TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

KIDE VUOJÄRVI

COMPARISON OF RSA AND ELLIPTIC CURVE CRYPTOSYSTEMS

Bachelor of Science

Examiner: Mikko Nurminen

Submitted for review on 22 November 2018

ABSTRACT

KIDE VUOJÄRVI: Comparison of RSA and elliptic curve cryptosystems
Tampere University of Technology
Literary review of two public key cryptosystems, 22 pages
November 2018
Software Engineering
Major: Software Engineering
Examiner: Mikko Nurminen

Keywords: cryptography, cryptosystems, public key, asymmetric, RSA, ECC, elliptic curves, digital signatures, key exchange, performance, security level, key length

The thesis was a literary review of two public key cryptosystems, RSA and elliptic curve cryptosystem (ECC). Its main focus was on their key generation as well as their use in key exchange and digital signatures, looking into the efficiency of these operations.

RSA is a scheme based on the hardness of integer factorization with large numbers, using exponentiation as its base operation. ECC is based on a discrete logarithm problem implemented on a geometric curve, and it uses point addition as its base operation.

Elliptic curves provided the same level of security with keys that are shorter by an order of magnitude, but RSA had more generally usable encryption possibilities as well as more optimised implementations for digital signatures.

The results showed elliptic curves to have a significant edge against RSA in the future of public key cryptography, but RSA still outclasses ECC in digital signatures.

CONTENTS

TERMS AND ACRONYMS

Alice, Bob	General names used for two people wishing to establish a secure connection
Oscar	A general name used for an attacker wishing to eavesdrop on Alice and Bob
AES	Advanced Encryption Standard, symmetric block cipher established in 2001
ECC	Elliptic Curve Cryptography
RSA	Public-key encryption system designed by Rivest, Shamir and Adleman
NIST	National Institute of Standards and Technology
HTTPS	Hypertext Transfer Protocol Secure
CPU	Computer Processor Unit

1. INTRODUCTION

Cryptography has been used by humans for millenia. Even the ancient Romans realized that a way to hide a message from prying eyes is important. Julius Caesar was famous, among other things, for using the Caesar cipher, where he encrypted his battle plans to his generals by shifting the alphabet three spaces over, so that an "A" became a "D", "B" became an "E" and so on. When these messages were received, they were easy to understand by the generals, but if a message was intercepted, the plans would not be revealed to the attacker.

In time, more sophisticated ciphers were created, such as the Playfair cipher in 1854 and the Enigma machine during the 2nd World War, which still worked on swapping letters with each other. After the development of digital computers, the alphabet shifting was replaced by bit shifting. Modern cryptographic algorithms have continued in this path, though they are now split into symmetric and asymmetric cryptosystems.

The Rivest-Shamir-Adleman cryptosystem (RSA) and elliptic curve cryptosystem (ECC) are two of the most used asymmetric algorithms currently in use. Both of them are in constant use throughout the internet, and they both have their advantages. This said, elliptic curves have been on the rise for over a decade, yet many applications and websites still use the aging RSA system.

This thesis attempts to find out the most common uses of these cryptosystems, as well as their efficiency and performance in these usages. A further research question is why RSA is used despite the fact RSA keys are an order of magnitude greater than ECC keys of equal strength.

The thesis will begin with the definitions of symmetric and asymmetric algorithms in chapter 2 and dive into the computer science and mathematics behind cryptography in section 2.2 and 3. The definitions of RSA and ECC will be presented in chapters 4 and 5, and their functionalities will be compared in chapters 6. Chapter 7 speaks about the performances of different operations of the cryptosystems, attempting to answer the research questions of the thesis.

2. CRYPTOSYSTEMS

Cryptosystems are a way of providing cryptographic tools for a user concerned with the security of their communication. These tools include *encryption and decryption*, meaning the obfuscation and deobfuscation of a message, *digital signatures*, meaning a way to produce a unique and verifiable proof of identity, as well as *key exchange*, meaning a way to securely exchange encryption keys for other algorithms over an insecure channel.

2.1 Symmetric and Asymmetric Cryptographic Algorithms

Cryptographic algorithms can be separated into two categories, symmetric and asymmetric, depending on the number of keys they use. In a symmetric algorithm everyone wishing to either encrypt or decrypt a message has to use the same key. This key is often known as a *shared secret*, because it is something known to all involved parties. Advanced encryption standard (AES) and triple data encryption standard (3DES) are examples of symmetric algorithms currently in use [13][19]. Both symmetric and asymmetric are based on modular arithmetic in integer rings, as presented in sections 3.1 and 3.2.

An asymmetric algorithm has two keys, one of which is known as a public key and the other a private key. Each person creates their own pair of keys and shares the public key. This public key can now be used to encrypt data, but the same data can only be decrypted with the private key. It is also impossible to calculate the private key from the public key.

