Overview of Elliptic Curve Cryptography on Mobile Devices

Ariel Hamlin
ariel.hamlin@tufts.edu
Professor Hescott

Abstract:
As mobile devices become smaller and more prevalent in daily life, the need for a computationally cheap, but still secure, cryptosystem rises. Elliptic curve cryptography (ECC) allows for much smaller key sizes: a 163-bit key in ECC has equivalent strength to a 1024-bit key in RSA. Additionally, unlike RSA and other cryptosystems based on the discrete log problem, the best solution for ECC takes exponential time. The algorithm, based on one-way operation of adding and doubling within the field, is faster and more energy efficient than conventional cryptosystems. The combination of security and smaller computational demands is ideal for highly mobile, resource-starved environments. The mobile platform also offers a unique method for developing the seed values for ECC parameters.

1. 0 Introduction

The use of smartphones, in the US alone, is up nearly 25% in the past year, and this is just a segment of the mobile computing population. By 2016, it is estimated that nearly 60% of the consumer market will have some type of smartphone [1]. The growth is explosive, as is the need for a tailored encryption scheme for the mobile platform.

ECC was developed in 1985 independently by Neal Koblitz and Victor Miller. Both men saw the application of the elliptic curve discrete log problem (ECDLP) as a replacement for the conventional discrete log problem (DLP) which is used in DSA, and the integer factorization problem found in RSA. For both problems, sub-exponential solutions have been generated; the same which cannot be said for ECDLP [2]. In addition to offering increased security for a smaller key size, operations of adding and doubling can be optimized successfully on a mobile platform [3]. ECC offers a viable replacement to the most common public-key cryptography algorithms on mobile devices.

2. 0 Why care about ECC?

Cell phones are an integral part of today's society, especially as technology allows mobile devices to become an extension of the workplace, an entertainment system, a banking center, or a favorite store [1]. Anything once done on a desktop computer has now migrated to a smartphone in some form or another. Take for example an average consumer, who does his banking online; in the past he used his home computer to transfer funds between accounts, but recently he has discovered the simplicity of mobile banking and started using his phone to transfer money while on the train to work. He has moved his sensitive data from a more stable, secure system to a smaller, more public platform. Without consideration of other technologies, he has two choices - wait longer as his device struggles with 1024-bit computations, or settle for a less secure, smaller key-size. However for financial institutions, the smallest RSA key-size allowed is 1024-bits, so the user is stuck waiting; and as he waits he can see his battery life dropping [4]. Other actions that the average consumer may undertake on their phone also requiring encryption (banking, sensitive corporate records in the form of email, or even medical records) face the same limitations.

The most serious restrictions of mobile devices can be narrowed to two categories: computational limitations and power limitations. Even as technology advances, mobile devices will always be less powerful than their non-mobile counterparts – an intrinsic quality of the mobile environment. As technology becomes faster, a race has developed between creating more powerful encryption schemes and the ability to crack these encryptions by using brute force – a race that mobile technology will always lose. In addition to computational limitations, power in a mobile environment is a type of electric gold. Most devices are in use all day and security should not be responsible for draining the battery. For example, a doctor sends tens of confidential emails a day from his phone, the same phone he relies on as an emergency contact option. The encryption of those emails should not prevent him from receiving a phone call about a patient who needs his help. The drive then is not towards longer key sizes that take longer and demand more power, but towards *smarter* encryption schemes [5] [6].

