Cryptography: An Introduction to Internet Security

October 23, 2018

Abstract

This essay aims to introduce the reader to the history of cryptography, its core concepts from elementary maths, and its applications in the modern digital age. RSA cryptography will be explored in more detail, as a specific application.

Contents

1 Introduction

1.1 Basic definitions

[21]

Definition 1.1. *A message in its original form, pending altering/encoding, is called the* ***plaintext****.*

Definition 1.2. ***Cleartext*** *refers to data that is stored or transmitted unaltered. The terms plaintext and cleartext are sometimes used interchangeably.*

Definition 1.3. *The key, or rule, used to encode plaintext is called a* ***cryptosystem****, or* ***cipher****. (Also spelled* ***cypher****.)*

Definition 1.4. *A message that has been altered using the cipher is called the* ***ciphertext****-this is assumed to be unreadable to an eavesdropper.*

Definition 1.5. *The process of changing plaintext into ciphertext is called* ***encryption****. This requires both the plaintext and a* ***key****.*

The process in the opposite direction, acquiring the plaintext from the ciphertext (which requires the ciphertext and a key), is called ***decryption****.*

Definition 1.6. *A* ***symmetric-key cryptosystem*** *is a cryptosystem that uses the same key to encrypt and decrypt a message. Both parties would need to know the key. A cryptosystem in which the keys for encryption and decryption are different is called a* ***public-key cryptosystem****. (In public-key cryptography, the encryption key is made public and the decryption key kept private).*

2 History

2.1 Introduction

Historically, cryptography has been used by armies at war to make secret communications seemingly unreadable, to prevent their enemies from intercepting vital information such as plans of attack. This usually consisted of simply encrypting the messages using a pre-determined key, such as the use of the *Caesar Cipher* (which is detailed later in this essay).

Another early example of cryptography was the *Pigpen Cipher*, which represented letters graphically using a key that was (in theory) known only to those sending the messages and their intended recipients. [1]This key consisted only of grids which were fragmented to represent each letter. It is also known as the *Freemason's Cipher*, due to the movement using it so frequently to encrypt their communications.

As will be explored later in more detail, these forms of cryptography were not particularly secure, as they were easy to guess based on knowledge of the languages being used and the potential contents of messages.

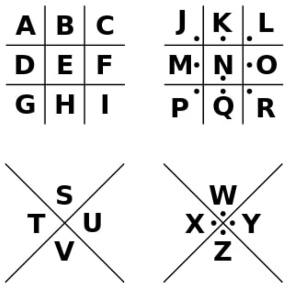


Figure 1: An example of the Pigpen Cipher.

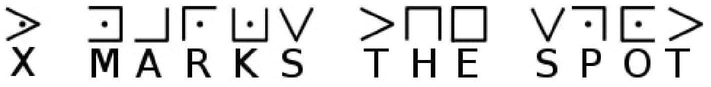


Figure 2: A message encoded using this Pigpen Cipher.

Later forms of cryptography made use of *rotor cipher machines*, such as the *Engima Code*, which made their decryption a lot more difficult. Modern cryptography makes use of discoveries in number theory and computer science, thus making modern cryptosystems more secure than any cryptosystems used prior to these discoveries. This will be studied more in depth later in this essay.

2.2 Caesar Cipher

It is known that Julius Caesar (13/06/100 BC - 15/03/44 BC) [2], the famous Roman politician and general, created a cryptosystem to encode his military messages; this way his opponents would, in theory, be incapable of deciphering his plans should they intercept any messages.

His cryptosystem was based on modular arithmetic, more precisely, it encrypted the alphabet by $+3 \mod 26$, after assigning each letter of the alphabet to a number from 0 to 25. In other words, the plaintext letter A was assigned to the letter D in the ciphertext, B to E, and so on. This is an example of a *polyalphabetic substitution cipher.* [3] [4]

2.2.1 How was this decrypted?

The decryption of this, providing the eavesdropper knew the key, would simply work in the opposite way, by *subtracting* 3 instead of adding; this is based on the logic of subtracting 3 from 0 giving 23, as per arithmetic mod 26.

