information

MDPI

Article

Secret Cryptographic Key Sharing Through the Integer Partition Function

Daniel Fernandes da Nobrega [1], Marcio Luís Munhoz Amorim [1], Sérgio F. Lopes [2,3], João Paulo Carmo [1], José A. Afonso [2,3,*] and Mario Gazziro [4]

[1] Group of Metamaterials Microwaves and Optics (GMeta), Department of Electrical Engineering (SEL), University of São Paulo (USP), Avenida Trabalhador São-Carlense, Nr. 400, Parque Industrial Arnold Schimidt, São Carlos 13566-590, Brazil; daniel.nobrega@alumni.usp.br (D.F.d.N.); marciolma@usp.br (M.L.M.A.); jcarmo@sc.usp.br (J.P.C.)
[2] CMEMS-UMinho, University of Minho, 4804-533 Guimarães, Portugal; sergio.lopes@dei.uminho.pt
[3] LABBELS—Associate Laboratory, University of Minho, 4710-057 Braga, Portugal
[4] Information Engineering Group, Department of Engineering and Social Sciences (CECS), Federal University of ABC (UFABC), Av. dos Estados, 5001, Santo André 09210-580, Brazil; mario.gazziro@ufabc.edu.br
* Correspondence: jose.afonso@dei.uminho.pt

Abstract

Secret key exchange is a necessary function for modern cryptography. The integer partition function is a mathematical function that arises from number theory. New methods for computing the integer partition function were developed and evaluated in the context of this paper, as well as new methods for using the integer partition function in a secret key exchange. The methods were categorized into single-variable and multiple-variable methods. The single-variable methods were found to be insecure. The multiple-variable methods were shown to be vulnerable to attacks that solve a linear system. These methods were implemented in microcontrollers using the C++ programming language. Experiments were conducted to evaluate the security of the developed methods in a wireless key exchange scenario. It was concluded that the security provided by the key exchange of the developed methods was low.

Keywords: cryptography; integer partitions; secret key exchange; number theory

Academic Editors: Zhigang Chu, Heming Jia and Emilio Matricciani

Received: 24 June 2025
Revised: 22 July 2025
Accepted: 24 July 2025
Published: 25 July 2025

Citation: Nobrega, D.F.d.; Amorim, M.L.M.; Lopes, S.F.; Carmo, J.P.; Afonso, J.A.; Gazziro, M. Secret Cryptographic Key Sharing Through the Integer Partition Function. *Information* **2025**, *16*, 637. https://doi.org/10.3390/info16080637

1. Introduction

One of the challenges faced by the information sector is the secure transmission of data. Cryptography enables secure communications in various applications, such as data transmission over the Internet. It becomes necessary when information needs to be exchanged between two entities without third parties being able to correctly interpret the messages, even if they have access to the transmitted data. Depending on the system, several cryptographic functions may be required, such as encryption, decryption, key sharing, pseudo-random number generation, digital signature, etc. Virtually all modern digital communication protocols, including TCP/IP, Wi-Fi, and Bluetooth protocols, have some encryption mechanism.

Cryptography is a telecommunications area that is expanding rapidly every day. With the goal of guaranteeing authenticity and privacy, new methods have been proposed as alternatives to existing protocols and systems. This advancement has been motivated not only by the demand for greater security but also by the latent threat of vulnerabilities in the most widely used protocols. Secret key exchange methods are a fundamental part of

modern cryptography, providing the basis mainly for the paradigm of public- and private-key encryption. However, the application of these methods in an insecure channel is a cryptographic problem where current implemented solutions may prove to be insufficient in the future.

The security of the Diffie–Hellman key exchange is based on the difficulty of the mathematical problem of the discrete logarithm [1], which assumes that it is difficult to find the exponent to which we raise a certain number, once the calculation has been carried out, within a finite group. It is, therefore, assumed that reversing the operation is too costly or impractical for an attacker targeting the communication. However, as time passes, new methods of performing this inverse operation continue to emerge. Notably, Shor's algorithm [2] can, in theory, factorize a large numbers efficiently using quantum computing techniques. Although this is still an area in its infancy, the assumption that the inversion of the exponentiation operation is difficult may be challenged in the coming decades with the advancement of quantum computing.

This paper focuses on sharing secret keys through a channel that can be accessed by third parties using novel methods based on integer partition. Experiments are also presented to analyze the cryptographic capabilities of the proposed scheme and its resistance to attacks. One of the main contributions of this paper is the proposal and evaluation of a new alternative to the Diffie–Hellman key exchange. To this end, we use integer partition functions, which have properties that make it possible to perform the cryptographic function of the key exchange. This paper also presents new formulas for obtaining the values of the integer partition functions, which are necessary in the practical application of the new key exchange alternative. The experimental results obtained from the implementation of these methods are presented and analyzed as a proof of concept of their viability for low-complexity cryptographic applications. Finally, ways to expand the scope of these methods to more robust applications are also proposed as directions for future work.

2. Literature Review

Cryptography consists of the use of practices and principles for transmitting information without this information being accessible to third parties, that is, in a way that only the transmitter (or sender) and the receiver (or recipient) of a message can correctly interpret its content [3]. This way, cryptography aims to protect private information from being known by potential malicious users or attackers. Although useful for protecting saved or transmitted data, this protective capacity of cryptography can be quite fragile. Therefore, careful studies on the mathematical and computational properties of the protocols and algorithms used in this area are necessary to apply or propose new techniques.

2.1. Cryptography Applications

Given its usefulness in ensuring authenticity and secrecy in a communication or data storage, cryptography has applications in several areas, including information security, telecommunications, financial systems, among others. Historically, cryptography has also been of great importance in the military sector due to the intrinsic need for communication security in this area. Therefore, cryptography research is an area in constant evolution.

There are currently several protocols for encrypting and decrypting e-mails, with PGP (Pretty Good Privacy) [4] being a widely used example for the secrecy and privacy requirement. The use of digital signatures to authenticate the sender of the message is also widespread in this area.

In general, protocols in the Internet protocol stack can implement cryptography to increase security. One of the most widely used protocols today that performs this function is TLS (Transport Layer Security) [5]. Encryption is also employed in website certificate

verification, where a public key cryptography system is used. Moreover, any website or service that registers its users with login and password usually implements some type of encryption to protect the data against attacks.

Pay TV and similar services also use encryption to control access to the service package [6]. The use of encryption protocols is also widespread on streaming platforms as a way to prevent unauthorized copying and distribution of content.

In mobile telephony, encryption has existed between the mobile phone and the radio base station since the second generation of communication technologies as a way to protect the privacy of communications. Subsequent generations, 3G, 4G, and 5G, have increased the security of communications, even reaching end-to-end encryption solutions [7]. Cryptographic authentication algorithms are also used to authenticate users of the mobile network.

The financial and banking system uses several encryption protocols [8]. Since it is an activity that requires constant authentication and data protection, cryptographic algorithms are used everywhere, from credit cards to bank transfers.

2.2. Security Attacks

The study of attacks is quite complex in itself since they are as numerous and complex as the systems they aim to compromise. Accordingly, we limit our discussion to examples of some attacks to consider when exchanging secret keys [3].

Brute-force attacks consist of going through the entire key space in search of the values that generated what was exchanged. If the secret or key is just a sequence entered by a person, it is possible that the secret is an easy-to-remember word or sequence, which reduces the key space considerably. Brute-force attacks are the simplest and also the easiest to prevent since simply increasing the key space can make this attack unfeasible.

There are also attacks that, through mathematical and computational methods, allow solving the problem whose computational hardness is the basis for the security of the cryptosystem. For example, given that the RSA system is based on factorization of large numbers, if an attacker manages to solve the factorization problem quickly, the entire cryptosystem would be at risk. This type of attack is considered in depth in this paper since we proposed key exchange schemes using mathematical problems as security guarantees.

Another type of attack is the man-in-the-middle (MITM), where an attacker places himself between the two entities that want to communicate and intercepts all messages, exchanging them for their own messages. This type of attack can be solved by implementing an authentication system. The most basic key exchange schemes, such as Diffie–Hellman, are not capable of preventing these attacks. The algorithms proposed in this paper also do not have this goal and should be implemented with the combination of other algorithms to prevent this type of attack.

2.3. Related Work

The integer partition function is a deep mathematical construct from number theory with various applications in cryptography [9]. To enable a modular use of secure protocols in this area, it is generally assumed that both the input and output are secretly shared between two or more parties [10]. For example, Nam et al. [11] explore password-based authenticated key exchange protocols in public network contexts, highlighting the challenges of establishing high-entropy secret keys from low-entropy sources, such as passwords. This complexity resonates with issues identified in the proposed methods as both require a comprehensive understanding of underlying mathematical functions to secure key exchanges effectively.

Another pertinent area of research is addressed by Afzal et al. [12], who assess the cryptographic strength of key schedule algorithms, quantifying their vulnerabilities. They

underscore the importance of rigorous evaluation of cryptographic strength in accordance with claims regarding low security in the outlined methods.

In addition, research on password-based three-party authenticated key exchange (3PAKE) protocols presents significant findings. For example, He and Chen [13] critically analyze the vulnerabilities of existing protocols, noting weaknesses that mirror those of the multiple-variable methods proposed in the abstract. Yoon and Yoo [14] also detail vulnerabilities in a password-based three-party authenticated key exchange protocol, emphasizing various attack strategies that could exploit weak key generation processes.

Previous studies have highlighted the implementation of cryptographic protocols based on proven mathematical constructs. Although some of these approaches contain nuances, they do not fundamentally diverge from traditional key exchange methods, which are also vulnerable to contemporary attack vectors [15,16].

3. Proposed Methods

3.1. The Integer Partition Function

Integer partitions consist of the decomposition of natural numbers into a sum of smaller natural numbers [17]. As an example, the number 4 can be represented as 4, $2+2$, $3+1$, $1+1+2$, or $1+1+1+1$. These are the integer partitions of 4, of which the summands are said to be a part. For example, 1 is a part of the partition $1+3$ of the number 4. Thus, the number 4 has four different partitions since changing the order of the parts does not imply a different partition (e.g., $3+1$ is the same partition as $1+3$).

By associating each integer with the number of possible partitions it has, we obtain the integer partition function $P(n)$. This quantity of possible partitions of a natural number has been the subject of study for many years, which has resulted in several important mathematical discoveries involving number theory throughout history, such as the Rogers–Ramanujan identities [18], and the recent discovery of exact formulas to find the value of these functions [19].

