STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, February 2026, pp 1323–1342.
Published online in International Academic Press (www.IAPress.org)

An Extended Grendel approach applied to blockchain signature as an alternative to Keccak permutation

Abdelkarim LKOAIZA [1,*], Seddik ABDELALIM [1], Asmaa CHERKAOUI [1], Ilias ELMOUKI [2]

[1] *Laboratory of Mathematical Analysis, Algebra and Applications (LAM2A), Faculty of Sciences Ain Chock (FSAC), University Hassan II of Casablanca, Casablanca, Morocco.*
[2] *MoNum, EHTP, Casablanca, Morocco*

Abstract In this paper, we present our own developed programming which helps to generate a sponge-based function while avoiding any call from hashing libraries. Then, we try to implement it in a blockchain signature by getting inspired from Keccak methods such as the recently inextinguishable Secure Hash Algorithm 3 (SHA-3) , but before this, we note that our main contribution here, is about introducing the Grendel permutation instead of the Keccak one as they both rely on sponge-based procedures, but the shuffling step is different. In fact, even our Legendre symbol considered here, extends the Euler criterion that is restricted to prime field, to the cases of the group of invertible elements $\mathbb{Z}/pq\mathbb{Z}$. To the best of our knowledge, this is the first time that such an approach is used in blockchain signature.

Keywords Grendel hashing, Blockchain signature, Keccak family, Quadratic reciprocity, Rescue-prime, Sponge function.

AMS 2010 subject classifications 94A60, 68M25, 68P25

DOI: 10.19139/soic-2310-5070-2755

1. Introduction

The iterative sponge construction method and permutation functions used by the Keccak family play a crucial role in converting inputs into output values of constant size, forming the fundamental basis of this construction [4]. Contrary to the conventional Keccak methodology, we suggest adopting a new technique called the Grendel permutation [13]. This innovative strategy incorporates arithmetization [10] as basic gates in the design of encryption, specifically designed for application in cryptographic proof systems [14]. The integration of this permutation strategy represents a variant of arithmetization, leveraging mathematical frameworks to enhance cryptographic protocols and strengthen their resistance to potential attacks. By introducing the Grendel permutation as an alternative to the traditional Keccak approach, we aim to explore new avenues in encryption design that may improve the security and efficiency of cryptographic systems. This evolution towards arithmetization-based techniques highlights a paradigm shift in cryptographic research, emphasizing the importance of utilizing mathematical structures to advance the field and address emerging security challenges.

*Correspondence to: Abdelkarim Lkoaiza (Email: karimlkoaiza6@gmail.com). Laboratory of Mathematical Analysis, Algebra and Applications (LAM2A), Faculty of Sciences Ain Chock (FSAC), University Hassan II of Casablanca, Casablanca, Morocco.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

2. Blockchain

Blockchain is an innovative technology for creating and utilizing a distributed ledger. Data is recorded on a decentralized node called a block. No third party is required to use this technology, which improves its reliability [8]. While blockchain represents a database or a new way of organizing data, its method of processing differs from centralized systems, it is an electronic recording system. It enables the processing and recording of transactions, allowing all parties to track information through a secure network. Its key characteristics include decentralization, it does not belong to any single entity but rather is owned by its users. It is also open-source and offers high levels of security and protection, as it is nearly impossible to penetrate the system [9]. Blockchain provides its users with a great degree of confidentiality and privacy protection, users are represented by encrypted codes, which prevents the identification of other users personal data [15].

2.1. Blockchain architecture

In a decentralized blockchain, a node creates a transaction using an electronic signature with a private key. These transactions are collected and broadcasted across the network for validation by peers, who check for sufficient balance and the absence of double spending [5]. Validated transactions are grouped into blocks, verified through a consensus mechanism, and then added to the chain. Each new block connects to the previous ones, updating the copies held by all network participants. The process heavily relies on hash functions to ensure data integrity and security.