While multiple algorithms with both schemes have been invented and implemented, the general trend seems to favor symmetric algorithms for speed, efficiency, and security. For example, the modern symmetric algorithms currently in use are faster by a factor of 1000 in comparison to RSA. Thus, ironically, public key algorithms are rarely used for actual encryption. [18, p.154]

The need for public key algorithms arises from the need to safely communicate over an unsafe network. Imagine that Alice and Bob live on different continents, but wish to communicate with each other privately. They decide to use AES for encryption because of its speed and security, but they need to share a key for the algorithm. A symmetric algorithm is useless if the shared key cannot be delivered to both parties without an attacker being able to snatch it and eavesdrop on their conversation, and visiting the other and agreeing on a key would be time consuming and expensive.

Instead, they decide to use an asymmetric cryptographic scheme to share the AES key. Alice generates a public and a private key and shares the public key with Bob. Bob then decides the key for AES, encrypts it with the public key, and sends the message to Alice

via the internet. Alice, being the only one with the private key, is the only person who can decrypt Bob's message. Although this decryption will most likely take some time, it is a reasonable trade-off, because this operation is only done once during the conversation, and now they can use a symmetric algorithm for the rest of their conversation. This way of using asymmetric cryptography to share a key for a symmetric algorithm is called a *hybrid cryptosystem*.

In this way, the asymmetric algorithms enable secure conversation over insecure channels. The key exchange is perhaps the most common use for asymmetric algorithms.

2.2 Key Lengths and Security Levels

To be able to compare different encryption algorithms, a way of measuring the security level of the algorithm is required. The security level of an algorithm is measured in bits, and Paar et al states that "an algorithm has the security level of n bit if the best known attack requires 2^n steps". This is especially simple for all symmetric algorithms, because their security level is always equal to their key length, but for asymmetric algorithms this is more complicated.[18, p.156]

Whereas symmetric algorithms are based on key substitution between rounds of diffusion and confusion, designed to make linear attacks impossible, current asymmetric algorithms are based on mathematical functions. These functions are designed so that knowing the starting values, it is easy to calculate the result, but very hard to get to the starting values from the result. These functions may be subject to different, more efficient attacks, and thus asymmetric algorithms require longer keys to achieve the same security level as symmetric ones. As seen in table 1, raising the security level of RSA from 80 bit to 256 bit requires huge amounts of effort and computational power.

Table 1. *Bit lengths of public-key algorithms for different security levels [18, p.156]*

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

Although predicting the computers of the future is quite ambitious, Paar et al. estimates the time frames in which the security levels are actually secure. Symmetric keys with length 64 bit or less can be cracked within a few days, while 112-128 bit keys are secure for several decades in the absence of quantum computers. Key lengths of 256 bit or more are estimated to be secure for several decades even with quantum computers. [18, p. 12] Currently, National Institute of Standards and Technology (NIST) suggests, that RSA key size should be 2048 bits, and 3072 bits if longer term security is required [2].

3. MATHEMATICS IN ENCRYPTION

There are some aspects of mathematics that are essential to be able to understand both RSA and ECC. In the following subsections, some of these aspects are presented.

3.1 Modular Arithmetic

In general mathematics, modulo operation is the search of the remainder of a given division.

$$a \equiv r \bmod m \tag{1}$$

In equation 1 a, r, m $\in \mathbb{Z}$ and m > 0 if $a - r$ is divisible by m. m is called the modulus and r is called the remainder. The $\equiv$ symbol means we can handle all the elements with equal remainders as though they were the same. For example $8 \equiv 5 \equiv 2 \bmod 3$ because each of them gives the same remainder when divided by m.

3.2 Integer Rings

We define a ring with the following rules:

1. A set $Z_m = \{0, 1, 2, ..., m-1\}$
2. Two operations "+" and "×" for all $a, b \in \mathbb{Z}_m$ such that:
   (a) $a + b \equiv c \bmod m, (c \in \mathbb{Z}_m)$
   (b) $a \times b \equiv d \bmod m, (d \in \mathbb{Z}_m)$

Although these rings are closed, the results of both of the given operations are always in the ring because of the modulo operation. Neutral elements for both addition (0) and multiplication (1) exist, but the multiplicative inverse exists only for some elements. If $a \in \mathbb{Z}_{>}$, a^{-1} is defined as follows:

$a \times a^{-1} \equiv 1 \bmod m$

If an inverse exists, we can use a division operator by multiplying with the inverse.

Rings are also associative and distributive, meaning $a+(b+c) = (a+b)+c$, $a \times (b \times c) = (a \times b) \times c$ and $a \times (b+c) = (a \times b) + (a \times c)$ for all $a, b, c \in \mathbb{Z}_m$.

3.3 Cyclic Groups

Some integer rings have special elements that, when raised to successive powers, will eventually produce every element in the ring. These rings are known as cyclic groups, and all the elements with this special property are known as *generators* or *primitive elements*.
$g^1 = g = 3$
$g^2 = g \cdot g = 9 \equiv 2 \bmod 7$
$g^3 = g^2 \cdot g = 6 \equiv 6 \bmod 7$
$g^4 = g^3 \cdot g = 18 \equiv 4 \bmod 7$
$g^5 = g^4 \cdot g = 12 \equiv 5 \bmod 7$
$g^6 = g^5 \cdot g = 15 \equiv 1 \bmod 7$
$g^7 = g^6 \cdot g = g \equiv 3 \bmod 7$

In the ring $\mathbb{Z}_7$ element $g = 3$ is primitive, as seen above.

