MES – Modern Encryption Standard

Awnon Bhowmik
Department of Mathematics and Computer Science
CUNY York College
94 – 20 Guy R. Brewer Blvd
Jamaica, NY 11451, USA

Unnikrishnan Menon
Department of Electrical and Electronics Engineering
Vellore Institute of Technology, Vellore
Tamil Nadu 632014, India

ABSTRACT
As mathematical theory has evolved and computing capabilities have improved, what initially seemed to be adequately difficult trapdoor functions, were deemed not to be later. In this paper, a new block-encryption scheme named Modern Encryption Standard (MES) is proposed based on the multiple concepts arising from number theory for a highly secure and fast cryptosystem that can be considered as an alternative to the existing systems. This is a block cipher like AES, but the inherent algorithm is quite different. The security of the proposed MES algorithm stands on the fundamentals of the Chinese Remainder Theorem, Cantor Pairing Function and the Prime Number Theorem for creating an ingenious trapdoor function. Breaking this algorithm proves to be quite a daunting obstacle to overcome for an unwelcome interceptor.

Keywords
AES, DES, NIST, MES, modern encryption, Modern Encryption Standard, 3DES, Triple DES, Chinese Remainder Theorem, Cantor Pairing Function, Shor's Algorithm, Pollard's Rho Algorithm

1. INTRODUCTION
In July 2017, National Institute of Standards and Technology (NIST) initially proposed retiring Triple Data Encryption Standard (3DES) following a security analysis and practical demonstration of attacks on 3DES in several real-world protocols. In November 2017, NIST restricted usage to 220 64-bit blocks (8 MB of data) using a single key bundle, so it could no longer effectively be used for TLS, IPsec, or large file encryption.

Advanced Encryption Standard (AES) was introduced in 2001 to replace 3DES. Data Encryption Standard (DES), the algorithm is based on, was retired in 2005. NIST is supposedly going to retire 3DES by 2023 (Henry, 2018), leaving only AES as the strongest candidate to be widely used in all of hardware and software security protocols. Owing to the research and rapid developments in the sector of cryptanalysis, modern cryptographers are posed with the constant challenge to develop superior cryptosystems so that it takes an interceptor to spend a certain period in the system which allows the concerned authorities to track them down. This means they must be able to develop enigmatic trapdoor functions in order to achieve this.

2. TRAPDOOR FUNCTION
A trapdoor function is a highly useful concept in modern cryptography. These are functions that are easy to compute in one direction but extremely hard to compute in reverse if certain parameters or critical information for reversal is lacked. The main novelty of this cryptosystem is the use of the fundamental Chinese Remainder Theorem as a trapdoor function. The algorithm starts off with a secret message that needs to be encrypted (called the Plaintext).

3. CHINESE REMAINDER THEOREM
Statement: Let $m_1, m_2, \cdots, m_n$ be n arbitrary integers. If all n_i's are pairwise coprime, and it $a_1, a_2, \cdots, a_n$ are integers such that $0 \leq a_i < m_i$ for every i, there then is one and only one x such that for the system of linear congruence equations

$$x \equiv a_1 \bmod m_1$$
$$x \equiv a_2 \bmod m_2$$
$$\vdots$$
$$x \equiv a_n \bmod m_n$$

Then the solution is given by

$$x \equiv k \operatorname{lcm}(m_1, m_2, \cdots, m_n)$$

This theorem implies we can represent an element of $\mathbb{Z}_{pq}$ by one element of $\mathbb{Z}_p$ and one element of $\mathbb{Z}_q$, and vice versa. In other words, we have a bijection between $\mathbb{Z}_{pq}$ and $\mathbb{Z}_p \times \mathbb{Z}_q$.

