Discrete Logarithms and Elliptic Curves in Cryptography

Derek Olson and Timothy Urness
Department of Mathematics and Computer Science
Drake University
Des Moines, IA 50311
derek.olson@drake.edu and timothy.urness@drake.edu

Abstract

Since ancient times, there has been a tug-of-war taking place between code makers and code breakers. Only within the last fifty years have the code makers emerged victorious (for now that is) with the advent of public key cryptography. This paper surveys the mathematical foundations, shortcomings, and novel variants of the "first" public key cryptosystem envisioned by Whitfield Diffie, Martin Hellman, and Ralph Merkle in 1976. The system they developed, Diffie-Hellman key exchange, relied on the difficulty of taking discrete logarithms in the finite fields $\mathbb{Z}_p$, where p is prime. While relatively secure, methods known as the index calculus exist to crack Diffie-Hellman key exchange in less than exponential running time. This has led to the use of elliptic curves in analogous cryptosystems. The basic theory underlying these elliptic curve cryptosystems is presented as well as a comparison of these systems with standard RSA encryption.

1 Introduction

Since antiquity, humans have been using codes to communicate secretly with one another. Whether sending battle plans to distant armies in ancient Rome or paying a utility bill over the Internet, the ability to communicate securely, quickly, and easily has consumed generations of human thought. The term cryptography, which is derived from the Greek word kryptos, meaning hidden, is used to describe the creation of these secure communication channels.

Along with this ability to communicate securely, the desire to break or crack this security has naturally become a valuable commodity leading to a battle between the code makers and code breakers, which has been raging for millenia and is well chronicled in [1]. Up until the mid 1970s, the code makers and code breakers were locked in a virtual stalemate. The code makers developed numerous means of communicating securely—ranging from Roman Caesar ciphers that shifted each letter in the alphabet a certain number of letters to the German enigma machine of World War II—only to have them broken by the code breakers. The common element of both of these systems and, in fact, every cryptosystem up to the 1970s, was the dependence upon a *private key*. This meant that before establishing secure communications, the two parties had to have first agreed to a shared key—some piece of information such as a codeword or number—that had to be kept secret at all costs. If the Germanic Roman legion were communicating with the Sabine Legion, each would have to know how many places to shift the letters in the message to encode and decode the message successfully. The private key in this case is the shift amount resulting in only 25 possible keys and a trivial cryptosystem to crack. Fast forwarding to the start of the 20th century, the German enigma machine allowed for approximately 10,000,000,000,000,000 possible keys in its original form to about 159,000,000,000,000,000,000 possibilities in its late World War II form yielding a much more secure cryptosytem [1]. The problem that arose was that this private key had to be distributed to every party involved in the communications, and seemingly the only way to distribute the key securely was by physically exchanging keys through couriers or face to face meetings, which was was cost prohibitive for governments and the richest private industries let alone the ordinary person.[1]

This problem became known as the *key distribution problem*, and many thought it was not solvable. Then in 1974 a partnership began that has, for the present, shifted the balance of the struggle firmly in the grasps of the code makers [1]. That partnership was between Whitfield Diffie and Martin Hellman and later included Ralph Merkle. The three men envisioned a means of encoding and decoding messages that did not involve a private key but instead relied upon a *public key* that could be broadcast to adversaries as well as allies. The system they developed only allowed for the exchange of keys—not textual messages—but it paved the way for bona fide public key cryptosystems such as RSA and ElGamal encryption [1].

[1]Though the idea of physically exchanging keys sounds absurd to us now, it was not even 50 years ago when banks sent their most trusted employees around the globe with private keys for the next week of banking activity held in a "padlocked briefcase" [1].

Known as Diffie-Hellman-Merkle key exchange or just Diffie-Hellman key exchange, their system still provides a very secure means of exchanging keys between two parties and relies on the difficulty of a mathematical problem known as the discrete logarithm problem. This paper explores just how difficult this problem is by detailing a well-known algorithm called the index caclulus to crack Diffie-Hellman key exchange in subexponential time [2]. It then details alternative cryptosystems based on the use of elliptic curves for which no subexponential algorithms have yet been discovered. As we shall see, these elliptic curve based cryptosystems offer significant advantages over even RSA encryption, which could soon lead to elliptic curve cryptography (ECC) becoming the standard in the cryptography industry.

