A Thorough Analysis of Elliptic Curve Cryptography

Jack Valladares
Georgia College & State University
Milledgeville, GA 31031, United States
john.valladares@bobcats.gcsu.edu

ABSTRACT

In this work, the Elliptic Curve Cryptography Algorithm is analyzed. The Elliptic Curve Algorithm is one that has great promise and potential in the world of System Security but is also met with a lot of distrust and still has a lot of room for evolution. Therefore, this work highlights the algorithm itself, briefly touches upon its history and major use cases, and then discusses the major strengths and potential obstacles of the algorithm in order to provide a holistic analysis of the algorithm from a perspective of system security.

I. PUBLIC KEY CRYPTOGRAPHY

Cryptography algorithms are the backbone of systems security. Finding ways to encrypt data and make it illegible to anybody without proper authorization in a lightweight, efficient, and secure manner is a part of the trade that continues to be a seemingly endless struggle between cryptographer and hacker, and a puzzle with many different existing algorithms that all have their own merits and downfalls. One such algorithm is Elliptic Curve Cryptography (ECC), a relatively new player in the field that comes with great promises of quick runtimes and small bit sizes, but one that still does not have the mainstream trust of its more well-established counterparts.

In order to understand what ECC is, as well as what it is up against, it is important to first know what a Public-key cryptosystem even is. Public-key cryptography allows one to accomplish two different security goals: encryption/decryption and nonrepudiation, which prevents the owner of the data from claiming they never sent the data and keeps it from being altered. Essentially, these goals combined means that a Public-key cryptosystem allows a sender to send securely encrypted data that can only be viewed by an authorized party, and a receiver to view these encrypted messages and have assurance that the information has not been altered and that the sender's name is attached to it. The structure of this type of cryptography involves the use of two different "keys" – a public key and a private key. Each user has an associated public key, which can be accessed and used by any sender to encrypt and send them a message. Once the encrypted message is sent, the receiver's private key comes into play. This key is not publicly available, and acts as the variable necessary to decrypt the message. Therefore, any potential interceptors cannot tamper with or read the message unless they hold the associated private key, so only the receiver has the toolset to decrypt a message sent to them. With this structure in mind, the goal of any good public-key cryptosystem is to create a strong and quick trapdoor function. That is to

say, a function that is easy to compute one way, but nearly impossible the other without the right variables.

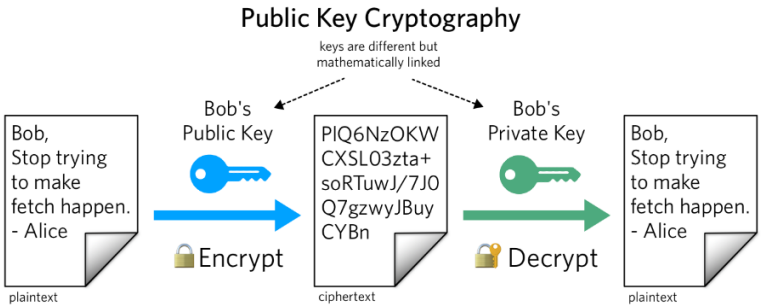
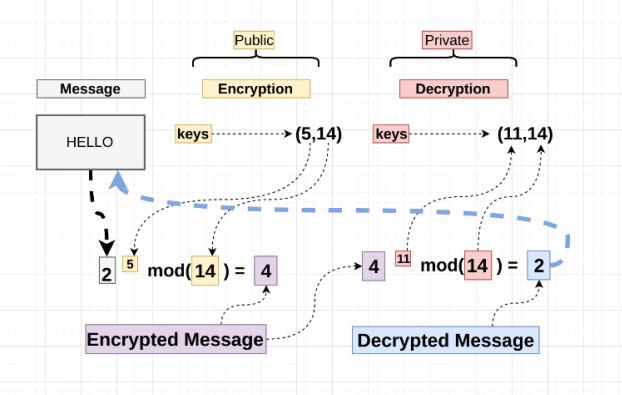


Figure 1.1 The Basic Structure of Public Key Cryptography (Robinson, 2018)

A message that is encrypted via a public key should be uncrackable without the associated private key or a billion years of computation.