3.4 Euclidean Algorithm

Euclidean algorithm is an algorithm designed to find the greatest common divisor of two integers. Again, the proof is bypassed here, but with its extended counterpart, it is very useful in cryptography. Here we have a Python implementation of the algorithm:

```
def gcd(a, b):
  while b != 0:
    a, b = b, a % b
  return a
```

Where $a, b \in \mathbb{Z}$. Usually $a > b$, but if $a < b$, the first iteration will swap the values with each other. The algorithm lessens the values iteratively and when the modulo of b reaches zero, the value of the greatest common divisor is a.

An important use of the Euclidean algorithm is discovering whether two numbers have no common divisors, meaning their greatest common divisor is 1. These kinds of numbers are called *relatively prime*, or *co-prime*.

3.5 Extended Euclidean Algorithm

The extended Euclidian algorithm is a version of the Euclidian algorithm which stores the temporary results as sum of the original parameters. It can be used to calculate the inverses of numbers in a cyclic group.

3.6 Euler's Phi Function

Euler's phi function is a function $\Phi(m)$ that finds the number of relatively prime numbers in $\mathbb{Z}_m$. It can be shown that the $\Phi(p) = p - 1$ where p is prime, and for non-primes it can be shown that the value of the phi function can be found by applying the phi function to its factors.

If the factorization of an integer is defined by

$m = p_1^{e_1} \cdot p_2^{e_2} \cdot ... \cdot p_n^{e_n}$,

the phi function can be defined as

$$\Phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1}). \tag{2}$$

For example, if $m = 1440 = 32 \cdot 45 = 2^5 \cdot 3^2 \cdot 5 = p_1^{e_1} \cdot p_2^{e_2} \cdot p_3^{e_3}$

$\Phi(m) = (2^5 - 2^4)(3^2 - 3^1)(5^1 - 5^0) = 16 \cdot 6 \cdot 4 = 384$,

meaning 384 integers < 1440 are co-prime with it.

It is worth mentioning that without knowing the factorization of the integer m, calculating the phi function of a large number becomes very cumbersome.

3.7 Fermat's Little Theorem and Euler's Theorem

Fermat's little theorem, in the equation 3, is a small yet important discovery in modular arithmetic. It states that if you raise $a \in \mathbb{Z}$ to power p, where p is a prime, it is equivalent to $a \bmod p$.

$$a^p \equiv a (\bmod p) \tag{3}$$

Now, as most integer rings in cryptography are of a prime order, this simplifies the calculations significantly, because from it follows that

$$a^{p-1} \equiv 1 (\bmod p) \tag{4}$$

Now, whenever there is a calculation involving a^n where $n > p$, the calculation can be simplified by subtracting $p - 1$ from the exponent.

A more generalized form of this theorem is Euler's theorem, presented in equation 5:

$$a^{\Phi(m)} \equiv 1 (\bmod m) \tag{5}$$

Euler's theorem is used in RSA key generation, as it enables public-private key pairs to be calculated so that when both keys are applied to a plaintext, the result will be the original text.

3.8 Discrete Logarithm Problems

Let $\mathbb{Z}_m$ be a cyclic group presented in section 3.3, an element $g \in \mathbb{Z}_m$ be a generator for this group and an integer x be a private key. In $\mathbb{Z}_m$, finding out the value of g^x is relatively simple, even if the m is a large number. There are fast algorithms that can calculate g^x in $O(logn)$ time, but given an integer $a \in \mathbb{Z}_m$, it is hard to find out x so that $g^x = a \bmod m$. Determining x if g, a and m are known is called a *discrete logarithm problem in* $\mathbb{Z}_m$. [9, p.12]

4. RSA

RSA is the most used public key cryptosystem currently in general use [24, p.109]. It was proposed by cryptologists Rivest, Shamir and Adleman in 1978 [21], and multiple variations of it have been built since then. Due to national security reasons, the proliferation of RSA was limited by the US arms regulation as an auxiliary military equipment until 1992 [6, Category XI]. Even after multiple decades and several attacks deviced against it, RSA with a long enough key is still a secure enough system to use [3].

4.1 General Working Model

When using RSA, Alice creates a public and a private key. The public key consists of a public exponent and a modulus, and the private key is a private exponent. This public key can now be shared over an insecure channel. When Bob wishes to contact Alice, all he needs to do is find Alice's public key and encrypt his message by raising it to the public exponent. After this has been done, Bob can send the message over an insecure channel, since only Alice will be able to decrypt the message.

In theory, all three of these numbers are very, very large, as the suggested length of the modulus is in between 1024-4096 bits, which means the key size is also of the same length.

4.2 Source of Security and Key Generation

The basis of the security in RSA is that currently computers are not good at factoring numbers, especially large ones [21]. At the present, largest factored numbers are in the general area of 768 bits, while some special cases have been solved with as many as 1061 bits. [5] [14]

The key generation is done in five phases [18, p.176].