ECC has many advantages in comparison to other asymmetric cryptography schemes. The foremost among them is the relatively small key size compared RSA and Diffie-Hellman when normalized in respect to symmetric key sizes suggested by NIST. Below is a table from the NSA's Case for Elliptic Cryptography [4] showing the disparity:

Symmetric Key Size (bits)	**RSA and Diffie-Hellman Key Size (bits)**	**Elliptic Curve Key Size (bits)**	**Ratio of DH Cost : EC Cost**
80	1024	160	3 : 1
112	2048	224	6 : 1
128	3072	256	10 : 1
192	7680	384	32 : 1
256	15360	521	64 : 1

Not only to the key growth rate slower than RSA and Diffie-Hellman, but per the cost to compute each bit there is measurably more security generated. While sub-exponential algorithms exist to break both DSA [7] and Diffie-Hellman [3] (number field sieve algorithm and Pollard's-rho algorithm respectively), no known sub-exponential algorithm exists to break ECC. The most successful algorithm for ECC is again is Pollard's-rho algorithm, which is at best $\sqrt{\pi n/4}$ steps, where n is the number of points on the curve. The cost of the algorithm based on size n is presented below [3]:

Size of n (in bits)	$\sqrt{\pi n/4}$	**MIPS years**[1]
160	2^{80}	8.5×10^{11}
186	2^{93}	7.0×10^{15}
234	2^{117}	1.2×10^{23}
354	2^{177}	1.3×10^{41}
226	2^{213}	9.2×10^{51}

[1] MIPS stands for million instructions per second. Over the course of a year this amounts to approximately 31.5 trillion instructions.

In addition to the increased security, encryption using ECC schemes is much faster than schemes such as RSA. When the two schemes were run on the same mobile platform, ECC, specifically ECDSA, ran over a hundred times faster RSA at a similar level of security. [8]

For the average consumer, ECC represents an encryption scheme which, for lower computation and power cost, generates greater security for his mobile interactions

3. 0 How do Elliptic Curves work?

3.1 Groups, Rings, Fields

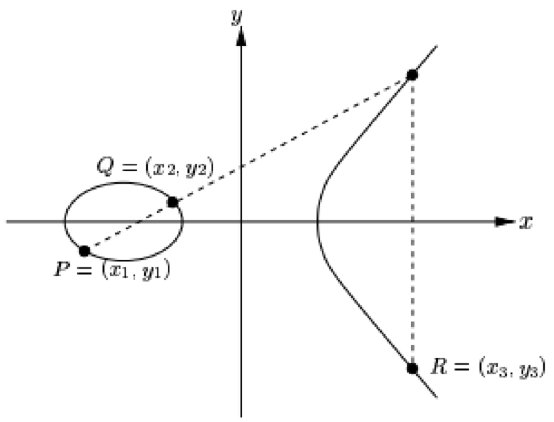
Elliptic curves are curves with the operations of adding and doubling defined over either an odd or even field. An understanding of groups and rings is needed to generate the definition of a field. Groups are a closed set; a set in which an operation of two elements in the group generates another element in the set. Each group has an identity element, akin to the number 1 in multiplication of real numbers, where an element times the identity element yields the original element. For each element in the group there exists an inverse such that the element times it inverse yields the identity element. A set with two operations is defined as a ring, and shares similar properties with groups. Like groups, rings have identity elements, one for each operation, and inverses for each element in the field for the first operation. For example, a set of integers is a ring in which the two operations are addition and multiplication. A field is a ring in which the elements and the multiplication operator form a group. A field can either be even or odd (F_2^m or F_p) where the number of elements is either two raised to the power of m for even fields, or power of a prime for odd fields [2]. Elliptic curves exist in both types of fields, and encryption algorithms are identical with the exception of the adding and doubling operations, as such the focus of this tour will be on odd fields.

3.2 Elliptic Curves over F_p

An elliptic curve is described by the following equation: $y^2 = x^3 + ax + b$ defined over the field F_p where p > 3 is an odd prime. The coefficients a and b are elements in the field F_p such that $4a^3 + 27b^2 \neq 0\ (mod\ p)$. The notation $E(F_p)$ represents the set of all points on the elliptic curve that are an element in the field, including the infinity point which is denoted by O. [3]

3.3 Adding and Doubling Operations

Point addition and doubling is the backbone of ECC. Point addition can be described graphically: point R, with R represented by R = P + Q where P and Q are both points on the elliptic curve, can be found by drawing the secant line between P and Q. The point where the line intersects the elliptic curve is taken and reflected across the curve's horizontal line of symmetry, which much of the time is the x-axis. The resultant point is the sum of P and Q. The operation is demonstrated below [3]:



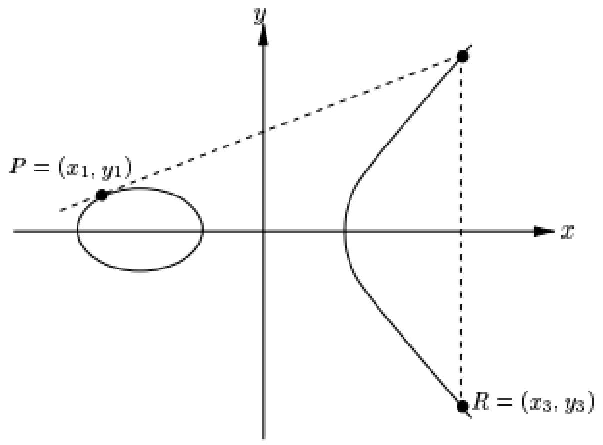
If two points having the same x-coordinate are added, the sum of the points will be infinity, or the identity point O. Adding can also be defined by the following equations:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}\ (mod\ p)$$

$$x_3 = \lambda^2 - x_1 - x_2\ (mod\ p)$$

$$y_3 = \lambda(x_1 - x_3) - y_1\ (mod\ p)$$

Doubling is very similar to adding, but only deals with a single point on the curve. Given P, R= 2P can be found by drawing the tangent line to P, and reflecting the intersection of that line with the curve across the horizontal line of symmetry – demonstrated below [3]:



Doubling is defined by the following equations:

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad (mod\ p)$$

$$y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 \quad (mod\ p)$$

3.4 Weak Curves and NIST Recommended Curves

There are two types of weak curves, supersingular and anomalous, that make solving the ECDLP in subexponential time possible. Supersingular curves allow the ECDLP to be reduced to the classic DLP in a multiplicative group, which has subexponential attacks. Additionally, curves with repeated roots, which are often supersingular curves themselves, such as $y^2 = x^3 - 3x + 2$, can be factored down to $y^2 = (x - 1)^2(x + 2)$, violating the properties of a group. Anomalous curves are even more dangerous, allowing polynomial attacks by reducing the ECDLP to the DLP in an additive group [8].

To prevent the unknown use of a weak curve, the NIST (National Institute of Standards and Technology) generated a list of 15 curves for general use. In addition to these curves, there is a list of ten recommended finite fields where the primes that generate the field are all Mersenne primes. This allows more efficient multiplication within the field [3].

4.0 Encryption Schemes

For most encryption schemes that rely on either integer factorization or the discrete log problem, there is an analogous scheme in ECC that relies on the ECDLP. This paper is by no means comprehensive survey of the possible ECC schemes that exist and will focus instead on three of the more popular algorithms – Elliptic Curve Diffie-Hellman (ECDH), Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic Curve Integrated Encryption Scheme (ECIES).

4.1 ECDH

<u>Public Information:</u>

i. Elliptic Curve:

- Coefficients a and b
- p, a large prime

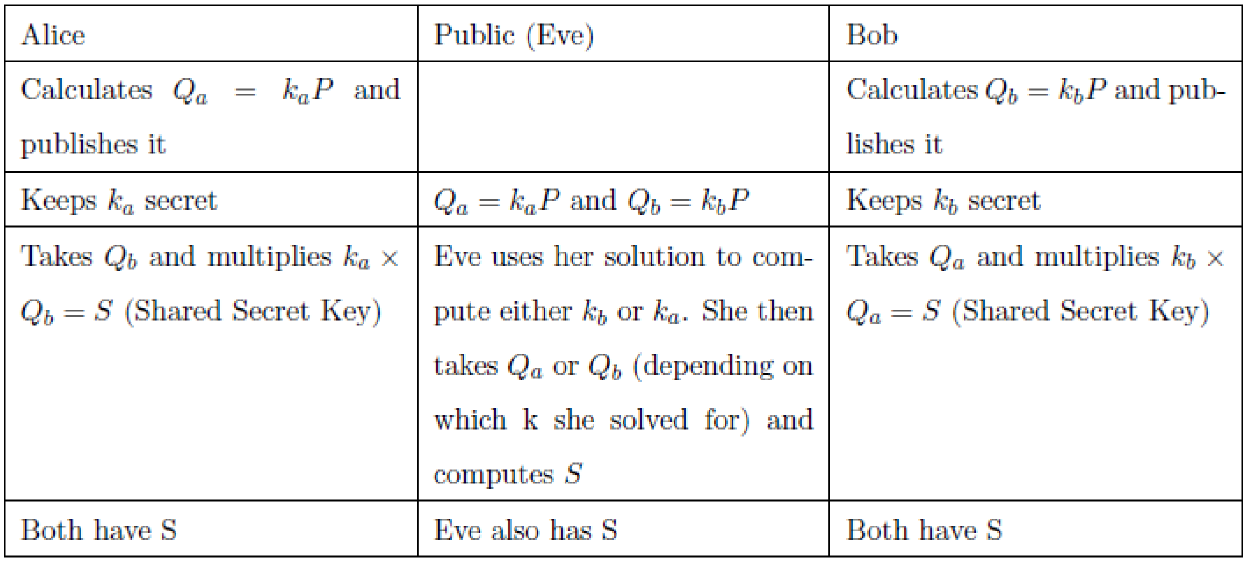
ii. Point P on Elliptic Curve, whose order is large value n

iii. $Q_a = k_aP$ and $Q_b = k_bP$

<u>Private Information:</u>

i. k_a and k_b, where k is some large integer value

<u>Key Generation:</u> [7]



Alice	Public (Eve)	Bob
Calculates $Q_a = k_aP$ and publishes it		Calculates $Q_b = k_bP$ and publishes it
Keeps k_a secret	$Q_a = k_aP$ and $Q_b = k_bP$	Keeps k_b secret
Takes Q_b and multiplies $k_a \times Q_b = S$ (Shared Secret Key)	Eve uses her solution to compute either k_b or k_a. She then takes Q_a or Q_b (depending on which k she solved for) and computes S	Takes Q_a and multiplies $k_b \times Q_a = S$ (Shared Secret Key)
Both have S	Eve also has S	Both have S

4.1 ECDSA

Public Information:

i. A finite field F_p such that $\#E(F_p) = fr$, where r is a large prime and f is a small integer (usually 1,2 or 4)

ii. A point, G, on the curve

iii. $Q_a = aG$

iv. The elliptic curve E

- Coefficients a and b
- p, a large prime

Private Information:

i. An integer a

ii. The message m (m is usually an integer or a hash of the document)

Key Generation: [3]

Alice	Public	Bob
Alice chooses an integer a and computes $Q_a = aG$, she then makes Q_a public		
She chooses a random integer k such that $1 < k < r$ and computes $R = kG = (x, y)$		
She then computes $s = k^{-1}(m + ax) \bmod r$		
She attaches to the document her signature which consists of (m, R, s)	$\Longrightarrow$	Bob receives Alice's signed message and downloads her public information
		He computes $u_1 = s^{-1}m \bmod r$ and $u_2 = s^{-1}x \bmod r$
		He then computes $V = u_1G + u_2Q$
		He declares the signature valid if $V = R$

4.3 ECIES

Public Information:

i. The elliptic curve, E, on finite field F_p.

ii. A point, G, on the curve.

iii. Two hashing functions H_1 and H_2.

iv. A symmetric encryption function ENC with a matching decryption function DEC.

v. A public value for both Alice and Bob accordingly, $Q_b = K_AG$ and $Q_b = K_BG$.