As public-key cryptography is used in message-sending, which is a massive category that ranges from email to text messaging, it is only to be expected that ECC has several different contemporaries, each with their own design philosophies and strengths. Rivest-Sharmir-Adelman (RSA) is a notable cryptosystem in this category, and also happens to be among the oldest and most well-established. RSA follows the principle of prime number factorization, which is an extremely popular methodology in modern-day cryptography. Two prime numbers are multiplied and the modulus value, or remainder, is given and used to encrypt the input, called the plaintext. This algorithm is as popular as it is intuitive. Multiplication is a relatively low effort operation and satisfies the requirement of being easy to do one way. Finding two specific multiplicands given the result of the operation, however, is a lot harder. This only becomes exponentially more difficult as the numbers get larger, and a computer might need to sift through billions of values to figure out which two prime numbers were used in the operation. Diffie-Hellman is another common public key algorithm that uses this same principle and was developed around the same time as RSA.



The RSA Algorithm (Stories 2017)

II. The Elliptic Curve Algorithm

ECC does not use prime number factorization, and instead, relies on a different approach to accomplish its trapdoor function. This approach is built on the Elliptic Curve Discrete Logarithm problem, which, ironically, does not involve any actual ellipses. Instead, they, "are so named because of the fact that ellipses are formed by quadratic curves. Elliptic curves are always cubic and have a relationship to elliptic integrals in mathematics where the elliptic integral can be used to determine the arc length of an ellipse" (Shankar 39). Two of these curves are typically used in an Elliptic Curve Algorithm. The standard way to generate the first curve is through the equation $y^2 = x^3 + ax + b$, and then the discriminant is drawn out as $D = -(4a^3 + 27b^2)$. It is important to note that this discriminant cannot be zero, or else two or more roots would have coalesced, which would create a single curve instead of two and compromise the security of the algorithm.Once the curve is generated, a prime finite field called F*p* is created, with *p* representing an odd, prime number. A finite field, which is otherwise known as a Galois Field, can, in simple terms, "be defined as a set of numbers that we can add, subtract, multiply and divide together and only ever end up with a result that exists in our set of numbers. This is particularly useful for [cryptography] as [one] can deal with a limited set of extremely large numbers" (Kohli). Any operation performed on this set of *p* values also returns another value in the set, called the closure property, which essentially creates a loop of values that can be used or returned. If a prime finite field called F(5) contains values {0, 1, 2, 3, 4}, for example, any operation performed on 0, 1, 2, 3, or 4 will result in one of those same numbers. Building on this concept, prime fields have a prime number, represented by *p*, of values. The elliptic curve is drawn out over field F*p*, giving it a clear start and end point.

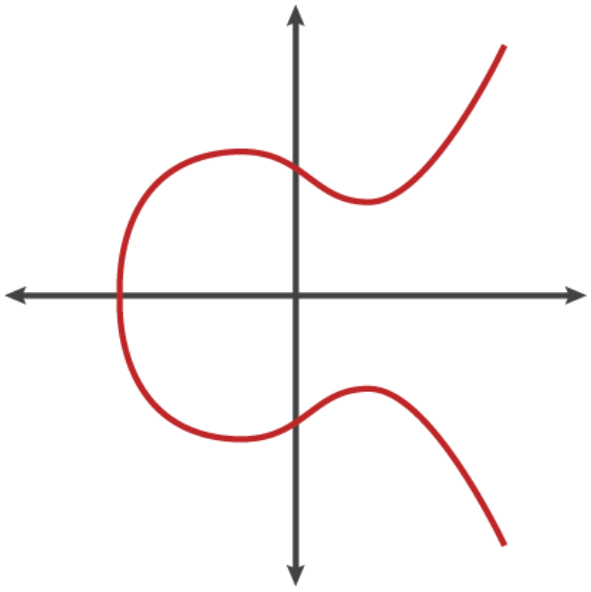


Figure 2.1 A simple ECC Curve Visualized (Sullivan, 2013)

