IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/44AI/2024/4445824/
Vol. 2024 (22), Article ID 4445824

Strong Cryptography - Is It Easy to Generate And Break A Cryptographic Key? *

Marcin SOBOTA

Silesian University of Technology, Gliwice, Poland

Correspondence should be addressed to: Marcin SOBOTA, marcin.sobota@polsl.pl

* Presented at the 44th IBIMA International Conference, 27-28 November 2024 Granada, Spain

Abstract

The motivation for this study is to deepen the understanding of the strength and significance of encryption keys used in asymmetric cryptographic systems, particularly by comparing them with physical and biological magnitudes. Existing literature primarily focuses on the technical aspects of cryptography, such as algorithms and system implementations, but lacks a comprehensive comparison of the scale of cryptographic keys with other scientific measures, leaving a gap in current research.
This paper adopts an interdisciplinary approach, employing mathematical analysis, physical data, and biotechnological models to evaluate the complexity of key generation, primality testing, and the thermodynamic constraints of breaking cryptographic keys. It also analyzes the potential threats posed by emerging computational technologies, such as quantum and DNA computing, and their implications for the security of modern cryptographic systems.
The findings demonstrate that the length and generation processes of cryptographic keys provide a very high level of security, and comparisons with physical and biological magnitudes reveal their immense scale. The study concludes that future cryptography must incorporate new computational paradigms to address the challenges posed by technological advancements.

Keywords: cryptography, asymmetric cryptography, thermodynamics, prime numbers, biotechnology, primality testing, quantum computers,

Introduction

Cryptography [1, 2, 3, 4] is a science that has allowed humans to protect information from prying eyes for thousands of years. However, it has never had as much influence on daily life as it does today. Just a few decades ago, cryptography was used mainly for military purposes and remained a secretive and inaccessible field for ordinary citizens. Times have changed, though. Today, without the advancements in cryptography, daily life as we know it would not be possible. Practically at every turn, whether we realize it or not, we rely on cryptographic systems, both symmetric and asymmetric. Online banking, mobile telephony, ATM cards, digital signatures, and electronic IDs would not function without cryptographic methods. Yet, when using these methods and selecting keys of appropriate lengths, do users stop to think about the magnitudes they are dealing with? Likely not. For most, these values are simple numbers. But when we compare these numbers with the physical magnitudes of the world around us, we start to realize just how powerful these numbers are.

This article approaches cryptography and its keys from a different perspective, aiming to help users understand the powerful tool they are using and the immense size of the keys they operate with.

Generating cryptographic keys

Cryptographic keys are one of the most important elements used in cryptographic systems. These systems are designed in such a way that the algorithms they rely on are open and well-known (Kerckhoffs` principle). The security of the system depends on the chosen algorithm, but the algorithm itself is not secret. What is secret is the

Cite this Article as: Marcin SOBOTA, Vol. 2024 (22) “Strong Cryptography - Is It Easy to Generate And Break A Cryptographic Key? " Communications of International Proceedings, Vol. 2024 (22), Article ID 4445824, https://doi.org/10.5171/2024.4445824

key (or one of the key pairs) used. The length and quality of the key ((ignoring factors like storage method or usage time) determine the system`s overall security. Therefore, how the key is generated is of utmost importance.

Prime numbers

One of the most popular asymmetric algorithms, RSA, relies on prime numbers [5]. Given its widespread use, one might question whether there's a risk of eventually exhausting prime numbers. So, let's consider: how many prime numbers are there?

The number of prime numbers in the interval [1, n] is given by the function (1), as empirically suggested by Gauss:

$$\pi(n) \sim Li(n) \tag{1}$$

where Li(n) represents the logarithmic integral, and "∼" indicates asymptotic equality, understood as (2):

$$\lim_{n\to\infty} \frac{\pi(n)}{Li(n)} \tag{2}$$

If the logarithmic integral is expanded as a series, we obtain an approximation (3):

$$\pi(n) \approx \frac{n}{\ln n} + \frac{n}{ln^2 n} + \frac{2n}{ln^3 n} + \cdots = \sum_{i=1}^{\infty} \frac{(i-1)!n}{ln^i n} \tag{3}$$

Although Gauss proposed this hypothesis, it was not proven until the late 19th century by Hadamard and de la Vallée Poussin.
The simplest approximation of the function π is the first term of this series (4):

$$\pi(n) \approx \frac{n}{\ln n} \tag{4}$$

In this case, we also have asymptotic equality (5):

$$\lim_{n\to\infty} \frac{\pi(n)}{\frac{n}{\ln n}} \tag{5}$$

Using these equations, we can easily calculate that there are approximately 10^{153} prime numbers with 512 bits or fewer. Such a large number is difficult to grasp, so let's try to compare it to something more familiar. It is estimated that the entire Universe consists of about 10^{77} atoms [6]. If each atom were to receive a trillion new prime numbers every nanosecond, the atoms would have "used up" only about 10^{116} prime numbers from the Big Bang until today. Thus, approximately 10^{153} primes would still remain, rendering the 'consumption' of primes practically negligible. Consequently, the probability of two users randomly selecting the same prime number for their keys is statistically negligible.