This may, at first glance, seem to give very little help in the way of decoding a message that uses this cipher. How would one go about finding out the key? This will be explained later in more depth, but is directly linked

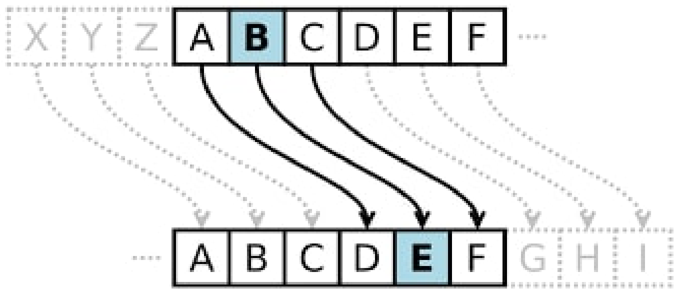


Figure 3: The Caesar Cipher. [5]

to the probability of each letter appearing in sentences in the language of the plaintext; for example, e is widely cited as the most frequently used letter in the English language [6]. Once the eavesdropper has determined the frequency of each of the ciphertext letters, they can make a series of educated guesses about certain associations of plaintext to ciphertext letters- this will then reduce the possibilities of various keys drastically, and help them to decode at least the vast majority of a message.

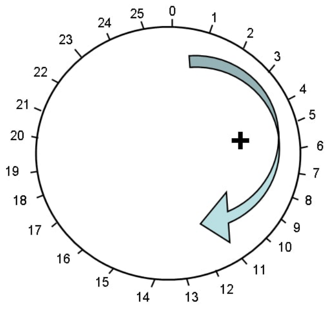


Figure 4: A modular addition circle modulo 26. [7]

2.3 Enigma Code

Invented during World War I and implemented by the Germans during World War II, the Enigma machine is a highly complex example of a polyalphabetic substitution cipher. The machines were a series of rotors which, when operated by a user, would encrypt messages using a constantly changing key. The different parts of the machine were programmed separately according to a set of specifications that differed for every machine; to decrypt a message the eavesdropper or recipient would have to be in possession of a machine set to the same specifications.

The options for adjusting the machines were: [8]

- Wheel order - the choice and order of rotors when building the machine.

- Ring settings - where each alphabet ring was positioned.
- Wiring - plugboards were a part of the machines that connected pairs of letters together, so the wiring of these was changed on every machine.
- Starting position of the rotors - these were different for every message, and chosen by the operator of the machine.

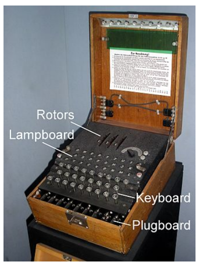
The machines were then operated with easily-destroyable books of keys, which were again specific to each configuration of settings. These keys were changed regularly, to make the number of possible encryptions even higher.

Without knowing the set-up of the machine used to encrypt the plaintext, it was extremely difficult to decrypt any message. This system of machines was the closest that any cryptosystem had ever come to *perfect secrecy* (explained in more depth later).

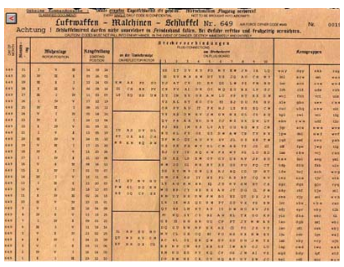
2.3.1 How was this decrypted?

Due to the complex nature of the encryption system, this cryptosystem was incredibly difficult to decrypt. Though the system in itself actually contained a few cryptographic weaknesses, it was actually a range of other factors that contributed to the Allies breaking the code in World War II. These consisted of German procedural and organisational errors, and the capture of Enigma information and hardware by the Allies. [9]

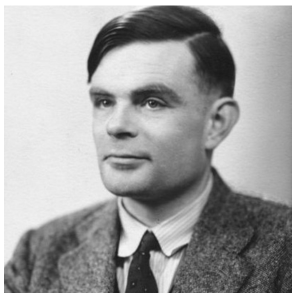
Much of the work on the Enigma code that enabled its decryption was completed by Alan Turing, a Cambridge University mathematician and logician [10]. His efforts are recorded in many books such as 'Alan Turing: The Enigma' by Andrew Hodges, as well as the 2014 American historical drama film 'The Imitation Game', starring Benedict Cumberbatch. [11]



(a) An enigma machine.



