Digital Cryptoeconomics Powered by Digital Cryptocurrency

Kefa Rabah
Global Open University, Vancouver Canada
Email: kefa.rabah@gmail.com

Received: September 4, 2016
Published: September 12, 2016

Abstract
In this paper we present some intuition about the deep mathematical relationship that exists between public and private keys using elliptic curve cryptography. Also presented are some examples of the math behind digital signatures and its verification using elliptic curve digital signature algorithm (ECDSA), and how it’s applied to creation and mining of Bitcoin digital cryptocurrency. A simple mathematical procedure necessary to create the one-way “trap door” function required to preserve the information asymmetry, which defines ownership of a bitcoin is also given. We’ll also show how quantum computing could be a threat to Bitcoin security and the best practices to preserve its security.

Keywords: Cryptography, cryptocurrency, elliptic curve cryptography, Bitcoin, digital currency, ECDSA, Hashing, IoT

© 2016 by the author(s); Mara Research Journals (Nairobi, Kenya)

1. INTRODUCTION

Digital currency or digital money (cybermoney) is an Internet-based medium of exchange distinct from the traditional physical monies (such as banknotes and coins) that exhibits properties similar to physical currencies, but allows for instantaneous transactions and borderless transfer-of-ownership. Both virtual currencies and cryptocurrencies are types of digital currencies, but the converse is incorrect. Like traditional money these currencies may be used to buy physical goods and services but could also be restricted to certain communities such as, for example, for use inside an online games or social network (Wikipedia, 2016).

Digital currencies or cybermonies have already changed the financial world somehow as we know it, and that’s just the beginning: they have the potential to do way more in the future and bring changes which nobody expects. Bitcoin definitely stands out from all of them as it’s the most recognizable option. When it comes to cryptocurrencies, there are lots of different innovative ideas that should not be underestimated or ignored. Indeed, there are already a number of competitors to bitcoin, including namecoin, litecoin, mastercoin, peercoin, darkcoin and ripple.

However, with bitcoin, apart from being the most famous among the cybermonies, also carries a reputation for instability, wild fluctuation, and illicit business ventures undertaken by cybercriminals; there are some legitimate fear that it has the power to eliminate jobs and to upend the concept of a nation state. It implies, above all, monumental and wide-reaching change-for better and for worse. But it is here to stay, and you ignore it at your peril. Simply put, it’s important that one get acquainted with the knowledge and how to navigate your way in the age of cybereconomy. The digital currency world is expected to look very different from the traditional paper currency world we know today.
However, the jury is still out on the question of outlook of the future global cryptoeconomic in the age of cryptocurrency cybermoney. To many thinkers and scholars, cybermoney is clearly poised to launch a different kind of revolution, one that could reinvent traditional financial and social structures while bringing the world’s billions of “unbanked” individuals into a new global economic order. Thus, cryptocurrency holds the promise of a financial system without a middleman, one owned by the people who use it and one safeguarded from the devastation of a 2008-type crash that occurred in the US.

Finally, in the advent of the 4th industrial revolution (Industry 4.0, the cyberspace era) coupled with the now common place IoT (Internet of Things), we surely stand at the brink of technological revolution, one which is expected to fundamentally alter the way we live, work, do things, run businesses and relate to one another (Schwab, 2016). With it come the scale, scope, complexity, and transformations that will be unlike anything humankind has experienced before. The end-result as things unfold, is still unknown to us, however, we must endeavor to proactively respond to it in an integrated and comprehensive manner, in a way that involve all stakeholder of the global polity, from the public and private sectors to academia and civil society. With all things digital and interconnected smart cities powered by Internet of Things, come the challenges of keeping out the bad guys, the cybercriminals. Here, cybersecurity and the way we secure our data as it transverse the enterprise network infrastructure and the Internet will become more critical.

However, we can take heart that there are positive outcomes that can come out of it – the integration of smart cities, IoT and Blockchain. The Blockchain technology was invented in a whitepaper in 2008 by Satoshi Nakamoto, an alias, and was first implemented in 2009 in Bitcoin, a cryptocurrency. Blockchain—also known as distributed ledger technology—was not at first the centre of attention, but rather was the under-the-hood invention that enabled the digital currency and payments system to work without the need for a trusted central authority by using a distributed, cryptographically secure, and crypto-economically incentivized consensus engine. Section 2 below explains what blockchains are and how they work (Potts, Davidson & De Filippi, 2016).

It has become well established that the blockchain has an almost infinite range of functional applications outside of pure financial transactions. This is particularly pertinent when it comes to the IoTs where technology is being used to disrupt a range of traditional industries. One such example, is the in the cargo freight transactions, an industry traditionally choked in paperwork, typically completed in triplicate, coupled with the physical slow couriering process of documentation as proof-of-purchase. Apart from being incredibly wasteful – such a process comes with problems in document authentication, payments, time delays and couriering; not to mention the risk of losing the physical documents. Here is where we can make use of smart cities combined with Blockchain and IoT to create new, decentralized platforms that can spawn powerful applications for cargo freight management solutions with Bitcoin cryptocurrency payment systems (Lawrence, 2016). Many more applications can be envisaged using the innovative cyberspace powered digital solutions.

1.1 History and Overview of Digital Currencies

The concept of digital currency started while the Internet was still catching up with most parts of the world. The origin of digital currencies can be traced back to the 1990s Dot-com bubble era. Some of the digital currencies are discussed below:

**E-gold**: is one of the first digital currencies that came into existence, founded in 1996 and backed by gold by oncologist named Douglas Jackson, a lawyer managed to have over 5 million user accounts by 2009 (Mullan, 2014). E-gold grew so big, that even merchants had started accepting it – thus becoming a very successful venture until it entered the favorite list of cybercriminals and hackers. Continued attacks on the platform by cybercriminals and use of e-gold as a favored currency by extortionists and money launderers eventually led to its downfall.
**WebMoney:** In 1998 another digital currency, WebMoney came into existence. Except for the decentralization part, WebMoney is a form of digital currency for all practical purposes (Mullan, 2014, p. 68-78). The Moscow based company offers a wide range of financial services including peer to peer payment solutions, merchant services, online billing and payments and even internet based trading platforms. WebMoney became the next best thing after e-gold and attracted many users, both good and bad from e-gold once it was shutdown. However, WebMoney made changes to its services soon after that to prevent its usage for illegal activities. WebMoney currently supports a number of international currencies including GBP, USD, Russian Rubles and even bitcoin.

**Liberty Reserve:** several years later, another digital currency service Liberty Reserve, founded in 2006, let users convert dollars or euros to Liberty Reserve Dollars or Euros; and exchange them freely with another at 1% fee. Both services were centralized, reputed to be used for money laundering, and inevitably shut down by the US government (Cloherty, 2013; Trautman, 2014).

**Q or QQ Coins:** is another digital currency which is used as a type of commodity-based digital currency on Tencent QQ’s massaging platform and emerged in early 2005 (Wang & Mainwaring, 2008, April). The Q coins were so effective in China that they were said to have had a destabilizing effective on the Chinese Yuan or RMB currency due to speculation (ATO, 2006).

**Perfect Money:** another digital currency was launched in 2007 (Moore, 2103). It is a digital currency platform that works with multiple currencies including USD, EUR, GBP, BTC and more. Like most of the digital currency platforms in the past, death of one established platform will result in a surge of its successor. The same thing happened with Perfect Money; customers from Liberty Reserve flooded it after the former was shut down by regulators. Perfect money also offers services similar to that of Liberty Reserve sans the lack of verification. However, Perfect Money is not available in the United States and for US citizens located anywhere in the world.

**Bitcoin:** in recent years, we have witnessed the emergent of cryptocurrencies, which has prompted renewed interest in digital currencies, with bitcoin, introduction in 2009, becoming widely used and accepted digital currency (Grinberg, 2012; Doguet, 2012; Courtois & Bahack, 2014). In 2016, a city government of Zug Switzerland first accepted digital currency payment of city fees. The city of Zug added bitcoin as a means of paying small amounts, up to SFr 200, in a test and an attempt to advance Zug as a region that is advancing future technologies. In order to reduce risk, Zug immediately converts any bitcoin received into the Swiss currency (Uhlig/jse, 2016).

