve cryptography (ECC), which appeared on the scene in 1985 **[11]**, both of which are the subject of this paper.

3.0 Elliptic Curve Cryptography (ECC)

Elliptic curves are mathematical constructions from number theory and algebraic/geometric entities that have been studied extensively for the past 150 years, and from these studies has emerged rich theory, which in recent years have found numerous applications in cryptography and integer factorization **[3]**. It

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

was Lenstra in his 1984 manuscript on integer factorization in which he indicated that elliptic curves could be used to attack cryptosystems **[12]**. Thereafter, Koblitz and Miller separately wondered if elliptic curves in this respect might not also form the basis of a potential cryptosystems – and there were sound mathematical reasons for this conjecture to back it up. In one aspect the ECC were to be implemented under finite field, and there is only ever one multiplicative group for a given prime.

In 1985, Dr. Victor Miller – then at IBM Research Lab at Yorktown Heights **[13]** and, Dr. Neil Koblitz at the University of Washington **[14]**, independently invented elliptic curve cryptography (ECC). It is a new branch in cryptography that uses an old, interesting and difficult topic in mathematics or, particularly, algebra: elliptic curves over finite fields. The elliptic curve approach is a mathematically richer procedure than traditional cryptosystems e.g., RSA, DH, ElGamal, DSA etc. The ECC from the very beginning was proposed as an alternative to established public-key systems such as the conventional cryptosystems. This is because elliptic curves do not introduce new cryptographic algorithms, but they implement existing public-key algorithms using elliptic curves **[15]**. In this way, variants of existing schemes can be devised that rely for their security on a different underlying hard problem, e.g., ECDH, EC ElGamal, ECDSA, ECDLP etc **[11,12]**.

Variant public-key cryptosystems provide crypto-services by relying on the difficulty of different classical mathematical problems, hence provide the services in different ways, and therefore, each algorithm have their own advantages and trade-offs. The efficiency of a crypto-algorithm depends on the key length and the calculation effort that is necessary to provide a prescribed level of security. The US National Institute of Standards in Technology (NIST) recommends that the security level of key management should match the level of the bulk cipher – which is equivalent to an implementation of AES with a 256-bit – currently recommended for classified US-government communications **[5,12]**. Under these circumstances one has option to use 512-bit ECC, 15360-bit RSA, or 15360-bit DH. If we assume that the computing power increases by Moore's law (i.e., it doubles every 18-months), then the evolution of the key lengths for secure communication will be as shown in **Fig. 1**. For constrained devices, however, one must take great care in choosing which protocol to implement as large keys are known to have great impact on the performance levels of wireless devices.

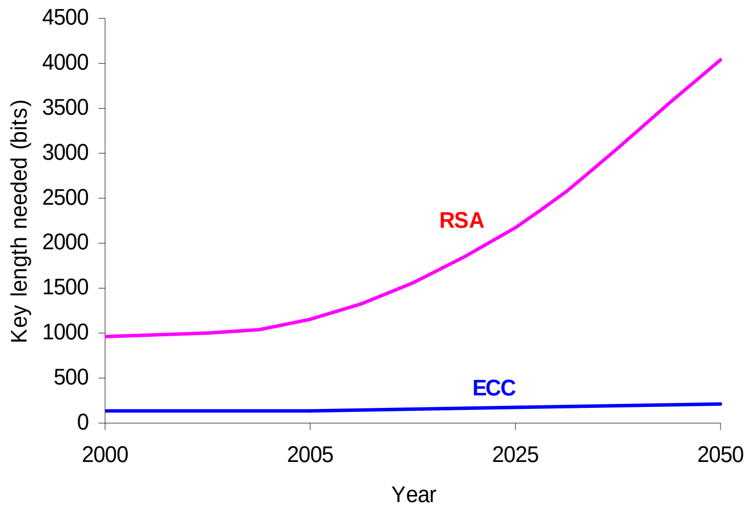


Fig. 1 Proposed the minimum key sizes (in bits) to be regarded as safe for RSA and ECC (*After* Lenstra *et. al*., **[12]**).

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

Today, 1024-bit RSA keys are standard, but the German Information Security Agency (GISA) recommends the usage of 2048-bit keys from 2006 on. The fact that most application specific integrated chips (ASICs) on smartcards and most wireless devices cannot process keys extending 1024-bit shows that there is need for alternatives of which ECC can be such an alternative. **Table 1** show typical performance levels of a constrained device using ECC, RSA and DH crypto-algorithms, for key generation, encrypt/verify (with public-key) and decrypt/sign (with private-key). We can observe that in general, ECC had the best timings for a general purpose crypto-algorithm. However, RSA is good for situations where only public-key verification is required, and is backed by only one way to implement it as an added advantage. Nevertheless, constrained devices like smartcards usually have to store the (long) secret key onboard and have to process a digital signature rather than verify one. Hence, there is a clear advantage in using ECC in terms of efficiency. Currently, the major problem still dogging the ECC-implementation is the lack of standardization. However, with ECC one can work with different sets of numbers, different (elliptic) curves, and a variety of representations of elements on the curve with consequent advantages and disadvantages – and one can certainly construct the most efficient for each application. Moreover, there are many elliptic curves of varying sizes for any given prime which is a great advantage in favor of ECC. A further positive change in favor of ECC is the involvement of major standardization organizations like: ASC (ANSI X9.62, X9.63)**,** IEEE (P1363), ISO/IEC 15946, RSA labs, and Certicom **[16-21]**.

Table 1: Comparisons of ECC vs. RSA vs. DH*.

	ECC (256 R1)	RSA (3072)	DH (3072)
Key Generation	166 ms	N/A**	38 s
Encrypt/Verify	150 ms	52 ms	74 s
Decrypt/Sign	168 ms	8 s	74 s

*Timings were taken on BlackBerry 7230 at 128-bit security level. Timings at a 256-bit security level would show even greater differences between ECC, RSA and DH. **Time was too long to measure **[22]**.

The main advantage of elliptic curve cryptosystems (ECC) is that they use smaller parameters (e.g., encryption key) than the traditional crypto-algorithms, as can be observed in Fig. 1 and Table 1. The reason is that the underlying mathematical problem on which their security is based, the EC discrete logarithm problem (ECDLP), appears to require more time to solve than the analogous problem in groups generated by prime numbers on which the traditional cryptosystems are based **[3]**. For groups defined on ECs (where the group elements are points on the EC), the best algorithm for attacking this problem takes time exponential in the size of the group, while for groups generated by prime numbers there are algorithms that take subexponential time. This implies that one may use smaller parameters for the ECC than the parameters used in RSA or DSA, obtaining the same level of security. A typical example is that a 160-bit key of an ECC is equivalent to RSA and DSA with a 1024-bit modulus. As a consequence, smaller keys result in faster implementations, less storage space, as well as reduced processing and bandwidth requirements. ECC has also shown itself as extremely well suited to the unique demands of the wireless communications environment where bandwidth is at a premium and device power and processing resources are constrained. For example, an Application Specific Integrated Circuit (ASIC) built for performing elliptic curve operations over the field $\mathbb{F}_{2^m}$ **[23,24]** has only 12,000 gates and would occupy less than 5% of the area typically designated for a smartcard processor, which compares with 50,000 gates for 512-bit numbers, and 90,000 gates for 593-bit numbers.