(b) An example of a monthly key sheet. [12]



(c) Alan Turing.

2.4 Why not these ciphers?

As previously described these ciphers, and similar developments, are unfortunately insecure. With modern applications for cryptography such as banking, online money transfers, emails, social media, and many more, the demand for perfectly secure encryption is ever increasing. Digital data transfers are increasingly vital to today's society, therefore no chances can be taken on our data being secure from prying eyes.

3 Computational Number Theory

Computational number theory is the combination of elements from number theory, and computation theory. This works in two directions; computing methods from computation theory are applied to solve number-theoretic problems, and vice-versa. [22]

This essay focuses more on using computing techniques to solve number-theoretic problems, in order to create applications in modern public-key cryptography.

This section is primarily comprised of research from *'Computational Number Theory and Modern Cryptography', Song Y. Yan*, including (but not limited to) the theorems and examples presented in the *Primality Testing* and *Integer Factorisation* subsections.

3.0.1 Primality Testing

Primality testing is one of the two main issues that arise within the computation of prime numbers. There are many algorithms for this that run in exponential-time, random polynomial-time, zero-error probabilistic polynomial-time and deterministic polynomial-time, but the details of which are beyond the scope of this essay. The notion of primality testing will be explored at a basic level, as motivation for its applications in RSA cryptography.

Definition 3.1. *The **Primality Test Problem** (PTP) can be represented as follows:*

$$PTP := \begin{cases} \textit{Input: } n \in \mathbb{Z}_{>1} \\ \textit{Output: } \begin{cases} \textit{Yes: } n \in \textit{ Primes} \\ \textit{No: Otherwise} \end{cases} \end{cases}$$

The following theorem is fundamental to primality testing, as it greatly reduces the amount of computational power needed to run the required algorithms.

Theorem 3.1. *Let $n > 1$. If n has no prime factor $p \in \mathbb{Z}$, such that $1 < p \leq \lfloor \sqrt{n} \rfloor$, then n is prime.*

The following proof is a generalisation of a proof found from [13]. $\lfloor\sqrt{n}\rfloor$ from the above is rewritten as simply $\sqrt{n}$, as it does not affect the proof in any way.
Proof:
Let n be composite s.t. $n \geq 0$.
We can write $n = ab$ where $a, b \in \mathbb{Z}$ and $1 < a, b < n$.

WLOG, suppose that $a \leq b$.

Let $a > \sqrt{n}$. Then $b > a > \sqrt{n}$. However, if $b > a > \sqrt{n}$ is true, then:

$n = ab > \sqrt{n}\sqrt{n} > n$,

which is clearly a contradiction.

So $a \leq \sqrt{n}$.

It follows that $\exists$ some prime p s.t. $p \mid a \Rightarrow p \leq \sqrt{n}$ (as $p \leq a$), $\Rightarrow p \mid n$.
■

This eliminates the need to test for all possible prime factors of n up to n, instead allowing the testing of all possible prime factors of n up to $\lfloor\sqrt{n}\rfloor$.

The simplest such test makes use of the *Sieve of Erathosthenes* to find all prime numbers up to $\sqrt{n}$. It runs trial divisions of all possible factors of n up to $\lfloor\sqrt{n}\rfloor$. It is defined as follows:

Definition 3.2. ***Primality test by trial divisions:***
Test(p_i) $:= p_1, p_2, \ldots, p_k \leq \lfloor\sqrt{n}\rfloor, p_i \nmid n$.
Then, if n passes Test(p_i), then n is prime.

This typically runs in exponential time, so is clearly not particularly useful when applying it to aid in decryption of a ciphertext, as extremely large numbers will be used in encryption.

Other, quicker primality testing methods have been developed in modern years, including the *Miller-Rabin test* and the *AKS test*, which build on work by Lucas and Pratt in the area of primitive roots as a basis for primality testing, and the proving of primality (respectively). This can be studied in more depth in *Computational Number Theory and Modern Cryptography, Song Y. Yan* on pages 159 - 190, and in other relevant texts (including those included in the references of this essay), but will be omitted for the remainder of this essay.