It is important to mention the fact that bitcoin protocol is open source, thus preventing anyone from having a monopoly over the whole system. Further, the security and transparency associated with bitcoin is worth mentioning as well. That is, bitcoin seems to have hit the right chord, satisfying all the apprehensions about digital currency that have been built over the years of experience with centralized, proprietary digital currency platforms. Following the popularity of bitcoin, there are new digital currencies that have come up, including national currencies. Ecuador has adopted its own digital currency and even Barcelona; Spain has plans on including digital currency as its legal tender (Dwyer, 2015).

The question we’re now left with to ponder is how long can global central banks continue to ignore Blockchain as a defacto currency of trade. Of-late there a kind of warming-up of the central bankers in blockchains, and which could be an enormous benefit to global economy. And as a matter of fact, the first major country brave enough to issue currency on a blockchain, that nation will sure enough gain a substantial competitive advantage over its competitors. An obvious strength is Cybersecurity, especially in the light of attacks on Swift, the most common global money financial transfer protocol (Long, 2016). Blockchains use secure decentralized IT architecture, which is harder to attack compared to centralized,
single-point-of-failure systems. Further, the Bitcoin blockchain is powered by a robust IT architecture. Moreover, its robustness has seen it survived cybercriminal warzone of Internet security, and to-date has a $9 billion capitalization sure to tempt hard-core hackers for almost seven years without a single successful attack on its core protocol. Thus, central banks that will be brave enough to issue money on blockchain will its financial systems suddenly becoming transnational, and in-effect, capital will flow in, allowing it to become the key hub of global payment systems (Long, 2016).

1.2 Overview of Cryptoeconomics

The science of cryptography has been in existence for over millennia but in a formal and systematized form for just a couple of decades – can be simply defined as the study of communication in an adversarial environment (Rabah, 2004). Similarly, we can define cryptoeconomics as a concept that goes one step further, i.e., the study of economic interaction in an adversarial environment (Davidson, De Filippi & Potts, 2016; Ernst, 2016). To distinguish itself from the traditional economics, which certainly studies both economic interaction and adversaries, cryptoeconomics generally focuses on interactions that take place over network protocols. Particular domains of cryptoeconomics, include: Online trust and reputation systems; Cryptographic tokens / cryptocurrencies, and more generally digital assets; Self-executing "smart" contracts; Consensus algorithms; Anti-spam and anti-sybil attack algorithms; Incentivized marketplaces for computational resources; Decentralized systems for social welfare / mutual aid / basic income; Decentralized governance (for both for-profit and non-profit entities).

In the last couple of years we have witnessed an increased in prominence of cryptoeconomics, which to a large extent as a result of the growth of cryptocurrencies and digital tokens, and which brings a new, and interesting dimensions to cryptography (Potts, Davidson & De Filippi, 2016). While before, cryptography was, by and large, a purely computational and information theoretic science, with strong guarantees built on security assumptions that there are close to absolute; once money enters the picture the perfect world of mathematics must interact with a much more messy reality of human social structure, economics incentives, partial guarantees and known vulnerabilities that can only be mitigated, and not outright removed. While a cryptographer is used to assumptions of the form “this algorithm is guaranteed to be unbreakable provided that the underlying math problems remain hard”, the world of cryptoeconomics must content with fuzzy empirical factors such the difficulty of collision attack, the relative quantity of altruistic, profit-seeking and anti-altruistic parties, the level of concentration of different kinds of resources, and in some even sociocultural circumstances (Ernst, 2016; Davidson, De Filippi & Potts, 2016).

In contrast, the traditional applied cryptography, security assumptions tend to look something like this: i) No one can do more than $2^{79}$ computational steps; ii) Factoring is hard (i.e., superpolynomial) (Rabah, 2005, 1); iii) Taking $n^{th}$ roots modulo composites is hard; iv) The elliptic curve discrete logarithm problem cannot be solved faster than in $2^{n/2}$ time (Rabah, 2005, 2).

In cryptoeconomics, on the other hand, the basic security assumptions that we depend on are alongside those of cryptographic assumptions, roughly as follows (Ernst, 2016):

- No set of individuals that control more than 25% of all computational resources is capable of colluding
- No set of individuals that control more than 25% of all money is capable of colluding
- The amount of computation of a certain proof of work function that can be accomplished with a given amount of money is not superlinear beyond a point which is reasonably low
- There exist a non-negligible number of altruists and a non-negligible number of crazies or political opponents of the system, and the majority of users can be reasonably modeled as being close to economically rational
• The number of users of a system is large, and users can appear or disappear at any time, although at least some users are persistent
• Censorship is impossible, and any two nodes can send messages to each other relatively quickly.
• It is trivial to generate a very large number of IP addresses, and one can purchase an unlimited amount of network bandwidth
• Many users are anonymous, so negative reputations and debts are close to unenforceable

In this respect, it’s important to note that there are additional security assumptions that are specific to certain problems. Thus, quite often it will not even be possible to definitively say that a certain problem has been solved. Rather, it will be necessary to create solutions that are optimized for particular empirical and social realities, and continue further optimizing them over time (Ernst, 2016).

1.3 Overview of Cryptocurrency

Cryptocurrency involves the medium of digital transactions (Rosca, 2016). By definition, this is a method of exchange that broadly incorporates the use of software, which is encrypted, so as to help in the operations of market transactions. Cryptocurrencies are designed to be inherently rare, and their inflation grows at a slow, controlled rate – that is, they’re not designed to become a popular option, therefore, they keep their inflation on a controlled and slow growth rate. This eventually may make them more stable, in complete contrast to all government controlled currencies, where inflation is an actual problem because of irresponsible actions, like money reprints or central banks and financial institutions can simply “add a few zeros” to the end of their bank notes/accounts as needed.

To-date we have witnessed three eras of currency, namely the commodity-based, politically-based and now the maths-based. The first maths-based cryptocurrency was actually DigiCash, which was invented in 1990s (Tanaka, 1996). The next was bitcoin, established in 2009 by anonymous investors subsequently identified as Satoshi Nakamoto who created the bitcoin protocol and reference software (Nakamoto, 2008, 2009; Grinberg, 2012). The market of transactions is plainly seen by the network users and they are based on some rules that are coded into specific algorithms. Therefore, cryptocurrency involves a high-level use of cryptography, mainly for security purposes and for providing measures that help in controlling counterfeiting circumstances.

That is, Bitcoins are created and held electronically, while encryption is used to create and verify the transactions. Bitcoin is based on a distributed network that uses blockchains. Data is stored by all individuals on the network and every node has a copy of the ledger of transactions, called the blockchain – a sequence of blocks, which holds the complete record of transactions like a public ledger and which is one of the most unique parts. Bitcoin is decentralized, with no central institutions controlling it, and cryptography is used to secure transactions and control the creation of new units. Because it does not have a connection or tie with any given country, the central bank does not have control over its operations in or its value determination in anyway. For instance, in Bitcoin, the value is mostly determined by the demand and supply in the marketplace, meaning this exchange medium is golden and very precise when used for transactions. This has made bitcoin the largest cryptocurrency when considered in terms of market volume, acceptance, and market capitalization, with some of other mediums of currency exchange like Litecoin, and Ripple coming behind.

Bitcoins are not actually currency, in the real-sense; a currency by definition needs to be a medium of exchange, a unit of account and a store of value. Therefore, it does not currently satisfy these three functions, although if it becomes less volatile coupled with greater stability, could lead to more merchants accepting it – thus, it eventually could become money. However, the key reasons for companies’ and
individuals’ continuing interest in using Bitcoins, are that transaction costs are low, they are fast to use for transfer for payment and they can be used to hedge against hyper-inflation. What’s the future like? It’s envisaged that new bitcoins are created through mining, where a cryptographic puzzle is solved (Eyal & Sirer, 2014, March; Kroll, Davey & Felten, 2013, June). The power of professional miners is growing as miners become more high tech and fewer people want to mine individually. Thus, if someone was to hold 51% of all Bitcoins, they might be able to alter the protocol and destroy the cryptocurrency or use computational factors to gain advantages.

In terms of speed and hashing, cryptocurrency is considered as having a high computing power and it can mine numerous hashes in second. Then, the hash created can be used in getting original data out, thus becoming more practicable to users everywhere. That is, in its operation cryptocurrency enables various computer networks to sustain collective book-keeping systems through the cloud, or Internet system, and the process is commonly referred to as blockchain. This process is public and can be accessed by every user registered under a given cryptocurrency. The blockchain allows all transactions to be logged, including various details concerning participants, date and the amount recorded for every single transaction done. As a result, the cryptocurrencies and the blockchain technology can be essential in handling money, payments and other credit awards activities, such as during elections (Davidson, De Filippi & Potts, 2016).