An additional thought experiment involves imagining the storage of all 512-bit prime numbers on a medium. Calculations suggest that if one terabyte of data could be stored on a disk weighing one gram, recording all 512-bit prime numbers would require a medium whose mass would exceed the Chandrasekhar limit—the critical mass of a star, beyond which it collapses into a black hole. This would mean the storage medium itself would collapse under its own weight, making any data retrieval impossible.

Primality Testing

When discussing the use of prime numbers in cryptography, it is impossible not to mention primality testing [7, 8, 9]. Cryptography relies heavily on large prime numbers, which are used in asymmetric encryption methods. It is important to remember that the prime numbers used are generated randomly. Specifically, numbers are generated randomly, and then it is checked whether the generated number is prime. The only parameter that needs to be provided is related to the length of the number we want to generate (the length of the number is related to the length of the key to be used).

Notably, verifying a number through deterministic methods is not feasible time-wise. If deterministic methods were used, generating such numbers would be impossible (the problem becomes a factorization problem), because the time required to perform factorization grows subexponentially with the size of the number. In fact, a number

would be considered prime if no factors were found. Therefore, probabilistic primality tests are used, which provide a quick answer to whether a number is prime (e.g., the Miller-Rabin test). Furthermore, each subsequent run of the test significantly reduces the probability of error. For example, the probability that the Miller-Rabin test incorrectly classifies a composite number as prime is ½. If the test is repeated 100 times, the probability of error drops to $(1/2)^{100}$. In such cases, one can be almost certain that the generated number is prime.

It should be remembered that, in addition to the Rabin-Miller test, there are other probabilistic primality tests, such as the Fermat test or the Solovay-Strassen test. However, due to factors such as computational complexity, efficiency, probability of error, and ease of implementation, the Rabin-Miller test remains the most widely used test.

Cracking cryptographic keys

Cryptanalysis is a branch of mathematics focused on studying and breaking cryptographic algorithms. Its purpose is to find weaknesses in encryption systems to access hidden information without having the key. Some techniques aim to derive the secret key using widely known information—for example, factoring a composite number into prime factors in RSA. These studies demonstrate the complex mathematical and technical challenges in breaking modern cryptographic systems.

Thermodynamics

Thermodynamics [10] is a science encountered daily, which also significantly affects cryptographic key security. Why? Because performing computations on a computer requires energy. Computing power is one limitation, while the energy needed to perform these calculations is another. Below, we will examine the barriers that thermodynamics imposes on the process of breaking large cryptographic keys.

Changing a single bit in a system requires an energy expenditure that can be described by the formula (6):

$$W = kT \tag{6}$$

where:

k – Boltzmann constant, equal to $1.38 * 10^{-23} \frac{J}{K}$

T – absolute temperature of the system.

Assuming we are dealing with an ideal (lossless) computer operating in an environment with a temperature equal to the average temperature of the universe 3.2 K, changing one bit of information would require an energy expenditure of (7):

$$W = 4.4 * 10^{-23} J \tag{7}$$

The annual energy output of the Sun is approximately $1.2 * 10^{34} J$, meaning that using all of the Sun's energy output over the course of a year for computations would allow for about $2.7 * 10^{56}$ bit changes, or roughly 2^{187} calculations. This implies that examining a 192-bit key space would require the energy output of the Sun over a span of 32 years.

By comparing this energy expenditure to the demands of an average household, we find it could power not only all households on Earth but also around a quadrillion Earth-like planets. These are theoretical calculations; however, it is worth noting that, back in 1999, an international team of scientists successfully broke a 512-bit RSA key. The winning team, associated with the National Institute for Mathematics and Computer Science in Amsterdam, broke the key using 292 computers operating across eleven different locations, including the Netherlands, Canada, the UK, France, Australia, and the United States. This setup included 160 Silicon Graphics computers and Sun workstations with processors running at 175-400 MHz, eight SGI Origin 2000 machines at 250 MHz, 120 Pentium II PCs at 300-450 MHz, and four Digital/Compaq machines clocked at 500 MHz.