3.0.2 Integer Factorisation

The **Integer Factorisation Problem** (IFP) is the basis for many forms of public-key cryptography, due to its current inability to be solved with any polynomial-time algorithm. This makes any form of cryptography which uses it as a structure incredibly secure, as it provides a *trapdoor function*: a one-way function for which the trapdoor (inverse) is nearly-impossible to find without knowing the inputs to the function. [14]

Definition 3.3. *The* ***Integer Factorisation Problem*** *(IFP) can be represented as follows:*

$$\mathit{IFP} := \begin{cases} \textit{Input: } n \in \textit{Composites} \\ \textit{Output: } f \textit{ such that } f \mid n \;\&\; 1 < f < n. \end{cases}$$

This problem could easily be solved with the existence of an algorithm to test the primality of an integer n, another algorithm to find a non-trivial factor f of a composite integer n, as one could then recursively call the primality testing algorithm and the integer factorisation algorithm to find the prime factorisation of $n = p_1^{\alpha_1}\, p_2^{\alpha_2} \cdots p_k^{\alpha_k}$.

The details of this will again be omitted in favour of greater exploring its applications, as the basic algorithms mimic the trial structure of those described previously in the *Primality Testing* section of this essay; more information on integer factorisation algorithms, along with specific examples, can be found on pages 191-233 of *'Computational Number Theory and Modern Cryptography', Song Y. Yan'.*

4 Modern Cryptography

4.1 Perfect Secrecy

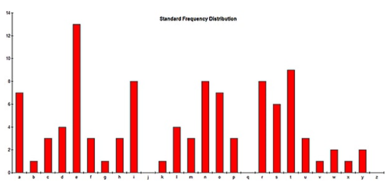
Definition 4.1. *Evaluate a cryptosystem with a random key, K such that $K \in 0, 1, ..., s$ with the set of possible plaintexts $P = 0, 1, ..., t$, and the encryption function $f(clear, key)$.*
For every possible cleartext i, let X_i denote the corresponding ciphertext. That is, $X_i = f(i, K)$.

Perfect secrecy *requires the probability distributions of the random variables $X_0, X_1, X_2, ..., X_t$ to be the same.*

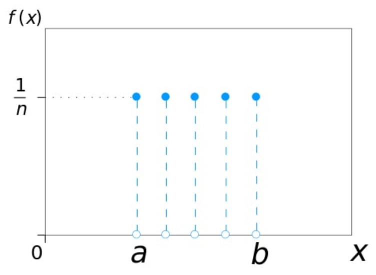
4.1.1 Probability Theory

An issue that arises when determining the security of a cryptosystem is whether the eavesdropper will be able to guess the message based on a series of logical connections. For example, if the eavesdropper knew what kind of message was being encrypted (i.e. war plans, a phone number, a password of a certain length), they could try a series of guesses at the key used (e.g. modulo addition) and apply these to the ciphertext.

They would be left with a selection of potential plaintexts, one of which may be the actual plaintext. Due to the probability of certain letters occurring in different languages not being a *uniform distribution*, these guesses are made a lot easier with only a small amount of knowledge about the plaintext. [15] [16]



(a) A frequency distribution of letters in the English language, based on a sample random text of 100 letters. [15]



(b) A discrete uniform distribution between parameters a and b, with n=5 (i.e. the number of possible outcomes).

The aim with perfect secrecy is to make the chance of any letter, number or symbol appearing in the message completely equal. This would ensure that, should an eavesdropper try to use every possible key and combination to guess the plaintext from the ciphertext, their selection of possible plaintexts following this process would be unintelligible, leaving them no closer to decrypting the message than before. Thus the aim is to create a *uniform distribution* with the probability of any symbol from the sample space showing up.

5 Integer Factorisation Based Cryptography

5.1 Number Theory Basics

Definition 5.1. *An integer b is said to* ***divide*** *an integer c if c/b is a whole number.*

Definition 5.2. *Two integers r and s are* ***relatively prime*** *if $\nexists\ q \in \mathbb{Z}_{>1}$ such that $q \mid r$ and $q \mid s$.*

Definition 5.3. *$n \in \mathbb{Z}_{>1}$ is* ***prime*** *if its only positive divisors are 1 and itself.*

Definition 5.4. ***Euler's phi function***, *$\varphi(x)$, is defined by the rule $\varphi(x) =$ the number of mod-x representatives that are relatively prime to x.*

This gives rise to two equivalent formulas for $\varphi(x)$:

1. *If m is a prime number then $\varphi(m) = m - 1$.*
   (This holds because every number from 0 to $m - 1$ is relatively prime to m apart from 0.