Tightening the parameters of the curve this way adds to the security of the algorithm and prevents it from certain exploits, which will be relevant in a later section. Over the prime finite field, a dot function is used to draw line PQ, which is a tangent line drawn across these points. The x-axis reflected point of the third intersection of line PQ is found, which is denoted as point

R. This process is repeated a multitude of times with the output of each calculation. The next "hop" would use line PR in its dot function and so on. The closure property of the finite prime field also means that the lines all loop through the acceptable values – if it goes too far, it just ends up back in the beginning. The public key of this algorithm is the starting point and the final ending point. If it started with point P and ended on point E after its final calculation, the public key would be points P and E. The private key is the number of hops used to get from point P to point E. This algorithm is a trapdoor function because it is extremely easy to get to the end point given a starting point and a number of hops, but magnitudes harder to find the number of hops given a starting point and an endpoint. Even though the number of hops tends to be an exponentially large number, given the starting point, " one can use an exponentiation trick to find the ending point quite quickly. For example . . . 2P = P dot P and then 4P = 2P dot 2P. This allows [one] to get up to those crazy high calculations exponentially faster." (Wagner) However, even if you know the start point and the end point, it is almost impossible to figure out how many hops were used to get between the two without running trial and error through every single hop along the way. These numbers can get as high as 2^256, so a wannabe hacker would need to compute all of those hops individually to eventually find the one that lands on the end point, which would take a ludicrous amount of time. These properties are why ECC has such a strong amount of security given a low key size and processing overhead, and keeps the algorithm safe from attacks such as brute force attacks.

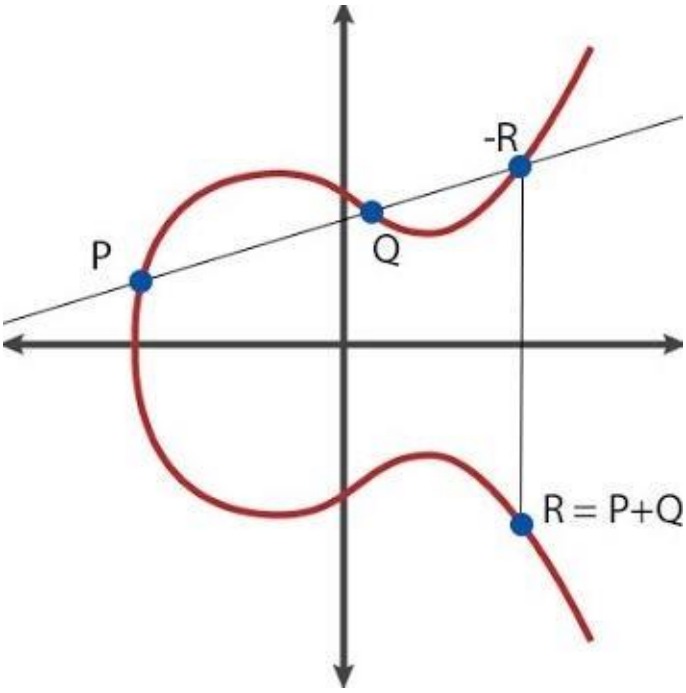


Figure 2.2 Line PQ and Points -R and R (Sullivan, 2013)

III. How does ECC Improve upon other public-key systems?

The ECC Algorithm improves upon the standard prime number factorization of other public-key algorithms in several different ways. The most glaring of these is that ECC allows for smaller key size and faster key generation than RSA and Diffie-Hellman at the same level of security thanks to the nature of the algorithm. These numbers are not by a slim margin, either, as,

“according to some researchers, ECC can yield a level of security with a 164-bit key that other systems require a 1,024-bit key to achieve” (Verma and Ojha). This key size difference only continues to increase exponentially as security requirements increase, too. The NSA requires that top-secret documents be encrypted by an elliptic curve key of at least 384 bits. An equivalent RSA key would require 3072 bits. Information availability, or quick accessibly to information for authorized users, is one of the pillars of system security. Key size becomes a huge limiting factor when given a lower processing overhead, such as in the case of mobile devices and microcomputers, and smaller key size is vital to quickly accessing information that is locked behind closed doors. Thus, the ECC has proven to be preferable as an algorithm that meets this goal.