In addition to the function $P(n)$, other functions associated with the integer partition problem are also studied. This is the case of the integer partition functions into k parts, $P(n,k)$. These are the operations on each side of an integer that have exactly k parts. For example, the number 5 has two partitions of three parts: $2+2+1$ and $3+1+1$, that is, $P(5,3)=2$.

This article focuses on new ways to obtain formulas for $P(n,k)$ and the use of these functions in cryptography. A formula for $P(n)$ is also presented, resulting from the recurrence relation found for $P(n,k)$.

3.2. Secret Key Exchange: Single-Variable Options

The cryptographic application of the three options presented is the sharing of a secret over an insecure channel between two entities, A and B. The three options are strategies for sharing a secret, each using a different formula involving several $P(n,k)$, but each function calculation $p(n,k)$ is the same in each of them.

This secret can then be used, for example, as the key of a symmetric cryptographic system. All options assume that the values n_p, n_g, k_p, and k_g are agreed upon between A and B through this insecure channel and are, therefore, public. We also evaluated that the values c_A and c_B are integers chosen randomly, respectively, by A and B, being taken from the set of keys $C=\{c\in\mathbb{N}|c_{min}\leq c\leq c_{max}\}$, thus being a secret variable for A and another for B. Also, c_B is the index up to which B will perform the sum of the list of values A shared.

n_p, n_g, k_p, and k_g are variables necessary to establish the secret exchange, assumed to be "public". In a system that uses this cryptographic scheme, they would compose

"public keys". They are used in the calculation of $P(n, k)$, but they do not compose the "private keys", which are c_A and c_B taken from the set C. In analogy with the original Diffie–Hellman, n_p, n_g, k_p, and k_g would be like the "g" to which the secret keys a and b (g^a and g^b) must be raised to share and then arrive at $g(ab)$.

3.2.1. Single-Variable Option 1

So, A computes a list $L_A = [l_A(c_{min}), \ldots, l_A(c_{max})]$ of values, corresponding to the possible values of the term:

$$l_A(c_B) := \sum_{i=1}^{c_B} P(n_g + c_A \cdot k_g + i \cdot k_g - 1, k_g - 1) \quad (1)$$

varying c_B in all values of the set of keys C. Since c_B is assumed to be smaller than c_{max} and larger than c_{min}, this list will have $c_{max} - c_{min} + 1$ values. A sends this list of values to B over the insecure channel (n_p, n_g, k_p, and k_g can be considered constants for each secret key sharing, but they do not have fixed values that are predetermined or that have been analyzed to be the best).

Similarly, B calculates its list of values $L_B = [l_B(c_{min}), \ldots, l_B(c_{max})]$ based on the possible values of

$$l_B(c_A) := \sum_{i=1}^{c_A} P(n_g + c_B \cdot k_g + i \cdot k_g - 1, k_g - 1) \quad (2)$$

varying c_A in all values of the set of keys C. Since c_A is also assumed to be smaller than c_{max} and larger than a c_{min}, this list will have $c_{max} - c_{min} + 1$ values. B sends this list of values to A over the insecure channel.

A then calculates the values of $P(n_g + c_A \cdot k_g, k_g)$ and decrements from it the index value c_A of the list received from B, i.e., $l_B(c_A)$. Similarly, B computes the values of $P(n_g + c_B \cdot k_g, k_g)$ and subtracts from them the index value c_B of the list received from A, i.e., $l_A(c_B)$. Since these are the operations on each side of the equation, the final values computed by A and B are the same and are not directly obtained from the lists shared over the insecure channel. To have access to the data, an attacker would need to know either c_A or c_B, which are kept secret. These are the secret keys exchanged.

3.2.2. Single-Variable Option 2

Following the same steps as described for option 1, in option 2, A computes the list of values L_A that it sends to B using the term:

$$l_A(c_B) := \sum_{i=1}^{c_B} P(n_p - k_p, k_p + c_A + i) \quad (3)$$

and B computes the list of values L_B that it sends to A using the term:

$$l_B(c_A) := \sum_{i=1}^{c_A} P(n_p - k_p, k_p + c_B + i) \quad (4)$$

So, A computes the values of $P(n_p + c_A, k_p + c_A)$, and B computes the values of $P(n_p + c_B, k_p + c_B)$, from which each one removes the index values c_A and c_B of the list received from the other.

3.2.3. Single-Variable Option 3

In option 3, A computes the list of values L_A to send to B using the term:

$$l_A(c_B) := \sum_{i=1}^{c_B} P(n_g + c_A \cdot k_g + i \cdot k_g - 1, k_g - 1) + P(n_p - k, k_p + c_A + i) \tag{5}$$

and B computes the list of values L_B to send to A using the term:

$$l_B(c_A) := \sum_{i=1}^{c_A} P(n_g + c_B \cdot k_g + i \cdot k_g - 1, k_g - 1) + P(n_p - k, k_p + c_B + i) \tag{6}$$

Then, A computes the values of $P(n_g + c_A \cdot k_g, k_g) + P(n_p + c_A, k_p + c_A)$, and B computes the values of $P(n_g + c_B \cdot k_g, k_g) + P(n_p + c_B, k_p + c_B)$, from which they remove the index value, respectively, c_A and c_B of the list received from the other.

3.2.4. Limitations and Susceptibility to Attacks

In this section, we analyze each one-variable option and point out its limitations and susceptibility to attacks.

Option 1 uses only two equations for partitioning integers into k parts, one corresponding to k_g and one corresponding to $k_g - 1$. Therefore, its implementation is simple, and increasing the number n_g does not result in a large increase in computational time since this time continues to be the time required to replace the variable n_g in an equation similar to a polynomial.

However, since the equations are known to everyone and the number k_g is public, by sharing an element of the list sent by A, it is possible to solve the equation corresponding to this element and reveal c_A since this equation has only one variable and it is known that the root sought is an integer. Let us take the following examples.

- Example 1

Let us take $n_g = 10$, $k_g = 4$, $c_A = 3$, $c_B = 4$, and $c_{max} = 8$. A performs the operation: $\sum_{i=1}^{c_B} P(10 + 3 \cdot 4 + i \cdot 4 - 1, 3)$, for the eight possible values of c_B, and sends this list of values to B.

$$L_A = [52, 122, 213, 327, 467, 636, 836, 1070] \tag{7}$$

B similarly sends the following list:

$$L_B = [70, 161, 275, 415, 584, 784, 1018, 1289] \tag{8}$$

A then chooses the third element from the list sent by B and computes the secret with $P(10 + 3 \cdot 4, 4) - \sum_{i=1}^{3} P(10 + 4 \cdot 4 + i \cdot 4 - 1, 3)$, finding -191. B chooses the fourth element from the list sent by A and computes the secret, $P(10 + 4 \cdot 4, 4) - \sum_{i=1}^{4} P(10 + 3 \cdot 4 + i \cdot 4 - 1, 3) = -191$.

However, if an attacker has access to the list that A sends, they can deduce the value of c_A by using the equation $P(10, 3)$, equating it to the first value in the list:

$$\begin{gathered} P(10 + c_A \cdot 4 + 4, 3) = P(14 + 4c_A, 3) = \\ = \frac{1}{12} \cdot (14 + 4c_A)^2 + \frac{1}{3} \cdot \delta_3(14 + 4c_A, 0) + \frac{1}{4} \cdot \delta_2(14 + 4c_A, 1) - \frac{1}{3} = 52; \rightarrow c_A = 4 \end{gathered} \tag{9}$$

We see that by replacing n_g or k_g with a larger number, or even increasing the value of c_{max}, the problem would persist since the equations of $P(n, k)$ are similar to polynomials and known by everyone. It is also noted that the number c_B would be revealed to an observer of the channel by the same procedure.

Therefore, option 1 is insufficient to perform the secret exchange since by revealing c_A and c_B, the secret exchanged is revealed to a potential attacker.

Option 2 uses several different "k". Therefore, its use is conditioned on such equations being known or on there being another algorithm that can obtain $P(n,k)$ in a reasonable time since the ideal would be to have a large number of possible values of c_A and c_B. The recurrence equation could be used to obtain values of $P(n,k)$ for small n ($<10^5$), but these values would not be useful for generating cryptographic keys of considerable size.

However, even if this obstacle is overcome, there is an attack that allows an observer of the channel to know c_A and c_B. Since we assume that k_p and n_p are known to everyone, an observer would simply compute a list of values for $P(n_p - k_p, X)$, varying the value X, and compare it with the lists that are exchanged. Let us take the following example.

- Example 2

Let us take $n_p = 30$, $k_p = 4$, $c_A = 3$, $c_B = 4$, and $c_{max} = 8$. A performs the operation: $\sum_{i=1}^{c_B} P(30-4, 4+3+i)$, for the eight possible values of c_B, and sends this list of values to B.

$$L_A = [288, 540, 752, 921, 1054, 1155, 1232, 1288] \tag{10}$$

B similarly sends the following list:

$$L_B = [252, 464, 633, 766, 867, 944, 1000, 1042] \tag{11}$$

A then chooses the third element of the list sent by B and computes the secret with $P(30+3, 4+3) - \sum_{i=1}^{3} P(30-4, 4+4+i)$, finding 376. B chooses the fourth element from the list sent by A and computes the secret $P(30+4, 4+4) - \sum_{i=1}^{4} P(30-4, 4+3+i) = 376$.

However, the attacker can generate the following list of possible first values:

$$L_{atk} = [282, 300, 288, 252, 212, 169, 133, 101] \tag{12}$$

Thus, by comparing the values, the attacker determines $c_A = 3$ and $c_B = 4$.

As in the case of option 1, increasing the values of n_p and k_p will have no effect on the attack. One can increase the value of c_{max}, causing an attacker to have to calculate and compare more values for $P(n_p - k_p, X)$, but this would come at an additional computational cost for A and B, proportional to this possible increase in c_{max}.

Therefore, option 2 is insufficient to perform the secret exchange.

Option 3 combines the advantages of options 1 and 2. Concerning the part coming from option 1, we can use large numbers n_g without greatly increasing the computational time used. By adding the part coming from option 2 in the generation of the exchanged list of values, we also have the change of not using only one equation of $P(n,k)$, but two, and an attacker would not know a priori which equation is in each position of the list since the equation varies with the values of c_A (or c_B).

However, the increase in the complexity of the key continues to occur only with the increase in the value of c_{max}, which in turn causes an increase in the amount of calculations necessary to generate the lists since the size of the lists is c_{max}, a factor that greatly limits our key space. This leads us to explore other possibilities to increase the complexity of the system.