2. *If m is the product of two distinct primes p and q then*
   $\varphi(m) = (p - 1) \cdot (q - 1)$.
   (This is stated without proof as the proof is not pertinent to this essay).

Definition 5.5. *If two numbers b and c have the property that their difference $b-c$ is integrally divisible by a number m (i.e., $(b-c)/m$ is an integer), then b and c are said to be* ***congruent modulo m****. The number m is called the modulus, and the statement "b is congruent to c (modulo m)" is written as $b \equiv c \pmod m$. [20]*

Theorem 5.1. ***The Substitution Principle of Congruences:*** *For any true modular congruence, replacing any subexpression with a congruent subexpression yields another true modular congruence.*

5.2 The RSA Cryptosystem

The RSA public-key cryptosystem was created by Ron Rivest, Adi Shamir, and Leonard Adleman,and published by them in 1977. It is based on the lack of solution for the *factoring problem*, the process in which a composite number is factorised into two (or more) large prime numbers. [23]

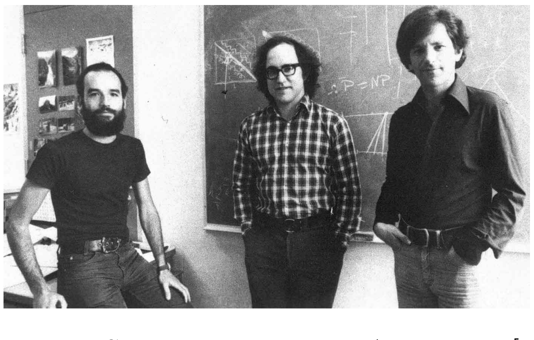


Figure 7: Shamir, Rivest, and Adleman. [17]

This problem creates a perfect trapdoor function to utilise, as without some knowledge of the primes used to create the public key, there is currently no given method in which a solution/decryption can be found, other than trying every possible solution (which is infeasible in polynomial time due to the size of the primes involved).

Breaking RSA encryption is known as the *RSA problem.* It is unknown whether this is as computationally difficult as the factoring problem.

Definition 5.6. *The **RSA cryptosystem** is as follows:*

$$\begin{cases} C \equiv M^e \text{ (mod } n) \\ M \equiv C^d \text{ (mod } n) \end{cases}$$

where

1. *M is the plaintext;*
2. *C is the ciphertext;*
3. *$n = pq$ is the modulus, with p and q large and distinct primes;*
4. *e is the public encryption exponent (key) and d the private decryption exponent (key), with $ed \equiv 1 \pmod{\varphi(n)}$. $\langle n, e \rangle$ should be made public, but d (as well as $\varphi(n)$) should be kept secret.*

This clearly creates a trapdoor function, $f : M \to C$, as with modern technology it is easy to compute in polynomial time, but its inverse $f^{-1} : C \to M$ is difficult to compute without knowledge of the private decryption key d, so would need to factor n and compute $\varphi(n)$ to find d. For those who know d, i.e. the sender and the intended recipient(s) of the message, f^{-1} is as easy to compute as f is.

In practice, suppose that the sender sends a message M to the recipient. This (intended) recipient will have already chosen a one-way trapdoor function f as described above, and published the public key (e, n). Any unintended recipients to the message will now, in theory, know/have access to (e, n).

M would be split into blocks of $\lfloor \log n \rfloor$ bits or less, adding zeros onto the right side of the final block to make it the same size. Each block is treated as an integer $x \in \{0, 1, 2, \ldots, n-1\}$.

The sender computes

$$y \equiv x^e \pmod{n}$$

and sends y to the recipient, who, knowing the private key d, computes

$$x \equiv y^d \pmod{n},$$

where $ed \equiv 1 \pmod{\varphi(n)}$.