3.1 Direct Construction
The computational steps for Chinese Remainder Theorem are as follows

1. Compute $M = \operatorname{lcm}(m_1, m_2, \cdots, m_n)$
2. Let $M_i = \frac{M}{m_i}$ be the product of all moduli but one. Since m_i's are pairwise coprime, M_i and m_i are coprime.
3. Applying *Bezout's Theorem*, there exist integers K_i and k_i such that
$$K_i M_i + k_i m_i = 1$$
4. Then the solution is given by
$$x = \sum_{i=1}^{n} a_i M_i K_i$$

There have been numerous papers (Lynn, n.d.; Chung-Hsien, Jin-Hua, & Cheng-Wen, 2001; Ding, Pei, & Salomaa, 1996; USA Patent No. 8,280,041, 2012; Grossschadl, 2000) where the authors have used Chinese Remainder Theorem to generate a private key or to encrypt a given message. It usually involved finding a solution for a set of two simultaneous congruence relations. This paper involves finding a solution to a set of two or many more such equations and a unique key generation paradigm based on the prime number theorem.

4. CANTOR PAIRING FUNCTION

This is an elegant function proposed by the Russian mathematician George Cantor that takes in two natural numbers and turns it into a single number. This function is a primitive recursive pairing function. (Szudzik, 2006)

$$\pi: \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

And is defined by

$$\pi(x, y) = \frac{1}{2}(x + y)(x + y + 1) + y \qquad (2)$$

Due to the way its defined, this is a one-to-one and onto function, which means it is invertible. This consequently means that given a single number, it can be readily mapped back to a unique (x, y) ordered pair.

In order to retrieve an ordered pair (x, y) from a given t, the following transformations are used

$$\omega = x + y$$

$$t = \frac{1}{2}\omega(\omega + 1)$$

$$z = t + y$$

From the second equation, cross multiplying gives a quadratic in ω

$$\omega^2 + \omega - 2t = 0$$

Solving it gives us

$$\omega = \frac{\sqrt{8t + 1} - 1}{2}$$

which is a strictly increasing and continuous function when t is non-negative real. Since

$$t \leq z = t + y < t + (\omega + 1) = \frac{(\omega + 1)^2 + (\omega + 1)}{2}$$

This implies that

$$\omega \leq \frac{\sqrt{8z + 1} - 1}{2} < \omega + 1$$

And thus

$$\omega = \left\lfloor \frac{\sqrt{8z + 1} - 1}{2} \right\rfloor$$

Finally calculate x and y from z as follows

$$\omega = \left\lfloor \frac{\sqrt{8z + 1} - 1}{2} \right\rfloor$$

$$t = \frac{\omega^2 + \omega}{2}$$

$$y = z - t$$

$$x = \omega - y$$

5. PRIME NUMBER THEOREM

Positive integers that are divisible by 1 and itself, are known as *prime numbers*. The sequence begins like the following...

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, \cdots$$

has held untold fascination for mathematicians, both professionals and amateurs alike. A result that gives an idea about an asymptotic distribution of primes is known as the *prime number theorem* (Goldstein, 1973).

Let $\pi(x)$ be the prime-counting function that gives the number of primes less than or equal to x, for any real number x. For example, $\pi(10) = 4$ because there are four prime numbers $(2, 3, 5 \text{ and } 7)$ less than or equal to 10. The prime number theorem then states that $\frac{x}{\ln x}$ is a good approximation to $\pi(x)$, in the sense that the limit of the quotient of the two functions $\pi(x)$ and $\frac{x}{\ln x}$ as x increases without bound is 1.

$$\lim_{x \to \infty} \pi(x) \frac{\ln x}{x} = 1$$

known as the asymptotic law of distribution of prime numbers. Using asymptotic notation this result can be restated as

$$\pi(x) \sim \frac{x}{\ln x}$$

The *logarithmic integral* provides a good estimate to the prime counting function.

$$\pi(x) \sim \mathrm{li}(x)$$

In order to get an idea of the distribution of primes, it is important to count the number of primes in a given range and find the percentage of primes. The following table demonstrates this.