2.2. Secure blockchain hashing

Blockchain technology relies on robust algorithms such as SHA-1, SHA-2, SHA-256, and SHA-3 to ensure security and reliability [1]. Our innovative approach draws inspiration from the permutation function used in the sponges of the Rescue-prime family [14], integrating the arithmetic on rings of invertible elements over $\mathbb{Z}_{pq}$, while remaining faithful to the overall framework of sponges seen in SHA-3 of the Keccak family[10]. Our main contribution focuses on the mixing step, given the similarities between sponge functions in both families. By introducing a new sponge function that incorporates the *Modified Grendel* permutation [13], our goal is to provide a secure and optimized solution for validating blocks and digitally signing transactions within the realm of blockchain technology. Leveraging the properties of the Legendre symbol L_{pq}, this function is meticulously designed to enhance security measures and deliver efficient performance, akin to the advancements brought about by SHA-3 within the Keccak family. This innovative approach promises to strengthen security and operational efficiency within blockchain technology, thus paving the way for revolutionary advancements in the field.

The algorithm used to generate the hash must have the following properties [6]:

- **Deterministic**: The Grendel hash function always produces the same output for the same input, ensuring consistency and predictability.
- **Fast computation**: The Grendel hash function is designed to be computed quickly, ensuring efficiency in its use.
- **Resistance to preimage attacks**: It is difficult to retrieve the original input from the resulting hash, ensuring data integrity.
- **Hash change with small input modification**: Even slight changes in the input result in significant changes in the resulting hash, ensuring sensitivity to modifications.
- **Collision resistance**: The probability of two different inputs producing the same hash is very low, ensuring hash uniqueness.

- **Non-modifiability of hashes**: Hashes generated by the Grendel hash function cannot be modified once calculated, ensuring their integrity and authenticity.
- **No correlation between input and output bits**: Input bits are not directly linked to output bits of the hash, making it difficult to deduce the input from the output.
- **Random behavior**: The Grendel hash function incorporates elements of random behavior, rendering any attempt to predict the output associated with a given input futile.

3. Grendel sponge construction

The sponge construction is a framework for specifying functions on binary data with arbitrary output length [3]. It employs three main components: the Grendel permutation, the Grendel sponge, and the Grendel hash. Internally, the Grendel hash function utilizes the Grendel sponge as illustrated in Figure 1, with a fixed output length. The absorbing and squeezing phases will be explained thereafter in subsection of the Grendel Hashing. Additionally, a padding rule may be applied to the input. If the input length is fixed by the context, no padding rule is necessary. However, if the input length is variable, it is padded as follows: a single 1 bit is appended first, followed by zero padding until the input length is a multiple of the padding rate denoted by r.

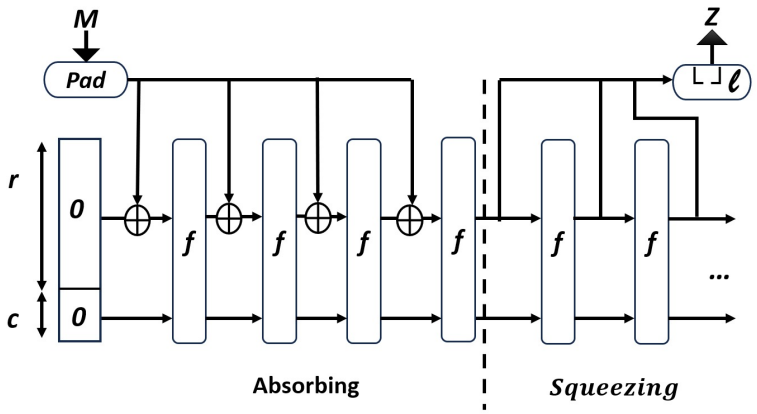

Figure 1. Grendel sponge steps

3.1. L_{pq} symbol

We propose a new S-box design specifically crafted for arithmetic-oriented ciphers, incorporating the L_{pq} symbol as a key element. An in-depth analysis is conducted on the differential and algebraic characteristics intrinsic to this particular implementation. Considering prime numbers represented as $p, q = 3 \mod 4$, and an integer $a \in \mathbb{Z}$ that is not divisible by either p or q. By invoking Bezout's theorem, it is established that there exist u, v such that $up + vq = 1$. The L_{pq} notation is formally introduced as part of this discussion. In fact, L_{pq} is defined by:

$$L_{pq}(a) = a^{\frac{(p-1)(q-1)}{4}} = \begin{cases} 1 & \text{if } a \text{ has a square root mod } p \text{ and } q\ , \\ vq - up & \text{if } a \text{ has a square root mod } p \text{ and does not have a square root mod } q, \\ up - vq & \text{if } a \text{ has a square root mod } q \text{ and does not have a square root mod } p, \\ -1 & \text{if } a \text{ does not have a square root mod } p \text{ and } q\ , \end{cases} \tag{1}$$

We could observe that the Legendre symbol considered here, extends the Euler criterion, by adding the cases when a has either a square root mod p while not having a square root mod q, or when a has a square root mod q while not having a square root mod p.

3.2. Extended Grendel permutation

We note that no Grendel approach has been applied to blockchain digital signature. This is to say that the word *extended* concerns the explanation stated just above. The extended Grendel permutation is carried out as follows:
Substitution: Every element within the list undergoes replacement with a novel numerical value based on a prescribed rule.
Permutation: The components of the list undergo rearrangement in a particular sequence utilizing a method of matrix multiplication.
Injection of round constants: Throughout each cycle of the permutation process, distinctive numerical values are incorporated to strengthen the security measures.

```
Algorithm 1.

def Extended_Grendel _Permutation(a):
     for i=0 to n-1 do:
         for j=0 to m-1 do:
            x_j =x_j^r*L_pq(x_j)
         y=M_1x
         z=M_2x
         for j=0 to m-1 do:
            x_j = z_j*u*p + y_j*v*q
         for j=0 to m-1 do:
            x_j=x_j+C_{im+j}
     return x
```

Grendel permutation applied to $\boldsymbol{x} \in \mathrm{U}(\mathrm{Z}_{pq})^m$
The pseudocode of Algorithm 1 formally describes this operation as it mutates a registered x in place.

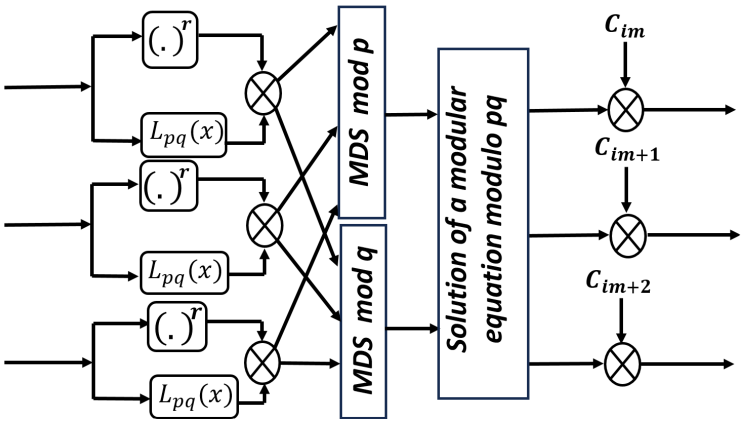

Figure 2. Grendel permutation steps. MDS: Diffusion Matrix

Figure 2 illustrates a single-round diagram for the specific case where m = 3.

3.3. Construction and Verification of an MDS Linear Layer

Setting and notation. Let p be a prime number, $\mathbb{F}_p$ the finite field of order p, and $g \in \mathbb{F}_p^*$ a primitive element. Fix $m \geq 1$ such that $2m \leq p-1$, and set

$$\alpha_j := g^j \qquad (0 \leq j \leq 2m-1), \qquad V_{i,j} := (\alpha_j)^i \quad (0 \leq i \leq m-1,\ 0 \leq j \leq 2m-1).$$

In other words, $V \in \mathbb{F}_p^{m \times 2m}$ is a Vandermonde matrix evaluated at the $2m$ pairwise distinct points $1, g, \ldots, g^{2m-1}$.