2 Diffie-Hellman Key Exchange

The original Diffie-Hellman key exchange algorithm was published in 1976 [3]. The mathematical justifications of the algorithm require only a basic understanding of abstract algebra, and the process is quite simple. Adding to its appeal are the efficient numerical algorithms for performing the various steps. However, as mentioned previously, a well known shortcoming of Diffie-Hellman key exchange is the existence of subexponential running time algorithms to crack it. Both the process of Diffie-Hellman key exchange and the algorithms to break it are offered in the next two sections.

2.1 The Algorithm

In order to explain Diffie-Hellman key exchange, we will employ the usual scenario involving Alice, Bob, and Eve. Alice and Bob are trying to share a secret message m, while Eve, an adversary with less than honorable intentions, is attempting to intercept this message. Alice must somehow *encrypt* the message m in such a way that Bob can *decrypt* the message to recover the original message m, but in such a way that Eve cannot discover m. The algorithm due to Diffie, Hellman, and Merkle that allows for the exchange of keys is as follows and can be found in most introductory texts on cryptography including [2].

1. Alice and Bob choose a large prime N and an integer $p \bmod N$. Both of these can be public information, meaning Eve knows this value.
2. Alice chooses a secret integer m, and Bob chooses a secret integer n. Neither Alice nor Bob knows the other's integer.
3. Alice computes $Q_m \equiv p^m \bmod N$, and Bob computes $Q_n \equiv p^n \bmod N$.
4. Alice sends Q_m to Bob, and Bob sends Q_n to Alice.
5. Alice calculates $Q \equiv (Q_n)^m \bmod N$ while Bob calculates $Q' \equiv (Q_m)^n \bmod N$.

What turns out to be the case is that $Q \equiv Q'$ so upon completing this algorithm both Alice and Bob have a shared value Q, or a key, that can be used to communicate securely. The equality of Q and Q' over $\mathbb{Z}_N$ can be seen in

$$Q \equiv (Q_n)^m \equiv (p^n)^m \equiv p^{nm} \equiv p^{mn} \equiv (p^m)^n \equiv (Q_m)^n \equiv Q' \bmod N, \tag{1}$$

where $(p^n)^m = p^{nm} = p^{mn}$ holds in an arbitrary group for integers m and n. In this case, the group (actually a field) is $\mathbb{Z}_N$. Also of note is that only steps one, three, and five appear to require significant computation time. However, one can use the binary expansion of the exponent and the "Fast Powering Algorithm" so that computing $b^y \bmod N$ requires at most $2\log_2(y)$ multiplications in steps three and five [2]. Step one requires the generation of a large prime integer which can be a substantial task, but in practice this can be performed by a trusted third party [2]. Hence Diffie-Hellman key exchange is both simple to understand and efficient to implement.

It should be noted that although Alice and Bob can use Diffie-Hellman key exchange to share a key with another, they are not actually sharing some message, m. Once they have a shared key, they must then use this key in another cryptosystem to exchange information. The first such system to use Diffie-Hellman key exchange to send messages was the ElGamal system put forth in [4].

2.2 Cracking Diffie-Hellman

With Alice and Bob using Diffie-Hellman key exchange to generate a shared key to be used in a public key based cryptosystem, the question is whether an adversary, Eve, can also determine the shared key and hence steal their communique. Using the example in the previous section, Eve can attempt to solve $p^x \equiv Q_n \bmod N$ or $p^y \equiv Q_m \bmod N$ for either x or y and then compute either $(Q_m)^x = Q$ or $(Q_n)^y = Q'$. In other words, Eve must find the integer x (y) such that raising p to the xth (yth) power yields $Q_n \bmod N$ ($Q_m \bmod N$). Over the real numbers, this value would be known as the base p logarithm, and over the finite field $\mathbb{Z}_N$, this value is called the base p discrete logarithm. We denote this as $x = \log_p(Q_n)$. Finding this value is known as the discrete logarithm problem, and the difficulty of doing so is what underlies Diffie-Hellman key exchange.