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

Today ECC has evolved into a mature public-key cryptosystem. The U.S. government recently endorsed it as an alternative public key algorithm **[25]**. Meanwhile, in September 2004, NIST instituted a validation system for the Elliptic Curve Digital Algorithm (ECDSA) as approved in FIPS 186-2. This is a significant achievement for an ECC-based algorithm, as it is the first validation system awarded to it. The implication is that now the accredited third parties can test vendor implementations of ECDSA, thereby ensuring proper security and interoperability. This should ultimately encourage further adoption of ECDSA. In this respect the future of ECC today and its potential for the future looks good and it is a technology which is here stay – and may become the crypto-algorithm of choice in the not so distance future. In this work we will concentrate on elliptic curve cryptography focusing on EC-ElGamal crypto-schemes.

4.0 Design and Implementation of ECC

There are many important decisions that one should make before starting to implement an elliptic curve cryptosystems (ECC). These include the type of the underlying finite field, the algorithms for implementing the basic algebraic operations, the elliptic curve to be used as well as its generation, and finally the elliptic curve protocols. The fields usually used are either prime field (denoted by $\mathbb{F}_p$, where p is prime) or binary fields, $\mathbb{F}_{2^m}$. Selecting a prime field (which will be our concern here) implies proper choice of p, since all basic operations will be modulo that prime and the security of the system will depend on its size. The larger the prime, the more secure but slower the cryptosystem.

The basic algebraic operations of the EC group that must be implemented are addition of points on an EC, and scalar multiplication (multiplication of a point by an integer). The latter is the operation upon which the ECDLP is based **[26]**. There should also be a way for generating secure elliptic curves and create random points on them. The generation of such curves can be accomplished with three methods, namely: the point counting method, Schoof's algorithm **[27]**; the method based on constructive Weil descent and; the Complex Multiplication (CM) method **[4]**. The latter two methods build ECs of *suitable order* (where order refers to the number of elements in the group defined by the EC), i.e., the order satisfies certain conditions necessary for the cryptographic strength of the EC. The former method does not necessarily produce ECs of a suitable order. Once all these basic operations have been implemented, one can start developing cryptographic protocol of choice.

4.1 ECC Arithmetic Over Finite Field

The use of elliptic curve groups over finite fields as a basis for a cryptosystem was first suggested by Koblitz, **[3,14]**. An elliptic curve can be defined over any field (e.g., real, rational, complex). However, elliptic curves used in cryptography are mainly defined over finite fields **[9,15,26]**. Finite fields, also called Galois fields, are fields consisting of a finite number of elements. The cost, speed and feasibility of elliptic curve cryptosystems depend on the finite field $\mathbb{F}_q$, where $q = p^m$, on which it is implemented. There are usually two finite fields to work on: prime finite field $\mathbb{F}_p$ (i.e., $m = 1$) when p is a prime number > 3 and, binary finite field $GF(2^m)$ or $\mathbb{F}_{2^m}$.

An elliptic curve $E(\mathbb{F}_p)$ consists of elements (x, y) of the form:

$$E: \quad y^2 = x^3 + ax + b \quad \text{with} \quad x, y, a, b \in \mathbb{F}_p = \{1, 2, 3, \ldots, p-2, p-1\} \qquad (1)$$

together with a single element denoted by O called the "*point at infinity*". Abstractly, a finite field consists of a finite set of objects called *field elements* together with the description of two operations – *addition*,

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

multiplication – that can be performed on pairs of field elements. These operations must possess certain properties: the addition operations (which is the counterpart to modular multiplication in traditional public-key cryptosystems) and; multiplication or multiple additions (which is the counterpart to modular exponentiation in the conventional cryptosystems) **[4]**. The *order*, n, of a finite field is the number of elements in the field. There exists a finite field of order q if and only if q is a prime power, then there are, however, many efficient implementations of the field arithmetic in hardware or in software **[27]**. Now given a message, m, we must first choose a large integer, k in the range $[1, n-1]$, and a suitable elliptic curve, $E(\mathbb{F}_p)$ defined as above. We must then embed the message m onto a point, P, on the curve. This is not as straightforward as it looks and involves the use of quadratic residues and probabilistic algorithms. In the next sections we are going to develop the methodology to achieve that task.

4.2 Construction of the ECC Arithmetic Over Finite Field

The core of the ECC is when it is used with Galois Field it becomes a one way function i.e., the math's needed to compute the inverse is not known. Let an elliptic curve group over the Galois Field $E_p(a,b)$ where $p > 3$ and is prime, be the set of solutions or points $P = (x, y)$ such that $(x, y \in E_p(a,b))$ that satisfy the equation: $y^2 = x^3 + ax + b \pmod p$ for $0 \le x < p$ together with the extra point O called the point at infinity. The constants a and b are non-negative integers smaller than the prime number p and must satisfy the condition: $4a^3 + 27b^2 \neq 0 \pmod p$. For each value of x, one needs to determine whether or not it is a *quadratic residue*. If it is the case, then there are two values in the elliptic group. If not, then the point is not in the elliptic group $E_p(a,b)$. The number of points on $E_p(a,b)$ is denoted by $\#E(F_p)$. Since 50% of integers $\bmod\, p$ are quadratic residues, the number of points will be roughly $p+1$, counting the point at infinity. Hasse's theorem states that the number of points on an elliptic curve (including the point at infinity) is $\#E(\mathbb{F}_q) = p + 1 - t$ where $|t| \le 2\sqrt{p}$; Here, t is called the trace of E and, $\#E(\mathbb{F}_p)$ is called the *order* of an elliptic curve E **[11]**. In other words, the order of an elliptic curve $E(\mathbb{F}_q)$ is roughly equal to the size p of the underlying field. In fact, the general theory says that there will be about p points (x, y) with error bounded by $O(\sqrt{p})$. The order n of a point $P \neq O$ on an elliptic curve is a positive integer such that $nP = O$ and $mP \neq O$ for any integer m such that $1 \le m < n$. The order n of a point must divide the order N of the elliptic curve. In fact, it is true for any group. If the elliptic curve order $N = \#E(\mathbb{F}_q)$ is a prime number, then the group is cyclic, and obviously all points except the point at infinity O are of order N.

The order of the elliptic curve can be determined by Schoof's algorithms or its variants, which is required if one chooses a random curve **[11]**. Random selection of a and b to determine the curve order gives a large prime factor which can improve the security of the crypto-algorithm scheme. Alternative one can choose a point P of a large prime order n: A simple method is usually applied in cryptographic practices when n is a large prime. Then the factor $\ell = \#E(\mathbb{F}_q)/n$ will not be divisible by n. Choose a random point $Q \neq O$ on the elliptic curve E, then verify whether the point $P = \ell Q$ has order n. This can be done simply by checking that $nP = O$. (Since n is prime, there is no other positive integer $m < n$ such that $mP = O$) If it is true, then $P = \ell Q$ is the point we need; otherwise, choose another point Q and repeat. This is the technique deployed by D. Shank in the Shank's baby step/giant step (BSGS) algorithm **[29]**.

The order of the group is known to all parties; we can either generate a curve at random and counts its order (Schoof's algorithm) **[28]**, choose an order and use a constructive algorithm to derive a curve (method of complex multiplication is most common). In this work will use the naïve approach and also

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

elliptic curve builder (ECB), a free open source software **[30]**. Here we will design systems to use prime order group (sub group) of points on the elliptic curve.