Table 1: Prime density and approximation to logarithmic integral

Search Size x	# of primes	Density (%)	$\mathrm{li}(x)$	$\frac{\mathrm{li}(x) - \pi(x)}{\pi(x)} \times 100$
10	4	40	6.16	54.14
10^2	25	25	30.13	20.50
10^3	168	16.8	177.61	5.72
10^4	1229	12.3	1246.14	1.39
10^5	9592	9.6	9629.81	0.39
10^6	78498	7.8	78625	0.17
10^7	664579	6.6	664918	0.05
10^8	5761455	5.8	5.76×10^6	0.01

Following is a visual demonstration

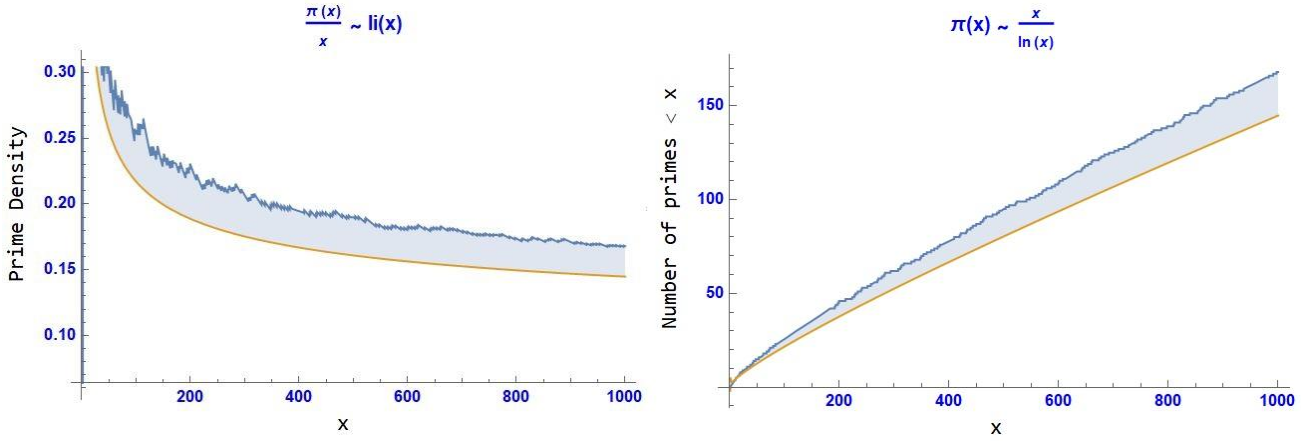

Fig. 1: Demonstration of the Prime Number Theorem

6. PROPOSED ALGORITHM

The following is the algorithm underlying this cryptosystem.

1. **Encryption**

a. An input string (plain text) is taken and split into blocks, padding is applied. Padding is used to maintain consistency of the modified plain text. User provides the input string and block size.

b. All characters in the blocks are converted to their corresponding ASCII values.

c. In every block, every i^{th} value is combined with the $(i+1)^{\text{th}}$ (adjacent) value using the *Cantor pairing function*, which takes two numbers as input and spits back a single, unique number. Now, each block has $\frac{\text{block size}}{2}$ number of elements. Suppose the list is

$$A = \{a_1, a_2, \cdots, a_{\text{blocksize}/2}\}$$

2. **Key Generation**

a. *The Chinese Remainder Theorem* requires that

$$0 \leq A_i < M_i$$

Where $|A| = |M| = \frac{\text{block size}}{2}$. This key M must be generated. In order to ensure that this condition is met, the worst-case scenario is considered where two characters with maximum ASCII value of 128 appears adjacent to each other. For these two numbers, the Cantor pairing function returns

$$C(128{,}128) = 33024$$

All values of M must be larger than 33024, and an upper bound U is also required so a finite loop can be run over this range $[\pi(128{,}128), N]$.

$$C(128{,}128) \leq M_i \leq U$$

$$\begin{cases} M = \{p_1, p_2, \cdots, p_n\}, \forall p_i \in \text{Prime} \\ |M| = N = \dfrac{\text{block size}}{2} \end{cases}$$

b. This list of primes M is shuffled and the solution to the following system of linear congruence equations generates the cipher text.

$$x \equiv A_i \bmod M_i$$

3. **Decryption**

a. Cipher text x and list of primes M must be known for successful decryption. The solution to the following system of congruence equation is required

$$A_i \equiv x \bmod M_i$$

b. Each A_i is passed through the reverse cantor pair function to get back two adjacent a_i's that initiated the encryption procedure.