2. The Mechanics of Maths Behind Bitcoin Cryptocurrency

As we have observed earlier, Bitcoins are not stored either centrally or locally and so no one entity is their custodian. They exist as records on a distributed ledger called the blockchain, copies of which are shared by a volunteer network of connected computers. To “own” a bitcoin simply means having the ability to transfer control of it to someone else by creating a record of the transfer in the blockchain. What grants this ability? Access to an Elliptic Curve Digital Signature Algorithm (ECDSA) using private and public key pair. What does that mean and how does that secure bitcoin? But before that we need to understand the theory behind elliptic curve cryptography.

2.1 Elliptic Curve Cryptography

Koblitz (1987) and Miller (1985) proposed a method by which public key cryptosystems can be constructed on the group of points of an elliptic curve (EC) over finite field. EC cryptosystems are based on operations involving points on an elliptic curve over a finite field. Popular choices for the underlying finite field are \( F_p \), the integers modulo \( p \) for a (large) prime number \( p \), and \( F_{2^m} \), a finite field of characteristic two and dimension \( m \). Point addition and point multiplication over ECC (Rabah, 2005, June).

A point, say \( P = (x, y) \) with \( x, y \in \mathbb{F}_p \). The point of interest for the EC cryptography (ECC) will satisfy equation of the form, \( y^2 = x^3 + ax + b \), where \( a, b \in \mathbb{F}_q \), and all the operations are over \( \mathbb{F}_q \). Points in \( \mathbb{F}_q \) satisfying the elliptic curve equation, together with a postulated point at infinity are referred to as rational points on the elliptic curve. Since the underlying field is finite, the number of rational points on the curve is finite, and will be denoted by \( \#E(\mathbb{F}_q) \).

It turns out that an addition operation can be defined on the elliptic curve points, and that the points, together with this operation, from an abelian group. By Hasse Theorem, the size of the abelian group is known to fall in the interval

\[ q + 1 - 2\sqrt{q} \leq \#E \leq q + 1 + 2\sqrt{q} \]
here the prime $q = p$ and $\#E(F_q) = q + 1 - t$ where $|t| = 2\sqrt{q}$ is the trace of the curve.

Note: $E(F_q)$ is called the group of rational points i.e., points that are rational over the ground field $F_q$. Not to be confused with $E(Q)$. The knowledge of the rational points on the curve is essential to cryptosystems because we often desire curves $E$ with a large prime dividing $\#E(F_q)$. So we can either construct curves with the right properties (like $\#E(F_q)$ containing a large prime) or generate curves randomly and check if they have the desired properties. Hence, efficient computation of $\#E(F_q)$ is an important question. Both approaches are useful depending on the application. For example EC factoring algorithms search for curves where the number to be factored, decomposes smoothly.

For an effective curve cryptosystems, the coefficients $a$ and $b$ are chosen so that the elliptic curve group has a large cyclic subgroup of prime size $p$, i.e., $\#E(F_q)$ can be written as $\#E(F_q) = h.p$, where $h$ is a small integer and $p$ is prime, i.e., order $p$ is then chosen as the generator of the cyclic subgroup, and all ECC protocols are based on computing $Q = kP = P + P + \cdots + P$ ($k$ -times). With well chosen curve parameters, if $k$ and $P$ are given it is fairly easy to compute $kP$, but the inverse problem, i.e., recover $k$ from $P$ and $kP$ is computationally unfeasible as per current algorithm knowledge. The inverse problem is known as the elliptic curve variant of the discrete logarithm problem (ECDLP) (Rabah, 2005, 6).

Moreover, if the elliptic curve cryptosystems satisfy both MOV-conditions (Cohen, Miyaji, & Ono, 1998; Menezes, Okamoto & Vanstone, 1993) and FR-conditions (Frey & Rück, 1994), and avoid $p$-divisible elliptic curves over $F_p$ (Rabah, 2005, June) then only known attacks are the Pollard $\rho$-method(Rabah, 2005, 6) and the Pollig-Hellman method (Pohlig & Hellman, 1978). Hence, with current knowledge, we can construct elliptic curve cryptosystems over a smaller definition field than the discrete-logarithm problem (DLP)-based cryptosystems like the ElGamal cryptosystems (Rabah, 2005,4) or the DSA (Rabah, 2005,3) and RSA cryptosystems (Rivest, Shamir & Adleman, 1983). The security of the elliptic curve hinges on the intractability of the DLP in the algebraic systems. Unlike the case of the DLP in the finite fields, or the problem of factoring integers, there is no subexponential-time algorithm known for the elliptic curve discrete logarithm problem (ECDLP). The best algorithm known to date takes exponential time (Rabah, 2005, 6).

The primary advantage of elliptic curve cryptosystems is thus their high strength relative to key size. This will tend to improve the attractiveness of the elliptic curve cryptosystems relative to other public-key cryptosystems as computing power improvements warrant general key size increases. For example, an EC cryptosystems with a 160-bit key are thus believed to have the same security as both the ElGamal cryptosystems and RSA with 1024-bit key. Thus, the significantly shorter key size using ECC, results in sufficiently shorter certificates and systems parameters. These advantages manifest themselves in many ways including storage efficiencies, bandwidth savings and computational efficiencies. The computational efficiencies lead in turn to higher speeds and power efficiency. If the elliptic curve systems are implemented using the finite field $F_{2^n}$, further computational efficiencies are available. This is particularly true in hardware implementations where compact architectures exist for finite field multiplication. The encryption and signature algorithms described here can be used to provide a variety of cryptographic services including the following: Privacy (secrecy), entity authentication, information authentication, digital signatures (non-repudiation), and authenticated key exchange.

2.1.1 ECC Arithmetic Over Galois 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: \( y^2 = x^3 + ax + b \) (mod \( p \)) for \( 0 \leq x < p \) together with the extra point \( O \) called the point at infinity. For a given point \( P = (x_p, y_p) \), \( x_p \) and \( y_p \) are called the \( x \) and \( y \) coordinates of \( P \), respectively. The number of points on \( E_p(a, b) \) is denoted by \( \#E(F_p) \). The constants \( a \) and \( b \) are non-negative integers smaller than the prime number \( p \) and must satisfy the condition: \( 4a^3 + 27b^2 \neq 0 \) (mod \( p \)). 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 \( E_p(a, b) \). So there will be a lot of points modulo \( p \). In fact, the general theory says that there will be about \( p \) points \((x, y)\) with error bounded by \( O(\sqrt{p}) \).

2.1.2 Point Addition

Scalar multiplication i.e., the computation of \( kP \), where \( k \) is a random integer and \( P \) is an elliptic curve generation point, can be defined as the combination of additions of two points on an elliptic curve. Scalar multiplication of elliptic curve points can be computed efficiently using the addition rule together with the double-and-add algorithm or one of its variants. Its properties, computation and uses will, therefore, the core of ECC implementation. 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 and, Fig. 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 \( P_1 = P_2 \): \( x_3 = \lambda^2 + \lambda + a, \ y_3 = \lambda^2 + (\lambda + 1)x_3 \) where \( \lambda = x_1 + y_1/x_1 \) (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 \( \lambda = (y_1 + y_2)/(x_1 + x_2) \) (Point addition)
3. Return: \((x_3, y_3)\)

![Fig. 1: Elliptic curve](image-url)
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 an addition step usually requires eight additions, two multiplications, one squaring, three reductions mod \( f(x) \), and one inversion. A Doubling step usually requires four additions, two multiplications, two squaring, four reductions mod \( f(x) \), and one inversion. A Negation step requires one addition. The important contributors to the run time are the multiplications and inversions (Morain & Olivos, 1990). Just as modular exponentiation determines the efficiency of RSA cryptographic systems (Frey & Rück, 1994), scalar multiplication dominates the execution time of ECC systems. In all the protocols that are fundamental implementation of ECC, say ECDH, ECDSA, ECAES etc., the most time consuming part of the computations are scalar multiplications. Elliptic curves have some properties that allow optimization of scalar multiplications.

An elliptic curve is the set of solutions \((x, y)\) to the equation:

\[
y^2 = x^3 + ax + b \mod p \quad \text{where} \quad (x, y \in E_p(a, b))
\]

Any point satisfying this equation is considered a point on the elliptic curve. The elliptic curve can be described over any finite two-dimensional field, which offers extra efficiency in the operation of ECC.