Computational Efficiency the advantage inherent in ECC and is tied to the low processing overhead. Computational Efficiency is the efficiency at which the computations in the algorithm are performed. On both a hardware and a software level, Elliptic Curves use scalar multiplication, which is a much more viable technique than regular multiplications and the exponentiations in which they are involved. RSA and Diffie Hellman in particular use regular multiplication and exponentiations, and thus, “without any doubt . . . ECC is the stronger and the faster (efficient) amongst the present techniques” (Malik 1465).

RSA KEY LENGTH (BIT)	ECC KEY LENGTH (BIT)
1024	160
2048	224
3072	256
7680	384
15360	521

Figure 3.1 Equivalent RSA and ECC NIST Recommmended Key Sizes (“Elliptic Curve Cryptography - KeyCDN Support.”)

The strength of the algorithm is key, but it would be limited in many contexts if it required a high power consumption compared to other algorithms. Therefore, power consumption is important to look at when analyzing encryption algorithms, and ECC does extremely well in this department, as “ ECC requires less power for its functioning so it is more suitable for low power applications such as handheld and mobile devices” (Malik 1465). It performs especially well in client-side cases where there is client authentication overhead, and in server-side cases where there is no client authentication overhead, putting it ahead of RSA in both scenarios.

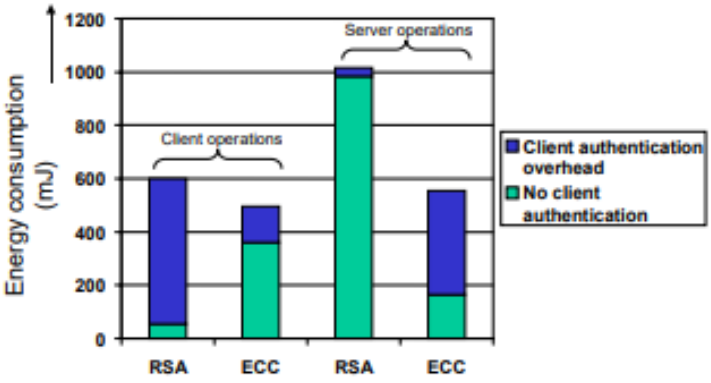


Figure 3.2 Power Consumption of RSA vs ECC in Different Contexts (Potlapally "Analyzing…")

The algorithm also has the advantage that it allows for a diverse selection of curves and finite fields over those curves. The algorithm is not static or limited to a specific set of safe numbers, but instead, "Different finite fields can be used for ECC according to security requirements [and] . . . many different curves can be chosen for the same field by different users" (Malik 1645). In cryptography, the less dynamic and unpredictable an algorithm is, the more susceptible it is to being cracked

In addition to the small key size and high security inherent in the algorithm, there is also precedence of the algorithm being improved upon and evolving over time, which is important in a world where attackers are constantly building new and improved methods and hardware is limited. Because public-key cryptography as a whole relies on exceptionally large numbers, even the fastest of these implementations tend to be very labor-intensive and slow compared to other techniques, and as such, implementing these cryptosystems on devices such as microcontrollers tends to be a challenge. One implementation that seeks to fix problem this in the case of ECC was implemented on the Dalton 8051 and a set of special hardware designed for the task. This hardware, "consists of an elliptic curve acceleration unit (ECAU) and an interface with direct memory access (DMA) to enable fast data transfer between the ECAU and the external RAM (XRAM) attached to the 8051 microcontroller" (Malik 1466).