1. Two primes, p and q are chosen.
2. The product n of these primes is calculated $n = p \cdot q$.
3. It is now easy to calculate $\Phi(n) = \Phi(p) \cdot \Phi(q) = (p-1)(q-1)$, as presented in section 3.6.
4. The public exponent $e \in 1, 2, \ldots, \Phi(n) - 1$ can now be selected such that $gcd(e, \Phi(n)) = 1$, using the Euclidean algorithm 3.4

5. The private exponent d is calculated using the extended euclidean algorithm 3.5 such that
   $d \cdot e \equiv 1 \bmod \Phi(n)$.

This hardness of factoring presents an opportunity for a strong cryptographic function. While multiplying two or more very large primes, say, of size 1024 bits or longer, can be done by a modern CPU in a relatively fast manner, finding out the primes from the resulting 2048+ bit product could not be accomplished quite as easily. [12] As Kleinjung et al. noted, the factoring of a 768 bit number took close to two years using hundreds of computers, and would have taken closer to fifteen hundred years if done with only one machine. This said, they did suggest abandoning 1024 bit keys for RSA within a few years due to still increasing efficiency of integer factorization. [14]

4.3 Encryption and Decryption

The message x being encrypted can be no longer than the size of the public modulus. This is because if the modulus n is exceeded, the resulting x is no longer unique, because $x + n \equiv x \bmod n$.

Both encryption and decryption of a message involve calculations with large integers. In encryption, the byte conversion of the message is raised to the public exponent. As this all happens in an integer ring, the encrypted message is the result of a modulo operation.

In decryption, the encrypted message is now raised to the private exponent, and as seen in equation 5, the original message will be received.

4.4 Implementation

Although the public exponent may be chosen to be an exponent with bit length equal to that of the modulo, in practice it is usually rather small, between 2 and 17 bits [18, p.183]. This because the exponentiation of the encrypted message is a rather taxing operation. Using a short public exponent decreases the amount of work required from the sender, although it often makes the private exponent significantly higher. This practice is a good trade-off, because the owner of the public key, say, a global corporation, can have efficient chips decided only for RSA decryption [4]. These chips make the operation manageable and quick for both parties.

5. ELLIPTIC CURVE CRYPTOGRAPHY

In 1985, Koblitz [15] and Miller [17] independently of each other suggested the use of elliptic curves in cryptography, though they entered the general use only in 2004.

An elliptic curve of this kind is defined as a cyclic group of integer pairs that satisfy the equation

$$y^2 = x^3 + ax + b, \tag{6}$$

along with a point of infinity O, defined to have the coordinates $(0, \infty)$.

$$P + O = P, \tag{7}$$

$$P + (-P) = O. \tag{8}$$

These points can be added to each other, as well as to themselves, and form a modular set.

5.1 Computing with Elliptic Curves

As with integer rings, elliptic curves have generators as well, but as expected, these generators are points on the curve. Scalar multiplication on a generator point P can be used to produce all the elements on a modular curve.

The use of the word *addition* as a name for the group operation is rather arbitrary, because the actual group operation "+" is actually something entirely different. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on the curve. The addition of these two points produces a third point, $R = (x_3, y_3)$ such that

$$P + Q = R$$
$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

$$x_3 = s^2 - x_1 - x_2 \bmod p$$
$$y_3 = s(x_1 - x_3) - y_1 \bmod p$$

where

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \bmod p; \text{ if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} \bmod p; \text{ if } P = Q \end{cases}$$

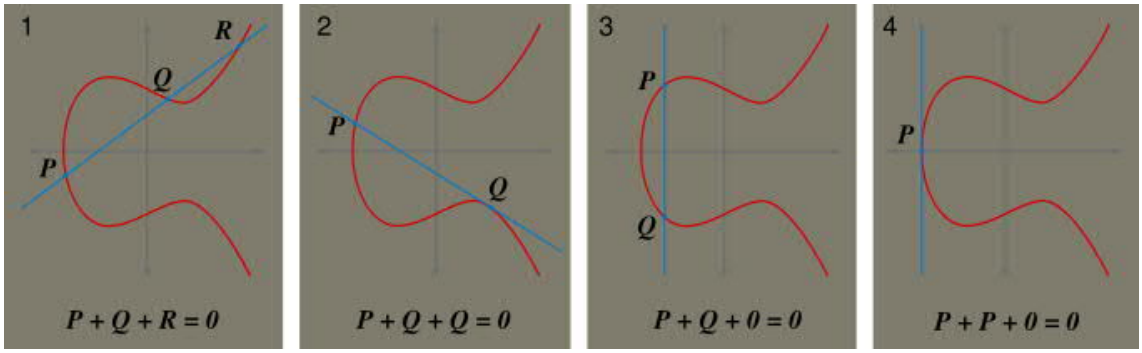

Figure 1. *Example elliptic curves [26]*

These operations seem rather unintuitive, but luckily there is a simpler geometric representation for these calculations.

On the four curves presented in figure 1, points P and Q are added. If $P \neq Q$, they define a line that intersects the elliptic curve in a third point R. R is then reflected over the x-axis, and the result of this reflection is the result of the "+" group operation. Thus $P+Q=-R$.