For nearly a decade, the discrete logarithm problem was thought to be sufficiently difficult to require exponential running time algorithms. In actuality, a method to calculate discrete logarithms and therefore solve Diffie-Hellman key exchange existed prior to the publication of Diffie-Hellman key exchange. This method is known as the index calculus and was put forth by Western and Miller in [5]. Their work included discrete logarithms for up to six digit prime numbers. For ease of understanding, we present the index calculus as it appears in [2].

The algorithm relies on several key concepts from number theory and on being able to write discrete logarithms as linear combinations of "smaller" logarithms. When enough of these linear combinations are found, the unknown, "smaller" logarithms can be solved for as if they were variables in a linear system.

The first concept from number theory that will be needed is that of a B-smooth number.

Definition 1. *A number x is B-smooth if each prime factor of x is less than or equal to B.*

For example, $36 = 2^2 \times 3^2$ is 3-smooth, 5-smooth, 7-smooth, and 31-smooth. Meanwhile, $49 = 7^2$ is 7-smooth but not 5-smooth. To start the algorithm, we will choose a value for B. The exact choice for B will be apparent at the end. The next step is to find a value of k such that $Q_n * p^{-k} \bmod N$ is B-smooth. We take as given that this can be done via a brute force approach starting with $k = 1, 2, \ldots$. The argument for why this can be done is a probabilistic one which is justified using polynomial fields in [6]. With this value of k, we will be able to write $Q_n * p^{-k}$ as the product of the primes less than or equal to B, y_i, with appropriately chosen exponents,

$$Q_n * p^{-k} \equiv \prod_{y_i \leq B} y_i^{e_{y_i}} \bmod N. \tag{2}$$

One useful property about taking discrete logarithms is that they also possess the properties familiar from taking logarithms on real numbers: the logarithm of a product is the sum of the logarithms of the multiplicands, and exponents may be taken outside of the logarithm as scalar multiples. Taking the base p discrete logarithm of equation (2) and applying these properties produces

$$\log_p(Q_n) - k \equiv \sum_{y_i \leq B} e_{y_i} * \log_p(y_i) \bmod (N-1). \tag{3}$$

Note that the modulus is now $N - 1$ because taking discrete logarithms is a mapping on Z_{N-1}. (The discrete logarithm of 0 is undefined just as the logarithm of 0 is undefined over the real numbers.) Also observe that we appear to be no further along than when we began. We sought a way to compute the base p discrete logarithm of Q_n and now need to compute the base p discrete logarithm of the prime numbers, y_i. This is where the insight of the index calculus comes in. We take a "random selection of exponents," j, and check to see if $p_j \equiv p^j \bmod N$ is B-smooth [2]. If it is not, then we disregard this value for j and try another random value. Upon finding a value for j such that $p_j \equiv p^j$ is B-smooth, we proceed to write p_j as the product of primes less than or equal to B,

$$p^j = p_j \equiv \prod_{y_i \leq B} y_i^{f_{y_i}(j)} \bmod N \tag{4}$$

where $f_{y_i}(j)$ is the exponent that appears on y_i in the prime factorization of j. Now taking the base p discrete logarithm of (4) gives us

$$\log_p(p_j) \equiv j \equiv \sum_{y_i \leq B} f_{y_i}(j) \log_p(y_i) \bmod (N-1). \tag{5}$$

If enough equations of the form of (5) can be found, then we will obtain a system of linear equations whose variables we are trying to solve for are of the form $\log_p(y_i)$

$$\begin{aligned}
j_1 &= \left[f_{y_1}(j_1) \log_p y_1 + \cdots + f_{y_k}(j_1) \log_p y_k\right] \bmod (N-1) \\
j_2 &= \left[f_{y_1}(j_2) \log_p y_1 + \cdots + f_{y_k}(j_2) \log_p y_k\right] \bmod (N-1) \\
&\vdots \\
j_k &= \left[f_{y_1}(j_k) \log_p y_1 + \cdots + f_{y_k}(j_k) \log_p y_k\right] \bmod (N-1).
\end{aligned}$$

The standard theory from linear algebra tells us we need as many equations as unknowns to solve this system. There is an unknown for each prime less than or equal to B, and the prime number theorem tells us that this number is approximately $\pi(B) \approx \frac{B}{\ln B}$. Once we solve for the unknown discrete logarithms of the primes less than B, these values can be substituted into equation (3) to discover $\log_p(Q_n)$. The question then becomes how many random integers, j, must be generated to find enough of these equations. The answer to this is provided by the following probabilistic result [2].