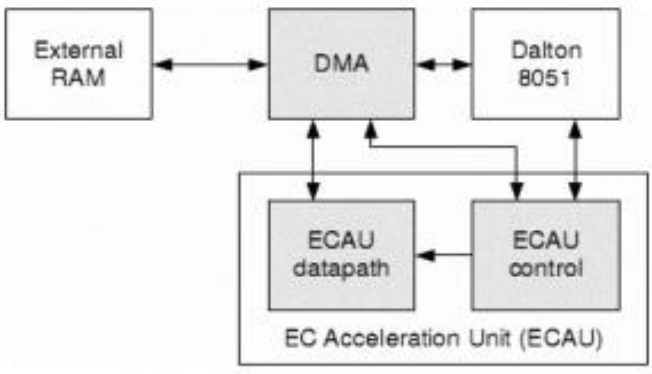


Figure 3.2 The Dalton 8051 ECC Hardware implementation (Malik 1466)

A software implementation was then built to suit this hardware implementation specifically, and as a result, the ECAU allowed for the multiplication to be performed over a field size of 2^{191} in only 118 msec when the Dalton 8051 was clocked at 12MHz. Over a field size of 2^{163}, times of under 100msec were reached.

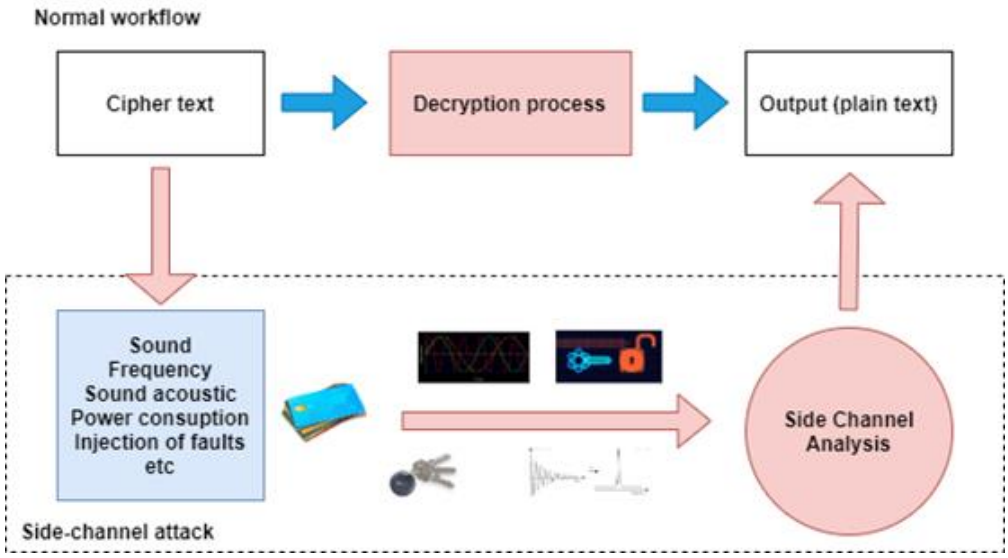
IV. The Use and History of the Algorithm

Given all of these advantages, it is no wonder ECC is currently seeing use in a variety of different contexts. The algorithm has been around since 1985 and was developed independently by Neal Koblitz and Victor Miller and was proposed as an alternative to existing cryptographic protocols. The two men independently developed the idea, which quickly gained interest and was believed to offer better security than other algorithms, and by the early 2000's, was made the NSA's standard suite B algorithm for signatures and encryptions.

Perhaps the most widespread use of ECC is on low-power devices, such as mobile devices. Thanks to its high security at a small key size as well as low power consumption and processing overhead, this is where the algorithm really stands out. Mobile devices such as smartphones are limited in computation and battery life, so robust key sizes and processing overheads like RSA can get out of hand when mobile applications require high levels of security. In addition to mobile devices, some other big names that use ECC include Bitcoin, which uses it to verify ownership of Bitcoins, iMessage, which has signatures that are provided by ECC, DNS information, which can be encrypted through DNSCurve, and SSL/TLS, which use it as the preferred method of authentication for secure web browsing. Even the US Military uses ECC for encryption of sensitive information.