A significant property of elliptic curve is that a line drawn between two specific points always cuts the curve three times. Using this property, mathematicians are able to define addition, the basis for the public key cryptosystem.

Addition between two points on the elliptic curve (call them \( P_1 \) and \( P_2 \)) is defined (using some deep mathematics) to be the negative of the point the line drawn between \( P_1 \) and \( P_2 \) cuts on the elliptic curve. This is more adequately explained on the graph.

The standard protocols in cryptography which make use of the discrete logarithm problem in finite fields, such as Diffie-Hellman key exchange, ElGamal and Massey-Omura, can all be made to work in the elliptic curve case.

### 2.1.3 Construction of an Elliptic Curve Over \( \mathbb{F}_p \)

Let the prime number \( p = 23 \) and consider an elliptic curve \( E: y^2 = x^3 + x + 4 \mod 23 \) defined over \( \mathbb{F}_{23} \). From Eq. (6), and let the constants \( a = 1 \) and \( b = 4 \). We first verify that:

\[
4a^3 + 27b^2 \mod p = 4(1)^3 + 27(4)^2 \mod 23 = 436 \mod 23 = 22 \neq 0 \quad (1)
\]

Therefore, \( E \) is indeed an elliptic curve. We then determine the quadratic residues \( Q_{23} \) from the reduced set of residue \( Z_{23} = \{1, 2, 3, \ldots, 21, 22\} \):

<table>
<thead>
<tr>
<th>( x^2 \mod p )</th>
<th>( (p-x)^2 \mod p )</th>
<th>=</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1^2 \mod 23 )</td>
<td>( 22^2 \mod 23 )</td>
<td>1</td>
</tr>
<tr>
<td>( 2^2 \mod 23 )</td>
<td>( 21^2 \mod 23 )</td>
<td>4</td>
</tr>
</tbody>
</table>
Therefore, the set of $(p - 1)/2 = 11$ quadratic residues $Q_{23} = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$. Now, for $0 \leq x < p$, compute, $y^2 = x^2 + x + 4 \mod 23$, and determine if $y^2$ is in the set of quadratic residues $Q_{23}$:

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2$</td>
<td>4</td>
<td>6</td>
<td>14</td>
<td>11</td>
<td>3</td>
<td>19</td>
<td>19</td>
<td>9</td>
<td>18</td>
<td>6</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>$y^2 \in Q_{23}$?</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$y_1$</td>
<td>2</td>
<td>11</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>11</td>
<td>5</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_2$</td>
<td>21</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>15</td>
<td>12</td>
<td>18</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x$</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2$</td>
<td>19</td>
<td>6</td>
<td>2</td>
<td>13</td>
<td>22</td>
<td>12</td>
<td>12</td>
<td>5</td>
<td>20</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>$y^2 \in Q_{23}$?</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>Yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_1$</td>
<td>11</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>9</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_2$</td>
<td>12</td>
<td>18</td>
<td>17</td>
<td>14</td>
<td>14</td>
<td></td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hence, the points in $E(F_{23})$ are $O$ and the following:

$$E_{23}(1, 4) = \begin{pmatrix} (0, 2) & (0, 21) & (1, 11) & (1, 12) & (4, 7) & (4, 16) & (7, 3) \\ (7, 20) & (8, 8) & (8, 15) & (9, 11) & (9, 12) & (10, 5) & (10, 18) \\ (11, 9) & (11, 14) & (13, 11) & (13, 12) & (14, 5) & (14, 18) & (15, 6) \\ (15, 17) & (17, 9) & (17, 14) & (18, 9) & (18, 14) & (22, 5) & (22, 18) \end{pmatrix}. \quad (2)$$

Figure 2 shows a scatterplot of the elliptic group $E_p(a, b) = E_{23}(1, 4)$. 
2.2 Algorithms for Elliptic Scalar Multiplication

Just as modular exponentiation determines the efficiency of RSA cryptographic systems (Rivest, Shamir & Adleman, 1978), scalar multiplication dominates the execution time of ECC systems. Scalar multiplication is the operation to compute $kP$, where $k$ is a random integer and $P$ is an elliptic curve generation point, say $(x_p, y_p)$, and it can be defined as the combination of additions of two points on an elliptic curve. That is, the calculations of the form: $Q = kP = P + P + \ldots + P$. Here $P$ is a fixed point that generates a large, prime subgroup of $E(F_q)$, or $P$ is an arbitrary point in such a subgroup and, $k$ is an integer in the range of $[1, n-1]$ where $n$ is the order of the elliptic curve $E$. Elliptic curves have some properties that allow optimization of scalar multiplications. The important contributors to the run time are the multiplications and inversions (Rabah, 2005, June; Koblitz, 1987; Menezes, Okamoto & Vanstone, 1993). 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.

To test the algorithm, let $P = (1,3) \in E_{31}(-1,9)$. Then $2P = (x_3, y_3)$ is equal to: $2P = P + P = (x_1, y_1) + (x_1, y_1) = (5,6)$. Next, test addition of two different points on the curve, i.e., $Q = (22,8) \in E_{31}(-1,9)$ and $R = (7,29) \in E_{31}(-1,9)$. Then $Q + R = (x_3, y_3) = (30,28)$, which we can observe also to be on the curve. However, for real implementation of ECC, we need to know the order of the elliptic curve group.

Let us now implement the scalar multiplication to form a subgroup of points $<P>$, where $<P>$ is the finite cyclic group $<P> = \{P, 2P, 3P, \ldots, nP\}$, with order $n$, by following the same additive rules, and a generator point $P$. For example, let $P = (1,3) \in E_{23}(-1,9)$ be a generator point which we use through repeated addition of point to generate all the point on the curve (see Table 3):
Table 3: Point addition/scalar point multiplicative values of $P$
(Note that: $kP = P_k$, so that $37P = P_{37} = \{O\}$ etc.).

<table>
<thead>
<tr>
<th>Value</th>
<th>Point Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1P = [1, 3]</td>
<td>2P = [5, 6]</td>
</tr>
<tr>
<td>3P = [12, 12]</td>
<td>4P = [22, 8]</td>
</tr>
<tr>
<td>5P = [16, 20]</td>
<td>6P = [23, 1]</td>
</tr>
<tr>
<td>7P = [17, 21]</td>
<td>8P = [7, 29]</td>
</tr>
<tr>
<td>9P = [28, 4]</td>
<td>10P = [4, 21]</td>
</tr>
<tr>
<td>11P = [0, 3]</td>
<td>12P = [30, 28]</td>
</tr>
<tr>
<td>13P = [9, 4]</td>
<td>14P = [6, 8]</td>
</tr>
<tr>
<td>15P = [25, 4]</td>
<td>16P = [24, 18]</td>
</tr>
<tr>
<td>17P = [10, 10]</td>
<td>18P = [3, 23]</td>
</tr>
<tr>
<td>19P = [3, 8]</td>
<td>20P = [10, 21]</td>
</tr>
<tr>
<td>21P = [24, 13]</td>
<td>22P = [25, 27]</td>
</tr>
<tr>
<td>23P = [6, 23]</td>
<td>24P = [9, 27]</td>
</tr>
<tr>
<td>27P = [4, 23]</td>
<td>28P = [9, 27]</td>
</tr>
<tr>
<td>29P = [10, 21]</td>
<td>30P = [24, 13]</td>
</tr>
<tr>
<td>33P = [25, 27]</td>
<td>34P = [9, 27]</td>
</tr>
<tr>
<td>35P = [10, 21]</td>
<td>36P = [24, 13]</td>
</tr>
<tr>
<td>37P = [0]</td>
<td>38P = [9, 27]</td>
</tr>
</tbody>
</table>

Observe from above algorithms that the addition of two elliptic curve points in $E(F_p)$ requires a few arithmetic operations (addition, subtraction, multiplication, and inversion) in the underlying field $F_p$. 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. 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 real practice $p$ is chosen to be a large prime number. Take, for example, a large group of points with prime number:

$$p = 6,277,101,735,386,680,763,835,789,423,207,666,416,083,908,700,390,324,961,279$$

Next imagine a square grid (cf. Fig. 3) with this many rows and columns. There is a curve defined over this space of the form: $E: y^2 = x^3 + ax + b \pmod{p}$ where $a$ and $b$ are two more carefully chosen large numbers so that the curve is not weak and $4a^3 + 27b^2 \neq 0 \pmod{p}$. This curve contains exactly $N$ points, where:


These points form a group, according to the rule above, which is ideal for elliptic curve Diffie-Hellman (ECDH) algorithm. Modern computers have no trouble dealing with numbers of this size, which are actually much smaller than those used in traditional DH and RSA cryptosystems. If you look at $p$ as binary number you’ll see that it has a very special form, $p = 2^{192} - 2^{64} - 1$, which makes computation much easier. It is interesting to note that $p$ and $N$ are so “close” to each other, relatively speaking; they only differ in the lower half of their bits. Elliptic curve theory predicts this.