4.3 Elliptic Curve Point Addition Algorithm

The basic condition for any cryptosystem is that the system is closed, i.e., any operation on an element of the system results in another element of the system. In order to satisfy this condition for elliptic curves it is necessary to construct nonstandard addition and multiplication operations. Capitals represent points on the curve while lower case represents integers. The addition of two points on an elliptic curve is defined in order that the addition results will be another point on the curve as presented in Algorithm 1.

```
Algorithm 1. Point Addition Equation
Input: $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$
Output: $P_1 + P_2 = P_3 = (x_3, y_3)$
1. If $P_1 = P_2$:      $x_3 = \lambda^2 - x_1 - x_2$,      $y_3 = \lambda(x_1 - x_3) - y_1$
        where $\lambda = (3x_1^2 + a)/2y_1$      (Point doubling)
2. Else if $P_1 \neq P_2$:      $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$,      $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$
        where $\lambda = (y_2 - y_1)/(x_2 - x_1)$      (Point addition)
3. Return: $(x_3, y_3)$
```

In either case, when $P_1 = P_2$ (doubling) and $P_1 \neq P_2$ (point addition), major operations are field multiplication and field inversion. (Squaring and field addition are enough ignorable because of its less computation time.) From these formulas of Algorithm 1, we can determine the number of field operations required for each kind of elliptic curve operation. We see that in affine coordinates, point addition step usually requires 6 addition/subtraction operations, three modular multiplications, and one inversion. A Doubling step usually requires 7 addition/subtraction operations, four modular multiplications, one squaring, and one inversion. A Negation step requires one addition.

Observe from above algorithm that the addition of two elliptic curve points in $E(\mathbb{F}_p)$ requires a few arithmetic operations (addition, subtraction, multiplication, and inversion) in the underlying field $\mathbb{F}_p$. The most basic operation is adding two points or doubling a point on an elliptic curve. It is more expensive computationally than a basic operation in a symmetric-key cryptosystem (a block encryption/decryption). But it is still much faster than a basic modular multiplication over a cyclic group whose order is of the same security level **[4]**. The methods, which included subtractions, are more attractive than the corresponding methods, which included divisions in calculating power in finite fields. The reason is division or inversion in finite fields is a more costly operation than multiplication, while subtraction is just as costly as addition in elliptic curve operations.

In the development and implementation of elliptic curve cryptography we are interested in the method for computing an equation of the form $m \cdot P$ where $m \in \mathbb{N}_{>1}$, and let $P \in E(\mathbb{F}_q)$ be a non zero point on some given elliptic curve *E*. For a positive integer m we let $[m]$ denote the *multiplication-by-m* map from the curve to itself. This map takes a point P to $P + P + \cdots + P$ (m summands). The notation $[m]$ is extended to $m \leq 0$ by defining $[0]P = O$, and $[-m]P = -([m]P)$. So for instance, as above, $[2]P = P + P$,

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

$[3]P = P + P + P$, and $[-3]P = -(P + P + P)$. This map is the basis of elliptic curve cryptography. Its properties, computation and uses will be, therefore, the core of this paper.

5.0 Security of an Elliptic Curve Over $\mathbb{F}_p$

Many of the security properties of elliptic curve cryptosystems depend on the order of the EC group and this is determined by generated EC. If this order is suitable, i.e., it obeys good properties, then there is guarantee for a high level of security. The order, n, of an EC is called suitable, if the following conditions are satisfied: (i) n must have a sufficiently large prime factor (greater that 2^{160}), (ii) $n \neq p$ and (iii) for all $1 \leq k \leq 20$, it should hold that $p^k \neq 1(\bmod n)$. Essentially, the only elliptic curves for which extension degree κ of $\mathbb{F}_{p^\kappa}$ is small are the so called "supersingular" elliptic curves, the most familiar examples of which are curves of the form: $y^2 = x^3 + ax$ when the characteristic p of $\mathbb{F}_p$ is $\equiv -1(\bmod 4)$ and curves of the form $y^2 = x^3 + b$ when $p \equiv -1(\bmod 3)$. The vast majority of elliptic curves, however, are non supersingular. These conditions ensure the robustness of cryptosystems based on the discrete logarithm problem for EC groups (ECDLP), since it is very difficult for all known attacks to solve this problem efficiently, if n obeys the above properties **[27]**. ECDLP asks for determining the value of k when two points $P, Q \in E(\mathbb{F}_p)$ are given such that P is of order n and $Q = kP$, where $1 \leq k \leq n-1$ and, P is the generator of the EC group and P has order of 160-bit or more. To the best of our knowledge, the ECDLP on E is *really hard* unless (i) $\#E(\mathbb{Z}/p\mathbb{Z})$ is "smooth", i.e., a product of small primes, because Pohlig-Hellman method **[31]** reduces the computation of k to problem of computing k modulo each prime factor of n; or (ii) E is "supersingular" in the sense that $p \mid \#E(\mathbb{Z}/p\mathbb{Z})$ or from Hasse's theorem, $\#E(\mathbb{F}_q) = q + 1 - t$, the trace $t = 0$. Thus, the key advantage of elliptic curve cryptosystems is that no subexponential algorithm is known that breaks the systems, provided that we avoid supersingular curves whose order has no large prime factor, while the DLP in $\mathbb{F}_q^*$ can be solved at most within a subexponential time **[27]**. To date the most efficient general algorithm to resolve the ECDLP is Pollard-ρ **[32]**, which takes about has the running time $\sqrt{\pi n/2}$ steps, where a *steps* here is the number of elliptic curve addition. In 1993, Oorschot *et al.*, **[33]** showed how the Pollard-ρ 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 found is $(\sqrt{\pi n/2})/r$, where n is the number of elliptic curve additions and r is the number of parallel processors running.

The computational complexity for breaking the elliptic curve cryptosystems using Pollard-ρ method is 3.8×10^{10} MIPS-years (i.e., millions of instructions per second times the required number of years) for an elliptic curve key size of only 150-bit **[34]**. For comparison, the fastest method to break RSA, using the *General Number Field Sieve* (GNFS) method to factor the composite integer $n(= p \cdot q)$ into the two primes p and q, requires 2×10^8 MIPS-years for a 768-bit RSA key and 3×10^{11} MIPS-years with a RSA key length 1024. In the RSA key is increased to 2048-bit, the GNFS-method will need 3×10^{20} MIPS-years to factor n whereas the elliptic curve key length of only 234-bit will impose a computational complexity of 1.6×10^{28} MIPS-years with Pollard-ρ method.

As an example to help us understand the magnitude of MIPS (Million Instruction Per Second), let us take a conservative estimate that an ASIC machine running at 40 MHz clock-rate is capable of performing

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

roughly 40,000 elliptic curve additions per second **[35]**. Then the number of elliptic curve additions that can be performed by a 1 MIPS machine in one year is: $(4\times10^4)\cdot(60\times60\times60\times365)\approx2^{40}$, see Fig. 2 for comparison of MIPS-year between RSA/DSA and ECC. In general, the key size can be assumed identical to the modulus size for each system. The total size of the system parameter and key pairs for RSA and DSA are much larger than ECC. They currently differ by a factor of four, which will become larger as the acceptable security level increases in future.