3.3. Secret Key Exchange: Multivariable Options

3.3.1. Multivariable Option 1

Here, we can create a system that uses as key a list of values for each individual A and B instead of just a number. In place of c_A and c_B, we would have lists of size s, $C_A =$

$[c_A(1),\ldots,c_A(s)]$ and $C_B = [c_B(1),\ldots,c_B(s)]$. Substituting each pair $(c_A(a), c_B(b)), 1 \leq a, b \leq s$, we have a total of s^2 equations, in the following form:

$$\begin{aligned} &P(n_p + c_A(a), k_p + c_A(a)) + P(n_g + c_A(a)\cdot k_g, k_g) - \\ &- \sum_{i=1}^{c_A(a)} \left[P(n_p - k_p, k_p + c_B(b) + i) + P(n_g + c_B(b)\cdot k_g + i\cdot k_g - 1, k_g - 1)\right] = \\ &= P(n_p + c_B(b), k_p + c_B(b)) + P(n_g + c_B(b)\cdot k_g, k_g) - \\ &- \sum_{i=1}^{c_B(b)} \left[P(n_p - k_p, k_p + c_A(a) + i) + P(n_g + c_A(a)\cdot k_g + i\cdot k_g - 1, k_g - 1)\right] \end{aligned} \tag{13}$$

If we add all the equations for a and b ranging from 1 to s, we obtain the following:

$$\begin{aligned} &s\cdot\sum_{a=1}^{s}\left[P(n_p + c_A(a), k_p + c_A(a)) + P(n_g + c_A(a)\cdot k_g, k_g)\right] - \\ &- \sum_{a=1}^{s}\sum_{i=1}^{c_A(a)}\sum_{b=1}^{s}\left[P(n_p - k_p, k_p + c_B(b) + i) + P(n_g + c_B(b)\cdot k_g + i\cdot k_g - 1, k_g - 1)\right] = \\ &= s\cdot\sum_{b=1}^{s}\left[P(n_p + c_B(b), k_p + c_B(b)) + P(n_g + c_B(b)\cdot k_g, k_g)\right] - \\ &- \sum_{b=1}^{s}\sum_{i=1}^{c_B(b)}\sum_{a=1}^{s}\left[P(n_p - k_p, k_p + c_A(a) + i) + P(n_g + c_A(a)\cdot k_g + i\cdot k_g - 1, k_g - 1)\right] \end{aligned} \tag{14}$$

The final secret is also equal to the following:

$$\begin{aligned} &\sum_{i=1}^{s}\left[P(n_p + c_A(i) + c_B(i), k_p + c_A(i) + c_B(i)) + P(n_g + c_A(i)\cdot k_g + c_B(i)\cdot k, k_g)\right] - \\ &- \sum_{a=1}^{s}\sum_{i=1}^{c_A(a)}\sum_{b=1}^{s}\left[P(n_p - k_p, k_p + c_B(b) + i) + P(n_g + c_B(b)\cdot k_g + i\cdot k_g - 1, k_g - 1)\right] - \\ &- \sum_{b=1}^{s}\sum_{i=1}^{c_B(b)}\sum_{a=1}^{s}\left[P(n_p - k_p, k_p + c_A(a) + i) + P(n_g + c_A(a)\cdot k_g + i\cdot k_g - 1, k_g - 1)\right] \end{aligned} \tag{15}$$

However, this last direct formula would require knowledge of both C_A and C_B and, therefore, cannot be used by any party in the communication.

Equation (14) reveals interesting properties for the exchange of lists. Although a priori s^2 exchanged lists are necessary, one for each equation of the type in option 3, when we add them, we can combine them into a single list, which goes from 1 to c_{max}. Each individual A or B can then add the terms of this index list corresponding to the elements of the list C_A or C_B itself. Consider the following multivariable example:

- Multivariable Example 1

Let us take $n_p = n_g = 30$, $k_p = k_g = 4$, $s = 2$, $C_A = [3,5]$, $C_B = [4,6]$, and $c_{max} = 8$. A performs the following operation:

$$\begin{aligned} &\sum_{i=1}^{c_B(b)}\sum_{a=1}^{2}[P(30 - 4, 4 + c_A(a) + i) + P(30 + c_A(a)\cdot 4 + i\cdot 4 - 1, 4 - 1)] = \\ &= \sum_{i=1}^{c_B(b)}[P(26, 4 + 3 + i) + P(30 + 3\cdot 4 + i\cdot 4 - 1, 3) + P(26, 4 + 5 + i)] + \\ &\quad + P(30 + 5\cdot 4 + i\cdot 4 - 1, 3) \end{aligned} \tag{16}$$

For the eight possible values of $c_B(b)$, it sends that list of values to B.

$$L_A = [903, 1795, 2684, 3577, 4494, 5447, 6457, 7534] \tag{17}$$

B similarly sends the following list:

$$L_B = [892, 1781, 2674, 3591, 4544, 5554, 6631, 7791] \tag{18}$$

A then chooses the third and fifth elements from the list sent by B and computes the secret with

$$\begin{aligned} 2 \cdot \sum_{a=1}^{2} [P(30 + c_A(a), 4 + c_A(a)) + P(30 + 4 \cdot c_A(a), 3)] - \sum_{a=1}^{2} \sum_{i=1}^{c_A(a)} P(30 - 4, 4 + 4 + i) = \\ = 2 \cdot [P(30 + 3, 4 + 3) + P(30 + 4 \cdot 3, 4) + P(30 + 5, 4 + 5) + P(30 + 4 \cdot 5, 4)] - \\ -(2674 + 4544) = 840 \end{aligned} \tag{19}$$

obtaining the number 840. B then chooses the fourth and sixth elements from the list that A sent and does similarly:

$$\begin{aligned} 2 \cdot \sum_{b=1}^{2} [P(30 + c_B(b), 4 + c_B(b)) + P(30 + 4 \cdot c_B(b), 4)] - \sum_{b=1}^{2} \sum_{i=1}^{c_B(b)} P(30 - 4, 4 + 4 + i) = \\ = 2 \cdot [P(30 + 4, 4 + 4) + P(30 + 4 \cdot 4, 4) + P(30 + 6, 4 + 6) + P(30 + 4 \cdot 6, 4)] - \\ -(3577 + 5447) = 840 \end{aligned} \tag{20}$$

Another advantage of this method is that each element of the exchanged list comes from a partition function equation with degree $k_g - 1$, plus a term of higher degree. In practice, we choose the value of n_g large enough so that these values can be considered difficult to reverse in the key space considered, also taking into account the terms that depend on k_p and n_p.

By increasing s, the number of keys available and possible for use increases, without necessarily increasing the amount of information available to potential observers of the channel. This ensures greater protection against attacks.

One of the attack methods would be to solve, with the c_{max} points from the list, a polynomial equation (with the δ-modular functions) with s independent variables, still having as a complicating factor that these points are increased by values that come from an unknown polynomial degree (part added by the term $P(n + c, k + c)$).

- Vulnerability to attack by linear system

Despite its advantages, including being robust to brute-force attacks, due to the greater number of keys, a vulnerability was found in the system based on multivariable option 1 to an attack strategy that looks at the problem proposed by the system in a different way, simplifying it. Instead of considering the problem as a system of s independent variables, we can consider it with one of c_{max} variables, writing it as follows:

$$\begin{aligned} \sum_{a=1}^{c_{max}} q_A(a) \cdot \left[P(n_p + a, k_p + a) + P(n_g + a \cdot k_g, k_g)\right] - \\ - \sum_{a=1}^{c_{max}} q_A(a) \sum_{i=1}^{a} \sum_{b=1}^{c_{max}} q_B(b) \cdot \left[P(n_p - k_p, k_p + b + i) + P(n_g + b \cdot k_g + i \cdot k_g - 1, k_g - 1)\right] = \\ = \sum_{b=1}^{c_{max}} q_B(b) \cdot \left[P(n_p + b, k_p + b) + P(n_g + b \cdot k_g, k_g)\right] - \\ - \sum_{b=1}^{c_{max}} q_B(b) \sum_{i=1}^{b} \sum_{a=1}^{c_{max}} q_A(a) \cdot \left[P(n_p - k_p, k_p + a + i) + P(n_g + a \cdot k_g + i \cdot k_g - 1, k_g - 1)\right] \end{aligned} \tag{21}$$

In this equation, the lists of numbers $Q_A = [q_A(1), \ldots, q_A(a), \ldots, q_A(c_{max})]$, known to A, and $Q_B = [q_B(1), \ldots, q_B(b), \ldots, q_B(c_{max})]$, known to B, represent, respectively, the number of times that the chosen variable of type c_A or c_B was equal to a and b, that is, how many of the elements of C_A are equal to each a and how many of the elements of C_B

are equal to each b. The lists Q_A and Q_B would, therefore, be sufficient to determine the secret exchanged instead of the sets C_A and C_B. The coefficients coming from the partition functions would be all possible, with the values a and b varying from 1 to c_{max}, and these would be known to everyone.

Due to these facts, it is more practical to think of multivariable option 1 as an exchange of lists Q_A and Q_B. We could then, instead of calculating s variables, just determine the quantities q of each possible value of the variables c, which reduces the calculation time, allowing us to use values of q as large as the value of the calculation space modulo M in which we will be inserted. This allows us to have a greater range of values and prevents brute-force attacks since the key space is assumed to be of considerable size. In this case, we would have $s = c_{max}$ variables.

However, the weakness of this option is that it is possible to construct a linear system from the exchanged values of c_{max} equations and c_{max} variables, and determine whose resolution provides the values of $q_A(a)$ or $q_B(b)$, thus reversing the process of building the exchanged lists and revealing the secret values.The exchanged values sent by A would be

$$\begin{aligned} L_A = \Bigg[& \sum_{i=1}^{1} \sum_{a=1}^{c_{max}} q_A(a) \cdot \left[P(n_p - k_p, k_p + a + i) + P(n_g + a \cdot k_g + i \cdot k_g - 1, k_g - 1) \right], \ldots \\ & \sum_{i=1}^{c_{max}} \sum_{a=1}^{c_{max}} q_A(a) \cdot \left[P(n_p - k_p, k_p + a + i) + P(n_g + a \cdot k_g + i \cdot k_g - 1, k_g - 1) \right] \Bigg] \end{aligned} \tag{22}$$

and similarly sent by B. We would then have a linear system of c_{max} equations and c_{max} variables (the variables q) for each transmitter, and an attacker would only need to solve one of them to obtain the secret since knowledge of one set Q and the list L of the other is enough to obtain the secret.