2.3 Elliptic Curve Discrete Logarithm Problem (ECDLP)

The fundamental mathematical operation in traditional public key cryptosystems, e.g., RSA is based on modular integer exponentiation, while crypto-algorithms like Diffie-Hellman, ElGamal, Digital signature Scheme (DSS), and SPEKE, the security depends directly on the relative difficulty of performing two group operations: exponentiation which states that: given $x$ and $y$, compute $z = x^y$; and discrete logarithm problem (DLP) which states that: given $x$ and $z$, compute $y$ such that $z = x^y$. In suitable groups, exponentiation is a much easier and efficient computational problem than DLP. When using group
of integers formed under modular multiplication, DLP becomes effectively impossible to compute when modulus is sufficiently large. This makes exponentiation a “one-way” function (Rabah, 2005, 6).

For example, in an (abelian) group $G$ (multiplicatively written) we can consider the equation $y = x^n$, $x, y \in G, n \in \mathbb{Z}$. If $x$ and $y$ are known real numbers and it is also known that $x$ is some power (say, $n$) of $y$, then logarithms can be used to find $n = \log_x(y)$ in an efficient manner. However, if $x$ and $y$ are given such that: $y = x^n = x \cdot x \cdot \ldots \cdot x$ ($n$-times), then in general it is technically much harder and hence the determination of $n$ cannot be carried out in a reasonable amount of time. This is equivalent to the well-known real logarithm, we call $n$ the discrete logarithm of $y$ related to the base $x$ (Rabah, 2005, 6). The operation “exponentiation” $x \rightarrow y := x^n$ can be implemented as a quick, efficient algorithm.

The elliptic curve discrete logarithm (ECDLP) is somewhat different and, by some measures, harder than DL in multiplicative integer groups. Implementations of ECDLP one can exploit this difference to provide increased speed and decreased key size for a given level of security. Most modern traditional public key crypto-algorithms can easily work with elliptic curves, or any suitable group, to take advantage of the best available implementations.

The core of elliptic curve arithmetic is an operation called scalar point multiplication, which computes $Q = kP$ (a point $P$ multiplied $k$ times resulting in another point $Q$ on the curve) (Rabah, 2005, 6). Scalar multiplication is performed through a combination of point-doublings (which add two copies of point together i.e., point $P$ to obtain $2P$) and point-additions (which add two distinct points together, i.e., point $P$ to $2P$ to obtain $3P$). For example, $11P$ can be expressed as $11P = 2 \cdot ((2 \cdot (2 \cdot P)) + P) + P$. Furthermore, when a point $P$ on an elliptic curve $E$ is given, there is a minimum positive integer $n$ such that $nP = O$, the identity point or point at infinity. Integer $n$ is called the order of the point $P$. It is known that $n$ is a divisor of the order of the curve $E$. Consider our earlier comment that these scalar multiples form a subgroup of points $<P>$. Here $<P>$ is the finite cyclic group $<P> = \{P, 2P, 3P, \ldots, nP\}$, with order $n$. The problem of calculating $k$ from a given points $P$ and $Q$ is called “the discrete logarithm problem over the elliptic curve (ECDLP)” The security of ECC relies on the hardness of solving the ECDLP.

2.4 Key Generation and Key Establishment Protocols

2.4.1 Key Generation Considerations

The key generation consideration are developed to be used with the following algorithms: ECES, ECSS, ECDSA, and ECKEP

System Setup

The underlying field $F_q$ is selected first and is common to all users. There are two basic options for generating the remaining parameters:

1. Key Option 1: Short Public Keys
   The elliptic curve parameters, namely the field elements $a$ and $b$ (which define the elliptic curve), and the base point $P$ of order $n$, are selected by the system administrator. The parameters $a$, $b$, $P = (x_p, y_p)$, $n$, are public information and are common to all users.
2. **Key Option 2: Long Public Keys**

Each user selects their own elliptic curve parameters, which also have to be part of the users’ public key. To make the public key shorter, the parameter $a$ is restricted to be $a = 0$ if the underlying field is $F_{2^m}$, and $a = 1$ if the underlying field is $F_p$.

### Key Generation

After system setup, each entity performs the following operations:

1. Select a random integer $d$ in the range $[1, n - 1]$.
2. Compute the point $Q = dP$.
3. Let $Q = (x_Q, y_Q)$ and compute $\tilde{y}_Q$.
4. For Key Option 1 (short public keys), the entity’s public key consists of the point $Q$, which is represented by $x_Q$ and $\tilde{y}_Q$.
   
   For Key Option 2 (long public keys), the entity’s public key consists of the point $P$ (the base point) which is represented by $x_p$ and $y_p$, and the point $Q$ which is represented by $x_Q$ and $\tilde{y}_Q$. (Note that the elliptic curve parameter $b$ can be easily computed from $x_p$ and $y_p$ as follows: if $q = p$ then $b = y_p^2 - x_p^3 - x_p$; if $q = 2^m$ then $b = y_p^2 + x_p y_p + x_p^3$). If ECDSA is to be used, then the public key must further include the order $n$ of $P$.
5. The entity’s private key is the integer $d$.

#### 2.4.2 Elliptic Curve Key Establishment Protocol (RCKEP)

This section describes a protocol whereby two parties A and B establish a shared secret key $S_{AB}$ called the **session key**. The session key may subsequently be used to achieve some cryptographic goal, such as privacy or authentication.

#### System Setup

It is assumed that A and B are using same elliptic curve parameters $F_q$, $E$, $P$ and $n$. A has private key $k_A$ and public key $Q_A = aP = (x_A, y_A)$. B has private key $b$ and public key $Q_B = bP = (x_B, y_B)$.

1. Entity A does the following:
   a) Select a random integer $k_A$, $1 \leq k_A \leq n - 1$
   b) Compute the point $R_A = k_A P$
   c) Compute the point $(x_i, y_i) = k_A Q_B$
   d) Compute the integer $s_A = k_A + ax_A x_i \mod n$
   e) A sends $R_A$ to B

2. Entity B does the following:
   a) Select a random integer $k_B$, $1 \leq k_B \leq n - 1$
   b) Compute the point $R_B = k_B P$
   c) Compute the point $(x_i', y_i') = k_B Q_A$
   d) Compute the integer $s_B = k_B + bx_B x_2 \mod n$
   e) A sends $R_B$ to A.

3. A does the following:
   a) Compute $(x_2', y_2') = bR_B$
   b) Compute the session key: $S_{AB} = s_A (R_B + x_B x_2 Q_B)$.
4. B does the following:
   
a) Compute \( (x_1, y_1) = bR_A \)
   
b) Compute the session key \( S_{BA} = s_B(R_A + x_A x_i Q_A) \)

**Implementation of Elliptic Curve Key Establishment Protocol (ECKEP)**

The ECC system wide parameters are the prime \( p = 751 \), elliptic curve \( E: y^2 = x^3 + x + 188 \) defined over \( \mathbb{F}_{751} \). Here \( E_p(a,b) = E_{751}(1,188) \) and order \( \#E(\mathbb{F}_{751}) = n = 769 \), which is prime order of the generator point \( P = G = (44,81) \) generating multiples of \( kP \) (for \( 1 \leq k \leq n - 1 \)) including point \( O \) located at infinity, is to be shared between the communicating partners.

**System Setup**

It is assumed that A and B are using same elliptic curve parameters \( E, \; P \) and \( n \).

A has private key \( a = 350 \) and public key \( Q_A = aG = 350G = (174,11) \).

B has private key \( b = 677 \) and public key \( Q_B = bG = 677G = (662,238) \).