V. Vulnerabilities of the Algorithm and Solutions to These Problems

ECC is not without its own set of vulnerabilities and challenges that need to be overcome, however, and like any other algorithm, every implementation is not created equal. One type of attack that can be devastating to an ECC cryptosystem is the side-channel attack. Side-Channel happen when an attacker analyzes information from the physical implementation of the cryptosystem. Side-Channel attacks can, "include a variety of attacks, such as simple timing attacks, simple power attacks, differential power attacks and fault analysis" (Veronika 2016). Timing attacks, for example, happen when the attacker actually measures the time difference between peaks in power consumption. An oscilloscope is used for attacks performed in this manner. The secret key can actually be found this way because variance in operations and input values will cause a large difference in the time between these peaks. Then there are power attacks, where are in the same vein as timing attacks but instead look at the shape and amplitude of the peaks.



A Diagram Demonstrating a Side-Channel Attack (Tavares 2020)

Side-Channel attacks can be combatted through the Montgomery Power Ladder. Powering algorithms are a vital part of public-key cryptosystems as a whole and are what allow the exponentiations to be performed. There are a number of these algorithms, but the Montgomery Power Ladder, developed by Peter Montgomery, was developed for getting Elliptic Curve Cryptography done in a fast, efficient, and secure way that could not be done with standard, binary algorithms of the past. The Montgomery Power ladder, in summary, speeds up scalar multiplication in the context of Elliptic Curve Cryptography and has since been expanded into a broader array of contexts. The standard algorithm that was used for this purpose was the square-and-multiply algorithm, which appears to be superior to the Montgomery Ladder upon a glance. However, when the properties of the algorithm are further analyzed, there are several clear advantages it has over its competition. Primarily, R0/R1 = g is maintained, which means that operations only need to be performed over the X-Axis, and thus, multiplications are saved in the process. In addition, the algorithm is inherently built to be parallelized, as the different multiplications performed can be evaluated independently when the formula is written out. Therefore, a parallelized Montgomery Ladder can be easily implemented with a consistent 200% speed factor over the standard algorithm. Finally, the two multiplications performed within each iteration of the algorithm share a common operand - R0 when Kj = 0 and R1 when Kj = 1. Therefore, the “common-multiplicand multiplication” method can be applied, which allows one to rewrite the multiplications involved with AND and XOR logic operators. This can lead to serious optimization within the right hardware types, giving the Montgomery Ladder and Elliptic Curve Cryptography a large advantage over the competition. These properties also give the algorithm huge security benefits in the form of being resistant to side-channel attacks without the need for dummy operations, thanks to its high regularity, and fault attacks, thanks to the lack of any dummy operations. Therefore, the Montgomery Power Ladder acts as a great way to keep ECC cryptosystems safe from Side-Channel Attacks of all different types, as well as makes the algorithm more efficient as a bonus. A limiting factor in implementing the Montgomery Powering Ladder, however, is that not every implementation of ECC supports the use of the ladder. Therefore, if one is afraid of Side-Channel Attacks, verifying which implementation they are using is a must for maximum security.

In addition to the Montgomery Powering Ladder, dummy adds can be used to combat certain types of Side-Channel Attacks. Dummy Adds are operations that act as an ignored variable. This equalizes the number of processor operations performed regardless of what the secret key value is, and thus, the differences cannot be measured. Adding an amount of entropy to the secret key, randomizing the projected coordinates, and disguising group points are other notable strategies that combat DPA-type side-channel attacks as well.

The other major existing type of attack that threatens ECC implementations is the twist-security attack. These attacks will succeed when several different conditions are met in the cryptosystem, and the result is the victim's private key being leaked. Usually, "during a twist attack, the malicious party shares a carefully selected public key that does not lie on the agreed-upon ECC curve and that will lead to a shared key that can be easily reversed. After the victim computes a shared key (computed out of the victim's private key and the malicious public key) and computes a hash out of the shared key, the malicious party is able to extract the victim's secret key" (Stolbikova 2016). These come in their own subcategories as well, such as, small-subgroup attacks, standard invalid-curve attacks and even invalid-curve attacks that target the aforementioned Montgomery ladders. Small-subgroup attacks work by using a small order point as the public key in order to just enumerate the private key. Standard invalid curve attacks are more dangerous and happen when the attacker finds a small order point on an elliptic curve with a different constant coefficient than the one being attacked. These can be combatted through the Montgomery Powering Ladder. However, this type of attack has another variant that also specifically targets the Montgomery Powering Ladder, requiring other measures to be taken to fully protect the cryptosystem.