Theorem 1. *If $B = e^{\sqrt{1/2(\ln N)(\ln\ln N)}}$, it requires approximately $e^{\sqrt{2(\ln N)(\ln\ln N)}}$ random numbers to find $\pi(B) \approx \frac{B}{\ln B}$ B-smooth numbers.*

Therefore, if we choose B to be approximately $e^{\sqrt{1/2(\ln N)(\ln\ln N)}}$ at the start of this algorithm, it will take roughly $e^{\sqrt{2(\ln N)(\ln\ln N)}}$ random integers to solve the discrete logarithm problem which leads us to

Theorem 2. *The index calculus method solves the Discrete Logarithm Problem in subexponential time: $\mathcal{O}\left(e^{c\sqrt{(\log N)(\log\log N)}}\right)$, where c is a constant.*

Using the index calculus, our adversary Eve is therefore able to solve the discrete logarithm problem and hence crack Diffie-Hellman key exchange in less than exponential time. Furthermore, methods employing B-smooth numbers can also be used to crack RSA encryption in subexponential time [2]. While no "easy," polynomial time algorithms exist for either of these encryption methods, the existence of the subexponential algorithms did spur the search for alternative, "harder" systems to crack.

3 Elliptic Curve Cryptography

Returning to the scenario of Alice trying to share a secret message with Bob, Diffie-Hellman key exchange does provide a reasonable amount of protection from eavesdroppers such as Eve. There are no polynomial running time algorithms Eve can use to quickly discover their information. But suppose that Alice and Bob are sharing vital information that requires even more security assurances—perhaps they are exchanging bank account numbers and pin numbers. These are likely to remain the same for some time so it would be desirable to have an encryption scheme as easy to implement as Diffie-Hellman key exchange but have no known subexponential algorithms to solve it. One such method that has garnered significant interest and may soon surpass RSA encryption as the standard high-powered encryption technique is the use of elliptic curves in cryptography, first pioneered by Koblitz and Miller [7]. We first describe what an elliptic curve is and then one way in which they are used in cryptography called elliptic curve Diffie-Hellman key exchange.

3.1 Elliptic Curves

Though the formal study of elliptic curves is best understood in terms of algebraic topology, a basic understanding can be gained using elementary group theory [2]. Once this is done,

the idea behind using elliptic curves in cryptography is to use the *elliptic curve group* in place of the group, $\mathbb{Z}_p$, in Diffie-Hellman key exchange.

Definition 2. *Let $\mathbb{F}$ be a field. An elliptic curve over $\mathbb{F}$ is a curve defined by an equation of the form*

$$y^2 = x^3 + ax + b,$$

where the discriminant, $-16\left(4a^3 + 27b^2\right) \neq 0$ and $a, b \in \mathbb{F}$.

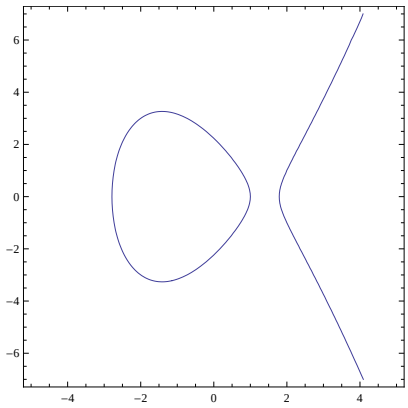


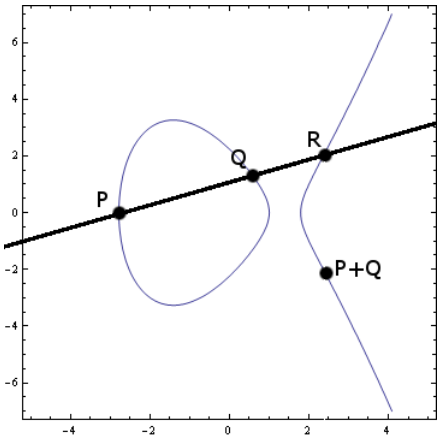
Figure 1: An elliptic curve given by $y^2 = x^3 - 6x + 5$ over $\mathbb{R}$.