If $P = Q$, meaning it's value is being doubled, there is only one point on the curve. In such a case the tangent line is calculated. This tangent has up to one intersection with the curve (zero, if the $P = O$), and the x-axis reflection of the point is the result. Thus $Q+Q=-P$.

The last two images show $P+(-P)=0$ and the special case of $P=-P$, the tangent line of the point P where the curve intersects the x-axis.

5.2 Discrete Logarithm Problem

While the number of elements within $\mathbb{Z}_m$ was easy to find out for RSA, knowing the number of points on an elliptic curve is not quite as easy to find out. Luckily, only the approximate number is required to find out suitable key lengths, and this approximation $\#E$ for elliptic curve E modulo a prime p can be found with Hasse's Theorem. [18, p.247]

$$p+1-2\sqrt{p} \leq \#E \leq p+1+2\sqrt{p} \tag{9}$$

The theorem states that the number of points is approximately in the range of the prime p. This means if we need a curve with 2^n elements, we only need a prime that is close to n bits long.

The discrete logarithm problem for elliptic curves is finding the integer d in equation 10.

$$P+P+\ldots+P=dP=T, \tag{10}$$

where P is a generator for E, T is another element on the curve and d is the factor of P such that $1 \leq d \leq \#E$.

5.3 Encryption and Decryption

While elliptic curves have multiple uses for cryptographic purposes, such as key agreement and pseudo-random number generation, they are mostly used for encryption indirectly. This is because the elliptic curve does not include all possible messages, due to only some of the points being in the modular ring [20]. The asymmetric-symmetric hybrid encryption is done by using a symmetric encryption scheme (for example AES) with key-sharing via ECC.

5.4 Implementation

Before ECC can be used, a curve with good cryptographic properties must be selected. Because this is a rather cumbersome process, standardized curves have been published and are in general use. [16] A complete example of such a curve will not presented here, because presenting such long numbers would not benefit the reader. An example of a prime specifying a 160-bit elliptic curve is
$p = 0xE95E4A5F737059DC60DFC7AD95B3D8139515620F$.

Software implementations of ECC are available and effective in most cases, but as seen in table 5, the verification of a ECC digital signature, Sect. 6.2, is a taxing operation. For servers having a heavy load of signatures to produce, a hardware solution is often more suitable.

This said, ECC key size makes it a good candidate for lightweight applications, for example RFID tags. These chips can be produced with a low amount of logic gate equivalents and are fast enough for a use of this kind.

6. APPLICATIONS

RSA and ECC both have multiple cryptographic applications, some of which are discussed below.

6.1 Key Exchange

Due to public key algorithms being rather cumbersome in comparison to symmetric cryptosystems, key exchange is one of the more used applications of both cryptosystems. Figures 2 and 3 show the workings of RSA and ECC key exhange, respectively.

In RSA Alice publishes her public key, $k_{pub} = (e,n)$. Bob generates a key, $m = k_{sym}$, for a symmetric algorithm as he pleases, encrypts it using the k_{pub} such that $y = m^e \mod n$, and sends y to Alice. Alice then decrypts the message y using her private key, $y^d \mod n = m$, and both parties can now safely encrypt and decrypt using the AES with their shared key k_{sym}. This information is more clearly presented in figure 2.

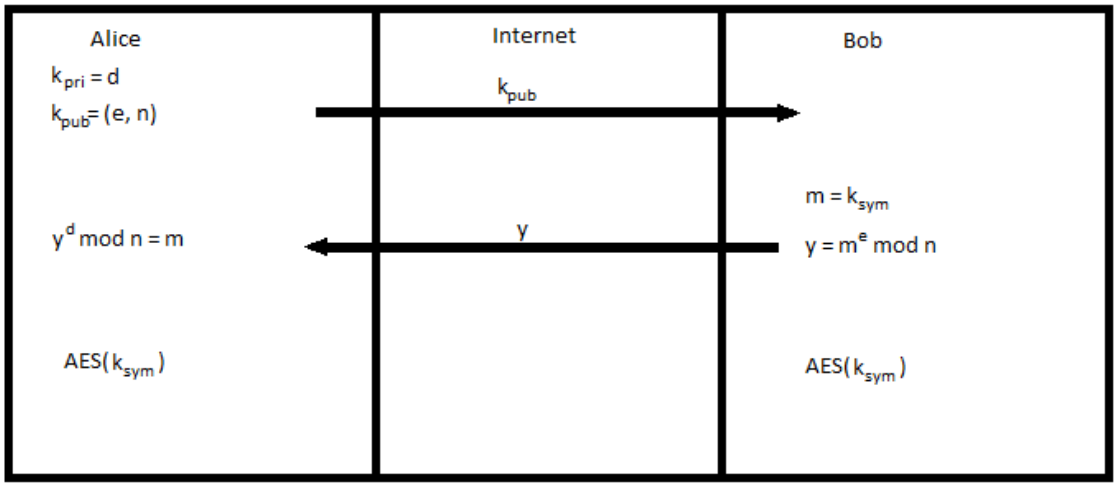

Figure 2. *RSA key exchange*

In ECC, however, Alice and Bob agree on a public number P, and a public elliptic curve E, where P is a generator for E. Generators are presented in section 3.3. Both Alice and Bob then decide a private key of their own, $k_{priA} = a$ for Alice and $k_{priB} = b$ for Bob. Both parties then multiply P with their private keys, so that $aP = A$ and $bP = B$, to create their public keys and send it to the other person. Both parties can now multiply the newly received public key by their own private key, landing on the same point $aB = bA = T_{AB}$, the shared secret. This information is more clearly presented in figure 3.