Discrepancies between these thermodynamic considerations and practical key-breaking efforts arise because brute-forcing the entire key space is not the only way to break a key. Various factorization methods (such as elliptic curve factorization, quadratic sieving, or number field sieving) can significantly accelerate the process. Moreover, these theoretical results assume a worst-case scenario where the key is found only after the entire key space has been searched. In reality, with a bit of "luck," a key could be found after searching only a small portion of the key space.

Biotechnology

Work is still ongoing to use biological systems for computation. One of the most popular ideas is to use a DNA chain as a data carrier that undergoes controlled modifications. A DNA computer is a collection of specially selected DNA strands, the combination of which is designed to solve a given problem. These computers' main advantage is their high parallelism, which should significantly accelerate problem-solving requiring extensive calculations. As early as 1973, Charles Bennett proposed a model of a programmable molecular computer. In 1994, Leonard Adleman demonstrated the possibility of using molecular particles to solve mathematical problems. Using DNA molecules, he solved a seven-vertex Hamiltonian path problem, which is considered an NP-complete problem. Adleman solved this problem by generating all possible DNA strands.

In practice, the first DNA computers were created in 2001. By 2003, an improved version of this type of computer achieved a molecular reaction speed of 330 TFLOPS in a 5-milliliter volume. The idea of using DNA as an information carrier also seems promising. The storage capacity of biological memory is much larger than that of today's storage media. Due to their small size, 1 mm^3 of DNA can store 10,000 TB of information (assuming that one pair of nucleotides represents one bit of information). One gram of DNA contains 10^{21} DNA bases, which allows the storage of 10^8 TB of data. This means that a few grams of DNA could store all the data available on Earth. Given DNA computers' expected computational power, it is natural to consider their potential in cryptanalysis. One of the co-creators of the RSA algorithm, Leonard Adleman, showed that a DNA computer the size of several test tubes could find a 256-bit key (using brute-force attack).

In 2023, a DNA-based integrated circuit was developed, with gates capable of forming up to 100 billion distinct circuits. This innovation opens vast possibilities for conducting various types of computations. The circuit's programmability and scalability allow for the application of multiple algorithms and enable handling an increasing number of tasks by adding resources [11].

However, it turns out that the computational power of such computers for cryptanalysis is not as powerful as one might expect. Beyond the significant challenges of biochemical algorithm implementation, the accuracy of calculations with DNA molecules remains far from ideal, estimates of computational capabilities for large keys rule out these methods as feasible. Beaver [12, 13], following Adleman's approach [14], estimated that a computer needed to factor a 1000-bit number would require a capacity of 10^{20} liters, which seems impressive when compared to the Earth's water resources, estimated at $2x10^{21}$ liters (1,385 mln km^3).

In addition to the aforementioned applications, biotechnology is used in:
- steganography,
- creation of molecular checksums,
- as an equivalent of a hash used to mark objects,
- personal identification systems.

Quantum computers

Quantum computers [15, 16, 17, 18] can be categorized based on the technology used to create the qubits. Some major types include:

1. Superconducting qubits - used by companies like Google and IBM.
2. Trapped ion quantum computers - developed by IonQ and Honeywell.
3. Photon-based quantum computers - as seen in Xanadu's work.
4. Quantum annealing – used by companies like D-Wave Systems, Volksvagen or Lockheed Martin.
5. Solid-state quantum computers - pursued by companies like Microsoft and Quantum Motion Technologies.

Each type offers unique strengths and challenges for quantum computing development.

Features of Quantum Computers:

1. Number of Qubits: This is the primary measure of a quantum computer's potential power. More qubits usually mean greater computational capability. For example, IBM Quantum System One operates with 127 qubits, while Google's Sycamore device uses 54 qubits.
2. Circuit Depth: Refers to the number of quantum operations (gates) performed on qubits during calculations. A higher depth allows for more complex computations.

3. Gate Fidelity: Measures the accuracy of quantum operations. Higher fidelity means fewer errors in computations.
4. Coherence Time: The time during which qubits maintain their quantum state. Longer coherence time allows for more complex calculations.

Of course, the most well-known algorithm that could be used on quantum computers is Shor's algorithm [19]. Shor's algorithm has a time complexity of $O((logN)^3)$, meaning the number of operations grows cubically with the bit length N. For an N of 2048 bits, approximately equivalent to 10^{617}:

1. First, we calculate the logarithm: $\log(10^{617}) = 617$.
2. Cubing this value gives 617^3=235263113.

Thus, Shor's algorithm could factorize such a number in a time proportional to around 235 million operations.

