DOI Link: https://doi.org/10.61586/0PbJL
Vol.43, Issue.7, Part.1, July 2025, PP.2-12

A METRIC-BASED COMPARISON OF WALSH, RSA, AND HYBRID WALSH-RSA ENCRYPTION TECHNIQUES

Sergo Episkoposian[1], Gevorg Margarov[1], Svetlana Grigoryan[1]

[1]National Polytechnic University of Armenia, 105 Teryan St, Yerevan, Armenia

Received May 2025 Accepted June 2025 Published July 2025

Abstract:
This paper presents a novel approach to encryption that combines RSA algorithm with the Walsh Transform. The RSA algorithm is used to securely exchange encryption keys, while the Walsh Transform is applied to the data prior to encryption.
Currently, our research team is investigating and developing algorithms to integrate the Walsh Transform into text processing methods. We are particularly focused on designing new hybrid encryption algorithms that combine the Walsh Transform with established cryptographic techniques such as RSA. This paper provides early findings on integrating RSA with the Walsh Transform [4] and examines the practical effectiveness of the proposed hybrid encryption model.

Keywords: Hybrid Cryptography, RSA, Walsh Transform, Walsh-RSA

INTRODUCTION

The use of cryptography dates back to ancient times. About 3,500 years ago, a Mesopotamian craftsman used cryptography, Spartans used cylinder-based encryption, and Indian spies employed coded messages in the 2nd century B.C. The Romans later developed the Caesar cipher, which worked by shifting letters in the alphabet. The first documented use of cryptography dates to 1900 B.C. in Egypt.

Modern cryptography falls into three categories:

Hash Functions: Creates irreversible mathematical "fingerprints" of data, mainly for ensuring message integrity.

Secret Key Cryptography (SKC): Uses one key for both encryption and decryption (symmetric encryption).

Public Key Cryptography (PKC): Uses separate keys for encryption and decryption (asymmetric encryption).

Public key cryptography was first publicly described in 1976 by Stanford's Martin Hellman and Whitfield Diffie [1], who introduced a system allowing secure communication without sharing secret keys.

This asymmetric approach uses two keys: a public key to encrypt messages and a private key to decrypt them, eliminating the need for pre-shared secret keys. The most widely used PKC algorithm is RSA, named after creators Ron Rivest, Adi Shamir, and Leonard Adleman. Introduced in 1977, RSA consists of key generation and encryption/decryption functions [10].

WALSH FUNCTIONS

To define the Walsh function, it is necessary to define the Rademacher function[5].

Definition 1. The Rademacher system is defined as

$$r_0(x) = \begin{cases} 1, x \in \left[0, \frac{1}{2}\right), \\ -1, x \in \left(\frac{1}{2}, 1\right], \end{cases} \quad r_0(x+1) = r_0(x),\ r_k(x) = r_0(2^k x), \qquad k = 1,2,\ldots,$$

i.e. to find the r_k Rademacher function, the interval $[0; 1)$ is split 2^{k+1} of equal subintervals, on each of which the $r_k(x)$ function takes +1 and -1 values successively.

The Walsh system [4,10] is defined as all possible finite products of Rademacher functions. More precisely, we define the Walsh system as follows:

Definition 2. $W_0(x) \equiv 1$. Let n be any natural number, represented as $n = \sum_{s=1}^{k} 2^{m_s}$, $m_1 > m_2 > \cdots > m_k$. The n-th Walsh function will be defined as follows:

$$W_n(x) = \prod_{s=1}^{n} r_{m_s}(x)$$

Walsh functions have several mathematical properties that make them suitable for use in cryptography.

The first property that needs to be mentioned is orthogonality, which means that each function is mathematically independent from the others.

The Walsh functions $W_n(x)$ are orthogonal on the interval [0, 1], and the orthogonality for all $m, n \in \mathbb{N}$ is expressed as:

$$\int_0^1 w_m(x) w_n(x) dx = \begin{cases} 1, & when\ m = n \\ 0, & when\ m \neq n \end{cases}$$

Walsh functions also exhibit unit energy which means that each function $w_k(x)$ has the norm L_2 equal to 1:

$$\|w_k(x)\|_2 = \sqrt{\int_0^1 w_k^2(x) dx} = 1$$

One of the other fundamental properties of Walsh functions is their completeness. It means that any function $f(x) \in L_2[0,1)$ can be expanded in a Walsh series:

$$f(x) = \sum_{k=0}^{\infty} c_k w_k(x)$$

where the coefficients c_k are defined as:

$$c_k = \int_0^1 f(x) w_k(x) dx$$

Walsh transforms are deterministic and reversible, allowing for the original data to be accurately recovered through inverse transformation.