These a_i's are converted to ASCII characters and padding sequences are removed.

7. DETERMINING THE UPPER BOUND

The prime number theorem gives us an estimate of the number of primes within a search size x. This asymptotic distribution is defined as

$$\pi(x) \sim \frac{x}{\ln x}$$

1) # of primes less than $n \Longrightarrow \pi(n) \sim \frac{n}{\ln n}$

2) # of primes in (a, b)

$$\pi(b) - \pi(a) \sim \frac{b}{\ln b} - \frac{a}{\ln a}$$

3) # of primes between $C(128{,}128)$ and an upper bound U

$$\pi(U) - \pi\big(C(128{,}128)\big) \sim \frac{U}{\ln U} - \frac{C(128{,}128)}{\ln C(128{,}128)}$$

4) It was stated in the proposed algorithm that

$$N = \frac{\text{block size}}{2}$$

This means

$$N = \frac{U}{\ln U} - \underbrace{\frac{C(128{,}128)}{\ln C(128{,}128)}}_{K}$$

$$N + K = \frac{U}{\ln U}$$

$$\ln U = \frac{U}{Q} \qquad \text{where } Q = N + K$$

$$U = e^{\frac{U}{Q}}$$

$$Ue^{-\frac{U}{Q}} = 1$$

$$-\frac{U}{Q}e^{-\frac{U}{Q}} = -\frac{1}{Q}$$

$$-\frac{U}{Q} = W_{-1}\left(-\frac{1}{Q}\right)$$

$$U = -QW_{-1}\left(-\frac{1}{Q}\right)$$

$$U = -\left[N + \frac{C(128,128)}{\ln C(128,128)}\right] W_{-1}\left[-\frac{1}{N + \frac{C(128,128)}{\ln C(128,128)}}\right]$$

Where W_n is the Lambert W Function means that it is possible to generate the upper limit for generating a list of primes for an arbitrary block size. Since the number of primes is positive, U must be positi (Weisstein, 2002; Corless, Gonnet, Hare, Jeffrey, & Knuth, 1996) also known as the product log function. This ve so consequently $W_n\left(-\frac{1}{Q}\right) < 0$. It means that the branch $n = -1$.

If $U = a + ib$, that is if $U \in \mathbb{C}$ then, considering $Re(U)$ generates the upper limit. Let x_1, x_2 be the possible solutions after applying the Lambert W Function. Since $U > C(128,128)$, it can be safely assumed that $U = \max(x_1, x_2)$.

Here is a table showing the initial number of values that must be checked to generate a list of primes. Then, five primes are chosen at random, from this *shuffled* list.

Table 2: Block size and upper bound

Block Size (bits)	x_1	x_2	$U = \max(x_1, x_2)$	$U - C(128,128)$
8	1.0003	33116	33116	730
16	1.0003	33208	33208	184
32	1.0003	33393	33393	369
64	1.0003	33761	33761	737
128	1.0003	34500	34500	1476
256	1.0003	35982	35982	2958
512	1.0003	38961	38961	5937
1024	1.0002	44975	44975	11951
2048	1.0002	57202	57202	24178

Following is a plot showing the connection between the block size and the length of the range of numbers in which to look for primes. It is observed to demonstrate a linear relationship.