Private Information:

i. Alice's secret integer K_A.

ii. Bob's secret integer K_B.

iii. The message m.

Key Generation: [7]

Alice	Public	Bob
Alice chooses an integer k_a and computes $Q_a = k_a G$		
She takes Bob's public key, $K_b G$, and computes $K_a K_b G$, this is used as the common key. She then uses the hash function $H_1(K_b G, K_a K_b G)$ to get outputs a_1 and a_2		
She then encrypts message m with ENC and sends Bob $(K_a G, ENC(m), t = H_2(K_a K_b G, a_2))$	$\Longrightarrow$	Bob receives Alice's message and using $K_a G$ computes $K_a K_b G$
		He then computes $H_1(K_b G, K_a K_b G)$ to get outputs a_1 and a_2
		He checks $t = H_2(K_a K_b G, a_2)$. If this does not match with the sent t, he rejects the message
		He decrypts the message with $DEC(ENC(m))$ to output m

5.0 Attacks on ECC

The attacks on ECC are many and varied, ranging from theoretical-math based attacks to implementation based attacks. The attacks covered in this section are those highlighted in the paper and make up a small portion of the possible attacks on ECC. More information on other paths of attack can be found in [3] and [2].

5.1 Pollard's rho Attack

Pollard's rho Attack is used to circumvent the ECDLP. The attack relies on find two distinct pairs (c', d') and (c'', d'') such that $c'P + d'Q = c''P + d''$. Once the pairs have been generated, the log of the point Q can be calculated using the equation $\log_P Q = (c' - c'')(d'' - d')^{-1}\ mod\, n$. The algorithm usually takes $\sqrt{\pi n/2}$ steps to generate a collision between the two pairs. Once the log of the k value has been calculated, the attacker can decrypt the message without any further difficulty.

Pollard's rho attack can be sped up by a factor of M, where M is the number of processors used, in the Parallelized Pollard's rho Attack. This attack, created by Van Oorscot and Wiener, allows the iterating functions running on separate processors to collide with each other, rather than just their own iteration. [2]

5.2 Power and Timing Attacks

Both paths of attack exploit the implementation of ECC on a mobile device. Power consumption changes with what data is being evaluated and what operations the data is undergoing. Differential power analysis uses this variation to generate an idea of what computations are necessary to generate the kP. Power analysis attacks require several thousand samples of power traces to generate a hypothesis. As a result, this venue of attack does not pose as significant risk to mobile devices. Of more concern is a similar attack that uses electromagnetic analysis, which only requires proximity rather than physical access to the device.

Timing attacks rely on the fact that different operations take varying amounts of time. For example, it takes longer to compute modulo values than simply adding two numbers together. If an attacker can determine the time it takes a device to execute each operation, such as on a smartcard, then they can make a guess as the nature of the key. Timing attacks can be thwarted by not providing adequate data to do the analysis. This is implicit with ECC due to the fact that for most encryption schemes the key is chosen anew. [2]

5.3 Insecure Random Number Generators (RNG)

The entire premise of ECC relies on the use of a secret key, k, where generating the inverse of the key computationally difficult. *k* is freshly chosen for each iteration of the encryption scheme, so predictable patterns in a RNG leaves ECC particularly weak. The process of designing and implementation a RNG or pseudo-RNG with a high enough level of entropy is very difficult, and thus “insecure random and pseudorandom number generators are perhaps the most common cause of cryptographic attacks on cryptographic systems” [7]. If a RNG is predictable, patterns in key generation emerge and the attacker can guess some information about the key, which makes solving the ECDLP much simpler.