Considering that the complexity of solving a linear system is $O(c_{max}^{2.807})$ (Strassen's Algorithm, [20]), and the complexity of generating the lists is $O(c_{max}^2)$, there is a difference of order between the cost of exchanging the lists and the cost that an attacker would have to decipher the secret. However, this difference is very small compared to existing cryptographic systems, which leads us to look for other options to perform the secret exchange.

3.3.2. Multivariable Option 2

To generate this option, we sought to generalize the previously shown equations to take advantage of the fact that the variables appear as summation limits. Therefore, we started from a more general equation, considering T as any function of four variables, or a set of random numbers organized in four dimensions.

$$\sum_{a=1}^{a_{max}} \sum_{b=1}^{b_{max}} \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} q_A(a) \cdot q_B(b) \cdot T(i, j, a, b) \tag{23}$$

Here, we will have three lists of variables for A and for B:

$$\begin{aligned} Q_A &= [q_A(1), \ldots, q_A(a), \ldots, q_A(a_{max})] \\ C_{A,sup} &= \left[c_{A,sup}(1), \ldots, c_{A,sup}(a), \ldots, c_{A,sup}(a_{max}) \right] \\ C_{A,inf} &= \left[c_{A,inf}(1), \ldots, c_{A,inf}(a), \ldots, c_{A,inf}(a_{max}) \right] \\ Q_B &= [q_B(1), \ldots, q_B(b), \ldots, q_B(b_{max})] \\ C_{B,sup} &= \left[c_{B,sup}(1), \ldots, c_{B,sup}(b), \ldots, c_{B,sup}(b_{max}) \right] \\ C_{B,inf} &= \left[c_{B,inf}(1), \ldots, c_{B,inf}(b), \ldots, c_{B,inf}(b_{max}) \right] \end{aligned} \tag{24}$$

Developing this equation in two different ways, we have

$$\sum_{a=1}^{a_{max}} q_A(a) \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} \sum_{b=1}^{b_{max}} q_B(b) \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} T(i,j,a,b) = \\ = \sum_{b=1}^{b_{max}} q_B(b) \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} \sum_{a=1}^{a_{max}} q_A(a) \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} T(i,j,a,b) \tag{25}$$

The secret exchange, in this case, would be conducted by exchanging matrices M_A and M_B. B calculates, varying a from 1 to a_{max} and j from 1 to c_{max}, to form M_B, the possible values of

$$m_B(a,j) := \sum_{b=1}^{b_{max}} q_B(b) \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} T(i,j,a,b) \tag{26}$$

A does similarly, and, upon receiving the matrix M_B, calculates the secret by adding each row from the $c_{A,inf}(a)$-th term up to the $c_{A,sup}(a)$-th term and multiplying this sum by $q_A(a)$, for each a, and adding everything together. B calculates the secret in the same way.

In the matrix exchange procedure, a total of $a_{max} \cdot c_{max}$ values (i.e., equation results) are revealed to potential observers of the channel from the matrix generated by B and $s_B \cdot c_{max}$ values from the matrix generated by A. Assuming $a_{max} = b_{max} = s$, we will have $3s$ unknowns for A and the same amount for B (s unknowns q and $2s$ unknowns c for each) in two systems of $s \cdot c_{max}$ equations. Unlike multivariable option 1, this would not be a simple linear system since there are multiplications between the variables and the variables c are at the upper and lower limits of the summations.

The values of the function $T(i,j,a,b)$ should, therefore, be chosen in such a way as to make it difficult for an attacker to solve these systems. This work did not extensively evaluate the options for choosing $T(i,j,a,b)$, which could be addressed in future work. It is also important, for practical purposes, to keep in mind the size of the exchanged matrices. In this option, each matrix would have $s \cdot c_{max}$ values.

A possible attack strategy against this option would be to assume values for the unknowns c (which range from 1 to c_{max}) and solve the resulting linear systems to obtain possible values for the unknowns q. Once in possession of the values of q, it is possible to check whether the choice made for c's was the correct one. If so, the attack is successful; if not, it is necessary to assume another set of values for the unknowns c. This attack is subject to having to check all possible values of the set of c's, which are greater than c_{max}^s. Thus, the difficulty in carrying it out is exponential with the growth of s, having to solve more than c_{max}^s linear systems of s variables to check all possible values of the set of c's.

- Vulnerability to linear system attack

After a few attempts, it was possible to observe that, like multivariable option 1, this option is also subject to being represented as a linear system and, therefore, also presents this vulnerability. Equation (23) can be further generalized to give rise to

$$\sum_{a=1}^{a_{max}} \sum_{b=1}^{b_{max}} \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} q_A(a,j) \cdot q_B(b,i) \cdot T(i,j,a,b) \tag{27}$$

Developing the equation in two different ways, we have

$$\sum_{a=1}^{a_{max}} \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} q_A(a,j) \sum_{b=1}^{b_{max}} \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} q_B(b,i) T(i,j,a,b) = \\ = \sum_{b=1}^{b_{max}} \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} q_B(b,i) \sum_{a=1}^{a_{max}} \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} q_A(a,j) T(i,j,a,b) \tag{28}$$

Although we have apparently increased the number of unknowns, this way of writing the equation allows us to see the lists C as nothing more than zeroing some of the variables q that are not in the intervals added by the auxiliary variables i and j.

$$\sum_{a=1}^{a_{max}} \sum_{j=1}^{c_{max}} q_A(a,j) \sum_{b=1}^{b_{max}} \sum_{i=1}^{c_{max}} q_B(b,i) T(i,j,a,b) = \\ = \sum_{b=1}^{b_{max}} \sum_{i=1}^{c_{max}} q_B(b,i) \sum_{a=1}^{a_{max}} \sum_{j=1}^{c_{max}} q_A(a,j) T(i,j,a,b) \tag{29}$$

Thus, solving this more general problem would be equivalent to solving the problem proposed by multivariable option 2. However, this more general problem is simply a linear system but with $s \cdot c_{max}$ unknowns. We are again faced with a "breaking" complexity of the secret of the order of $O((s \cdot c_{max})^{2.807})$, against a generation complexity of the secret of the order of $O((s \cdot c_{max})^2)$, due to the four summations present in the formula.

3.3.3. Multivariable Option 3

Multivariable option 2 does not restrict the choice of the function (or the set of four-dimensional values) $T(i,j,a,b)$; its construction remains open and can be explored. If we choose fixed values, we impose a necessary amount of memory of A and B to store such values, which would be proportional to $c_{max}^2 \cdot s^2$. An option to avoid such an imposition is to use a function that benefits from the structure of the Equation (25). For this reason, we chose the integer partition function as follows: first, we added the equations of the recurrence relations.

$$P(n_g + c \cdot k_g, k_g) + P(n_p + c, k_p + c) = P(n_g, k_g) + \sum_{i=1}^{c} P(n_g + i \cdot k_g - 1, k_g - 1) + \\ + P(n_p, k_p) + \sum_{i=1}^{c} P(n_p - k_p, k_p + i) \tag{30}$$

$$P(n_g + c \cdot k_g, k_g) + P(n_p + c, k_p + c) - P(n_g, k_g) - P(n_p, k_p) = \\ = \sum_{i=1}^{c} \left[P(n_g + i \cdot k_g - 1, k_g - 1) + P(n_p - k_p, k_p + i) \right] \tag{31}$$

This equation is valid for any natural c, including c_{sup} and $(c_{inf} - 1)$:

$$P(n_g + c_{sup} \cdot k_g, k_g) + P(n_p + c_{sup}, k_p + c_{sup}) - P(n_g, k_g) - P(n_p, k_p) = \\ = \sum_{i=1}^{c_{sup}} \left[P(n_g + i \cdot k_g - 1, k_g - 1) + P(n_p - k_p, k_p + i) \right] \tag{32}$$

$$P(n_g + (c_{inf} - 1) \cdot k_g, k_g) + P(n_p + c_{inf} - 1, k_p + c_{inf} - 1) - P(n_g, k_g) - P(n_p, k_p) = \\ = \sum_{i=1}^{c_{inf}-1} \left[P(n_g + i \cdot k_g - 1, k_g - 1) + P(n_p - k_p, k_p + i) \right] \tag{33}$$

Subtracting the two previous equations, we have

$$\begin{aligned} &P(n_g + c_{sup} \cdot k_g, k_g) + P(n_p + c_{sup}, k_p + c_{sup}) - \\ &- P(n_g + (c_{inf} - 1) \cdot k_g, k_g) - P(n_p + c_{inf} - 1, k_p + c_{inf} - 1) = \\ &= \sum_{i=c_{inf}}^{c_{sup}} \left[P(n_g + i \cdot k_g - 1, k_g - 1) + P(n_p - k_p, k_p + i) \right] \end{aligned} \tag{34}$$

Let then $\beta(a, b)$ be a matrix of different and independent numbers. We then replace n_g by $\beta(a, b) + j \cdot k_g$, n_p by $n_p + j$, k_p by $k_p + j$, c_{sup} by $c_{B,sup}(b)$ and c_{inf} by $c_{B,inf}(b)$. We put the results in such a way as to replace the term $T(i, j, a, b)$ on the B side of (25), obtaining

$$\begin{aligned} \sum_{a=1}^{a_{max}} q_A(a) \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} \sum_{b=1}^{b_{max}} q_B(b) \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} &\left[P(\beta(a, b) + j \cdot k_g + i \cdot k_g - 1, k_g - 1) + \right. \\ &\left. + P(n_p - k_p, k_p + j + i) \right] \end{aligned} \tag{35}$$

$$\begin{aligned} &\sum_{a=1}^{a_{max}} q_A(a) \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} \sum_{b=1}^{b_{max}} q_B(b) \left[P(\beta(a, b) + j \cdot k_g + c_{B,sup}(b) \cdot k_g, k_g) + \right. \\ &+ P(n_p + j + c_{B,sup}(b), k_p + j + c_{B,sup}(b)) - P(\beta(a, b) + j \cdot k_g + (c_{B,inf}(b) - 1) \cdot k_g, k_g) - \\ &\left. - P(n_p + j + c_{B,inf}(b) - 1, k_p + j + c_{B,inf}(b) - 1) \right] \end{aligned} \tag{36}$$