For an N of 4096 bits, approximately equivalent to 10^{1234}, and further $\log(10^{1234}) = 1234$ and cubing these values gives 1234^3=1879366904. It means that for a number of 4096 bits, Shor`s algorithm performs approximately 1.88 billion operations.

It is remarkably fast, especially compared to the most efficient classical factorization algorithm (General Number Field Sieve), whose asymptotic complexity is $O(\exp\left((\frac{64}{9} * \log N * loglog\ N)^{\frac{1}{3}}\right))$. For a 2048-bit number, this algorithm would need to perform approximately 10^{35} operations, while for a 4096-bit number, it would require around 10^{70} operations.

Currently, fastest classical supercomputer called Frontier [20] can perform approximately 10^{18} operations per second which means approximately $3.15*10^{25}$ operations in one year. It means that breaking a 2048-bit key could take billions of years, while breaking a 4096-bit key is difficult to express in terms understandable by human time perception.

Summary and conclusions

The comparisons presented in the article demonstrate the enormous numbers used in modern asymmetric cryptographic systems. Users, when faced with the choice of key length, are often unaware of the vast differences between them. Considering that a key length of 1024 bits (or a 2048-bit key, which guarantees a very high level of security for home users) provides sufficient security for private use, there is no need for home users to opt for keys of 4096 or longer. They are often unaware of the significant gap, from a cryptanalysis perspective, between keys whose lengths are merely doubled. Only the comparisons made in the article to physical quantities can provide a sense of the size of the numbers used and the limitations encountered when attempting to break large encryption keys. It seems that the cryptanalysis of asymmetric cryptographic systems will require the use of entirely new methods, such as quantum algorithms, to achieve success.

References

- Kutyłowski, M., & Strothmann, W. B. (1999). Kryptografia: teoria i praktyka zabezpieczania systemów komputerowych. Read Me.
- Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (2018). Handbook of applied cryptography. CRC press.
- Wobst, R. (2002). Kryptologia. Budowa i łamanie zabezpieczeń, Wyd. RM, Warszawa.
- Stokłosa, J., Bilski, T., & Pankowski, T. (2001). Bezpieczeństwo danych w systemach informatycznych. Wydawnictwo Naukowe PWN.
- Maurer, U. M. (1995). Fast generation of prime numbers and secure public-key cryptographic parameters. *Journal of Cryptology*, *8*, 123-155.
- Schneier, B. (2002). Kryptografia dla praktyków: protokoły, algorytmy i programy źródłowe w języku C. Wydawnictwa Naukowo-Techniczne.
- Rosenberg, B. (1993). The Solovay-Strassen Primality Test. *University of Miami. Department of Computer Science. Miami*.
- Ishmukhametov, S., & Mubarakov, B. (2013). On practical aspects of the Miller-Rabin primality test. *Lobachevskii Journal of Mathematics*, *34*, 304-312.

- McGregor-Dorsey, Z. S. (1999). Methods of primality testing. *MIT Undergraduate Journal of Mathematics*, *1*, 133-141.
- Whitman, A. M. (2020). *Thermodynamics: basic principles and engineering applications* (pp. 1-320). Springer.
- Lv, H., Xie, N., Li, M., Dong, M., Sun, C., Zhang, Q., ... & Fan, C. (2023). DNA-based programmable gate arrays for general-purpose DNA computing. Nature, 622(7982), 292-300.
- Beaver, D. (1994, November). Factoring: the DNA solution. In International Conference on the Theory and Application of Cryptology (pp. 419-423). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Beaver, D. (1995). Computing with DNA. Journal of Computational Biology, 2(1), 1-7.
- Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. science, 266(5187), 1021-1024.
- Abdelgaber, N., & Nikolopoulos, C. (2020, December). Overview on quantum computing and its applications in artificial intelligence. In *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)* (pp. 198-199). IEEE.
- Hu, F., Wang, B. N., Wang, N., & Wang, C. (2019). Quantum machine learning with D-wave quantum computer. *Quantum Engineering*, *1*(2), e12.
- Mücke, S., Heese, R., Müller, S., Wolter, M., & Piatkowski, N. (2023). Feature selection on quantum computers. *Quantum Machine Intelligence*, *5*(1), 11.
- Guzik, V., Gushanskiy, S., Polenov, M., & Potapov, V. (2015, October). Models of a quantum computer, their characteristics and analysis. In *2015 9th International Conference on Application of Information and Communication Technologies (AICT)* (pp. 583-587). IEEE.
- Shor, P. W. (1994, November). Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings 35th annual symposium on foundations of computer science (pp. 124-134). IEEE.
- Chen, S. (2024). A day with the world`s fastest supercomputer. Nature, 633, 23.