6.0 RNG on Mobile Devices

RNG fall into two categories, true random number generators (TRNG) and pseudo-random number generators (PRNG). TRNG uses entropy sources, which are purported to be truly random such as thermal or atmospheric noise, and polishing algorithms to generate random numbers. PRNGs on the other hand use a random ‘seed’ value to an algorithm to generate the random numbers. Numbers generated by a TRNG are considered more random than those generated by a PRNG because in a PRNG every bit in the pool is dependent on every other bit. Additionally, they are regarded as more secure because once the seed is known for a PRNG, all future numbers can be predicted. While more powerful, TRNGs take longer to generate random values than PRNGs. [9]

Either RNG needs a source of entropy from which they can pull their values or seed. On computers, a popular source is input from the user, such as keystrokes or mouse use. If the post-processing algorithm is fine enough, even though human actions may not be completely random, they can be used as a good source of randomness. On a mobile device, a similar source of entropy is needed especially for ECC schemes, which depend highly on new key generation for each encryption. Possible sources of randomness on mobile devices, specifically smartphones, include the accelerometer, compass, or touch patterns. Not many studies have been done on post-processing algorithms for these sources. Most likely, each would require a unique algorithm based on the patterns in the data, sampling rates, and reporting accuracy. For example, the accelerometer on an iPhone will show the walking patterns of the user if it is carried around in a pocket. Repeated touch patterns such as unlocking a phone, or playing a game, could also be an issue. A study measuring the entropy values of these sources, show that the compass is potentially the best option for use as source of randomness [10].

7.0 Conclusion

ECC has shown to be a fast, secure, and computationally cheap alternative to other encryption schemes on mobile devices. Several standards already exist for ECC such as American Nation Standard Institute (ANSI), National Institute of Standards and Technology (NIST), and Standards for Efficient Cryptography Group (SECG) to name a few [2]. There are several industry implementations of ECC schemes, some open source, others commercial.

Among the open source APIs are: OpenSSL, Crypto++, TinyOS, libecc, and Bouncy Castle. Commercial implementations include: Java SE 6, CNG for Windows Vista and Windows Server, MIRACL. Most are larger cryptography libraries with ECC utilities. It is difficult to develop implementations of ECC as Certicom owns most patents pertaining to elliptic curves [4]; a development team must lease the patents from Certicom or face copyright infringement chargers. This is the major roadblock towards the development of ECC as an industry standard. However, as computation power becomes cheaper and keys must become longer, coupled with the more prevalent use of mobile devices, it is predicted that Elliptic Curve Cryptography will see a rise in popularity and use.

Works Cited

[1] "The 'Smartphone Class': Always On, Always Consuming Content," 2 May 2012. [Online]. Available: http://www.emarketer.com/Article.aspx?R=1009014&ecid=a6506033675d47f881651943c21c5ed4.

[2] D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, New York, New York: Springer-Verlag, 2004.

[3] D. Johnson, A. Menezes and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm," [Online]. Available: http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf.

[4] National Security Agency, "Case for Elliptic Curve Cryptography," 19 Jan. 2009. [Online]. Available: http://www.nsa.gov/business/programs/elliptic_curve.shtml.

[5] G. Deepak and Pradeep, "Issues and Limitation of Mobile Computing," *Int.J.Computer Technology & Applications,* vol. 3, pp. 177-181, Jan-Feb 2012.

[6] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," [Online]. Available: http://www.cs.cmu.edu/~coda/docdir/podc95.pdf.

[7] Certicom Research, "Standards for Efficient Cryptography: Elliptic Curve Cryptography," Semptember 1999. [Online]. Available: http://www.secg.org/collateral/sec1.pdf.

[8] W. Chou, "Elliptic Curve Cryptography and Its Application to Mobile Devices," [Online]. Available: http://www.cs.umd.edu/Honors/reports/ECCpaper.pdf.

[9] Q. Zhou, X. Liao, K.-w. Wong, Y. Hu and D. Xiao, "True random number generator based on mouse movement and chaotic hash function," September 2009. [Online]. Available: http://www.sciencedirect.com/science.

[10] C. Lauradoux, J. Ponge and A. Roeck, "Online Entropy Estimation for Non-Binary Sources and Applications on iPhone," June 2011. [Online]. Available: http://hal.inria.fr/docs/00/60/90/65/PDF/RR-7663.pdf.