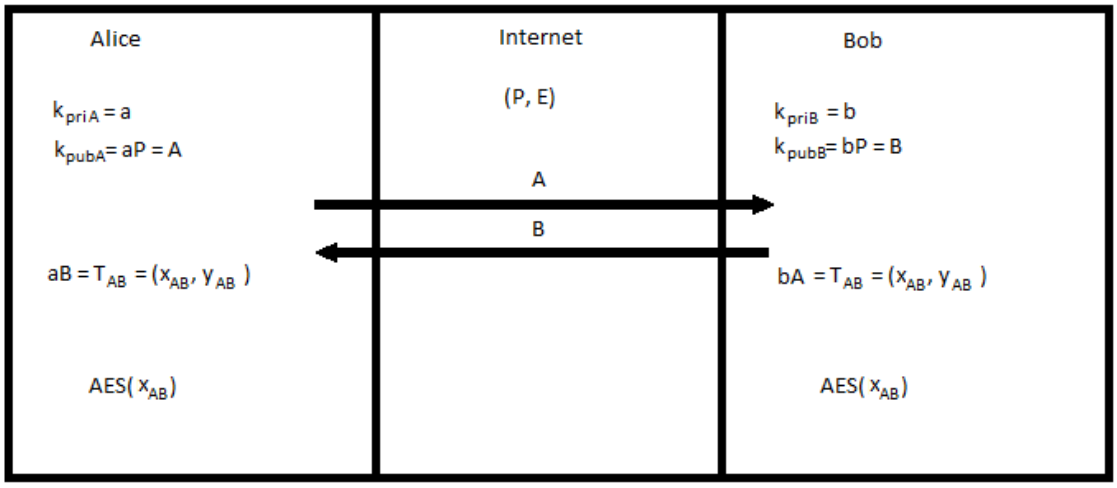

Figure 3. *ECC key exchange*

One of the coordinates of the shared point can now be used for a symmetric key, say, with AES. The single coordinate has a high probability of being as long in bits as the modulo, so it will provide adequate security. Because the coordinates of the point are linked to one another, only one of them is used. As either coordinate can be easily calculated if one is known, there is no reason to use both.

Now, as seen in table 1 the required key lengths for RSA and ECC for the same level of security are quite different. Naturally, longer keys lead to higher number of calculations, higher strain on the CPU, and all-in-all less efficient communication. For this reason ECC is used on chips and other smaller devices more often than RSA.

6.2 Digital Signatures

Digital signature is a way of providing proof of the sender of the message. This is achieved by using a private key to create a signature that can be unlocked with the public key. In RSA signing is actually this simple. Alice can calculate the signature:

$$s = sig_{k_{pr}}(m) = m^d \bmod n \tag{11}$$

Now, if the public exponent was chosen to be small, Bob can verify this signature is very quickly:

$$y = s^e \bmod n. \tag{12}$$

Now, if $y \equiv m \bmod n$ the signature is valid. Alice, being the only one who has the private key, is provably the sender of the message.

In ECC the computation of the signature is not quite as straightforward, but equally as secure. The Elliptic Curve Digital Signature Algorithm (EDCSA) was first proposed by

Scott Vanstone in 1992. [11]. The algorithm itself is omitted from this thesis, because what it does is more important than how it does it.

The Federal Information Processing Standards (FIPS) are a set of standards defining, among other things, cryptographic algorithms. Both RSA Digital Signature Algorith and EDCSA are currently defined in FIPS 186-4. [7]

6.2.1 Nonrepudiation

Nonrepudiation is defined as the assurance of not being able to deny something. It is important, say, for a car factory to be able to trust that the person who just ordered a brand new Cadillac with all the latest technology will actually be purchasing said car. If the order is signed with a digital signature, it can be shown that only the person buying the car could actually have made the order.

6.2.2 Identification

Digital signatures can be used to identify the bearers of private keys. These private keys may often lay in smart cards or RFID tags, which require a fast way to proof of the bearer's identity. [22]

7. PERFORMANCE

Hypertext transfer protocol (HTTP) is a very common way of transferring hypertext over the internet. The secure version of this protocol, HTTPS, was introduced in 1994 [25]. HTTP is an insecure protocol, because it basically sends all the data in plaintext, so that a malicious individual might capture all the sent messages. HTTPS encrypts all the communication, making this impossible.

Since July 2018 Google Security Blog announced that every site not using HTTPS will be marked unsafe [8], and the rising amount of encryption over the internet calls for efficient algorithms. Sites such as Google, Facebook, Twitter etc. get enormous amounts of visitors per second, meaning the performance of their public key algorithms is critical.