Any interceptors who were not meant to read the message would be unable to complete this step to find x without knowing d, as it would essentially involve factoring n.

Example 5.1. *Suppose the plaintext to be encrypted is "PLEASE WAIT FOR ME".*
Let $n = 5515596313 = 71593 \cdot 77041$. *Let also* $e = 1757316971$ *with* $gcd(e, n) = 1$.
Then $d \equiv 1/1757316971 \equiv 2674607171$ *(mod (71593-1)(77041 -1)).*
To encrypt the message, we first translate it into its numerical equivalent by the letter-digit encoding scheme defined by $A \to 01, B \to 02, \ldots, Z \to 26$ *and space* $\to 00$*:*

$$M = 1612050119050023010920000061518001305.$$

Then we split it into 4 blocks, each with 10 digits, padded on the right with zeros for the last block:

$$M = (M_1, M_2, M_3, M_4) = (1612050119, 0500230109, 2000061518, 0013050000).$$

Now, we have
$C_1 \equiv 1612050119^{1757316971} \equiv 763222127$ *mod(5515596313)* ,
$C_2 \equiv 0500230109^{1757316971} \equiv 1991534528$ *mod(5515596313)* ,
$C_3 \equiv 2000061518^{1757316971} \equiv 74882553$ *mod(5515596313)* ,
$C_4 \equiv 0013050000^{1757316971} \equiv 3895624854$ *mod(5515596313).*

That is,
$C = (C_1, C_2, C_3, C_4) = (763222127, 1991534528, 74882553, 3895624854)$.

To decrypt the ciphertext, we perform:
$M_1 \equiv 763222127^{2674607171} \equiv 71612050119$ *mod(5515596313)* ,
$M_2 \equiv 1991534528^{2674607171} \equiv 0500230109$ *mod(5515596313)* ,
$M_3 \equiv 74882553^{2674607171} \equiv 2000061518$ *mod(5515596313)* ,
$M_4 \equiv 3895624854^{2674607171} \equiv 0013050000$ *mod(5515596313).*

By padding the necessary zeros on the left of some blocks, we get
$M = (M_1, M_2, M_3, M_4) = (1612050119, 0500230109, 2000061518, 0013050000)$
which is "PLEASE WAIT FOR ME", the original plaintext message.

5.2.1 How is this decrypted?

If there exist efficient algorithms for the integer factorisation problem (IFP) and the discrete logarithm problem (DLP), then RSA can be completely decrypted in polynomial time (note that the DLP has been omitted from this essay for brevity).

Unfortunately, no such efficient algorithm exists yet, the search for which is the most "most important unsolved problem in computational number theory".
This section will explore some elementary attacks on RSA, based on some elementary number-theoretic properties and weaknesses that exist within RSA (weaknesses that can, however, be avoided if RSA is implemented properly).

Definition 5.7. *A* ***forward search attack****, or* ***guessing plaintext attack****, is an elementary algorithmic attack on RSA that is built from the process of guessing the values of the plaintext M.*

Suppose (e, n, C) is given, and the cryptanalyst wishes to find M.

That is, $\{e, n, C \equiv M^e \ (mod\ n)\ \} \longrightarrow \{M\}$. //
If the plaintext space $\eta = \{M_1, M_2, \ldots, M_k\}$ is small or predictable, the cryptanalyst can decrypt C by simply encrypting all possible plaintext messages $M_1, M_2, \ldots, M_k$ to get $C'_1, C'_2, \ldots, C'_k$ and check, at each step, if $C'_i = C$. If yes, then $M = M_i$, the plaintext M is found.

This is the same method that created weaknesses in the aforementioned *Caesar Cipher* and other historical ciphers, but is rendered impractical when applied to RSA when the message space η is very large.

Definition 5.8. *A* ***short plaintext attack*** *is closely related to the forward search attack. If the plaintext message M is small but the corresponding C can be as big as N (this is the general case for public-key cryptography), then the cryptanalyst can perform two sequences of the operations as follows:*

$$U \equiv Cx^{-e} \ (mod\ n)\ , \forall 1 \leq x \leq 19^9 V \equiv y^e \ (mod\ n)\ , \forall 1 \leq y \leq 19^9.$$

If for some of the pair (x, y), we have $U = V$, then $C \equiv (xy)^e \ (mod\ n)$. Then $M = xy$.