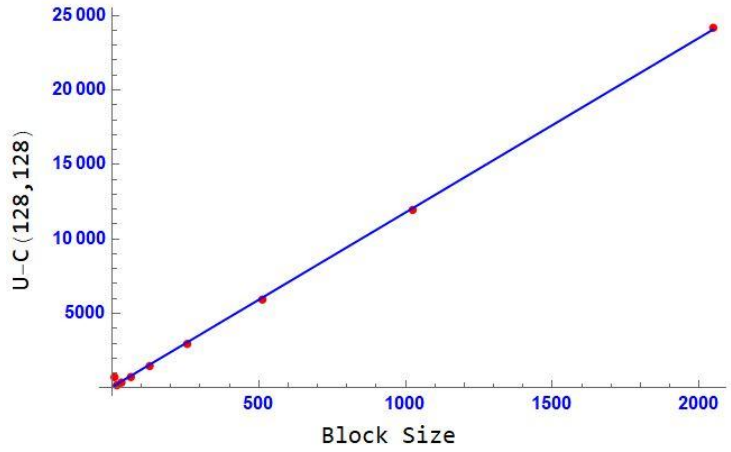

Fig. 2: Block size vs prime search range

8. GENERATING THE LIST OF PRIMES

For the convenience of runtime complexity, an infinite loop was run from $C(128,128)$. The SymPy module in Python3 was used to generate primes adjacent to each other. The loop was terminated after the generation of $\frac{\text{block size}}{2}$ prime numbers.

This method, however, poses a serious problem. If an interceptor gets to know of this design, he/she would easily break the system using the classic *Shor's Algorithm*.

On the other hand, if a set \$S\$ of randomly shuffled primes having sufficient gaps between each pair with cardinality of $\frac{\text{block size}}{2}$ is generated between $[C(128,128), U]$ where U is the

upper bound calculated in section 7, then it would be quite tough for an interceptor to regenerate.

9. FROM AN INTERCEPTOR'S PERSPECTIVE

9.2 Shor's Algorithm

Pollard's rho algorithm, invented by John Pollard in 1975 (Pollard, 2008) is used for integer factorization. This algorithm is known to work very fast for numbers with small factors but gets slower in cases where large factors are involved. The ρ algorithm's most remarkable success was the factorization of the Fermat number F_8 (Pollard, 2008). If the pseudorandom number $x = g(x)$ occurring in the Pollard ρ algorithm were an actual random number, it would follow that success would be achieved half the time, by the Birthday paradox in $O(\sqrt{p}) \leq O\left(n^{\frac{1}{4}}\right)$ iterations. It is believed that the same analysis applies as well to the actual rho algorithm, but this is a heuristic claim, and rigorous analysis of the algorithm remains open (Galbraith, 2012). Since the primes chosen in the MES are sufficiently large it stops any unwelcome interceptor by raising the toughness of integer factorization while maintaining reasonable run time.

10. EXPERIMENTAL ANALYSIS

10.1 String length vs Time for encrypt-decrypt cycle

For this test, block size of 128,192 and 256 were considered and the length of string(with repetition) was gradually increased in steps of powers of 10 while noting down the time it takes for successful encrypt-decrypt cycles

Table 3: Effect of string length on run time for different block size

String Length	Block Size		
	128	**192**	**256**
10	0.001	0.002	0.003
10^2	0.002	0.002	0.002
10^3	0.011	0.012	0.011
10^4	0.13	0.11	0.115
10^5	0.943	1.053	1.062
10^6	9.807	10.535	11.106
10^7	99.458	106.824	111.409

It was observed that if the length of string is n and a block size $\gg n$ is used, then the time increases slightly owing to the excessive padding operation requirements

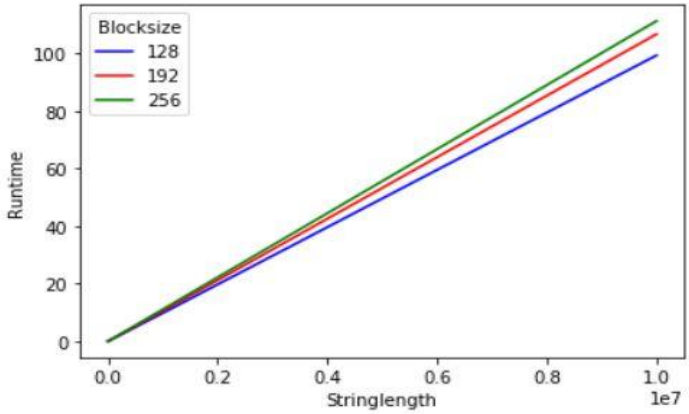

Fig. 3: Block size vs run time

In case $n \gg$ block size, choosing a very small value for the block size vs choosing a comparatively larger block size leads to very similar time of execution but improved security in case of larger block size, thus making it all the more difficult for an interceptor to crack the message while ensuring reasonable run time complexity on the authorized pipeline

10.2 Block size vs Time for encrypt-decrypt cycle

A function was written on Jupyter Notebook to test out how varying the block size has an impact on the time it takes to complete one successful encryption-decryption cycle. Note that for this test, a random alphanumeric string of 26000 characters were considered.