By performing similarly for side A, we have

$$\begin{aligned} &\sum_{a=1}^{a_{max}} q_A(a) \sum_{j=c_{A,inf}(a)}^{c_{A,sup}(a)} \sum_{b=1}^{b_{max}} q_B(b) \left[P(\beta(a, b) + j \cdot k_g + c_{B,sup}(b) \cdot k_g, k_g) + \right. \\ &\quad + P(n_p + j + c_{B,sup}(b), k_p + j + c_{B,sup}(b)) - \\ &\quad - P(\beta(a, b) + j \cdot k_g + (c_{B,inf}(b) - 1) \cdot k_g, k_g) - \\ &\quad \left. - P(n_p + j + c_{B,inf}(b) - 1, k_p + j + c_{B,inf}(b) - 1) \right] = \\ &= \sum_{b=1}^{b_{max}} q_B(b) \sum_{i=c_{B,inf}(b)}^{c_{B,sup}(b)} \sum_{a=1}^{a_{max}} q_A(a) \left[P(\beta(a, b) + i \cdot k_g + c_{A,sup}(a) \cdot k_g, k_g) + \right. \\ &\quad + P(n_p + i + c_{A,sup}(a), k_p + i + c_{A,sup}(a)) - \\ &\quad - P(\beta(a, b) + i \cdot k_g + (c_{A,inf}(a) - 1) \cdot k_g, k_g) - \\ &\quad \left. - P(n_p + i + c_{A,inf}(a) - 1, k_p + i + c_{A,inf}(a) - 1) \right] \end{aligned} \tag{37}$$

Although we still need to store the values of β, we have a reduction in the number of dimensions of the data structure required, as well as a reduction in the number of calculations required, due to the reduction in a sum in the general formula. The data structure that would need to be calculated would have a size proportional to $s^2 \cdot c_{max}$, and the complexity of generating the exchanged matrices would be $O(s^2 \cdot c_{max})$. The exchanged matrices, as in the previous option, remain with a number of values $s \cdot c_{max}$.

- Vulnerability to linear system attack

Since it is a more specific version of multivariable option 2, multivariable option 3 is also subject to the same attack. The possible advantage is in the order of complexity required to solve the linear system compared to the complexity of generating the matrices. The generation of the matrices is conditioned to increase only linearly with the increase in c_{max}, while the "breaking" of the option requires a complexity equal to $O((s \cdot c_{max})^{2.807})$.

There is then a gain of order 1 in relation to c_{max}, when comparing the generation time and breaking time of this option with that of the multivariable option 2.

3.3.4. Multivariable Options Comparison

Multivariable option 1 uses the same equation from single-variable option 3. However, it is expanded to use many values as secret keys, having a list of secret keys for A and B. Multivariable option 2 is a general form of sharing a secret using the summation limits as secret keys. Here, the function T does not necessarily need to be a partition function. Multivariable option 3 uses equations with the partition function to be implemented in the form of multivariable option 2.

4. Results and Discussion

4.1. Simulations

As measures of efficiency and effectiveness of the cryptographic systems presented, we can analyze the time required to exchange secrets and the time an attacker would need to carry out a successful attack. In this sense, simulations were performed to measure the time required to exchange a secret, varying several parameters. Attack scenarios were also simulated.

Whenever applicable, two distinct strategies that could be implemented in real systems were considered. One of them consists of A and B generating new matrices or lists for each new communication, not necessarily by generating new secrets, but possibly also by choosing a random parameter. For example, in multivariable options 1 and 3, a large and random n_g could be used for each communication as a way of generating different secrets for each exchange between two devices. Since this strategy implies a significant computational cost added for each exchange, a second strategy was also tested in which A and B have fixed lists or matrices that would be used in every communication, or at least that these lists or matrices change according to a low frequency. One option would be to draw new secret numbers every day or every time the device in question is turned on. In this second strategy, it was also considered that devices A and B would have previously calculated the necessary values, such as integer partition functions, and stored them in memory. Since there is a significant cost to producing the exchanged matrices and lists, the difference between these two strategies is critical to considering real applications.

In order to approximate real situations, a space of size 256 bits, that is, size 2^{256}, was chosen as the space to perform the operations. This choice is due to the fact that the secret calculated between A and B can be used later to establish secure symmetric encryption communication, with 256 bits being one of the possible sizes of the AES (Advanced Encryption Standard) protocol.

It is also important to note that, in the case of multivariable option 1, we have $s = c_{max}$, an equality that is not repeated in the other multivariable options.

In all simulations, with the exception of the values that are varied to assess time dependencies in relation to the variables, the values adopted were those in Table 1.

Table 1. Values used in simulations.

Variable	n_g	n_p	k_g	k_p	c_{max}	s
Value	10^{100}	85	11	10	2	16

To evaluate the efficiency of the options, two measures of interest were established:

- t_{ger}, the time to generate the list or matrix.
- t_{calc}, the time to calculate the secret once the matrices or lists have been exchanged.

4.1.1. Influence of s on t_{calc}

As can be seen in Figure 1, the three multivariable options presented the same linear growth behavior in calculation time after the transmission of lists or matrices with the increase in the value of s, with multivariable option 1 being faster because it is a list, not a matrix. This is due to the fact that, in all options, the number of calculations necessary to reach the secret after the exchange of matrices or lists increases linearly with s, this value being the limit of a sum present in the calculation of the secret, as can be seen in the equations of the secrets of each option.

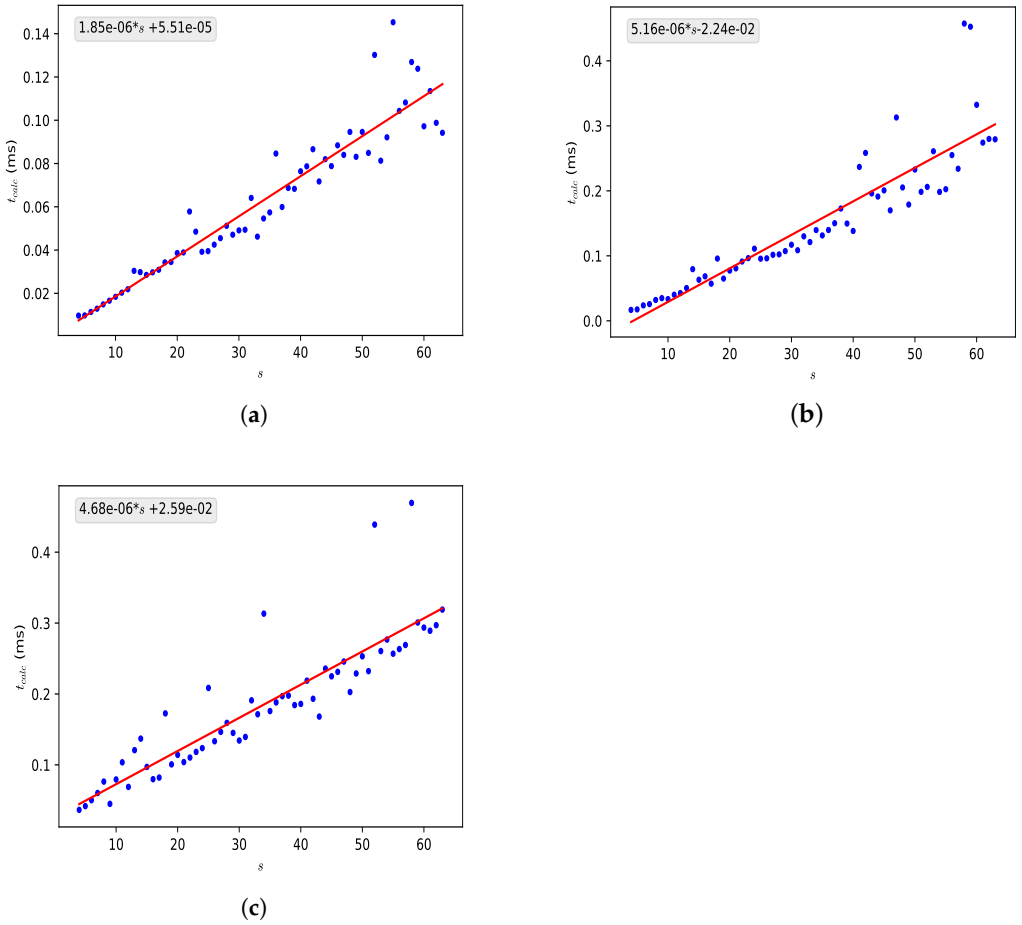
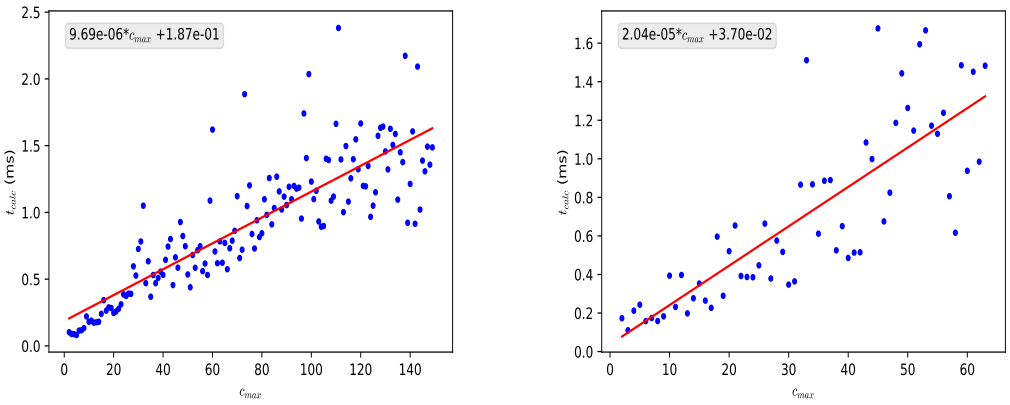


Figure 1. t_{calc} depending on s shows the computation time of the calculation of a secret key, given multiple values of the s parameter. (**a**) Multivariable option 1. (**b**) Multivariable option 2. (**c**) Multivariable option 3.

4.1.2. Influence of c_{max} on t_{calc}

We can see in the graphs in Figure 2 that the growth of t_{calc} in relation to an increase in c_{max} was linear. Observing the generating formulas of multivariable options 2 and 3, we see that this simulation result was expected. The large variance present in the graph is due to the fact that the choices of variables c, which were randomly chosen in each simulation, impact the number of calculations that must be made to arrive at the exchanged secret. Multivariable option 1 was not analyzed since in this case $s = c_{max}$, and this variation was already discussed previously.