Proposition 3.1. *For every subset $J = \{j_1 < \cdots < j_m\} \subset \{0, \ldots, 2m-1\}$, the submatrix $V[:, J]$ is invertible. In particular, V has rank m and generates a Reed–Solomon $[2m, m]$ code over $\mathbb{F}_p$, hence it is* MDS.

Proof
The submatrix $V[:, J]$ is a square Vandermonde matrix of size m on the nodes $\alpha_{j_1}, \ldots, \alpha_{j_m}$, which are pairwise distinct because g is primitive and $2m \leq p-1$. Its determinant is

$$\det(V[:, J]) \;=\; \prod_{1 \leq a < b \leq m} (\alpha_{j_b} - \alpha_{j_a}) \;\neq\; 0 \quad \text{in } \mathbb{F}_p,$$

hence $V[:, J]$ is invertible. Since every choice of m columns is independent, the code generated by the rows of V is a Reed–Solomon $[2m, m]$ code, thus MDS. □

Proposition 3.2. *There exists a matrix $U \in GL_m(\mathbb{F}_p)$ such that*

$$U\,V \;=\; [\,I_m \mid M\,],$$

where $M \in \mathbb{F}_p^{m \times m}$. Consequently, $[I_m \mid M]$ is MDS, and every square submatrix of M is invertible.

Proof

Write $V = [A \,|\, B]$ where $A := V[:, 0\!:\!m] \in \mathbb{F}_p^{m \times m}$ and $B := V[:, m\!:\!2m] \in \mathbb{F}_p^{m \times m}$. By Proposition 3.1, A is invertible. Set $U := A^{-1}$ and $M := A^{-1}B$. Then

$$UV = [I_m \,|\, M].$$

Left multiplication by $U \in GL_m(\mathbb{F}_p)$ preserves column independence, hence $[I_m \,|\, M]$ is MDS.

□

Remark. *In particular, for any $1 \leq r \leq m$, every $r \times r$ square submatrix of M is nonsingular: pick those r columns in M and complete them with $m-r$ columns from I_m; since $[I_m \,|\, M]$ is MDS, any m columns are independent, hence the corresponding $r \times r$ minor is nonzero.*

Lemma 3.3. *For the linear layer $\ell(u) = (u, Mu)$ over $\mathbb{F}_p^m$, if the $m \times 2m$ matrix $[I_m \,|\, M]$ generates an MDS code (hence an $[2m, m]$ code with minimum distance $d_{\min} = m+1$), then*

$$B(M) \;=\; \min_{u \neq 0} \big(wt(u) + wt(Mu)\big) \;=\; m+1.$$

Proof

Every vector of the form (u, uM) with $u \neq 0$ is a nonzero codeword of the linear code generated by $[I_m \,|\, M]$. Since this code is MDS with parameters $[2m, m]$, its minimum distance is $d_{\min} = 2m - m + 1 = m + 1$.

Therefore, for all $u \neq 0$,

$$wt(u) + wt(uM) \;=\; wt\big((u,\, uM)\big) \;\geq\; d_{\min} = m + 1,$$

so $B(M) \geq m + 1$. By the definition of $d_{\min}$, there exists $u^\star \neq 0$ such that $wt\big((u^\star,\, u^\star M)\big) = d_{\min} = m + 1$, hence $B(M) \leq m + 1$. Combining both inequalities yields $B(M) = m + 1$. □