Table 4: Effect of block size on run time

Block Size (bytes)	Run time (sec)
2	0.405
4	0.248
8	0.229
16	0.239
32	0.286
64	0.295
128	0.264
256	0.292

Note that since every time the MES algorithm is executed, a unique secret key is generated depending on the block size. The bar and scatter plot in the first two images in Fig. 3 show a linear relationship with negligible slope when optimized. An *ideal* system should resemble a horizontal line. This entropy in the real time is caused by multiple background processes that maybe running during the execution of this program.

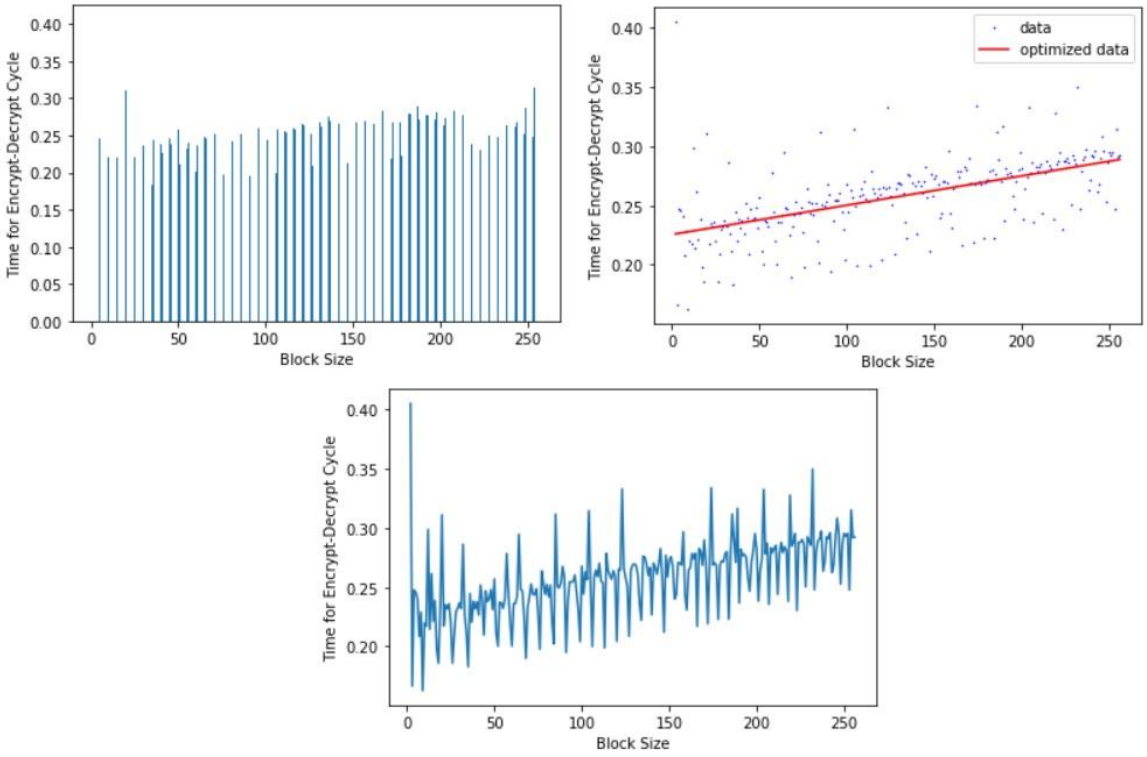

Fig. 4: Block size vs run time

11. CONCLUSION

For benchmarking string length vs run time, 128,192 and 256 were taken as the block sizes in the experiment since AES also uses these bitlength keys. However, it is crucial to note that AES and MES have a very different notion of block size and bit size. Unlike AES, the block size is not restricted to just 128,192,256 in MES. Instead it can be any even positive integer chosen by the user. The idea of this paper was to introduce a new algorithm for future use. Since AES and DES involves bit wise calculations and MES simply works with ASCII values, there cannot be an effective comparison. However, if MES can somehow be converted into bits (Ariyama & Toyoshima, 1995; Chen & Yao, 2011) and calculations involving byte substitution, bit shifting etc., are introduced, the run time should be much faster than AES and DES.

Use of ASCII values upfront is not the best approach for MES algorithm because it brings in computationally intensive task on the operations involved in the trapdoor function. However, if in future, the original message is converted into bits before applying MES then there would most likely be an exponential reduction in the complexity which can ultimately lead up to a great boost in performance of the algorithm to compete with industrially prevalent algorithms.

12. REFERENCES

[1]. Ariyama, K., & Toyoshima, H. (1995). Hardware implementation of Chinese remainder theorem using redundant binary representation. (pp. 552-561). Sakai,

Japan: IEEE. doi:https://doi.org/10.1109/VLSISP.1995.527526