(**a**) (**b**)

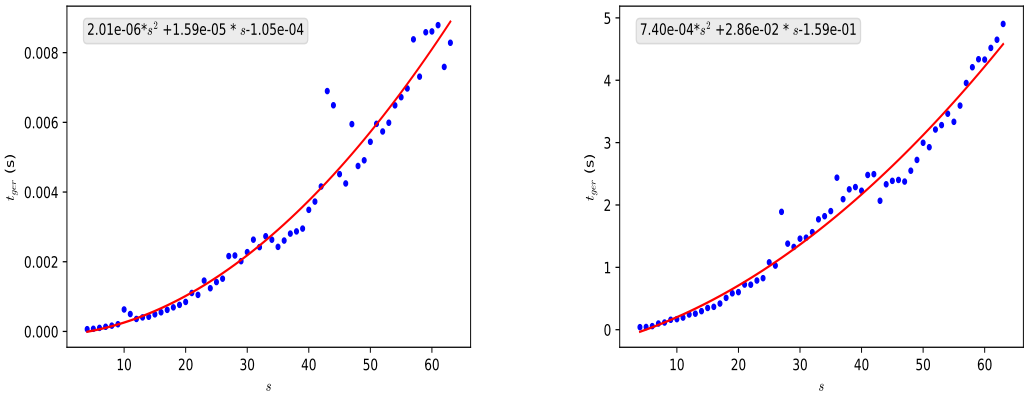
Figure 2. t_{calc} depending on c_{max} shows the computation time of the calculation of a secret key, given multiple values of the c_{max} parameter. (**a**) Multivariable option 2. (**b**) Multivariable option 3.

4.1.3. Influence of s on t_{ger}

Regarding the generation time of lists and matrices, as we can see in Figure 3, the growth was quadratic with the increase in s. In the case of multivariable option 1, we have $s = c_{max}$, with this value as the limit of two consecutive sums. In the case with the possibility of reusing the same list, there is a quadratic increase in total time with increasing s, while in the case of a constant matrix or list, this increase is linear (on each side of Equation (21), both A and B need to perform calculations whose quantities grow with s^2). Since, to calculate the values of a new matrix with each exchange, the growth is also quadratic, we have a quadratic result in this case in the same way.

Meanwhile, in multivariable option 2, we have a quadratic dependence on s since we have s as one of the dimensions of the exchanged matrix ($s \cdot c_{max}$) and s in the limit of a sum of each member of this matrix, as we see in Equation (25).

In multivariable option 3, in the case of a constant matrix in every exchange, we have s as one of the dimensions of the exchanged matrix and s as the limit of the sum of each member of the matrix. In the case where a matrix is calculated in every exchange, we have a dependence of s^2 on the size of the structure to be calculated, as we see in Equation (37), also resulting in quadratic growth.



(**a**) (**b**)

Figure 3. *Cont.*

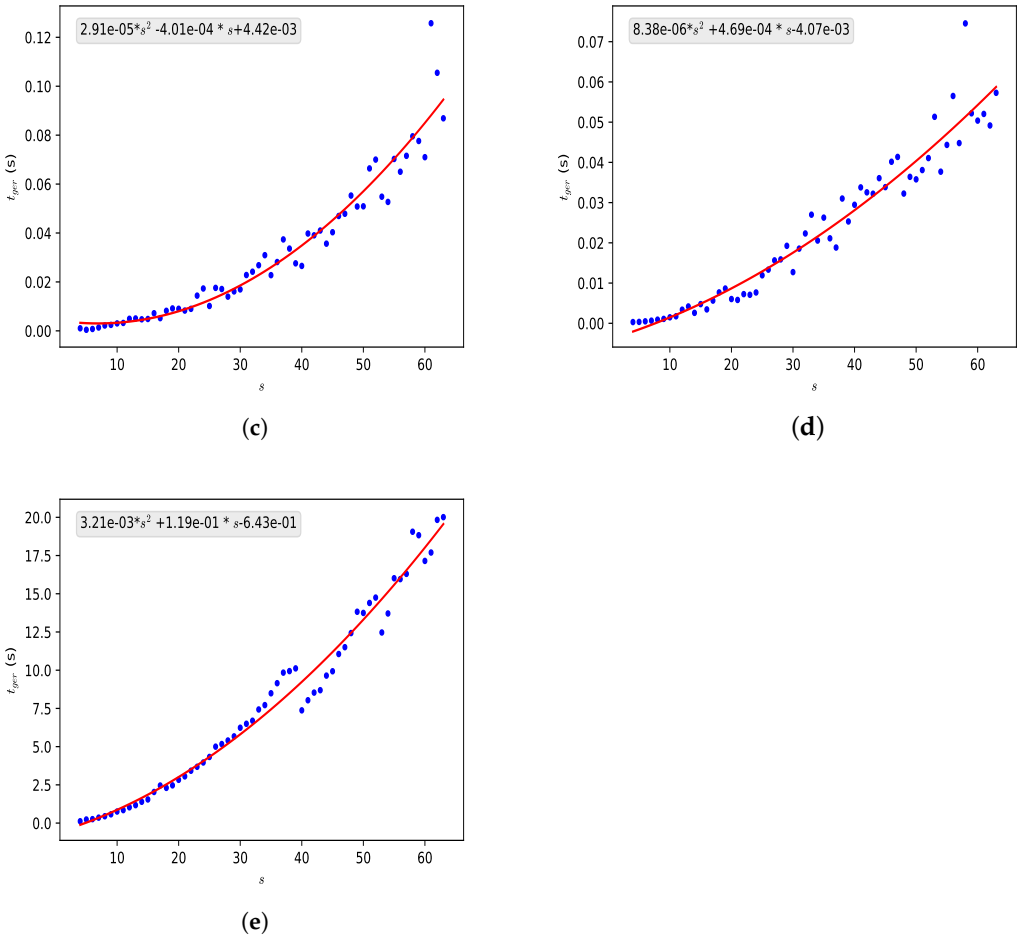


Figure 3. t_{ger} depending on s shows the computation time of the generation of the exchanged lists/matrices, given multiple values of the s parameter. (**a**) Multivariable option 1: Same list every time you swap. (**b**) Multivariable option 1: New list on each exchange. (**c**) Multivariable option 2. (**d**) Multivariable option 3: Same matrix every swap. (**e**) Multivariable option 3: New matrix on each swap.

4.1.4. Attack Simulations

To test the resistance of the developed options to the attacks evaluated, simulations were also performed. These attacks assume that an attacker would have access to the communication channel used to exchange matrices or lists. Except when variations are shown, the values of the variables required for each option were the same as in the previous simulations.

Due to the large key space in each multivariable option, it is not feasible to perform brute-force attack simulations. It is also important to note that there may be attacks that have not yet been discovered for each option, which were not studied in this work.

4.1.5. Summary of Variable Dependencies

We conclude this chapter with a summary of the impact of each variable considered in relation to the generation times of the matrices and lists and the calculation times of the secret after the exchange.

In Table 2, regarding t_{ger}, it can be seen that most of the variables did not show a clear correlation since, in these cases, it was considered that the calculation of the partitions would be carried out before the beginning of the exchange process. This table also shows that multivariable option 3 has a linear dependence in relation to c_{max}.

Table 2. Dependencies of t_{ger}, assuming the same array or list every swap.

Option	s	c_{max}	k_g	n_g	k_p	n_p
multivariable 1	$O(s^2)$	-	No correl.	No correl.	No correl.	No correl.
multivariable 2	$O(s^2)$	$O(c_{max}^2)$	-	-	-	-
multivariable 3	$O(s^2)$	$O(c_{max})$	No correl.	No correl.	No correl.	No correl.

In Table 3, we can observe several correlations between the various variables and t_{ger}, although it was not possible to identify clear correlations with the numbers n_g and k_p. This table also shows that the linear dependence of multivariable option 3 in relation to c_{max} is repeated. This dependence is noteworthy because it represents an advantage of this option in relation to multivariable options 1 and 2.

Table 3. Dependencies of t_{ger}, assuming a new array or list on each swap.

Option	s	c_{max}	k_g	n_g	k_p	n_p
multivariable 1	$O(s^2)$	-	$O(k_g^2)$	No correl.	No correl.	$O(n_p)$
multivariable 2	$O(s^2)$	$O(c_{max}^2)$	-	-	-	-
multivariable 3	$O(s^2)$	$O(c_{max})$	$O(k_g^2)$	No correl.	No correl.	$O(n_p)$

Regarding t_{calc}, all options presented the same behavior, a linear growth of t_{calc} both in variations of s and in variations of c_{max} (Table 4).

Table 4. Dependencies of t_{calc}.

Option	s	c_{max}
multivariable 1	$O(s)$	-
multivariable 2	$O(s)$	$O(c_{max})$
multivariable 3	$O(s)$	$O(c_{max})$

4.2. Application of Integer Partition Function Cryptography

In order to demonstrate the viability of the secret exchanges proposed in Section 3.2, an experiment was developed to simulate a situation in which this exchange is necessary. Using microcontrollers with integrated *Bluetooth*, a data transmission scenario was implemented between two devices (A and B) through an insecure channel, which was also observed by a third device (E).

In this experiment, the main objective was to verify whether, in an environment with limited resources, it is possible to perform information exchanges in a reasonable amount of time, considering a secure data transmission scenario of, for example, a body sensor or an automotive sensor that uses a wireless channel to communicate with other devices.

To evaluate multivariable options 1, 2, and 3, seen in the previous chapter, several parameters were analyzed. At first, due to the sensitivity of the possible applications to the amount of memory available in each microcontroller, the following parameters were considered:

- The memory space dedicated to each matrix (or list) to be sent, E_M;
- The memory space dedicated to storing private values, E_P.

The times required to exchange secrets were also analyzed, and the following parameters were specified:

- The time to generate the matrix (or list), t_{ger};
- The time to transmit the data, t_{trans};
- The time to calculate the secret, after transmission, t_{calc};
- The total time of the exchange, t_{total}.

Regarding the total exchange time, experiments were conducted with two different approaches: considering that there will be a new calculation of the matrix or list with each new exchange, or considering that there will be only one matrix or list that will be considered as calculated before the exchange begins. In the second case, the matrix or list would function as a public key for communication. However, this prevents the use of a random number in one of the variables (such as n_g for example) at the time of generating the matrix or list.

Through these experiments, we sought to determine whether any of the options presented are viable for use in a real situation, where the time and resources available to perform the secret exchange are limited.

4.3. Experiment Specifications

4.3.1. System Architecture

To perform the experiments, three ESP32 boards were programmed: Alice (transmitter *A*), Bob (receiver *B*), and Eve (attacker *E*), assembled as shown in Figure 4. The boards have the technical specifications reported in Table 5. The programming was carried out using the Arduino programming interface [21], compatible with the boards, and C++ was adopted as the programming language. The choice of using a third board equal to the first two as an attacker was made to simulate an attack from a device similar to those involved in the key exchange.