It is not immediately apparent how to define an operation on points on an elliptic curve that results in an algebraic group, but when the field in question is the real numbers, we can use geometry to help define this operation.

To find the sum of two (distinct) points, P and Q, we use the following steps

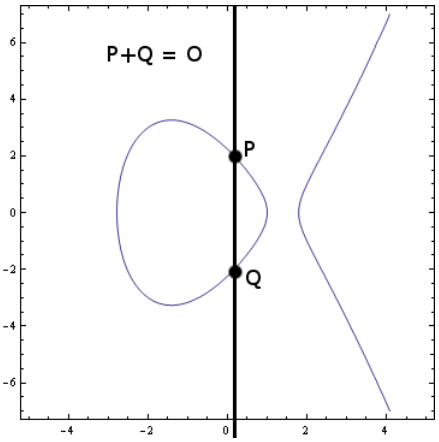
1. Construct the line determined by P and Q. For the moment, let us assume that this line is not vertical. Then the line will intersect the elliptic curve in a a third point, R. [2]

2. Reflect the point R across the x-axis, about which elliptic curves are readily seen to be symmetric. The resulting point is defined as the sum, $P + Q$.



[2]This is because substituting the equation for a line, $y = mx + c$, for y into the equation for an elliptic curve will produce a third degree polynomial in x that has two real roots since it passes through P and Q and hence must have a third real root.

There are two caveats to consider in this process. Should the points P and Q not be distinct, then the tangent line at $P = Q$ is used in place of the line connecting P and Q.[3] Should the two points lie on a vertical line, then there will be no way to recover a third point of intersection. For this reason a point at infinity, denoted $\mathcal{O}$, is added.



The point at infinity also serves a dual purpose as the identity in the group we are searching for: when adding a point, P, to $\mathcal{O}$, we construct the vertical line through P which intersects the elliptic curve in the point that is the reflection of P across the x-axis. To obtain the final sum $P + \mathcal{O}$, we then follow the second step above and reflect this point across the x-axis yielding the original point P. Using basic algebra, a simple algorithm to compute the sum of two points on an elliptic curve can then be defined. We present the algorithm as it appears in [2], referred to as the elliptic curve addition algorithm. A full derivation can also be found there.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on an elliptic curve.

Case 1: If $P = \mathcal{O}$, then $P + Q = Q$

Case 2: If $Q = \mathcal{O}$, then $P + Q = P$

Case 3: If $x_1 = x_2$ and $y_1 = -y_2$, then $P + Q = \mathcal{O}$.

Case 4: If $P \neq Q$,

i. Set $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$

ii. Set $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_1 - x_3) - y_1$

iii. $P + Q = (x_3, y_3)$

Case 5: Else, $P = Q$

i. Set $\lambda = \frac{3x_1^2 + A}{2y_1}$

ii. Set $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_1 - x_3) - y_1$

iii. $P + Q = (x_3, y_3)$

[3] In this case, the point $P = Q$ will correspond to a double root of the polynomial obtained when substituting the equation for the tangent line into the equation for the elliptic curve.

Using geometry in the case of elliptic curves over $\mathbb{R}$, it is not difficult to see that the set of points on an elliptic curve plus the point at infinity along with addition as described constitutes an algebraic group. What the equations in the elliptic curve addition algorithm allow us to do is define an algebraic group, called the elliptic curve group, over arbitrary fields.

Theorem 3. *Let $\mathbb{F}$ be a field. Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}$ so $a, b \in \mathbb{F}$. Then the set of points $(x, y) \in \mathbb{F} \times \mathbb{F}$ that satisfy the elliptic curve equation along with the point $\mathcal{O}$ form a group under the elliptic curve addition algorithm.*

For example, if we take $\mathbb{Z}_5$ to be our field with $y^2 = x^3 + 3x + 2$ as our elliptic curve, then a brute force algorithm can determine which of the 25 possible points in $\mathbb{Z}_5 \times \mathbb{Z}_5$ is on the elliptic curve. Doing so, we find this elliptic curve consists of the five points $(1, 1), (2, 1), (1, 4), (2, 4)$, and $\mathcal{O}$. In the next section, we see that the number of points on an elliptic curve and finding those points is intimately related to the difficulty underlying the elliptic curve version of Diffie-Hellman key exchange.