Another important property is their support for the Fast Walsh–Hadamard Transform (FWHT), which makes them highly efficient for both encryption and decryption processes.

HADAMARD MATRIX

A Hadamard matrix is a square matrix whose elements are either +1 or -1 and whose rows are mutually orthogonal. This means that the scalar product of any two distinct rows is zero. A Hadamard matrix of order n (where n equals powers of 2) is formed as:

$$H_n = \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix},$$

where $H_1 = [1]$.

For example:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

We can get Walsh functions using Hadamard Matrix.
Walsh functions are essentially the rows of the Hadamard matrix rearranged in order. The Hadamard matrix provides a simple, recursive structure for deriving Walsh functions[7].

WALSH-HADAMARD TRANSFORM

Consider a text represented as a data vector $x \in \mathbb{R}^N$, where $N = 2^n$. The Walsh-Hadamard transform [7] for encryption is performed as follows:

Direct transformation:

$$y = H_n x.$$

Inverse transformation: to decipher the data, the transposed matrix is used:

$$x = \frac{1}{N} H_n^T y.$$

Let us denote e as the vector of noise superimposed on the encrypted data:

$$\tilde{y} = y + e$$

When reversed, the recovered data will be:

$$\tilde{x} = \frac{1}{N} H_n^T \tilde{y} = x + \frac{1}{N} H_n^T e$$

If the noise vector e is small in the L_2 norm, the reconstruction error will also be small:

$$\|\tilde{x} - x\|_2 = \frac{1}{N} \|H_n^T e\|_2 \leq \frac{1}{N} \|H_n\|_2 \|e\|_2 = \|e\|_2$$

WALSH FUNCTIONS AND RSA ALGORITHM IN CRYPTOGRAPHY

The Walsh transform takes a different approach from typical encryption methods - it doesn't need asymmetric keys. Instead, both the sender and receiver use the same key to transform and reverse the data.

First, you represent your original data as a vector, then apply the Walsh transform by multiplying a Walsh matrix with your data vector, and get Walsh coefficients that serve as your encrypted data ([2], [3]).

Before applying the Walsh transform, you'll need to do some pre-processing work - things like formatting your data, normalizing it, or adding padding where needed. The transform itself breaks down your original data into coefficients that show how much each Walsh basis function contributes to your data.

Let us now examine the mathematical basis underlying this concept. Consider the original data as a $x = [x_1, x_2, \ldots, x_n]$ vector where the dimension represents the size of your data. We need to apply the Walsh transform to this vector by using a transformation matrix with the same dimensions. After that we get a vector of coefficients: $c = [c_1, c_2, \ldots, c_n]$

$$c = H * x .$$

Once the Walsh transform is applied, we get a vector of coefficients.

When it comes time to decrypt, the original data needs to be rebuilt using the significant coefficients. Here's how that works:

Let's say we have a $c' = [c_1', c_2', \ldots, c_k']$, vector of significant coefficients where k represents how many significant coefficients you're working with. To recover your data, you use the inverse Walsh transform, but only with those selected significant coefficients:

$$x' = H^T * c',$$

In this case, the H^T matrix represents the transposed version of the Walsh transform matrix.

RSA (Rivest Shamir Adleman) is the most widely used public key encryption algorithm [8,10]. It operates with integers modulo $n = p \cdot q$ and requires keys of at least 1024 bits for adequate security, with 2048-bit keys providing optimal security. RSA is primarily used for secure communication channels and authentication with identity service providers. While too slow for encrypting large data volumes, it is commonly used for key distribution. The following steps are used to generate public and private keys in RSA. [9]

Generate a public key (e) and a private key (d) by choosing two very large prime numbers p and q each around 256 bits (75 digits).

Multiply p and q and let the result be n. $n = p \cdot q$. Note that the factors p and q remain secret and n is public. Even if n is known it is not practically feasible to get back p and q since factorizing a very large number is computationally infeasible.

Generate a public key by choosing a number e, which is relatively prime to the "totient" function $\varphi(n) = (p-1)(q-1)$.

Generate a private key by choosing a number d, which is the multiplicative inverse of e mod $\varphi(n)$. Note that the public key is $< e, n >$ which is known to everyone and the private key is $< d, n >$ which is known only to the person who has to decrypt or sign the message.