1. Entity A does the following:
   
a) Select a random integer \( k_A = 56 \), and compute the point \( R_A = k_A G = 56G = (596,345) \)
   
b) Compute the point \( k_A Q_B = 56(677G) = 231G = (440,379) \)
   
c) Compute the integer \( s_A = k_A + ax_1 \mod n = 251 \)
   
d) A sends \( R_A = 56G = (596,345) \) to B

2. Entity B does the following:
   
a) Select a random integer \( k_B = 289 \), and compute the point \( R_B = k_B G = 289G = (265,386) \)
   
b) Compute the point \( k_B Q_A = 289(350G) = 411G = (166,538) \)
   
c) Compute the integer \( s_B = k_B + bx_2 \mod n = 268 \)
   
d) B sends \( R_B = 289G = (265,386) \) to A.

3. A does the following:
   
a) Compute \( aR_B = a(k_B G) = 350(289G) = 411G = (166,538) \)
   
b) Compute the session key: \( S_{AB} = s_A(R_B + x_B x_1 Q_B) = s_A(k_B + x_B x_1 b)G = 365G = (500,408) \).

4. B does the following:
   
a) Compute \( bR_A = b(k_A G) = 677(56G) = 231G = (440,379) \)
   
b) Compute the session key \( S_{BA} = s_B(R_A + x_A x_1 Q_A) = s_B(k_A + x_A x_1 a)G = 365G = (500,408) \)

Note that the session keys are equal since: \( S_{AB} = S_{BA} = 365G = (500,408) \)

**Message Embedding Scheme**

The method proposed by Koblitz represents a message as a point on an elliptic curve. Suppose \( E \) is an elliptic curve given by \( E: y^2 = x^3 + ax + b \) over a field \( \mathbb{F}_p \). Using the following steps to map a message \( m \) to a point on the curve:

1. Express the message \( m \) as a number: \( 0 \leq m \leq p/100 \)
2. Let \( x_i = 100m + i \) for \( 0 \leq i \leq 100 \). For \( i = 0,1,2,\ldots,99 \), compute \( s_i = x_i^3 + ax_i + b \), till we find
\[ s_i^{(p-1)/2} \equiv 1 \mod p \] (\( s_i \) is a square \( \mod p \)).

3. Compute the square root of \( s_i \) as \( y_i \). The point \((x_i, y_i)\) is the corresponding \( m \).

Using the following steps to map a point \((x_i, y_i)\) on the curve to a message \( m \):

1. Compute \( m = [x_i/100] \) (the greatest integer less than or equal \( x_i/100 \))

*Note:* As \( s_i \) can be regarded as a random number in \( F_p^* \), and for a random number \( s_i \), the probability of being square is approximately \( 1/2 \). So after trying 100 times, the probability of not being able to find a square is about \( 2^{-100} \). For the elliptic curve over \( F_p^* \), the method is similar.

### 3. The Message Digest Protocol

The Secure Hash Algorithm (SHA) is part of Secure Hash Standard (SHS) (Rabah, 2005, 3). It is a message-digest function that reads a variable-length input and produces a 160-bit hash of the input. The function that computes the hash is one-way, meaning that it is computationally infeasible to search for an input that evaluates to a given hash value. Because of this one-way property, SHA is used in Digital Signature Standard (Rabah, 2005, 3) to increase efficiency. Instead of signing a long message, only the hash of the message is signed. Likewise, the ECC provides a faster alternative for public key cryptography. Much smaller key lengths are required with ECC to provide a desired level of security, which means faster key exchanges, user authentication, signature generation and verification, in addition to small key storage needs (Rabah, 2005, 3). Some of the other currently approved hash functions are MD5, RIPEM-128, and RIPEM-160 among others.

#### 3.1 The Mechanics of the Hash Algorithm

It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest. This is a fingerprint for the data. A digest has two main properties: In the first case, if even one single bit of data is changed, then the message digest changes as well, and there is a very remote probability that two different arbitrary messages can have the same fingerprint. Secondly, even if someone was able to intercept transmitted data and its fingerprint, that person would not be practically able to modify the original data so that the resulting data has the same digest as the original one. Hashing functions are often found in the context of digital signature. For secure electronic signatures, in a public key infrastructure (PKI) procedure, a hash function must be collision-resistant, which means that it is computationally infeasible to find two different documents yielding the same hashcode (alternatively, it is infeasible to find a different document yielding the same hashcode as a given document). This is a method for authenticating the source of the message, formed by encrypting a hash of the source data. Public key encryption is used to create the signature so it effectively ties the signed data to the owner of the key pair that created the signature (Rabah, 2005, 3). Some of the currently approved hash functions are: SHA-1, MD5, RIPEM-128, and RIPEM-160 etc. The MD5 and SHA1 algorithms are the most commonly used in digital signature applications, where a large file must be “compressed” in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA (Rivest, Shamir & Adleman, 1978; Rabah, 2005, 3).

An typical example of Hash values of various Hash algorithm is shown in Table 4 (Rabah, 2005, 3).

<table>
<thead>
<tr>
<th>Table 4: Hash values output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1 hash of data is: e5e5137567a6a39d385d203222cc73052d9adb31</td>
</tr>
</tbody>
</table>

4. Elliptic Curve Digital Signature Algorithm (ECDSA)

Elliptic Curve Diffie-Hellman (ECDH) (Rabah, 2005,1) and Elliptic Curve Digital Signature Algorithm (ECDSA) Rabah, K. (2005,3) are the Elliptic Curve counterparts of the Diffie-Hellman key exchange and Digital Signature Algorithm, respectively.

The Elliptic Curve Digital Signature Algorithm (ECDSA) works in much the same way as DSA. The only significant difference between ECDSA and DSA is in the generation of $r$. It requires an elliptic curve $E$ over a finite field $F_p$ with a point $P$ of (large prime) order $q$, all of which is public. Each user is identified by a scalar multiple, $xP$, where $x$ is a (secret) number between 1 and $q$. To sign a message, the possessor of $x$ again secretly picks a number $k$ between 1 and $q$ and appends a pair of numbers $r$ and $s$. The number $s$ is exactly as it is for DSA: $k^{-1}(\text{SHA(M)} + xr) \mod q$. But this time $r$ is obtained by computing the point $kP$ on the elliptic curve $E$ over $F_p$: If $kP = (k_1,k_2)$ is the result of that computation, then $r = k \mod q$. To obtain a security level similar to that of the DSA, the parameter $n$ should have about 160 bits. If this is the case, then DSA and ECDSA signatures have the same bitlength (320 bits).

Verifying an elliptic curve signature is also similar to the verification process for DSA: The receiver computes $u$ and $v$ as before, but then computes the point $uP + v(xP)$ on the elliptic curve $E$ over $F_p$. If the $x$-coordinate of this point is not equal to $r \mod q$, the signature is rejected.

The computationally expensive steps in ECDSA are the scalar multiplications $kP$, $uP$ and $v(xP)$. (The computation of $xP$ is done only once, by the owner of $x$.) These computations aren’t hard in the P-versus-NP sense. They’re just tedious and time-consuming. But the special algebraic nature of elliptic curves presents some cost-cutting opportunities. Scalar multiplication, which a naive approach does by repeated doubling and straightforward addition (for example, $9P = 6P + 2P + P$, where $2P = P + P$ and $6P = 2P + 2P + 2P$), is, from the perspective of abstract algebra, a group endomorphism – that is, a mapping of the group $E_p$ into itself. As such, it can sometimes be re-expressed in terms of other endomorphisms that, while theoretically fancier, are easier to compute, much as an appropriate change of basis can turn an ugly matrix multiplication problem into a computational breeze.

Instead of using system-wide parameters, we could fix the underlying finite field $F_q$ for all entities, and let each entity select its own elliptic curve $E$ and point $P \in E(F_q)$. In this case, the defining equation for $E$, the point $P$, and the order $n$ of $P$ must also be included in the entity’s public-key. If the underlying field $F_q$ is fixed, then hardware or software can be built to optimize computations in that field. At the same time, there are an enormous number of choices of elliptic curves $E$ over the fixed $F_q$.

4.1 The Mechanics of ECDSA Algorithm

We now describe ECDSA as an example of a digital signature algorithm. ECDSA is the elliptic curve analogue of the DSA, which is most famous signature protocol. This protocol needs not only the elliptic curve operations, such as scalar multiplication, field multiplication and field inverse multiplication, but also
big integer multiplication, big integer inverse multiplication, modular operation and SHA-1, which is the 160-bit hash function.

In digital signature algorithm, the sender sends a message with the sender’s own unique signature and the receiver validates the received signature. Instead of providing a signature for the entire message, the message is first shortened to a fixed length by a hash function, then a short signature that is valid over the whole message is generated.