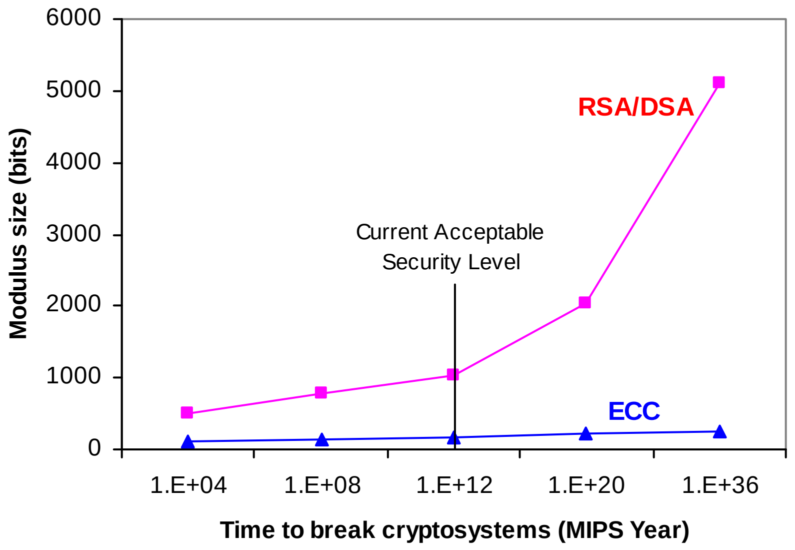


Fig. 2 Comparison of security levels of ECC and RSA/DSA.

Since the ECC key sizes are so much shorter than comparable RSA/DSA keys, the length of the public-key and private-key is much shorter in elliptic curve. This shorter key lengths for ECC translates into faster processing, and lower demand and bandwidth which makes ECC particularly useful in applications where memory, bandwidth, and/or computational power is limited (e.g., smartcards) and it is this area that ECC use is expected to grow.

6.0 The Cost of Elliptic Arithmetic Operations

Just as modular exponentiation determines the efficiency of RSA cryptographic systems **[9]**, scalar multiplication dominates the execution time of ECC systems. In all the protocols that are fundamental implementation of ECC, say ECDH, ECElGamal, ECDSA, ECAES etc., the most time consuming part of the computations are scalar multiplications. Hence, the dominant cost operation in elliptic curve cryptographic schemes is *scalar point multiplication*, namely computing kP where P is an elliptic curve point and k is an integer. This operation is the additive analogue of the exponentiation operation α^k in a general (multiplicative-written) finite group. The basic technique for exponentiation is the repeated square-and-multiply algorithms. Several algorithms exist for the fast and efficient implementation of the scalar multiplication operation. Most of them are based on the binary representation of the integer k. For the speed of the elliptic curve cryptosystems, the number of elementary field operations for point addition is important. Here we are just interested in "quadratic" field operations, i.e., we do not care about operations which can be done in linear time. One important observation is then the fact that negating a

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

point is “for free”, since for any non-zero point $P=(x,y)\in E(\mathbb{F}_q)$ the negative point is given as $-P=(x,-y)$ **[11]**.

6.1 Implementing ECC Point Multiplication using Binary Method

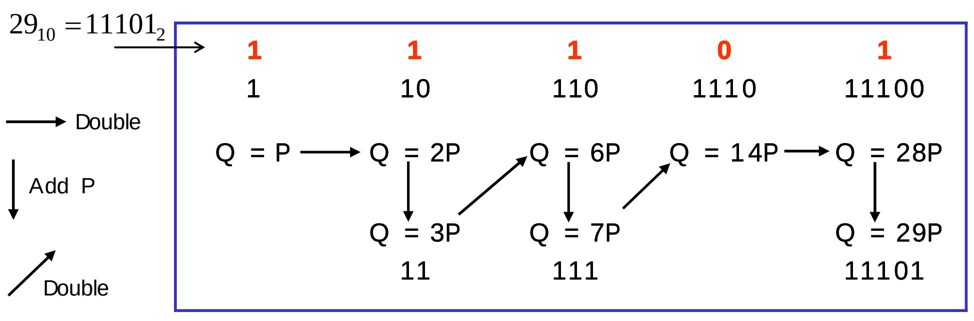
Consider scalar multiplication, kP where k and P represented a scalar multiplier and an elliptic curve generating point $P=(x,y)$, respectively. Now if were to implement point multiplication using a constrained wireless devices (e.g., smartcards, pagers, mobile phones etc.) or a piece of software that computes $Q=kP$, then one of the inputs suitable for this device/software would necessarily be the integer k which would be encoded in a certain format. Exactly how k is encoded would depend up on the application. A very common encoding method is the binary number system; however, there may be other better and more efficient alternatives **[36]**. The devices/software could also have the capability to convert between encoding formats.

Suppose that k is represented (i.e., encoded) as string of digits, $a_{\ell-1}\ldots a_1a_0$, such that:

$$k=a_{\ell-1}2^{\ell-1}+\cdots+a_2 2^2+a_1 2^1+a_0 2^0$$

where $a_{\ell-1}\ldots a_1a_0$ is a *radix 2* representation of k. We denote sums like the one above by $(a_{\ell-1}\ldots a_1a_0)_2$. If each a_i is in the set $\{0,1\}$, then we can compute $kP=Q$ using the left-to-right method of *binary algorithm*, and we can apply clever tricks at each stage of the partial computation **[36]**. Recall that $2P$ and $P+Q$ are usually expensive operations. The cost of the binary method is usually measured by counting the number of addition and doubling operations that are performed with $P\neq O$. If the representation $(a_{\ell-1}\ldots a_1a_0)_2$ has $a_{\ell-1}\neq 0$ then the number of such doubling operations is $\ell-1$ and the number of such addition operations is one less than the number of nonzero digits in $(a_{\ell-1}\ldots a_1a_0)_2$.

As an example, let’s consider $kP=29P$ where $k=29$ and (note that $29_{10}=11101_2$), which can be implemented in scalar multiplication as per Fig. 3, from which we can observe that point addition and doubling can be performed as: $2(2(2(2P+P)+P))+P$ i.e., four elliptic curve doublings and three elliptic curve additions.



$$k=a_{\ell-1}2^{\ell-1}+\cdots+a_2 2^2+a_1 2^1+a_0 2^0 \qquad \text{(binary representation of } k\text{)}$$

Fig. 3: An implementation of scalar multiplication in ECC using standard binary encoding.

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

We do not have to restrict ourselves to the digits $\{0,1\}$ with the binary method. We can observe from Fig. 3 that the computation of k would be more efficient if we had fewer nonzero digits. In general, we can have $k=(a_{\ell-1}\ldots a_1a_0)_2$ where each a_i can have $\{0,\pm1\}$. This method leads to an alternative binary method known as *signed binary method*, which allows for the operations $2P$, $P+Q$ and now $P-Q$. In an elliptic curve group, point *subtraction* can be done at essentially the same cost as point addition. A nonzero integer has many $\{0,\pm1\}$-radix 2 representations (an infinite number, in fact) and *any* one of these can be used in the procedure above. Efficient algorithms for group scalar multiplication have a long history and optimal scalar multiplication routines typically use a combination of the left-to-right *m*-array methods with sliding windows, addition/subtraction chains, signed representations, etc **[36a]**.

6.2 Non adjacent form (NAF) Algorithm

In 1960, Reitwiesner **[37]** explained how to construct an optimal solution and his results have been rediscovered many times **[38]**. He showed that every integer has a *unique* $\{0,\pm1\}$-radix 2 representation with the property that, *any two consecutive digits, at most one is nonzero*, moreover, he showed that no other $\{0,\pm1\}$-radix 2 representation of k can have fewer nonzero digits than this canonical representation.