Twist-Security attacks, though they can even bypass measures such as the Montgomery Powering Ladder in certain cases, can be mitigated in a relatively simple way. Since these attacks rely on manipulating the algorithm and finding mathematical weak-points, the key to combatting Twist-Security is carefully selecting curves that are not easily exploitable, as well as validating the parameters of the operation. With tight parameters and a strong curve, small order points and duplicate curves with different coefficients will not give attackers proper insight to figure out the secret key.

There is also the issue of potential future attacks. The realm of quantum computing is one that threatens ECC and cyber security as a whole. Even though, "quantum computing is already facing a large variety of problems, such as its poor decoherence rates, error correction issues, state preparation issues and problems with quantum gates, its advancement may bring additional challenges to ECC once it becomes a technological reality instead of the theoretical concept it is today" (Stolbikova 2016). Shor's and Grover's algorithms are two of the main threats faced by cryptology in the world of quantum computing. Shor attacks turn factoring, the foundation upon which trapdoor functions lie, into a trivial operation and make uncovering a secret key easy. Grover attacks, however, are what threaten ECC especially. Grover's algorithm allows for brute force attacks to be performed through a uniform superposition over every single possible input. Invalid states are destructively interfered, and therefore, inputs that satisfy the function will be narrowed down. ECC's small key size works against it in the context of quantum computing and gives it a low qubit (the quantum equivalent of a standard bit) requirement to break. Though there are no present solutions to this looming threat, this is not an immediate threat, as quantum

computing still has a lot of physical hurdles to overcome before it can even begin to become a viable threat.

Though these attacks can tear apart an ECC cryptosystem, the biggest vulnerability inherent in the algorithm is incompetence. The Elliptic Curve Algorithm is advanced, complex, and can be implemented in a large number of ways. A strong, secure implementation of ECC is theoretically possible, but is difficult to actually implement, especially in the hands of a novice or uncareful programmer. The results of poor implementation can lead ECC private key leaks in a number of different cases. This can happen when results are calculated incorrectly, and the input does not end up on the curve. Branch timing issues and cache timing errors can also occur and cause the private key to be compromised.

This issue is not just a theoretical bomb ticking under the surface of the algorithm. There are many actual cases of Elliptic Curve Cryptography gone wrong that have lead to catastrophic damages. A significant, memorable case of this is seen in, "that of the Sony ECDSA security disaster. Although Sony used ECDSA to sign software for their PlayStation game console, they did not properly implement the algorithm. Using static parameters instead of random ones made Sony's implementation of the algorithm solvable and subsequently useless .. . . [in addition,] there are examples of improper implementation of ECC in OpenSSL that resulted in common vulnerabilities, such as Common Vulnerability and Exposure (CVE)-2014-3572, CVE-2014-0076 and CVE-2008-5077. These vulnerabilities range from omission of the server key exchange message to malformed signatures. Worse, such issues can lead to an unauthenticated, remote attacker gaining access to Secure Sockets Layer (SSL) private keys" (Stolbikova 2016). A bad implementation of ECC is a very real threat that needs to be avoided at all costs, and can be combated through code review, static code analysis, and penetration testing. Strong techniques such as the Montgomery Power Ladder, tight parameters, and well-chosen curves need to be ingrained into the playbook of the algorithm in order to ensure widespread success and longevity, and to avoid disasters like the one that happened to Sony.

VI. Conclusion

Public Key Encryption is a staple in the world of information security and is vital to securely sending messages and upholding nonrepudiation. There are various different options each with their own pros and cons, ranging from RSA to Diffie-Hellman to Elliptic Curve Cryptography. Elliptic Curve Cryptography, though newer than and not quite as established as its competitors, uses a revolutionary algorithm based on Elliptic Curve Functions and accomplishes small key sizes and high security that make it perfect for implementation in a variety of different fields, and is already seeing use in areas as significant as Apple's iMessage, the United States Military's security systems, and Bitcoin. Moving forward, it is vital that this algorithm continues to evolve and sees competent implementations to keep up with the ever-evolving world of information security.