3.2 Elliptic Curve Diffie-Hellman Key Exchange

With the working knowledge of Diffie-Hellman key exchange that we already have, it is very simple to extend this process to involve the elliptic curve group in what is known as elliptic curve Diffie-Hellman key exchange. In the original Diffie-Hellman key exchange, a large prime N and an integer $p \bmod N$ were chosen to be shared between Alice and Bob. Both then chose secret, individual integers m and n. The very simple idea behind elliptic curve Diffie-Hellman key exchange is to replace $\mathbb{Z}_N$ by the elliptic curve group over $\mathbb{Z}_N$ and have Alice and Bob choose a shared elliptic curve, E, and a starting point, P, on that curve. The process then proceeds exactly as before with the exception of a different group operation—that of adding points on an elliptic curve via the addition algorithm.

1. Alice and Bob agree to use an elliptic curve, E, over a field $\mathbb{Z}_N$ with N prime and a starting point, P, on that curve.
2. Alice chooses an integer m and Bob an integer n.
3. Alice computes $Q_m = \underbrace{P + P + \cdots + P}_{m \text{ times}}$, and Bob computes $Q_n = \underbrace{P + P + \cdots + P}_{n \text{ times}}$.
4. Alice sends Q_m to Bob, and Bob sends Q_n to Alice.
5. Alice calculates $Q_{nm} = \underbrace{Q_n + Q_n + \cdots + Q_n}_{m \text{ times}}$ while Bob calculates $Q_{mn} = \underbrace{Q_m + Q_m + \cdots + Q_m}_{n \text{ times}}$.

Equation (1) holds for any group so it can be directly applied to the above to prove that Alice and Bob have a shared key $Q_{nm} = Q_{mn}$. This can then be used in a public key cryptosystem, but the question remains as to how secure this cryptosystem actually is. Presently, the answer to that question is that elliptic curve Diffie-Hellman key exchange is actually more

secure than either Diffie-Hellman key exchange or RSA encryption, as there are currently no known algorithms capable of solving elliptic curve Diffie-Hellman key exchange in anything less than exponential time for well chosen elliptic curves. The reason that methods like the index calculus cannot be used on elliptic curves is that points on an elliptic curve cannot be broken up into sums of "smaller" points as discrete logarithms were broken up into sums of smaller logarithms.

The elliptic curve must, however, be chosen judiciously, for there are algorithms that can crack elliptic curve Diffie-Hellman key exchange in less than exponential time when the number of points in the elliptic curve group over $\mathbb{Z}_p$ is equal to $p, p-1, p+1$, or is a product of small primes [7]. Because of this, the number of points on the elliptic curve must be checked prior to using it. As we did in the case of $y^2 = x^3 + 3x^2 + 2$ over $\mathbb{Z}_5$, this can be done by a brute force algorithm, but that would be impractical over fields of large prime order. In practice, a polynomial time algorithm due to Schoof, Elkies, and Atkin is used to compute the number of points in the elliptic curve group [7].

It is in fact this number of points for which the difficulty of cracking elliptic curve Diffie-Hellman key exchange depends. In order for Eve to breach its security, she must solve the discrete logarithm problem over an elliptic curve group. Without methods such as the index calculus, the best algorithms that Even can use have exponential running time dependent upon the number of points in the elliptic curve group—which itself depends on the size of the prime number N.

4 ECC vs. RSA

Elliptic curve Diffie-Hellman key exchange is but one example in an ever-growing field known as elliptic curve crytptography (ECC) that applies elliptic curves in various ways to cryptography. We have already seen an example of ECC that requires more time to solve than either RSA encryption or Diffie-Hellman key exchange. But there are other desirable properties that a cryptosystem must have for it to be practical. These include algorithm set-up costs, storage requirements, speed and efficiency, and standardization. We conclude this paper by offering an overview of the advantages and disadvantages of general elliptic curve cryptography versus RSA encryption.[4] This comparison will indicate that elliptic curve based cryptosystems could soon become the standard tool used in encryption applications.