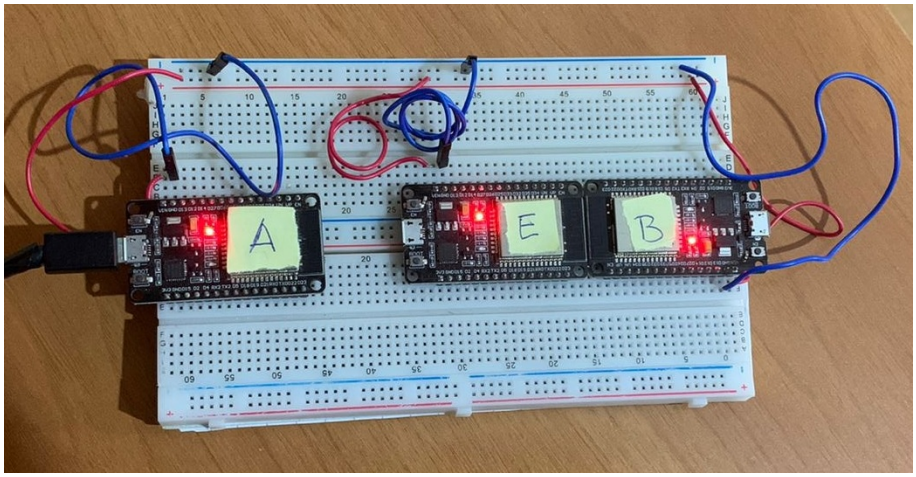


Figure 4. Experimental setup.

Table 5. Microcontroller specifications *Espressif Systems ESP32-WROOM-32* [22].

Processor	ROM Memory	SRAM Memory
240 MHz	448 KB	520 KB

In the situations presented, *A* initiates communication and establishes a Bluetooth Low-Energy channel with *B*. The secret exchange is then performed between *A* and *B* using one of the multivariable options presented in Section 3.3. Meanwhile, *E* observes

the same channel established by A and B and attempts to perform an attack that reveals the secret.

This communication capability between the three boards was achieved by configuring the boards A and B to automatically connect to any other board that initiated the connection, without requiring any type of authentication from devices that could read the data exchanged between the two. Thus, we would have a situation in which board E could read both the messages that A sends to B and the messages that B sends to A.

4.3.2. Simulation of an Insecure Channel Through Bluetooth

Although version 4.2 of Bluetooth presents means to establish a secure connection through cryptographic techniques, our goal in this experiment was to simulate an insecure channel simulation. Thus, the configuration of the devices was changed to completely open the transmitted data for third parties to observe, not respecting the specifications of the Bluetooth protocol.

This insecure channel works by configuring both devices A and B as server and client of each other. The program starts with both devices announcing their server capabilities. Then, both enable the client capability, which seeks the identification of the each other's server. After that, the double registration of A as a client of B and B as a client of A occurs. This allowed each device to subscribe to a service (communication channel) and be notified when there was a change in the characteristic, which is the data transmitted in that communication channel. This way, the exchange of messages between A and B was facilitated.

At the same time, the device E searches for the A and B and creates two client entities to connect to each one. Then, E registers to use the same service that B registered on A, and the same that A registered on B, thus having access to all the information exchanged between them.

Although this situation is not comparable to a real situation with the Bluetooth protocol, since it already implements advanced encryption and key exchange solutions that prevent this insecurity, this was a way to simulate a situation in which there is an insecure channel in some other technology with similar capabilities.

4.3.3. Algorithm Specifications

In order to simulate a scenario in which the secret exchanged between A and B would serve as a symmetric key in a subsequent transmission, the secret size was chosen to be 256 bits, which is one of the possible key sizes of the AES standard, only as an example, since the implementation of symmetric key cryptography was not the objective of this work.

The calculations performed by A, B, and E, as well as the values transmitted by them, were all performed using the default size of 256 bits, which means that all operations are performed $mod\ 2^{256}$.

Due to the specifications of the chosen devices, considerations regarding memory availability were made so that the values tested for s and c_{max}, among others, had to be relatively small, which is in line with our expectation of using this technology in devices with reduced capacity.

In the experiments of multivariable option 2, to simulate a function that was easy to calculate, the following function was chosen:

$$T^*(i, j, a, b) = (a + b) \cdot (i + j) \tag{38}$$

Due to time constraints, it was not possible to analyze whether this chosen function is the most appropriate, and it may have characteristics that facilitate some attack that is currently unknown. In its place, another function that can be calculated quickly may be used in the future.

Similarly, in multivariable option 3, instead of using a matrix of random numbers, $\beta(a, b)$ was replaced by the function:

$$\beta^*(a, b) = a + b \tag{39}$$

4.3.4. Attacks Evaluated

In none of the options was a brute-force attack considered since the space to which the secret belongs has 2^{256} elements, a quantity considered safe for the vast majority of contemporary applications [23]. Thus, to simulate attacks on the secret exchange options presented, focused strategies were chosen according to each option.

For multivariable option 1, as in the simulations presented in chapter Section 3.3, the attack by the linear system was chosen, considering the dimension of the system equal to the size of the exchanged list $s = c_{max}$. Since this is a known vulnerability of this option, it was expected that the board E would be able to carry out successful attacks for the chosen values of s.

For multivariable options 2 and 3, the attack by solving a linear system was chosen, as the case for the simulations in chapter Section 3.3, but with dimensions equal to the size of the exchanged matrices of $s \cdot c_{max}$. Although other attacks not currently known may emerge in the future, the analysis of these other attacks is beyond the scope of this work.

4.4. Experiment Results

Here, we present the results of the experiments performed using plates A, B, and E. Except for the tests in which each variable was analyzed separately, the values of the variables chosen were those reported in Table 6.

Table 6. Values used in simulations.

Variable	n_g	n_p	k_g	k_p	c_{max}	s
Value	10^{11}	5	11	1	2	2

Attacks

Secret key exchange attack tests were also performed, using the board E as an observer of the insecure channel constructed by Bluetooth communication, as was previously described. The results of these attacks were consistent with those of the simulations performed in Section 3.3. In both Figures 5 and 6, we see that the growth is polynomial, compatible with the expected growth in complexity when increasing the dimension of a linear system. In particular, it is clear that both the addition of more variables q and c, through the increase in s, and the increase in the possible range of values c generate the same difficulty in breaking.

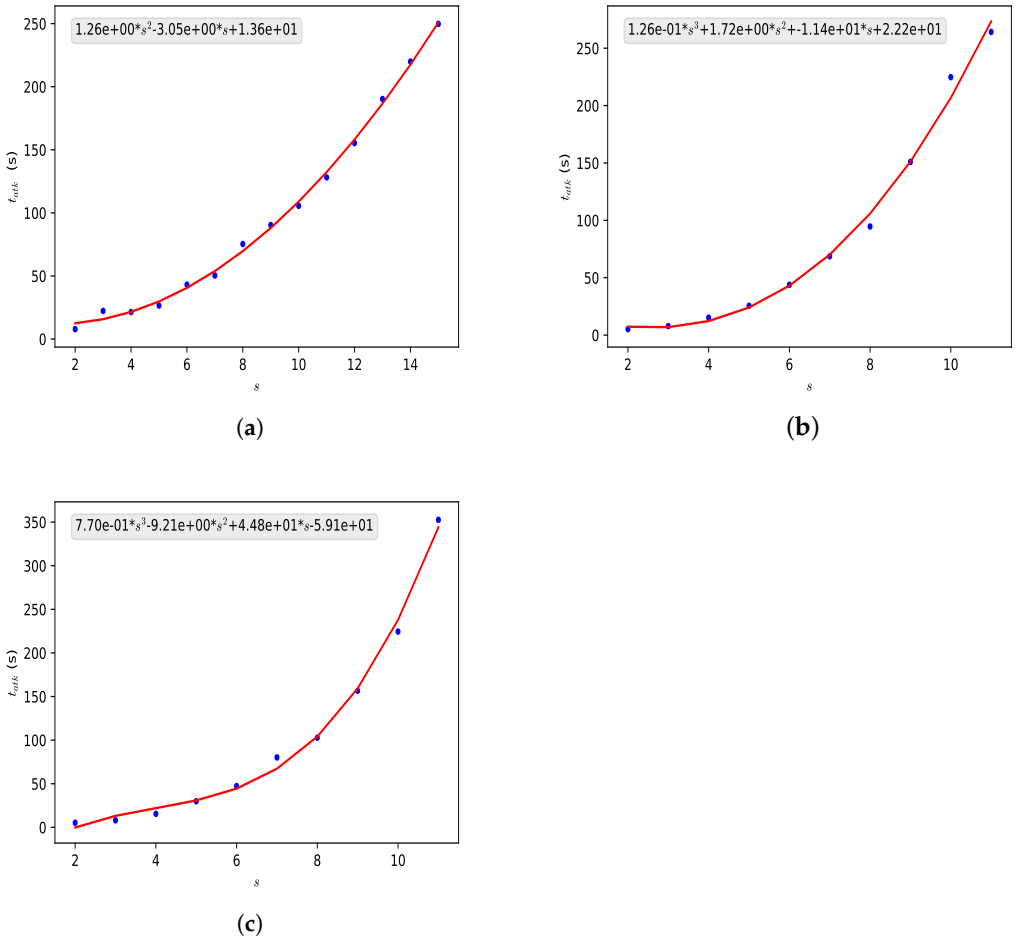


Figure 5. t_{atk} as a function of s. (**a**) Multivariable option 1. (**b**) Multivariable option 2. (**c**) Multivariable option 3.

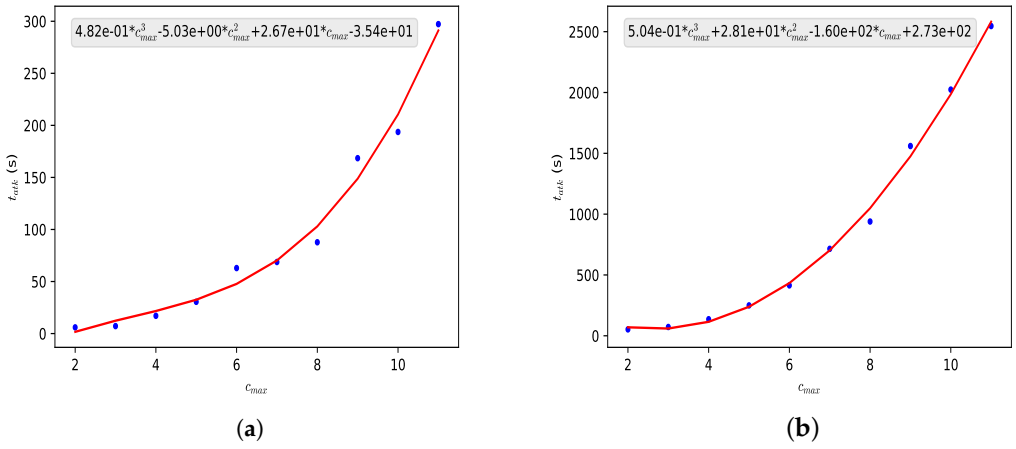


Figure 6. t_{atk} as a function of c_{max}. (**a**) Multivariable option 2. (**b**) Multivariable option 3.