Works Cited

Abdullah, Kawther Esaa, and Nada Hussein M. Ali. “Security Improvement in Elliptic Curve Cryptography.” International Journal of Advanced Computer Science and Applications (IJACSA), The Science and Information (SAI) Organization Limited, thesai.org/Publications/ViewPaper?Volume=9&Issue=5&Code=IJACSA&SerialNo=16.

“Elliptic Curve Cryptography - KeyCDN Support.” *KeyCDN*, 18 Oct. 2018, www.keycdn.com/support/elliptic-curve-cryptography#:~:text=It%20has%20been%20noted%20by,the%20same%20level%20of%20security.

Joye, Marc, and Sung-Ming Yen. “The Montgomery Powering Ladder.” Cryptographic Hardware and Embedded Systems - CHES 2002, 2003, pp. 291–302., doi:10.1007/3-540-36400-5_22.

Kohli, Kerman. “Learning Cryptography, Part 1: Finite Fields.” *Medium*, Loopring Protocol, 15 Aug. 2019, medium.com/loopring-protocol/learning-cryptography-finite-fields-ced3574a53fe.

Malik, Muhammad Yasir. “Efficient Implementation Of Elliptic Curve Cryptography Using Low-Power Digital Signal Processor.” *Arxiv*, 7 Feb. 2010, arxiv.org/ftp/arxiv/papers/1109/1109.1877.pdf.

Potlapally, Nachiketh R. “Analyzing the Energy Consumption of Security Protocols”, et al. *Palms.ee.princeton.edu*, Princeton University, palms.ee.princeton.edu/PALMSopen/potlapally03analyzing.pdf.

Robinson, Kelley. “What Is Public Key Cryptography?” *Twilio Blog*, Twilio, 1 Mar. 2021, www.twilio.com/blog/what-is-public-key-cryptography.

Shankar, T N, and K L University. “Cryptography with Elliptic Curves.” International Journal of Computer Science and Applications , May 2002, pp. 38–42.

Singh, Laiphrakpam Dolendro, and Khumanthem Manglem Singh. “Implementation of Text Encryption Using Elliptic Curve Cryptography.” Procedia Computer Science, vol. 54, 2015, pp. 73–82.

Stolbikova, Veronika. “Can Elliptic Curve Cryptography Be Trusted? A Brief Analysis of the Security of a Popular Cryptosystem.” ISACA, ISACA JOURNAL, www.isaca.org/resources/isaca-journal/issues/2016/volume-3/can-elliptic-curve-cryptography-be-trusted-a-brief-analysis-of-the-security-of-a-popular-cryptosyste.

Stories, Short Tech. “How Does RSA Work?” *Hacker Noon*, 23 June 2017, hackernoon.com/how-does-rsa-work-f44918df914b.

Sullivan, Nick. “A Simple Visualization of an Elliptic Curve.” Ars Technica, 24 Oct. 2013, arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/.

Tavares, Pedro. “What Is a Side-Channel Attack?” *Infosec Resources,* 26 Nov. 2020, resources.infosecinstitute.com/topic/what-is-a-side-channel-attack/.

Verma, Sharad Kumar, and Dr. D. B. Ojha. “A Discussion on Elliptic Curve Cryptography and Its Applications .” International Journal of Computer Science Issues, vol. 9, no. 1, ser. 1, Jan. 2012, pp. 74–77. 1.

Wagner, Lane. “Basic Intro to Elliptic Curve Cryptography.” Qvault, 24 Mar. 2021, qvault.io/cryptography/very-basic-intro-to-elliptic-curve-cryptography/.

Wohlwend, Jeremy. “ELLIPTIC CURVE CRYPTOGRAPHY: PRE AND POST QUANTUM.” Mit.edu, MIT, math.mit.edu/~apost/courses/18.204-2016/18.204_Jeremy_Wohlwend_final_paper.pdf.