Encrypt a message m $(< n)$, raise m to the power e under modulo n .The result is the cipher text (c).

$$c = m^e \bmod n$$

Decrypt the cipher text, raise the cipher to the power d under modulo n.

$$m = c^d \bmod n.$$

Sign the message by encrypting it with the private key $< d, n >$ and decrypting it with the public key $< e, n >$. [6]

HYBRID APPROACH

The hybrid encryption method proposed in this article combines the Walsh Transform and RSA encryption. In this approach, the Walsh Transform is integrated as a preprocessing layer before the application of the classical RSA encryption algorithm (Figure 1).

The very first step of the algorithm is the input data processing. It begins by applying the Walsh Transform to the input data. By using the orthogonal Walsh functions, the data is transformed from its original domain into the Walsh domain, and the original message becomes hidden in a transformed pattern of coefficients.
This initial transformation makes it resistant to potential cryptographic attacks. From a cryptographic standpoint, an attacker attempting to analyze or manipulate the data would first have to reverse the Walsh transform—without knowing the structure or transform parameters used—before even reaching the RSA-encrypted payload. After preprocessing, the algorithm moves on to RSA encryption, which brings its own robust security features such as reliance on modular exponentiation and the computational hardness of prime factorization, to protect the sensitive information.
During decryption, the process is reversed. We need to use the RSA algorithm to decrypt the ciphertext using the private key. This step unlocks the transformed data, which is inverted through the Walsh transform as a next step to recover the original plaintext message.

To demonstrate the functionality of the proposed hybrid encryption method that combines the Walsh Transform with RSA, we present a conceptual example that illustrates the algorithm step by step. The process begins with a plaintext message—in this case, the word "Test". Each character in the message is first converted into its corresponding ASCII numerical representation, resulting in a vector of values: [84, 101, 115, 116].
As the first step in the encryption process, the Walsh Transform is applied to this vector using a 4×4 Walsh matrix, which converts the original data into a new set of coefficients in the Walsh domain. The resulting transformed vector is [416, -18, -46, -16]. These values are then converted to a byte format and encrypted using a 2048-bit RSA public key. This encryption step produces a ciphertext expressed in hexadecimal format (e.g., '8a7d0c75e2...'), effectively concealing both the original content and the structural properties of the data.
During decryption, the ciphertext is first decrypted using the RSA private key, recovering the same Walsh-domain values: [416, -18, -46, -16]. These values are then passed through the inverse Walsh Transform to return to the original domain. The output of this inverse transformation is [84, 101, 115, 116], which closely matches the initial ASCII values. Finally, these numbers are rounded and converted back into characters, successfully reconstructing the original message, "Test".

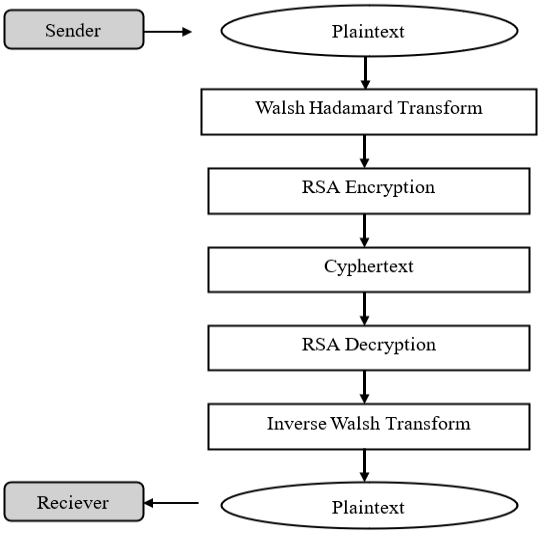


Figure 1. Hybrid approach

MEASURING CRYPTOGRAPHIC SECURITY

When we want to figure out how well cryptographic algorithms actually work and whether we can trust them, we look at several different measurements. These help us understand how secure they are, how fast they perform, and how random their output really is.

One of the first things we check is entropy. This tells us how unpredictable the system is, which is crucial for good encryption. Entropy is calculated using the Shannon's formula:

$$H = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i),$$

where H is the entropy in bits, p_i is the probability of the i-th symbol, n is the number of possible symbols.

Another key thing we look for when testing cryptographic algorithms is something called the Avalanche Effect. This basically means that even tiny changes to your input - like flipping just one bit - should create major, unpredictable changes in what comes out the other end.