The group operation of RSA, exponentiation, can be done cheaply in a CPU, whereas the point addition of ECC is very expensive [23]. While calculations in RSA are faster with equal key lengths, RSA requires significantly longer keys to achieve the same security level as ECC keys, as seen in table 1. The following subsections will discuss the performance of different operations of RSA and ECC.

7.1 Key Generation Performance

Although creating one strong public key might seem enough for a person or a service, in practice multiple keys are created. This is because if one private key is compromised, the leak doesn't grant the attacker access to all the victim's communication. Therefore the performance of key generation is of great importance. Values for times for generating keys of different lengths are presented in table 2.

Key Length (bits)		Time (s)	
RSA	ECC	RSA	ECC
1024	163	0.16	0.08
2240	233	7.47	0.18
3072	283	9.80	0.27
7680	409	133.90	0.64
15360	571	679.06	1.44

***Table 2**. Key Generation Performance [10]*

Even with the smaller key sizes the ECC is better than RSA, and especially with the more sizable ones the difference is evident. In generating keys for a similar security level, ECC is clearly superior.

It is also worth noting, that due to significantly shorter key lengths for similar levels of security, the number of transistors required for implementing ECC in hardware is way lower than that of RSA[23]. This to leads faster circuits and cheaper chips, especially if only short messages need to be secured.

Internet of Things (IoT) is a phrase used to describe small items that are connected to the internet, for example heart monitors, smart ovens and such. These objects often contain sensitive data, and therefore their security is paramount, but their size limits the security applications that can be implemented. In these devices, the shorter key lengths of ECC are a a great advantage, especially if in the future the keys have to be lengthened. [1]

7.2 Digital Signature Performance

Digital signatures have to be handled in two parts, signing and verifying. Time values for signature operations with different key lengths are presented in tables 3 and 4.

Key Length (bits)		Time (s)	
RSA	ECC	RSA	ECC
1024	163	0.01	0.15
2240	233	0.15	0.34
3072	283	0.21	0.59
7680	409	1.53	1.18
15360	571	9.20	3.07

Table 3. *Signature Generation Performance [10]*

As seen above, an ECC signing has a greater overhead than that of RSA. The increase in ECC signing time doesn't rise quite as fast as RSA's does, which may be more important in the future. Within the RSA key lengths between 2048 bits and 3072 bits, as suggested by NIST, RSA digital signatures are still faster than those of ECC with similar security levels.

Key Length (bits)		Time (s)	
RSA	ECC	RSA	ECC
1024	163	0.01	0.23
2240	233	0.01	0.51
3072	283	0.01	0.86
7680	409	0.01	1.80
15360	571	0.03	4.53

Table 4. *Signature Verification Performance [10]*

The performance of signature verification can be equally as important as the actual signing. Understandably, even if signing is really fast but verification takes minutes, the system is not really usable.

As mentioned in subsection 4.4, the public exponent of RSA systems is often chosen as a two to seventeen bit number. Calculating modular exponentiation with such a small exponent is extremely fast. This explains the great gap between signing and verifying a 15360 bit key RSA signing, most of the work is done on the side of the public key holder.

Even though RSA slows down in signing when keys get longer, RSA is still the more efficient choice for situations where a lot of signature verification needs to be done.

7.3 Overview of Performance

Sinha et al compared ECC and RSA in their study, and came up with table 5. They concluded that the key generation for RSA is significantly slower, and if RSA wants to keep up with the security level, the key sizes need to be greater. This means exponentially greater key generation times and weakening efficiency on it's part. They do also note, that since RSA has been around for quite a bit longer, ECC has more potential undiscovered attacks against it. [23]

Parameters	ECC	RSA
Computational overhead	Roughly 10 times than that of RSA can be saved	More than ECC
Key sizes	System parameters and key pairs are shorter for the ECC	System parameters and key pairs are longer for the RSA
Bandwidth saving	Considerable savings over the RSA	Much less saving than ECC
Key Generation	Faster	Slower
Encryption	Much faster than RSA	At good speed but slower than ECC
Decryption	Slower than RSA	Faster than ECC
Small Devices efficiency	Much more efficient	Less efficient than ECC

Table 5. *ECC and RSA: an overview [23]*

Still, as seen in tables 5 and 4, RSA is still more efficient than ECC in decryption and digital signatures. Therefore, it is unlikely that RSA will be abandoned in quite a while.

Both algorithms have a problem with their key sizes, as their computational complexity rises with the cube of the key length. [18, p. 157] If an efficient attack against ECC is found and the key size needs to be raised significantly, ECC may be too inefficient to use.

8. CONCLUSIONS

While RSA and ECC have many similar functionalities, and both of them are based on modular rings, their implementations differ greatly. RSA's basic operation, exponentiation, is quite straight forward in comparison to ECC's point multiplication, and this can be seen in their base level performance. This makes the RSA base operation way faster, but its advantage is lost because of the larger key size, as seen in table 2. The increasing key sizes may make RSA obsolete in the future.

RSA does still have it's uses, especially with digital signatures. Tables 3 and 4 show the undoubted superiority of RSA, especially with signature verification. It also has the advantage in public key encryption. Because an elliptic curve does not have all possible messages within the group, it is easier to encrypt and decrypt messages with RSA. If a hybrid scheme is used, ECC is again a better option.