Reitwiesner canonical representations have come to be called *non-adjacent forms* (NAFs). Taking $(a_{\ell-1}\ldots a_1a_0)_2$ to be NAF of k in the signed binary method minimizes the number of addition/subtraction operations. However, if the NAF of k was substantially longer than, say, $\{0,1\}$-radix 2 representation of k, then it would not be a good idea to use the NAF. Fortunately, the NAF of k is at most one digit longer than the $\{0,1\}$-radix 2 representation of k. The NAF algorithm is normally considered to be a more efficient method for the computation of kP., when compared to the binary scalar multiplication technique. The weight of NAF representation of a number of length ℓ is $\ell/3$.

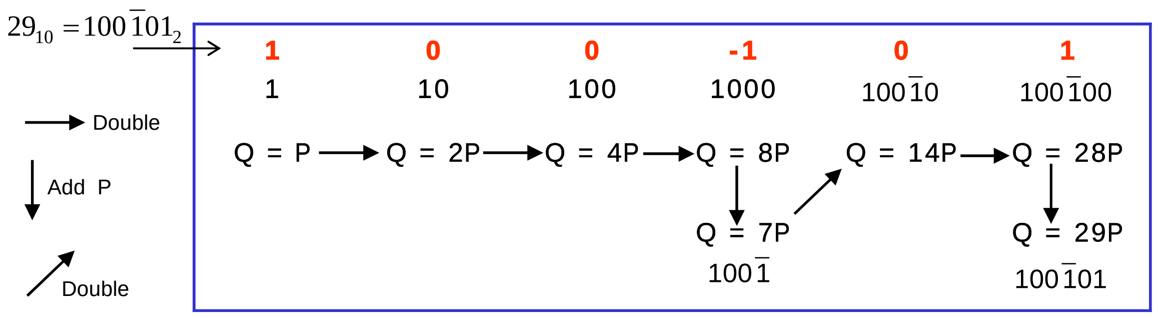
Let's us now look again at $k=(29)_{10}$, which we can implement as per Table 2. Therefore, the value of 29 in NAF form is $k=(29)_{10}=(100\bar{1}01)_2$. (Note that no two consecutive (adjacent) digits are non-zero). Using the pictorial format of Fig. 4, we can observe that compared ro standard binary representation, the NAF method computes $k=(29)_{10}=(100\bar{1}01)_2$ uses 5 doubling and 2 subtraction/addition operations.

Table 2: Illustration of computation of NAF(29)

Iteration #	**c**	ℓ	**u**
1	29	0	1
2	14	1	0
3	7	2	–1
4	4	3	0
5	2	4	0
6	1	5	1
$k=(29)_{10}=(100\bar{1}01)_2$			

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License



k = 29 can be found with 5 doubling and 2 addition/subtraction operations using NAF.

Fig. 4: An implementation of scalar multiplication in ECC using binary encoding using NAF methodology.

In general, given $P \in E$, the coordinates of kP can be computed in $O(\log k \log^3 q)$ bit operations. Table 3 gives general computational times for binary operations for $nP = Q$.

Table 3: Computational times for binary method, $n = \lceil \log_2 q \rceil$.

	EC Doubling	**EC Addition**	**Total**
Binary			
(Maximum)	n	n	$2n$
(Average)	n	$n/2$	$3n/2$
Signed Binary			
(Maximum)	n	$n/2$	$3n/2$
(Average)	n	$n/3$	$4n/3$

7.0 ElGamal Elliptic Curves Cryptosystems

Taher ElGamal was the first mathematician to propose a public-key cryptosystem based on the Discrete Logarithm problem (DLP) **[10].** He in fact proposed two distinct cryptosystems: one for encryption and the other for digital signature scheme in 1984, well before elliptic curves were introduced in cryptography. Since then, many variations have been made on the digital signature system to offer improved efficiency over the original system. The ElGamal public-key encryption scheme can be viewed as Diffie-Hellman key agreement protocol in key transfer mode **[39]**. Its security is based on the intractability of the discrete logarithm problem (DLP) and the Diffie-Hellman problem. However, the elliptic curve cryptosystems as applied to ElGamal protocols were first proposed in 1985 **[10]**.

7.1 EC-ElGamal Encryption Scheme

In implementing ElGamal elliptic curve cryptosystems, suppose that two entities, the Bank (B) and the customer Alice (A) wants to communicate between each other over an insecure communication network. Next let's assume that the two entities have decided to use the protocol of EC-ElGamal encryption protocol to implement their secure communication. One basic point to note is unlike ECDH protocol **[11]**,

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

this protocol does not create a common key, but using EC-ElGamal protocol a message $M = mP = (m_1, m_2)$, a point on elliptic curve, can be sent from Bank to Alice and vice versa, as per the Algorithm 2.

Algorithm 2. EC-ElGamal Encryption Protocol

Key generation: (A)

1. Select a random integer k_A from $[1, n-1]$.
2. Compute: $A = k_A P$
3. A's public key is $k_A P$ or (E, P, A), A's private key is k_A

Encryption: (B)

1. Select a random integer k_B from $[1, n-1]$.
2. Compute $B = k_B P$ such that $S_{BA} = k_B(k_A P) = (x_S, y_S)$
3. If $x_S = 0(\bmod p)$ and $y_S = 0(\bmod p)$ then go to step 2.
4. Compute: $c_{m1} = x_S m_1$ and $c_{m2} = y_S m_2$
   (Note the calculations are done, mod p)
5. Send (B, c_{m1}, c_{m2}) to A

Decryption: (A)

A receives (B, c_{m1}, c_{m2}) and does the following:

1. Compute $S_{AB} = k_A(k_B P) = k_B(k_A P) = (x_S, y_S)$
2. Compute $m_1 = c_{m1}/x_S$ and $m_2 = c_{m2}/y_S$,
   (Note the calculation are done, mod p)
3. Recover the message $M = (m_1, m_2)$

- Note that in step 2 we can also compute $hk_A P$ and $hk_B P$, which can resist the attack on small subgroup. Where h is a co-factor defined in P1363.

Performing the decryption, we reverse the embedding process to produce the message, M, from the point P. It is not trivial to find a point M for the message. Note that the difficulty in obtaining the private-key from the public-key is based on the discrete log problem (DLP) for elliptic curves, ECDLP.

7.1.1 *Simple Implementation of ElGamal Cryptosystem for Elliptic Curves*

Let the prime number $p = 37$ and consider an elliptic curve $E: y^2 = x^3 + x + 25 \bmod 37$ defined over F_{37}. Here $E_p(a,b) = E_{37}(1,25)$ with the order n of the elliptic curve as: $\#E(\mathrm{F}_{37}) = n = 29$, which is prime order. The curve has generator point given by $G \equiv P = (5,9)$ such that the multiples kG of the generator point G are (for $1 \le k \le n-1$) including point O located at infinity.

As an example, suppose the Bank (B) chooses public-key set: $B = k_B G = 17G = (20,4)$, where the

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

secret-key $k_B = 17$, giving rise to its public-key ring (E, G, A), which is kept in the public-key server.

Next let Alice selects a random secret-key $k_A = 21$ such that: $A = k_A G = 21G = (17,16)$ and computes: $S_{AB} = k_A(k_B G) = 21(17G) = 9G = (16,17) = (x_S, y_S)$.