In the ECDSA, Alice (A) generates the signature with her secret-key and Bob (B) verifies the signature with A’s public-key. The algorithm below is the ECDSA protocol which A signs the message $m$, and B verifies A’s signature.

**The key generation:**

In ECC, the system parameters such as the prime $p$, elliptic curve $E$, base point $G = (x, y)$, and order $n$ of the point $G$ need to be shared between the sender and the receiver.

**Key generation: Alice (A)**

1. Select a random integer $s$, where $1 \leq s \leq n - 1$,
2. Compute $Q = sG$ the public-key of the sender.
3. A’s public-key $Q$; A’s private-key is $s$.

**The signature is generated as follows:**

**Signature generation: (A)**

1. Hash the message, and obtain the hash value $f = \text{hash}(m) = \text{SHA-1}(m)$.
2. Generate a random number $u$, where $1 \leq u \leq n - 1$
3. Compute $R = uG = (x_g, y_g)$ and $c = x_g \mod n$.
4. If $c = 0$ then go to step 1.
5. Compute $d = u^{-1}(f + sc) \mod n$
6. If $d = 0$ then go to step 1.
7. Output $(c, d)$ as a signature of $m$.
8. Sends the message $m$ and the signature $(c, d)$ for the message $m$, to B.

**The signature is validated as follows:**

The receiver (B) receives the message $m$ and the signature $(c, d)$. Then, the receiver B performs the following procedure to validate the signature:

1. Verify that $c$ and $d$ are integers in $[1, n - 1]$.
2. Hash the message, and obtain the hash value $f = \text{SHA-1}(m)$.
3. Compute $h = d^{-1} \mod n$.
4. Compute $h_1 = fh \mod n$ and $h_2 = ch \mod n$.
5. Compute $P = h_1G + h_2Q = (x_p, y_p)$ and $c' = x_p \mod n$.
6. If $c = c'$, then the signature is valid. Otherwise, it is invalid and rejects the signature.

As can be seen in the above algorithms, we need various operations for implementing the ECDSA protocol. However, we are only concerned with the elliptic curve operations, because these operations dominate the execution time in the protocol schemes. In this protocol, there are two operations, which are: the scalar...
multiplication in signature generation step 3 and the signature validation step 5 and, the point addition in the signature validation step 5.

The key generation:
In ECC, the system parameters such as the prime \( p = 751 \), elliptic curve \( E \), base point \( G = (45,97) \), and order \( n = 727 \) of the point \( G \) need to be shared between the sender and the receiver.

\[ E: y^2 = x^3 - x + 188 \mod 751, \text{ here the prime } p = 751, \#E(\mathbb{Z}/p\mathbb{Z}) = 727 \]

Key generation: Alice (A)
1. Select a random integer \( s = \text{random}(727) = 335 \), and computes \( Q = sG = 335(45,97) = (591,647) \) her public-key of the sender. A’s public-key \( Q \); A’s private-key is \( s \).

The signature is generated as follows:

Signature generation: (A)
1. Hash the message, and obtain the hash value \( f = \text{hash}(m) = \text{SHA-1}(m) \). We will simulate this by selecting the message \( M \) as a random number modulo \( n \): \( M = 543 \), where \( 1 \leq M \leq n - 1 \).
2. Generate a random number \( u = 671 \).
3. Compute \( R = uG = 671(45,97) = (426,550) = (x_R, y_R) \) and \( c = x_R \mod n \) i.e., \( c = 426 \mod 727 = 426 \).
4. Compute \( d = u^{-1}(f + sc) \mod n = (671)^{-1} \cdot (543 + 335 \cdot 426) \mod 727 = 285 \)
5. Output \( (c, d) = (426,285) \) as a signature of \( m \).
6. Sends the message \( m \) and the signature \( (c, d) \) for the message \( m \), to B: \( (M, c, d) = (543,426,285) \).

The signature is validated as follows:
The receiver (B) receives the message \( m \) and the signature \( (c, d) \) i.e., \( (m, c, d) = (543,426,285) \). Then, the receiver B performs the following procedure to validate the signature:
1. Verified: \( (c, d) = (426,285) \) are integers in \([1, n - 1]\).
2. Hash the message, and obtain the hash value \( f = \text{SHA-1}(m) \), i.e., \( M = 543 \).
3. Compute \( h = d^{-1} \mod n = (285)^{-1} \mod 727 = 426 \)
4. Compute \( h_1 = fh \mod n = 543 \cdot 426 \mod 727 = 132 \) and \( h_2 = ch \mod n = 426 \cdot 426 \mod 727 = 453 \).
5. Compute \( P = h_1G + h_2Q = 132G + 453(335G) = 671G = (426, 550) = (x_p, y_p) \Rightarrow c' = x_p \mod n = 426 \).
6. \( c = c' = 426 \), true \( \Rightarrow \) the signature is valid.

How secure is ECDSA? Like all cryptosystems based on the unproved assumptions of complexity theory, that question will have no definitive answer unless some breakthrough shows the answer to be “No.” In 1998, Joseph Silverman, an elliptic curve expert at Brown University, sent shock waves through the elliptic curve crypto community when he proposed a new method for attacking the \( xP \) problem. Silverman’s “xedni” algorithm (“xedni” is “index” spelled backward) uses sophisticated techniques in algebraic geometry to pry loose the \( x \). Fortunately (for cryptographers), the xedni algorithm turns out to require exponential time (Silverman & Suzuki, 1998; Silverman, 2000).

5. Using ECDSA to Implement Bitcoin Cryptocurrency Protocol
In a protocol such as bitcoin, one needs to select a set of parameters for the elliptic curve and its finite field representation that is fixed for all users of the protocol. The parameters include the equation used, the prime modulo of the field, and a base point that falls on the curve. The order of the base point, which is not
independently selected but is a function of the other parameters, can be thought of graphically as the
number of times the point can be added to itself until its slope is infinite, or a vertical line as we had seen
earlier. The base point is selected such that the order is a large prime number.

For its security, Bitcoin uses very large numbers for its base point, prime modulo, and order. In fact, all
practical application of ECDSA uses enormous values. The security of the algorithm relies on these values
being large, and therefore, impractical to brute force or reverse engineer from the ECDLP point of view.

A good example for ECC implementation in case of bitcoin:

Elliptic curve equation \( y^2 = x^3 + 7 \)

Primo modulo = \( 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 = \) FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F

Base point = 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

Order = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

Note that: Public key = private key * Base point

Furthermore, it’s important to note that the base point does not change, therefore, one public key maps to
one private key. This is important. Implies Bitcoin has enormous field. In real practice, who chose these
numbers, and why? A great of research, and a fair amount of intrigue, surrounds the selection of
appropriate parameters. After all, a large, seemingly random number could hide a backdoor method of
reconstructing the private key. In brief, this particular realization goes by the name of secp256k1 and is part
of a family of elliptic curve solutions over finite fields proposed for use in cryptography ((Roy,
Mukhopadhyay & Bengal, 2012).

As we have observed in earlier ECDSA operation in practice – going from private key to public key – is
computationally easy, since (public = private key * base point), in comparison to trying to work backwards
to deduce the private key from the public key (recall ECDLP), which while theoretically possible is
computationally infeasible due to the large parameters used in actual elliptic curve cryptography. Thus,
going from the private key to public key is by design and a one-way trip.

As with the private key, the public key is normally represented by a hexadecimal string. However, the
question is, how do we get from a point on a plane, described by two numbers, to a single number? In an
uncompressed public key the two 256-bit numbers representing the \( x \) and \( y \) coordinates are just stuck
together in one long string. Alternatively, we can also take advantage of the symmetry of the elliptic curve
to produce a compressed public key, by keeping just the \( x \) value and noting which half of the curve the
point is on. From this partial information we can recover both coordinates.

**Bitcoin Addresses**

In practical application Bitcoin address looks (Rabah, 2005,3), see Table 4:

```
cddf358db?bbec17432e6670a2754bac9529595a
```

The address is not a public key. An Address is an RIPEMD-160 hash of an SHA256 hash of a public key.
SHA256 and RIPEMD-160 are also algorithms. Unlike ECDSA, which is used to generate key pairs,
RIPEMD-160 generates a hash. Think of an algorithm like a machine. You put in “stuff” and, hopefully, new “stuff” comes out.