4.5. Discussion of Results and Alternatives

4.5.1. Comparison Between Options

Tables 7 and 8 make a comparison between the multivariable options 1, 2, and 3, according to the experimental results, in relation to the growth that each variable presents regarding the increase in s. $t_{total,1}$ and $t_{total,2}$ represent, respectively, the strategies of using

a previously determined matrix or list for all communications and calculating a new matrix for each communication.

Table 7. Computational complexity as a function of s.

Option	E_M	E_P	t_{ger}	t_{trans}	t_{calc}	$t_{total,1}$	$t_{total,2}$	t_{atk}
m.var. 1	$O(s)$	$O(s)$	$O(s^2)$	$O(s)$	$O(s)$	$O(s)$	$O(s^2)$	$O(s^3)$
m.var. 2	$O(s)$	$O(s)$	$O(s^2)$	$O(s)$	$O(s)$	$O(s)$	$O(s^2)$	$O(s^3)$
m.var. 3	$O(s)$	$O(s)$	$O(s^2)$	$O(s)$	$O(s)$	$O(s)$	$O(s^2)$	$O(s^3)$

There is also the variation in computational complexity in relation to the increase in c_{max}, which can be observed in Table 8.

Table 8. Computational complexity as a function of c_{max}.

Option	E_M	E_P	t_{ger}	t_{atk}
m.var. 2	$O(c_{max})$	$O(c_{max})$	$O(c_{max}^2)$	$O(c_{max}^3)$
m.var. 3	$O(c_{max})$	$O(c_{max})$	$O(c_{max})$	$O(c_{max}^3)$

In the experimental values obtained, we can see that multivariable option 2 presented the shortest time to generate the value matrix, which occurs because the other two options calculate lists and matrices from integer partition functions, while multivariable option 2 performs simple sums.

It can also be verified, through the graphs of variation of s, that the transmission times of options 2 and 3 were approximately twice that of option 1 because with $c_{max} = 2$, we have matrices being transferred whose size is twice the list of option 1. If the value of c_{max} in options 2 and 3 increased, we would see a linear growth in this transmission time.

The calculation time of option 1 after the exchange of matrices was considerably shorter than the other two options because it presented a sum of only one dimension, while the other options needed to add matrices.

Regarding the total time for secret exchange, all options presented very high times (several seconds) if we consider the efficiency needs of key exchange protocols when we consider the strategy of generating a list or matrix for each new communication. However, if we adopt the strategy of keeping a matrix or list in memory and using it for several communications, the exchange time is restricted to milliseconds, a large part of which is due to the transmission time. More comparative studies are needed to verify the suitability of this key exchange in relation to the options available now.

4.5.2. Results Regarding Attacks

The secret key exchange methods demonstrated in this paper propose mathematical problems as a way to add security to this exchange. This paper explored attacks that focus on extracting the secret values (lists Q and C) from the communication devices through simple mathematical methods. There are other attacks that this work did not analyze, such as man-in-the-middle attacks, which can only be prevented by more complete systems.

It is also possible that there are unknown mathematical methods that make the presented options unsuitable for any type of application. In particular, the vulnerabilities presented by multivariable options 1, 2, and 3, due to being broken through the resolution of linear systems, make them relatively insecure for applications that wish to keep the exchanged messages secret for a long period of time. The security of the options can be increased, mainly in option 3, with little additional computational cost, but the performance

is not comparable to existing solutions, which protect the exchanged information for long periods of time.

Although only classical computing algorithms are applied, it is known that due to the existence of the HHL (Harrow, Hassidim, and Lloyd) algorithm, the exchanges presented would be even more vulnerable to attacks using quantum computing [24].

5. Conclusions

In order to present a new method for sharing secret keys in an insecure channel, this paper analyzed several ways of performing this sharing using integer partition functions.

Some examples of insecure communication channels are HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), plain email (SMTP, POP3, and IMAP without TLS/SSL), public Wi-Fi networks, SMS (Short Message Service), Bluetooth (older/weak implementations), and physical media without protection (USB drives/external hard drives).

A proof-of-concept example of some of the key exchange options was shown, using a set of microcontrollers equipped with *Bluetooth* connection as the application object. Experiments were then carried out to prove the validity of the conclusions drawn through simulations and to explore the challenges of transposing the theoretical solution to a concrete application.

This paper presented a new way to solve the problem of the number of integer partitions that a natural number can have. Therefore, we can conclude that the proposal to approach this topic from a new perspective was successful, which allowed the development of the techniques discussed in the other chapters.

As a result, we obtained a new formula for calculating the function $P(n)$, a function of interest to number theory. This formula comes, in turn, from the development of formulas for calculating the functions $P(n,k)$, related to $P(n)$. During the study of these formulas, ways were sought to make them easier to calculate, which led to the development and study of the δ-modular function, which relates the remainder of the division of two integers by a third integer.

The results obtained regarding the integer partition function may be useful in other areas of mathematics, such as statistics. Furthermore, the use of the δ-modular function to simplify calculations with floor and ceiling functions may be useful for several other related areas.

Author Contributions: Conceptualization, J.P.C.; methodology, D.F.d.N.; software, D.F.d.N.; validation, M.L.M.A.; formal analysis, M.G.; investigation, D.F.d.N. and J.P.C.; writing—original draft preparation, D.F.d.N. and M.G.; writing—review and editing, S.F.L. and J.A.A.; supervision, J.P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data are available at https://doi.org/10.5281/zenodo.15532198.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Schneier, B.; Diffie, W. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
2. Shor, P. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134. [CrossRef]

3. Boneh, D.; Shoup, V. A Graduate Course in Applied Cryptography. 2020. Available online: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf (accessed on 10 July 2024).
4. Callas, J.; Donnerhacke, L.; Finney, H.; Shaw, D.; Thayer, R. OpenPGP Message Format. 2007. Available online: https://www.rfc-editor.org/rfc/pdfrfc/rfc4880.txt.pdf (accessed on 10 July 2024).
5. Rescorla, E. *The Transport Layer Security (TLS) Protocol*; Version 1.3; Internet Engineering Task Force: Fremont, CA, USA, 2018. [CrossRef]
6. Graf, R.F.; Sheets, W. *Video Scrambling & Descrambling: For Satellite & Cable TV*; Newnes: Boston, UK, 1998.
7. Njoroge, F.; Kamau, L. A Survey of Cryptographic Methods in Mobile Network Technologies from 1G to 4G. 2018. Available online: https://www.researchgate.net/publication/328902626 (accessed on 10 July 2024).
8. Grigg, I. *Financial Cryptography in 7 Layers*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 332–348. [CrossRef]
9. Rajalakshmi, V.; Mala, G.S.A. Integer partitioning based encryption for privacy preservation in data mining. In Proceedings of the First International Conference on Security of Internet of Things (SecurIT '12), Kollam, India, 17–19 August 2012; pp. 246–251. [CrossRef]
10. Agarwal, A.; Boyle, E.; Chandran, N.; Gilboa, N.; Gupta, D.; Ishai, Y.; Kelkar, M.; Ma, Y. Secure Sorting and Selection via Function Secret Sharing. In Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security (CCS '24), Salt Lake City, UT, USA, 14–18 October 2024; pp. 3023–3037. [CrossRef]
11. Nam, J.; Choo, K.K.R.; Park, M.; Paik, J.; Won, D. On the Security of a Simple Three-Party Key Exchange Protocol without Server's Public Keys. *Sci. World J.* **2014**, *2014*, 479534. [CrossRef] [PubMed]
12. Afzal, S.; Yousaf, M.; Afzal, H.; Alharbe, N.; Mufti, M.R. Cryptographic Strength Evaluation of Key Schedule Algorithms. *Secur. Commun. Netw.* **2020**, *2020*, 3189601. [CrossRef]
13. He, D.; Chen, J. Cryptanalysis of a three-party password-based authenticated key exchange protocol using Weil pairing. *Int. J. Electron. Secur. Digit. Forensics* **2012**, *4*, 244–251. [CrossRef] [PubMed]
14. Yoon, E.J.; Yoo, K.Y. Cryptanalysis of a simple three-party password-based key exchange protocol. *Int. J. Commun. Syst.* **2011**, *24*, 532–542. [CrossRef]
15. Pak, K.; Pak, S.; Ho, C.; Pak, M.; Hwang, C. Anonymity preserving and round effective three-party authentication key exchange protocol based on chaotic maps. *PLoS ONE* **2019**, *14*, e0213976. [CrossRef] [PubMed]
16. Farash, M.S.; Attari, M.A. An enhanced and secure three-party password-based authenticated key exchange protocol without using server's public-keys and symmetric cryptosystems. *Inf. Technol. Control* **2014**, *43*, 143–150. [CrossRef]
17. Andrews, G.E.; Eriksson, K. *Integer Partitions*; Cambridge University Press: Cambridge, UK, 2004.
18. Ramanujan, S.; Rogers, L. Proof of certain identities in combinatory analysis. *Proc. Camb. Philos. Soc.* **1919**, *19*, 3.
19. Bruinier, J.H.; Ono, K. Algebraic formulas for the coefficients of half-integral weight harmonic weak maass forms. *Adv. Math.* **2011**, *246*, 198–219. [CrossRef]
20. Strassen, V. Gaussian elimination is not optimal. *Numer. Math.* **1969**, *13*, 354–356. [CrossRef]
21. Arduino. Arduino IDE. 2021. Available online: https://www.arduino.cc/ (accessed on 10 July 2024).
22. Systems, E. ESP32 Series Datasheet. 2021. Available online: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (accessed on 10 July 2024).
23. NIST. Advanced Encryption Standard (AES). 2001. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf (accessed on 10 July 2024). [CrossRef]
24. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [CrossRef] [PubMed]