Encryption: She then selects a message point: $M = (M_1, M_2) = (5,28) = 28G$, and computes: $c_{m1} = x_S m_1 \bmod p = 16(5) \bmod 37 = 6$ and $c_{m2} = y_S m_2 \bmod p = 17(28) \bmod 37 = 32$, and send $(A, c_{m1}, c_{m2}) = (21G, 6, 32)$ to the Bank (B).

Decryption: Bank receives the message $(A, c_{m1}, c_{m2}) = (21G, 6, 32)$, and computes:

$S_{AB} = k_B(k_A G) = 21(17G) = 9G = (16,17) = (x_S, y_S)$ which it uses to recover *M*, i.e.,

$m_1 = c_{m1}/x_S \bmod p = (6/16) \bmod 37 = 5$ and $m_2 = c_{m2}/y_S \bmod p = (32/17) \bmod 37 = 28$

and recovers the original message chosen point $M = (m_1, m_2) = (5,28) = 28G$.

7.2 Digital Signature Scheme

Digital signatures and message digests are used to guarantee the integrity of communications over a network. Random numbers are often used in digital signature algorithms to make it difficult for a malicious party to forge the signature. Many signing algorithms, including the U.S. Government's Digital Signature Standard (DSA) also require random sources to ensure the security of the signing keys

In public-key cryptography, communicating entities can use their private keys to encrypt a message and the resultant ciphertext can be decrypted back to the original message using the individual entity's public key **[1]**. Evidently, the ciphertext so created can play the role of a manipulation detection code (MDC) accompanying the encrypted message, i.e., provide data integrity protection for the message. Here, the public key decryption process forms a step of verification. Since, it is considered that only the owner of the public key used for the MDC verification could have created the MDC using the corresponding private key. Thus, this usage of public key cryptosystem can model precisely the property of a signature, a digital signature, for proving the authorship of the message **[11,26,40,41]**.

Diffie and Hellman were the first researchers to envision the notion of digital signature scheme with their invention of asymmetric key crypto-algorithm through use of key distribution and shared keys – the DH key exchange algorithm. This systems of key distribution leading to shared keys under public key crypto-algorithm, finally meant that only a single entity is able to create a digital signature of a message which can be verified by anybody, it is easy to settle dispute over who has created the signature. This allows the provision of a security called non-repudiation, which means no denial of connection with message. Non-repudiation is a necessary security requirement in electronic commerce application **[11,26,40,41]**.

7.3 EC-ElGamal Digital Signature Scheme

The ElGamal signature algorithm **[10]** is similar to the encryption algorithm in that the public-key and private-key have the same form; however, encryption is not the same as signature verification, nor is decryption the same as signature creation as in RSA. Algorithm 3 shows the implementation of EC-ElGamal signature scheme.

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

Algorithm 3. EC-ElGamal Signature Scheme

Key Generation: Entity A (Alice) selects a random integer k_A from the interval $[1, n-1]$ as her private key, and computes $A = k_A G$ as her public key, which she places in the public-key server.

Signing Scheme

1. Selects random integer k from the interval $[1, n-1]$
2. Computes $R = kG = (x_R, y_R)$, where $r = x_R \bmod n$; if $r = 0$ then goto step 1;
3. Compute $e = h(M)$, where h is a hash function $\{0,1\}^* \to F_n$
4. Compute $s = k^{-1}(e + k_A r) \bmod n$; if $s = 0$ then goto step 1

   (R, s) is the signature message M.

Verifying Scheme

1. Verify that s is an integer in $[1, n-1]$ and $R = (x_R, y_R) \in E(F_q)$.
2. Compute $V_1 = sR$.
3. Compute $V_2 = h(M)G + rA$, where $r = x_R$.
4. Accept if and only if: $V_1 = V_2$.

Consistency

$V_1 = sR = skG\{(h(M) + k_A r) \bmod n\}G$, $V_2 = h(M)G + rA = [h(M) + rk_A]G$. And because $G's$ order is n, $kG = jG$ where $j \equiv k \bmod n$. Hence, $V_1 = V_2$.

7.3.1 *Simple Implementation of EC ElGamal Signature Scheme*

Let the prime number $p = 37$ and consider an elliptic curve $E:\ y^2 = x^3 + x + 25 \mod 37$ defined over F_{29}. Here $E_p(a,b) = E_{37}(1,25)$ with the order n of the elliptic curve as: $\#E(F_{37}) = n = 29$, which is prime order. The curve has generator point given by $G \equiv P = (5,9)$ such that the multiples kG of the generator point G are (for $1 \le k \le n-1$) including point O located at infinity.

Let Alice selects a random secret-key $k_A = 21$ such that: $A = k_A G = 21G = (17,16)$. Next she chooses integer k in the interval [1, n-1] and computes $R = kG = 13G = (19,13) = (x_R, y_R)$, and also computes $r = x_R \bmod n = 19 \bmod 37 = 19$.

Suppose now Alice wants to send the message, $M = 17G = eG$, where e lies in the interval $[1, n-1]$. She next computes: $s = k^{-1}(e + k_A r) \bmod n = (13)^{-1}(17 + 21 \cdot 19) \bmod 29 = 3$.

$(R, s) = (13G, 3)$ is the signature of the message M.

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

Verifying

1. Verify that s is an integer in $[1, n-1]$
2. Compute $V_1 = sR = 3R = 3(13G) = 10G = (15,14)$
3. Compute $V_2 = h(M)G + rA = 17G + 19(21G) = 10G = (15,14)$, where $r = x_R \bmod 37 = 19$
4. Accept signature, since $V_1 = V_2 = 10G = (15,14)$

8.0 Securing online Content Management Site using EC-ElGamal Scheme - *A Practical Application*

For this part let's get real and simulate real application. Assume we have a website: www.bauxicat.com, an online content management site that specializes in selling downloadable online consumer items. Now the question is? How can we set up a public-key implementation of elliptic curve EC-ElGamal cryptosystems (ECEG) to protect our site, so that only the registered parties can download music from the site? We start by setting up an elliptic curve cryptographic system. Let's take as an example our usual communicating partners, Alice and Bob. Alice and Bob love to swap music files stored in their computers amongst themselves.

When Alice became a member of Bauxicat's music content rights management (BMCRM), she was prompted to download and install BMCRM software on her computer. While registering, Alice and BMCRM both exchanged their public keys. Bauxicat's public-key point is $k_M G = P_M$. During installation session, the BMCRM software sneakily generated a private-key, a music point private-key integer m, in the range $[1, n-1]$, and which it stealthily hid in bits of files (e.g., `darkside.dll`, `w4gs.dla` and `BoxView.key`). BMCRM public music point is $mG = M$. (For more details on data hiding techniques, see ref. **[1-5]**.) In order for Alice to play the latest UB20's music `bemylove.wma`, she must use her private key to decrypt the file. Bauxicat created the license using the EC-ElGamal public-key cryptosystem using a predefined elliptic curve group over the Galois Field $E_p(a,b)$; Alice's license file can now be used to unlock `bemylove.wma`, but *only* in her computer, since the license file is not transferable and hence Bob's computer has no access to this file, as there is no shared key between Bob and Bauxicat. So, whenever she shares her music files including the license files etc. Bob gets very crossed because he can't play `bemylove.wma`. This is mainly because Bob's computer doesn't know Alice's computer's private-key (i.e., the integer m), therefore, Bob's computer can't decrypt the license file to allow him access and enjoy the music, his beloved song `bemylove.wma.`

8.1(a) *Implementation of Practical Application of ECEG Scheme*

Here we simulate the real ECEG scheme in practice. Let an elliptic curve group over the Galois Field $E_p(a,b)$ where $p > 3$ and is prime, be the set of solutions or points $P = (x, y)$ such that $(x, y \in E_p(a,b))$ that satisfy the equation: $y^2 = x^3 + ax + b \pmod{p}$ for $0 \le x < p$ together with the extra point O called the point at infinity. Here the curve $E(\mathrm{F}_P)$ is cyclic and any point other than O is a generator of all points on curve. For example, $G = (x, y)$ is a generator point such that the multiples kG of the generator point G (for $1 \le k \le n-1$).