This attack is much more efficient than the previous attack, as it only needs to perform $2x10^9$ calculations to compare the elements in the two sequences up to 10^9 times, as opposed to trying all 10^{17} values of M.

These attacks can be **prevented**, however, with methods such as a *salting process* (i.e. adding some random digits to the end of the plaintext message M before encryption), or a *paddling process* (i.e. adding random digits to the beginning and end of M prior to encryption), so that a large random plaintext M can be formed; these randomly added digits can simply be removed after decryption.

Another way in which RSA is strengthened is through *RSA signatures*, which involves the addition of a digital signature from the sender to the message, so that the recipient can check the signature upon decryption to see whether the message has been tampered with in any way.

5.2.2 Is RSA the best modern cryptosystem?

As there currently do not exist any effective algorithms to solve RSA in polynomial time, RSA remains as the industry standard in cryptography. There are, however, concerns that the factorising problem could be solved in the near future, even within the next 10 years, which makes RSA a potentially unsafe choice to encrypt new technologies with.

New developments in cryptography have included the creation of *Elliptic Curve Cryptography* (ECC), which is based on the utilisation of the algebraic properties of elliptic curves. This is more difficult than RSA to implement, but is much more secure.

In fact, work completed by Lenstra, described under the term *global security*, allows one to compute how much energy is needed to break a cryptographic algorithm and compare that with how much water that energy could boil. By this measure, breaking a 228-bit RSA key requires less energy than it takes to boil a teaspoon of water. Comparatively, breaking a 228-bit elliptic curve key requires enough energy to boil all the water on earth. For this level of security with RSA, one would need a key with 2,380 bits. [18]

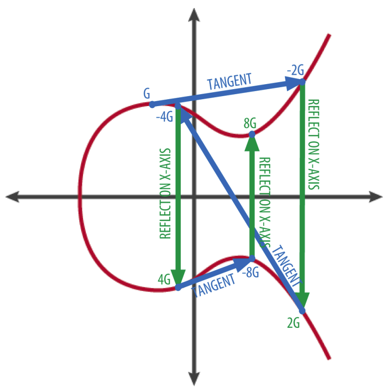


Figure 8: A basic diagram showing the algebraic properties of elliptic curves and how progressive tangents are taken to encrypt a message. [19]

References

[1] https://en.wikipedia.org/wiki/Pigpen_cipher.

[2] https://en.wikipedia.org/wiki/Julius_Caesar.

[3] https://en.wikipedia.org/wiki/Caesar_cipher.

[4] https://en.wikipedia.org/wiki/Polyalphabetic_cipher.

[5] https://www.geeksforgeeks.org/caesar-cipher/.

[6] http://letterfrequency.org/.

[7] http://www.cs.virginia.edu/~evans/dragoncrypto/day1.html.

[8] https://en.wikipedia.org/wiki/Enigma_machine.

[9] https://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma.

[10] https://www.biography.com/people/alan-turing-9512017.

[11] https://en.wikipedia.org/wiki/The_Imitation_Game.

[12] https://plus.maths.org/content/exploring-enigma.

[13] https://proofwiki.org/wiki/Composite_Number_has_Prime_Factor_not_Greater_Than_its_Square_Root.

[14] https://en.wikipedia.org/wiki/Trapdoor_function.

[15] https://crypticcodes.weebly.com/lesson-2.html.

[16] https://en.wikipedia.org/wiki/Discrete_uniform_distribution.

[17] https://en.wikipedia.org/wiki/RSA_(cryptosystem).

[18] https://arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/2/.

[19] https://crypto.stackexchange.com/questions/48657/how-does-ecc-go-from-decimals-to-integers.

[20] Yehuda Lindell Johnathan Katz. *Introduction to Modern Cryptography: Second Edition.* CRC Press, 2015.

[21] Philip N. Klein. *A Cryptography Primer: Secrets and Promises.* Cambridge University Press, 2014.

[22] Song Y. Yan. *Computational Number Theory and Modern Cryptography*. Higher Education Press, 2013.

[23] V. Yaschenko. *Cryptography: an Introduction*. American Mathematical Society, 2002.