In Internet of Things, the shorter key lengths and simpler implementation of ECC provide superiority over RSA, especially in the future. IoT is a growing field, and more research on security applications on IoT devices is required.

ECC, being the newer algorithm, may still have undiscovered attacks against it. Because of how long RSA has been available, it is unlikely that a new attacks will be discovered.

Most of the signs seem to point toward the superiority of ECC in the public key cryptography field. The efficiency of lower key lengths is significant, and as it gains more recognition, it is quite possible that more and more systems will turn to use it.

REFERENCES

[1] M. Bafandehkar, S.M. Yasin, R. Mahmod, Z.M. Hanapi, Comparison of ecc and rsa algorithm in resource constrained devices, in: IT Convergence and Security (ICITCS), 2013 International Conference on, IEEE, 2013, pp. 1–3.

[2] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, Nist special publication 800-57, NIST Special publication, Vol. 800, Iss. 57, 2007, pp. 1–142.

[3] D. Boneh *et al.*, Twenty years of attacks on the rsa cryptosystem, Notices of the AMS, Vol. 46, Iss. 2, 1999, pp. 203–213.

[4] E.F. Brickell, A survey of hardware implementations of rsa, in: Conference on the Theory and Application of Cryptology, Springer, 1989, pp. 368–370.

[5] G. Childers, Factorization of a 1061-bit number by the special number field sieve, Cryptology ePrint Archive, Report 2012/444, 2012. Accessed: (2018-11-07), https://eprint.iacr.org/2012/444.

[6] Electronic Privacy Information Center, International traffic in arms regulations, 1992. [Online; accessed 2018-11-22]. Available: https://epic.org/crypto/export_controls/itar.html

[7] P. FIPS, 186-4, Digital Signature Standard (DSS), 2013.

[8] Google Online Security Blog, A secure web is here to stay, 2018. [Online; accessed 2018-11-22]. Available: https://security.googleblog.com/2018/02/a-secure-web-is-here-to-stay.html

[9] D. Hankerson, A.J. Menezes, S. Vanstone, Guide to elliptic curve cryptography, Springer Science & Business Media, 2006.

[10] N. Jansma, B. Arrendondo, Performance comparison of elliptic curve and rsa digital signatures, nicj. net/files, 2004.

[11] D. Johnson, A. Menezes, S. Vanstone, The elliptic curve digital signature algorithm (ecdsa), International journal of information security, Vol. 1, Iss. 1, 2001, pp. 36–63.

[12] J. Jonsson, B. Kaliski, Public-key cryptography standards (PKCS)# 1: RSA cryptography specifications version 2.1, Techn. rep., 2003. Accessed: (2018-11-08), https://www.rfc-editor.org/info/rfc3447.

[13] P. Karn, P. Metzger, W. Simpson, The ESP triple DES transform, Techn. rep., 1995.

[14] T. Kleinjung, K. Aoki, J. Franke, A.K. Lenstra, E. Thomé, J.W. Bos, P. Gaudry, A. Kruppa, P.L. Montgomery, D.A. Osvik *et al.*, Factorization of a 768-bit rsa modulus, in: Annual Cryptology Conference, Springer, 2010, pp. 333–350.

[15] N. Koblitz, Elliptic curve cryptosystems, Mathematics of computation, Vol. 48, Iss. 177, 1987, pp. 203–209.

[16] M. Lochter, J. Merkle, Elliptic curve cryptography (ECC) brainpool standard curves and curve generation, Techn. rep., 2010.

[17] V.S. Miller, Use of elliptic curves in cryptography, in: Conference on the theory and application of cryptographic techniques, Springer, 1985, pp. 417–426.

[18] C. Paar, J. Pelzl, Understanding Cryptography, 2nd edition, Springer, 2010, 239-255 p.

[19] N.F. Pub, 197: Advanced encryption standard (aes), Federal information processing standards publication, Vol. 197, Iss. 441, 2001, p. 0311.

[20] G. Raju, R. Akbani, Elliptic curve cryptosystem and its applications, in: Systems, Man and Cybernetics, 2003. IEEE International Conference on, IEEE, 2003, Vol. 2, pp. 1540–1543.

[21] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, Vol. 21, Iss. 2, 1978, pp. 120–126.

[22] C.P. Schnorr, Efficient identification and signatures for smart cards, in: Conference on the Theory and Application of Cryptology, Springer, 1989, pp. 239–252.

[23] R. Sinha, H.K. Srivastava, S. Gupta, Performance based comparison study of rsa and elliptic curve cryptography, International Journal of Scientific & Engineering Research, Vol. 4, Iss. 5, 2013, pp. 720–725.

[24] P. Thorsteinson, G.G.A. Ganesh, NET security and cryptography, Prentice Hall Professional, 2004.

[25] C. Walls, Embedded software: the works, Elsevier, 2012.

[26] Wikipedia, the free encyclopedia, example elliptic curves, 2007. [Online; accessed 2018-11-19]. Available: https://en.wikipedia.org/wiki/File:ECClines.svg