Alice-Bob Session

In order to setup a communicate session, both Alice and Bob selects random integers k_A and k_B in the range $[1, n-1]$, as their private-keys (kept secret). Next they compute $A = k_A G$ and $B = k_B G$

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

respectively, which are kept in the public-key server, for access to anyone wishing to communicate with them. Alice and Bob may both set up a common session key $k_B(k_A G) = k_A(k_B G) = S_{AB} = (x_t, y_t)$, using say ECDH key agreement scheme **[41]**. In order to send a message $P_w = wG = (w_1, w_2)$, Alice and Bob decides to use EC-ElGamal crypto-scheme. Alice's message encrypted code is: $(A, x_t w_1, y_t w_2)$, where A is Alice's public-key, $x_t m_1$ and $y_t m_2$ are the encrypted message points, respectively.

Alice-Bauxicat Session

Now let's take a look at the interaction session between www.bauxicat.com and Alice. Common session key between Alice and Bauxicat is: $S_{\mathrm{AM}} = k_M(k_A G) = k_A(k_M G) = (x_S, y_S)$. (Notice that there is no common session key between Alice, Bob and Bauxicat.) Alice license file contains the music file encrypted code: (A, c_{m1}, c_{m2}), where A is Alice's public-key and; $c_{m1} = x_M m_1 \bmod p$ and $c_{m2} = y_M m_2 \bmod p$, are the encrypted music file points, respectively, i.e., $C_m = (c_{m1}, c_{m2})$.

The parameters for the elliptic curve featuring in Alice-Bob and Alice-Bauxicat communication are as follows:

```
p = 1320289767260163936968809125287978244373455215509863645302239
a = 1162323547384410627544889938378379573059142481934431481713754
b = 1779810017097048764346213441416898875281472097072488425243 94
```

```
\\Order U = R*S with R Prime

U = 1320289767260163936968809125287668483156160622361375915462606
N = 50780375663852459114184966357218018582929254706206765979331
S = 26

G = (x, y)  \\generator point of order U or R #E, depending on usage.

Where
X = 421526132993100674499186543743618072759975050035788560965760
Y = 128328308509316221529625258350750388808791994724791315155306

kA = 334311750631561905457965400776703294649473217653573125698 78 \\A-privK
kB = 256383031470922306459293241950961639677034964585481276883 07 \\B-privK
kM = 76890575490361905311421227475296277143280671446822963221 73 \\Baux-pri
 h = 489755924509465995334598808914033652447914954951259707273 95 \\hash
 m = 156679036347553777694060580627516889904924892525861014647 28 \\music
```

Using the above parameters, we get:

$A = k_A G$

```
  =(1046237401838541784386570891605033755990364036475346158793795,
     7648157409460615790704228808700422599867830911285619904427 59)
```

$B = k_B G$

```
  = (3500141421566290867894476165349322388763159708296052978293 41,
      868669466493347324822781059307841031180954060427712082610019)
```

Now Alice's computer had sneakily loaded BMCRM license file private key m into memory which

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

automatically computes music point to allow her access to music session:

$$M = mG = (m_1, m_2)$$
$$= (999763484420734549974894041302516417845073296646850793974405,\ 581597048491656075323344204698946987620758290571681899918110)$$

which is however scrambled (encrypted) using BMCRM public key $P_M = k_M G$ and Alice's public-key $A = k_A G$ into a music message point: (P_M, c_{m1}, c_{m2}). Where

$$c_{m1} = x_S m_1 \bmod p$$
$$= 490601104006035591800992160138697044588359971330393820478429$$

$$c_{m2} = y_S m_2 \bmod p$$
$$= 473716465977412145869796100310969200274995786235830506384103$$

Next Alice's computer automatically uses BMCRM public-key P_M to compute the session key which allows her access to music file, i.e.,:

$$S_{\mathrm{AM}} = k_A(k_M G) = k_M(k_A G) = (x_S, y_S)$$
$$= (132360347480825062457073474967138895660627459094427262882650,\ 1317902145775285502197954022294494418734839785023473873875523)$$

and then recovers the music point $M = (m_1, m_2) = (x_M, y_M)$

$$m_1 = x_M = (c_{m1}/x_S) \bmod p$$
$$= 999763484420734549974894041302516417845073296646850793974405$$

$$m_2 = y_M = (c_{m2}/y_S) \bmod p$$
$$= 581597048491656075323344204698946987620758290571681899918110$$

and which maps the music point $M = (x_M, y_M)$ back into the original point *M*. The crucial parameter here is the x-coordinate, i.e.,:

$$x_M = 999763484420734549974894041302516417845073296646850793974405$$

which is the top secret BMCRM magic "content key" that unlocks `bemylove.wma`. Note that if Alice was aware of the license file secret-key k_M that her computer stealthily generated, she could easily compute the music point *M* herself, and hence unlock `bemylove.wma` which she could now securely share with Bob without the knowledge of BMCRM.

NB: The number crunching done above is a game played by Big Boys, so we need number crunching software to crunch our computation undertaken above. The number crunching here was done using PARI, a Free open source, number cruncher **[42]**.

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

8.1(b) ***EC-ElGamal Signature Verification***

Before Alice's computer can start to interact with BMCRM software she wants to make sure that the music file is authentic before commencing playing the music. The set of values (R, s) which is the signature message for music point M was coded using the music-point private-key $m = h(M)$, to simulate a hash function, as in previous simple example of EC-ElGamal signature algorithm performed earlier. To verify the authenticity of the music file she must do the following operation using EC-ElGamal signature algorithm:

She must first verify that s is an integer in $[1, n-1]$ and $R = (x_R, y_R) \in E(F_q)$; i.e.,:

$R = kG = (x_R, y_R)$,

$= (6181706743341074260518550755374940867384057817666249610400 66,$
$\quad 4919667881372171696439662730555074821213552290340140326092 35)$

where:

$r = x_R = 618170674334107426051855075537494086738405781766624961040066$

and

$s = k^{-1}(h + k_A r) \bmod n$

$= 16865816466355493621338004093920537981853515802121287786830$

then her computer automatically computes:

$V_1 = sR = s(kG)$

$= (10070208873520265484706930775747254177901172277372738968372 62,$
$\quad 603505051264566480597048509211429068641356456714339060885715)$

and

$V_2 = hG + r(k_A G)$

$= (1007020887352026548470693077574725417790117227737273896837262$
$\quad 603505051264566480597048509211429068641356456714339060885715)$

and verifies that: $V_1 = V_2$. So her computer goes ahead and plays the music.