Changing a single bit in the input should flip about half of the bits in the output. This kind of behavior is really important because it creates strong diffusion and stops attackers from spotting patterns they could exploit.
The Avalanche Effect is typically measured by the following formula:

$$Avalanche = \frac{Total\ Number\ of\ Output\ Bits}{Number\ of\ Changed\ Output\ Bits} \times 100\%$$

The Chi-Square Test is another way we check how random and evenly distributed the output from cryptographic systems really is - things like encrypted text or key streams. What it does is compare how often certain symbols (like bits or bytes) actually show up versus how often they should appear if everything was truly random.
The test is calculated using the formula:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i},$$

where χ^2 is the Chi-Square statistic, O_i is the observed frequency of the i^{th} symbol, E_i is the expected frequency of the i^{th} symbol, n is the number of distinct symbols.
To have a clear picture of how well each encryption method actually works, we put them through testing using the three key measures discussed above. The results are presented in Table 1.
First, we looked at entropy, which basically tells us how random and unpredictable the encrypted data is - the higher the number, the harder it would be for someone to crack the code. Next, we examined the avalanche effect, which shows how sensitive each method is to tiny changes in the original data. If you change just one letter in your message, a good encryption system should completely scramble the entire result. The higher the percentage, the better the system performs in this regard.
Finally, we used the chi-square test to check how evenly the encrypted bytes are distributed.
RSA-Walsh combination demonstrated better results than both the Walsh Transform and RSA when used alone. The hybrid approach displayed more random results, showed enhanced sensitivity to input changes, and created more uniformly distributed encrypted data. This suggests that combining these two methods really does create more reliable encryption system.

Table 1. Performance Comparison of Encryption Algorithms
(For "*Hello, this is a secret message*." plaintext.)

	Walsh Transform	RSA	RSA+W
Entropy	2.9830 bits per byte	3.9689 bits per byte	3.9872 bits per byte
Avalanche Effect	13.53% bits changed	32.20% bits changed	33.79% bits changed
Chi-Square Test	256.00	270.00	262.00

CONCLUSION

The evaluation of three encryption algorithms—Walsh transform, RSA, and the combined Walsh-RSA hybrid—reveals that the Walsh-RSA method delivers high security performance across key cryptography metrics, as shown in Table 1.

The Walsh transform alone shows good results in byte distribution uniformity with a Chi-Square value of 256.00, yet exhibits limitations in entropy generation (2.9830 bits/byte) and avalanche characteristics (13.53%), both essential measures of cryptographic strength.

RSA encryption substantially enhances these weaker areas, producing higher entropy (3.9689 bits/byte) and improved avalanche properties (32.20%), which indicate greater randomness and input sensitivity. The chi-square value of 270.00 for RSA indicates a slight deviation from perfect uniformity, suggesting minor statistical biases in byte distribution, but still reflects a generally strong level of ciphertext randomness.

The results indicate that the Walsh-RSA hybrid approach further optimizes discussed critical parameters, delivering peak entropy levels (3.9872 bits/byte) and maximum avalanche response (33.79%) while preserving near-optimal Chi-Square performance (262.00).

With the results, discussed above, the hybrid Walsh-RSA approach demonstrates encryption effectiveness compared to each method individually.

REFERENCES

1. Diffie, W., & Hellman, M. (1976). New Directions in Cryptography. IEEE Transactions on Information Theory, 22, 644-654.
2. Episkoposian, S. A. (2023). Application of Walsh system in data encryption. Tuijin Jishu/Journal of Propulsion Technology, 44(3), 494–502.
3. Episkoposian, S. A. (2023, September 12–13). Securing information with the Walsh transform. In Proceedings of the III International Scientific and Practical Conference: Questions, Hypotheses, Answers – Science XXI Century (pp. 10–14). Toronto, Canada.
4. Episkoposian, S. A., & Grigoryan, S. A. (2025, February). Hybrid cryptographic algorithm based on AES, RSA and Walsh transform. *Theoretical and Applied Technological Science Review*, *3*(1), 736–745.
5. Fine, N. J. (1949). On the Walsh functions. *Transactions of the American Mathematical Society*, *65*(3), 372–414. https://doi.org/10.1090/S0002-9947-1949-0032833-2
6. Kota, C. M., & Aissi, C. (n.d.). Implementation of the RSA algorithm and its cryptanalysis. University of Louisiana at Lafayette, College of Engineering.
7. Kunz, H.O. (1979). "On the Equivalence Between One-Dimensional Discrete Walsh–Hadamard and Multidimensional Discrete Fourier Transforms". IEEE Transactions on Computers. 28 (3): 267–8.
8. Nisha, S., & Farik, M. (2017, July). RSA public key cryptography algorithm – A review. International Journal of Scientific & Technology Research, 6(7).
9. Padmavathi, B., & Ranjitha Kumari, S. (2013, April). A survey on performance analysis of DES, AES, and RSA algorithm along with LSB substitution technique. International Journal of Science and Research (IJSR), 2(4). Online ISSN: 2319-7064.
10. Rivest, R.; Shamir, A.; Adleman, L. (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM. 21 (2): 120–126.