In a paper published on the RSA website in 1998 [8], Robshaw and Yin analyzed the setup costs, security, implementation, and performance of elliptic curve cryptography versus RSA encryption. The setup costs for ECC are more intensive because it involves first obtaining an underlying field, then judiciously choosing an elliptic curve and starting point. Meanwhile, initiating RSA consists only of generating a large prime integer. We have already noted that elliptic curve Diffie-Hellman key exchange is more secure than RSA en-

[4] An overview of RSA can be found in [2].

cryption since there exist no subexponential algorithms to crack the elliptic curve version. This gives ECC a significant advantage in storage requirements as secure public keys need only be 160 bits to offer the same security as 1024 bit RSA keys [8]. Elliptic curve addition can also be performed quite efficiently using the "Double-and-Add" algorithm [2] making it similar in efficiency to RSA, which uses the aforementioned "Fast-Powering" algorithm. Although the exponentiation required in RSA encryption can be performed faster, a similar paper to the Robshaw and Yin paper on the NSA website [9] claims that ECC is actually more efficient to implement than RSA when factoring in the gains in security.

One distinct disadvantage of elliptic curve cryptography is the proprietary nature of it. Certicom, a Canadian based research company, currently possess well over 100 patents on elliptic curve cryptography making usage and wide-spread implementation difficult [9]. Furthermore, there has been a lack of standardization with ECC, whereas RSA was standardized very early on [8]. Despite this, elliptic curve cryptography has shown enough promise for the following to appear on the NSA website, "For protecting both classified and unclassified National Security information, the National Security Agency has decided to move to elliptic curve based public key cryptography" [9].

5 Conclusion

This paper has surveyed the history of public key cryptography from the first key exchange system developed by Diffie, Hellman, and Merkle to the more recent methods dependent upon elliptic curves. Though we have isolated the lack of a subexponential running time algorithm to break elliptic curve Diffie-Hellman key exchange as a major attraction to this form of encryption, it must be remembered that there is no proof that one does not exist. With the inherent secrecy involved in developing cryptographic applications, it could be that one already exists. For now, ECC provides what appears to be the next evolution in the code maker's toolbox, possessing many of the requirements needed in an encryption algorithm. But until an absolutely secure cryptosystem can be developed, the proverbial tug-of-war between the code makers and code breakers will no doubt continue.

References

[1] S. Singh, *The Code Book: The Evolution of Secrecy*. New York, NY: Doubleday, 1999.

[2] J. Hoffstein, J. Pipher, and J.H. Silverman, *An Introduction to Mathematical Cryptography*. New York, NY: Springer, 2008.

[3] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. IT-22, no. 6, 644-654, 1976.

[4] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Information Theory*, vol. 31, no. 4, 469-472, 1985.

[5] A.E. Western and J.C.P Miller, *Tables of Indices and Primitive Roots*. Cambridge: Cambridge University Press, 1968.

[6] M. E. Hellman and J. M. Reyneri, "Fast computation of discrete logarithms in GF(q)," in *Advances in Cryptology: Proceedings of CRYPTO 82*, D. Chaum, R. Rivest, and A. Sherman, Eds. New York, NY: Plenum Press, 1983, pp. 3-13.

[7] M. Stein, *Elementary Number Theory: Primes, Congruences, and Secrets: A Computational Approach*. New York, NY: Springer, 2008.

[8] M.J.B. Robshaw and Y.L. Yin, "Overview of Elliptic Curve Cryptosystems," RSA Laboratories, 1997. [Online]. Available: http://www.rsa.com/rsalabs/node.asp?id=2013 [Accessed: Dec. 10, 2009].

[9] National Security Agency, "The Case for Elliptic Curve Cryptography," National Security Agency, Jan. 15, 2009. [Online]. Available: http://www.nsa.gov/business/programs/elliptic_curve.shtml [Accessed: Dec. 10, 2009].