9.0 Conclusions and Future Work

We have shown that elliptic curve ciphers require less computational power, memory, and communication bandwidth giving it a clear edge over the traditional crypto-algorithms. To date elliptic curve cryptography is gaining wide acceptance, especially in wireless and hand-held devices, when compared to the conventional cryptosystems (DES, RSA, AES, etc.), which tend to be power hungry. However, while the performance advantages are impressive with ECC, the data security industry need to ensure that the security system, using elliptic curve algorithm has been studied extensively in the public forum and, also specified by major standards worldwide. But we think that elliptic curve cryptography is here today and is without question the next generation of public-key cryptography of choice. The ElGamal cryptosystem, as we have already seen, requires a high level of mathematical abstraction to implement. One significant

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

practical problem if this system is to be useful is how can it be packaged in a user-friendly way so that developers can incorporate it into their applications with minimal knowledge of its inner workings – and that is subject of next work.

7.0 References

1. Kefa Rabah, "*Data Security & Cryptographic Techniques – A Review*" PJIT 3(1) 2004, (106-132).

2. K. Rabah, "*Steganography – The Art of Hiding Data*". ITJ 3(3) 245-269, July 2004.

3. Koblitz, N.: "*A course in Number theory and Cryptography*", Springer Verlag , 1994.

4. Menezes, A, P. Van Oorschot, and S. Vanstone, 1997. "*Handbook of Applied Cryptography*". CRC Press.

5. U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology. Announcing request for candidate algorithm nominations for the Advanced Encryption Standard (AES). World Wide Web, September 12, 1997. http://csrc.nist.gov/encryption/aes/pre-round1/aes_9709.htm.

6. W. Diffie and M. E. Hellman, "*New Directions in Cryptography*". IEEE Transaction on Information Theory, v. IT-22, n. 6, Nov. 1966, pp. 644-654.

7. W. Diffie and M. E. Hellman, "*Multi-user Cryptographic Techniques*". Proceedings of AFIPS National Computer Conference, 1976, pp. 109-112.

8. Rabin, M. O., 1979. "*Digital Signature and Public-Key Functions as Intractable as Factorization,*" MIT Laboratory of Computer Science, Technical report, MIT/LCS/TR-212, Jan.

9. Rivest, R., A. Shamir, and L. Adleman, 1978. "*A Method for Obtaining Digital Signatures and Public Key Cryptosystems*". Comm. of the ACM 21(2): 120-126. Feb.

10. ElGamal, T., 1985. "*A public-key cryptosystem and a signature scheme based on discrete logarithms*". IEEE Trans. Info. Theory". **31**, pp. 469-472.

11. K. Rabah, "*Theory and Implementation of Elliptic Curve Cryptography*". J. Applied Sci., 5 (4): 604-633, 2005.

12. Arjen K. Lenstra, Eric R. Verheul, "*Selecting Cryptographic Key Sizes,*" Journal of Cryptology: the Journal of the International Association for Cryptologic Research, (1999).

13. Miller, V, 1986. *Use of elliptic curves in cryptography*. In *CRYPTO '85*, pages 417-426.

14. Koblitz, N, 1987. "*Elliptic curve cryptosystems*", Mathematics of Computation, Vol. 48, pp. 203-209.

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

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

16. ANSI X9.62, "*The Elliptic Curve Digital Signature Algorithm (ECDSA*)", American Bankers Association, 1999.

17. IEEE P1363a: http://grouper.ieee.org/groups/1363/P1363a/

18. ANSI X9.42 and X9.63.
http://cio.doe.gov/Publications/profile2000/Profile2000_AppendixA.htm

19. ISO/IEC: http://grouper.ieee.org/groups/1363/Research/Other.html

20. RSA labs: http://www.rsasecurity.com/rsalabs/node.asp?id=2013

21. Certicom www.certicom.com

22. BlackBerry^{TM}: http://www.blackberry.com/products/handhelds/index.shtml

23. A. Daly, W. Marnane, T. Kerins and E. Popovici. "*Fast Modular Division for Application in ECC on Reconfigurable Logic*". Field-Programmable Logic and Applications - FPL 2003, (LNCS 2778):786–795, Sept 2003.

24. J. Wolkerstorfer. "*Dual-Field Arithmetic Unit for GF(p) and GF(2m)*". Cryptographic Hardware and Embedded Systems - CHES 2002, (LNCS 2523):500–514, Aug 2002.

25. National Institute of Standards and Technology, "NIST: FIPS Publication 186-2: Digital Signature Standard (DSS)," January 2000.

26. L. Basham, D. Johnson, T. Polk, *Representation of Elliptic Curve Digital signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public-Key Infrastructure Certificates*, Internet Draft, June 1999, Available at: http://www.ietf.org.

27. Odlyzko, A., 1984. *Discrete logarithms in finite fields and their cryptographic significance*. In *Advances in Cryptology Eurocrypt'84*, pages 224-314. Springer-Verlag.

28. Schoof, R., "*Elliptic curves over finite fields and computation of square roots mod p*". Math. Comp., **44**, pp. 483-494 (2998

29. D. Shanks: *Class number, a theory of factorization and general*, Proc. Symposium of Pure Math. *20* , Amer. Math. Soc. 1970, pp 415-440.

30. ECB (Elliptic Curve Builder) – Ellipsa - http://www.ellipsa.net/index.html

31. Pohlig, S. C. and M. E. Hellman, 1978. "*An improved algorithm for computing logarithms over GP(p) and its cryptographic significance*". IEEE Trans. Info. Theory, **24**, pp. 106-110.

32. John M. Pollard, "*Monte Carlo methods for index computation* (mod *p*)", pp. 918-924, Mathematics of Computation, 32, 1978.

33. P. Van Oorschot and M. Wiener. Parallep collision search with cryptanalytic applications. *Journal of Cryptology*, 12:1-28, 1999.

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License

34. W. Stallings. "*Cryptography and Neywork Security: Principles and Practice.*" Prentice-Hall, Upper Saddle River, New-Jersey, 2nd, 1999.

35. G.B. Agnew, R. C. Mullin, and S.A. Vanstone. An implementation of elliptic curve cryptosystems over $\mathbb{F}_{2^{155}}$. *IEEE Transactions on Selected Areas in Communications*, 11:804-813, 1993.

36. Solinas, J. A. efficient arithmetic on Koblitz curves. Designs, Codes and Cryptography 19 (2000), 195{249.

36a. D. E. Knuth, "*The Art of Computer Programming*." Vol. 2, Seminumerical Algorithms Addison-Wesley, Reading, Mass., 1969.

37. G. W. Reitwiesner. Binary Arithmetic, in *Advances in Computers, Vol. 1*, Academic Press, 1960, pp. 231.308.

38. J. Jedwab AND C. J. Mitchell. Minimum Weight Modi_ed Signed-Digit Representations and Fast Exponentiation, *Electronic Letters* **25** (1989), 1171.1172.

39. W. Diffie, P. van Oorschot and M. Wiener, "*Authentication and authenticated key exchanges*", Designs, Codes and Cryptography, **2** (1992), 107-125.

40. Kefa Rabah, "*Secure Implementation of Message Digest, Authentication and Digital Signature*". In Press, To Appear in ITJ.

41. K. Rabah, "*Implementation of Elliptic Curve Diffie-Hellman and EC Encryption Schemes*". ITJ 4(2), pp. 132-139, 2005.

42. PARI http://pari.math.u-bordeaux.fr/

© April 2003, Kefa Rabah, Global Open Versity, Vancouver Canada

www.globalopenversity.org A GOV Open Knowledge Academic Access License