Under the assumption $2m \leq p - 1$ and g is primitive, Algorithm 2 returns a matrix $M \in \mathbb{F}_p^{m\times m}$ such that $[I_m \mid M]$ is a systematic generator of a Reed–Solomon $[2m, m]$ code over $\mathbb{F}_p$ (hence MDS) [7], and the associated linear layer $\ell(x) = (x, Mx)$ achieves a branch number $B(M) = m + 1$.

```
Algorithm 2.

def get_mds_matrix(p, m):
    assert 2*m <= p-1
    Fp = FiniteField(p)
    g = Fp(2)
    while g.multiplicative_order() != p - 1:
        g = g + 1
    V = matrix([[g^(i*j) for j in range(0, 2*m)] for i in range(0, m)])
    V_ech = V.echelon_form()
    MDS = V_ech[:, m:].transpose()
    return MDS
```

3.4. Round constants

The round constants are generated deterministically by expanding a short seed phrase using SHAKE256. This method ensures a *nothing-up-my-sleeve* construction and prevents the intentional introduction of hidden weaknesses, assuming SHAKE256 is secure.

Specifically, we start from the string: `"Grendel-Modified-%i-%i-%i-%i"` where the parameters p, q, m, λ replace the placeholders. SHAKE256 expands this string into a stream of $nm \times \left(1 + \left\lceil \frac{\log_2(pq)}{8} \right\rceil\right)$ bytes. Each block of $w = 1 + \left\lceil \frac{\log_2(pq)}{8} \right\rceil$ bytes is interpreted in big-endian order, then reduced modulo pq to obtain the corresponding round constant.

```
Algorithm 3.

def get_round_constants(p, q, m, n, lam):
    w = 1 + math.ceil(math.log2(p*q) / 8)
    seed = f"Grendel-Modified-{p}-{q}-{m}-{lam}"
    shake = hashlib.shake_256(seed.encode())
    out_len = n * m * w
    buf = shake.digest(out_len)
    C = []
    for i in range(n * m):
        acc = 0
        off = i * w
        for b in buf[off:off + w]:
            acc = (acc << 8) | b
        x = acc % pq
        C.append(x)
    return C  # Index: k = i*m + j
```

3.5. Grendel hash

In practical implementations, the output is always truncated to a specified number of elements, with the output length provided as an argument to the function. The evaluation of the sponge function on a computer involves two phases: absorbing and squeezing as also introduced in Algorithm 4. Functions like Keccak (used in SHA-3) and Grendel hashing utilize these steps internally. In the *absorbing phase*, a chunk of r elements from the input is added to the top, r elements of the state, followed by applying the permutation to the entire state. In the *squeezing phase*, the top r elements of the state are appended to the output, and the permutation is applied to the entire state again. This process repeats until the output buffer accumulates the required number of elements specified by the output length, after which the result is returned and the algorithm terminates. The Grendel hash function employs the Grendel sponge internally, with a fixed output length. Additionally, an input padding mechanism may be applied. Specifically, if the input length is predetermined within the context, no padding rule is necessary. However, for variable input lengths, padding follows this protocol: appending a single 1 initially, then padding with zeros until the input length becomes a multiple of r.

```
Algorithm 4

def Extended_Grendel_sponge(m,len_output):
     satate=(0,...,0)
### absorbing
     for i=0 to [len(m)/r] do:
         for j=0 to min(r,len(m)-ir)do:
            state_j =state_j+m_{ir+j}
         state=Extended_Grendel_Permutation(state)
### squeezing
    output="" ### empty string
     for i=0 to [len_output/r] do:
        for j=0 to min(r,len_output-ir)do:
           output = output + state_j
        state=Extended_Grendel_Permutation(state)
    return(output)
```

The extended Grendel hash function in Algorithm 5, incorporates the extended Grendel sponge internally. Furthermore, a padding scheme may be implemented on the input data: initially, add a solitary 1, followed by the addition of zeros to extend the input length to a factor of r. The subsequent algorithm delineates this process in a structured manner

```
Algorithm 5

def Grendel_Modified _Hash(m):

     if  padding do:
         m=m+1
         while len(m)!= r do :
         m=m+0
    output=Grendel_Modified_sponge(m,len_output)
     return(output)
```