It’s important to note that the RIPEMD uses your public key to create a hash. A bitcoin address is smaller than a public key. That introduces another term, collisions which we came across earlier. When two unique inputs give the same output in a hash algorithm, it’s called a collision. Using an enormous numbers and a strong algorithm reduces collisions. But for Bitcoin it’s because we’re turning large numbers into smaller numbers. Furthermore, for Bitcoin, there are so many possible keys that collisions are astronomically unlikely. More-so, since there are only 21M bitcoins only a very miniscule fraction of keys can even claim a balance. So even if someone were to generate a key pair that collides with another – an astronomical feat – the other key likely wouldn’t have a balance. As of 6 February 2016, there are 15.2 million bitcoins circulation of a capped total of 21 million. Here's an up-to-date graph and other bitcoin network statistics. That means over 72% of all bitcoins are already in circulation. Currently there are 25 new bitcoins produced (mined) every 10 minutes (Kaiko, 2016).

6. The Mechanics of Bitcoins

Recall that your private key must be kept secret and must never be divulged to anyone else. It’s the key that unlocks the funds owed to you in the Bitcoin blockchain. Bitcoin has a scripting system that is used to define the parameters necessary to spend bitcoins. That is, when you build a transaction in addition to referencing previous transaction you’ve received, it contains a script with your private key’s signature and the matching public key. This is used to prove that the provided public key matches the private key used to make the signature. That is, if that private key hashes to (RIPEMD160), the Bitcoin Address in a previously unclaimed transactions, it can be spent. That’s a high-level view of some of the ways Bitcoin uses cryptography.

The Mechanics of Mining Bitcoins

It’s now very clear to you that Bitcoin digital currency is the most famous cryptocurrency, but how does new units get created or come into existence. New units of bitcoin currency are generated by “mining” – which is a computationally intensive task, and it requires a lot of processing power. Essentially, the computer is rewarded for solving the difficult math problems. The processing power is used to verify transactions, so all that number-crunching is required for the cryptocurrency to work. Bitcoin mining is so called because it resembles the mining of other commodities: it requires exertion and it slowly makes new currency available at a rate that resembles the rate at which commodities like gold are mined from the ground. As we had observed, a bitcoin is simply an SHA-256 hash (which is an extremely large number) in hexadecimal format. A person’s bitcoins are stored in a special file called a wallet, which also holds each address the user sends and receives bitcoins from, as well as password/private key known only to the user, which is required before the bitcoins can be spent.

In real practice, mining programs tap into your computer’s hardware resources and put them to work mining Bitcoin. In its essence, bitcoin is a cryptocurrency implemented entirely with open source specifications and software which relies entirely on a peer-to-peer network for both transaction processing and validation (Nakamoto, 2009). Bitcoin mining is the process of adding transaction records to the Bitcoin’s public ledger of past transactions or blockchain. This ledger of past transactions is called the blockchain as it is a chain of blocks. The blockchain serves to confirm transactions to the rest of the networks. Bitcoin nodes use the blockchain to distinguish legitimate Bitcoin transactions from attempt to re-spend coins that have already been spent elsewhere.

It’s important to note that Bitcoin mining is intentionally designed to be resource-intensive and difficult so that the number of blocks found each day by miners remains steady. Individual blocks must contain a proof of work to be considered valid. This proof of work is verified by other Bitcoin nodes each time they receive
a block. Bitcoin uses the hashcash proof-of-work function. A proof-of-work is essentially a piece of data which is difficult (costly, time-consuming) to produce so as to satisfy certain requirements. It must be trivial to check whether data satisfies said requirements. Producing a proof of work can be a random process with low probability, so that a lot of trial and errors are required on average before a valid proof of work is generated.

The primary purpose of mining is to allow Bitcoin nodes to reach a secure, tamper-resistant consensus. Mining is also the mechanism used to introduce Bitcoins into the system: Miners are paid any transaction fees as well as “subsidy” of newly created coins. These both serve the purpose of disseminating new coins in a decentralized manners as well as motivating people to provide security to the system. Additionally, the miner is awarded the fees paid by users sending transactions. The fee is an incentive for the miner to include the transaction in their block. In the future, as the number of new bitcoins miners are allowed to create in each block dwindles, the fees will make up a much more important percentage of mining income.

7. Quantum Computing a Threat to the Security of Bitcoin Cryptocurrency

Traditionally, classical high speed super computers works in the realm of massively parallel processing devices. By focusing of high speed data transfer with the computer and on the algorithms needed to break up these problems into small chunks, standard or near-standard processors could be brought to bear on the problems in vast numbers. By 2014, the world’s fastest computer, Tianhe-2, had around 34000 processors, all working in parallel. As of June 2016 the fastest supercomputer in the world is the Sunway TaihuLight in the People's Republic of China (PRC), with a Linpack benchmark of 93 PFLOPS, exceeding the previous record holder, Tianhe-2, by around 59 PFLOPS. It tops the rankings in the TOP500 supercomputer list. However, even so, these speeds could only be achieved by adding processors, a relatively expensive method that has limitations.

Enters quantum computing. By contrast, quantum computers work on fundamentally different principles. While classical cryptography employs various mathematical techniques to restrict eavesdroppers from learning the contents of encrypted messages, in quantum mechanics the information is protected by the laws of physics. In classical cryptography an absolute security of information cannot be guaranteed. The Heisenberg uncertainty principle and quantum entanglement can be exploited in a system of secure communication, often referred to as "quantum cryptography" (Bennett, Brassard & Ekert, 1992). Quantum cryptography provides means for two parties to exchange an enciphering key over a private channel with complete security of communication.

However, as with the traditional super computers, again time must be spent on writing an efficient algorithm, but this is applied to only a single processing array. This array operates at near to absolute zero to enroll the counter intuitive properties of fundamental particles. This allows the processor to try all possible answers in parallel, settling on the correct one in a highly efficient way. In effect, in less than a century, computers have completely transformed what humanity can envision and achieve. In this respect, quantum computers have the potential to again transform human capabilities.

One of the first areas that would be revolutionized would be cryptography, and with it cryptosystem like cryptocurrencies, for example, Bitcoin, cryptoassets and any of the distributed processing tasks which relay on strong encryption. Further, as you may recall, the algorithm used by many cryptocurrencies is a maths problem based on the properties of certain types of elliptic curves. These maths problems are easy to set, but difficult to crack. To do so using traditional computing tools requires long periods of time or vast numbers of processors, Moreover, as processing power increases they can easily be more difficult by using longer ‘keys’, strings of numbers used as inputs. This makes them ideal for cryptography where key size can be increased until the processing needed to crack them is entirely beyond reach.
Under quantum computing, elliptic curve cryptography is known to be vulnerable to Shor’s algorithm, a quantum algorithm, which if run on a quantum computer would be able to crack the code in short periods of time (Vandersypen et al., 2001; Ekert & Jozsa, 1996). Therefore, for cryptocurrency the most response would be the deployment of encryption methods that are not susceptible to a Shor’s attack. Although this may be possible to implement in an existing cryptocurrency like Bitcoin, it would more likely need the old blockchain to be abandoned in favor of an entirely fresh one. Further, not all cryptocoins are susceptible to mining centralization. There are already alterations to the basic decentralized bitcoins scheme which makes it impossible. These then would be the cryptocoins to survive. So quantum computing is a threat to Bitcoin, but not to the concept of decentralized cryptocurrencies. A move to a new dominant cryptocoins would be difficult to complete rapidly without what amounted to massive overnight currency crash. However, the market would quickly right itself. There would be winners and losers in such a flight, but the concept of decentralized cryptocurrencies would survive.

8. Conclusion
We presented a background to the history of cryptocurrency and its effect on cryptoeconomic. We also developed and presented some intuition about the deep mathematical relationship that exists between public and private keys using elliptic curve cryptography. Also presented were some examples of the math behind digital signatures and its verification using ECDSA and how it’s applied to creation and mining of Bitcoin digital cryptocurrency. A simple mathematical procedure necessary to create the one-way “trap door” function necessary to preserve the information asymmetry which defines ownership of a bitcoin was also given. We also looked at how quantum computing could be a threat to Bitcoin security and best practices to preserve its security, but not to the concept of decentralized cryptocurrencies. A move to a new dominant cryptocoins would be difficult to complete rapidly without what would amount to massive overnight currency crash. However, it’s expected that the market would quickly right itself. There would be winners and losers in such a flight, but the concept of decentralized cryptocurrencies would survive.

9. References


https://www.gtnews.com/articles/cryptocurrency-insights-key-trends-and-risks/


---

**Cite this article:**