[2]. Chen, J., & Yao, R. (2011). Efficient CRT-based residue-to-binary converter for the arbitrary moduli set. *Science China Information Sciences, 54*(1), 70-78. doi:https://doi.org/10.1007/s11432-010-4133-3

[3]. Chung-Hsien, W., Jin-Hua, H., & Cheng-Wen, W. (2001). RSA cryptosystem design based on the Chinese remainder theorem. *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*, (pp. 391--395).

[4]. Corless, R. M., Gonnet, G. H., Hare, D. E., Jeffrey, D. J., & Knuth, D. E. (1996). On the LambertW function. *Advances in Computational mathematics, 5*(1), 329--359. doi:https://doi.org/10.1007/BF02124750

[5]. Ding, C., Pei, D., & Salomaa, A. (1996). *Chinese remainder theorem: Applications in Computing, Coding, Cryptography*. World Scientific.

[6]. Douguet, M., & McKeeney, N. M. (2012, October). *USA Patent No. 8,280,041*.

[7]. Galbraith, S. D. (2012). Towards a rigorous analysis of Pollard rho. In S. D. Galbraith, *Mathematics of Public Key Cryptography* (pp. 272-273). Cambridge University Press.

[8]. Goldstein, L. J. (1973). A history of the prime number theorem. *The American Mathematical Monthly, 80*(6), 599--615.

[9]. Grossschadl, J. (2000). The Chinese remainder theorem and its application in a high-speed RSA crypto chip. *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)* (pp. 384--393). New Orleans, LA: IEEE. doi:https://doi.org/10.1109/ACSAC.2000.898851

[10]. Henry, J. (2018, August 3). 3DES is Officially Being Retired. Retrieved from https://www.cryptomathic.com/news-events/blog/3des-is-officially-being-retired

[11]. Lynn, B. (n.d.). *The Chinese Remainder Theorem*. Retrieved from Applied Crypto Group - Stanford: https://crypto.stanford.edu/pbc/notes/numbertheory/crt.html

[12]. Miszczak, J. A. (2015, October). Shor's factoring algorithm. *Quantiki*. Retrieved from https://www.quantiki.org/wiki/shors-factoring-algorithm

[13]. Pollard, J. M. (2008, October 24). Theorems on factorization and primality testing. *Mathematical Proceedings of the Cambridge Philosophical Society, 76*(3), 521-528. doi:https://doi.org/10.1017/S0305004100049252

[14]. Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, (pp. 124-134).

[15]. Szudzik, M. (2006). An Elegant Pairing Function. Washington, DC. Retrieved from http://szudzik.com/ElegantPairing.pdf

[16]. Weisstein, E. W. (2002). Lambert W-function.