4. Mathematical results

Lemma 4.1. *Let p and q be two prime numbers. An integer a is a quadratic residue in $U(\mathbb{Z}/pq\mathbb{Z})$ if and only if a is a quadratic residue in $\mathbb{Z}/p\mathbb{Z}$ and in $\mathbb{Z}/q\mathbb{Z}$.*

Proof
$\Rightarrow$ Let a be a quadratic residue in $U(\mathbb{Z}/pq\mathbb{Z})$. Then there exists $b \in U(\mathbb{Z}/pq\mathbb{Z})$ such that $a = b^2$. This implies that $pq \mid (b^2 - a)$, so we can write $b^2 - a = k \cdot pq$ for some integer k.

In particular, we have:

$$b^2 - a = k \cdot pq$$

which implies:

$$p \mid (b^2 - a) \quad \text{and} \quad q \mid (b^2 - a).$$

Thus, a is a quadratic residue in $\mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}/q\mathbb{Z}$.

$\Leftarrow$ Conversely, suppose that a is a quadratic residue in $\mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}/q\mathbb{Z}$ for distinct prime numbers p and q. Then, there exist b_1 and b_2 such that:

$$\begin{cases} a \equiv b_1^2 \pmod{p} \\ a \equiv b_2^2 \pmod{q}. \end{cases}$$

By the Chinese Remainder Theorem, there exists $t \in U(\mathbb{Z}/pq\mathbb{Z})$ such that:

$$\begin{cases} t \equiv b_1 \pmod{p} \\ t \equiv b_2 \pmod{q} \end{cases} \quad \text{Thus, we have :} \quad \begin{cases} a \equiv t^2 \pmod{p} \\ a \equiv t^2 \pmod{q} \end{cases}$$

This implies:

$$p \mid (a - t^2) \quad \text{and} \quad q \mid (a - t^2).$$

Therefore, the least common multiple of p and q divides $a - t^2$, i.e.:

$$pq \mid (a - t^2).$$

Hence, a is a quadratic residue in $U(\mathbb{Z}/pq\mathbb{Z})$.

□

Lemma 4.2. *Let p and q be two prime numbers strictly superior than 2. The following result is verified:*
1- For all $a \in U(\mathbb{Z}/pq\mathbb{Z})$, $a^{\frac{(p-1)(q-1)}{2}} \equiv 1 \pmod{pq}$.
2- The cardinality of the quadratic residues in $U(\mathbb{Z}/pq\mathbb{Z})$ is equal to $\frac{(p-1)(q-1)}{4}$.

Proof
1- Since a is invertible in $\mathbb{Z}/pq\mathbb{Z}$, it follows that a is relatively prime to both p and q. Therefore, $a \neq 0 \pmod{p}$ and $a \neq 0 \pmod{q}$. Consequently, we have:

$$\begin{cases} a^{(p-1)/2} \equiv \pm 1 \pmod{p} \\ a^{(q-1)/2} \equiv \pm 1 \pmod{q}. \end{cases}$$

From these congruences, we can deduce:

$$\begin{cases} a^{\frac{(p-1)(q-1)}{2}} \equiv (\pm 1)^{(q-1)} \equiv 1 \pmod{p} \\ a^{\frac{(p-1)(q-1)}{2}} \equiv (\pm 1)^{(q-1)} \equiv 1 \pmod{q}.. \end{cases}$$

Therefore:

$$\begin{cases} p \mid a^{\frac{(p-1)(q-1)}{2}} - 1 \\ q \mid a^{\frac{(p-1)(q-1)}{2}} - 1 \end{cases} \quad \text{Thus, we have:} \quad pq \mid a^{\frac{(p-1)(q-1)}{2}} - 1.$$

In conclusion:

$$a^{\frac{(p-1)(q-1)}{2}} \equiv 1 \pmod{pq}.$$

2- According to the Chinese Remainder Theorem, $U(\mathbb{Z}/pq\mathbb{Z})$ is isomorphic to $U(\mathbb{Z}/p\mathbb{Z}) \times U(\mathbb{Z}/q\mathbb{Z})$. An element $a \neq 0$ in $U(\mathbb{Z}/pq\mathbb{Z})$ is a square if and only if it is a square in $\mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}/q\mathbb{Z}$.
We have the number of squares in $\mathbb{Z}/p\mathbb{Z}$ is $\frac{p-1}{2}$ and in $\mathbb{Z}/q\mathbb{Z}$ is $\frac{q-1}{2}$. Therefore, the number of squares in $U(\mathbb{Z}/pq\mathbb{Z})$ is $\frac{(p-1)}{2} \cdot \frac{(q-1)}{2} = \frac{(p-1)(q-1)}{4}$. □

Proposition 4.3. *Let p and q be two prime numbers congruent to 3 modulo 4. There exists a $b \in U(\mathbb{Z}/pq\mathbb{Z})$ such that $a = b^2$ if and only if $a^{\frac{(p-1)(q-1)}{4}} \equiv 1 \pmod{pq}$. Moreover, there exist Bezout coefficients u and v such that $up + vq = 1$. Then, we get (1).*

Proof
$\Rightarrow$ We have $a = b^2$, thus $a^{(p-1)(q-1)/4} = b^{(p-1)(q-1)/2} = 1$ according to a 4.2.
$\Leftarrow$ We have $a^{(p-1)(q-1)/2} = 1$, then $a^{(p-1)(q-1)/4} = 1$ is a root of the equation $x^2 = 1 \mod pq$. Hence, we have 4 cases.

$$\begin{cases} x = 1 \mod p \\ x = 1 \mod q \end{cases} \Rightarrow x = 1 \mod pq \qquad \begin{cases} x = 1 \mod p \\ x = -1 \mod q \end{cases} \Rightarrow x = vq - up \mod pq$$

$$\begin{cases} x = -1 \mod p \\ x = 1 \mod q \end{cases} \Rightarrow x = up - vq \mod pq \qquad \begin{cases} x = -1 \mod p \\ x = -1 \mod q \end{cases} \Rightarrow x = -1 \mod pq$$

□

Lemma 4.4. *Let p and q be prime numbers congruent to 3 modulo 4, and let u and v be the Bezout coefficients such that $up + vq = 1$. Consider the set $H = \{1, -1, -up + vq, up - vq\} \subseteq U(\mathbb{Z}/pq\mathbb{Z})$.*
We claim that H is a subgroup of $U(\mathbb{Z}/pq\mathbb{Z})$. Moreover, the map $f : U(\mathbb{Z}/pq\mathbb{Z}) \to \{1, -1, -up + vq, up - vq\}$ defined by $f(a) = a^{\frac{(p-1)(q-1)}{4}}$ is a group homomorphism.

Proof
Given that $up + vq = 1$, it follows that $up = 1 - vq$. Squaring both sides, we get:
$(up)^2 = up - (vq)\, up \mod pq = up$. Therefore, $(up)^2 = up$. Similarly, we have $(vq)^2 = vq$.
We now show that $\{1, -1, -up + vq, up - vq\}$ is a subgroup of $U(\mathbb{Z}/pq\mathbb{Z})$ by using the observation that this set is closed under multiplication and that each element is its own inverse.
Moreover, $f(xy) = (xy)^{\frac{(p-1)(q-1)}{4}} = x^{\frac{(p-1)(q-1)}{4}} y^{\frac{(p-1)(q-1)}{4}} = f(x)f(y)$.

□

5. Security of the L_{pq} symbols

Statistical attacks, such as linear and differential cryptanalysis, analyze the non-random propagation of patterns in the values processed by a cipher. These patterns may manifest as linear relations or differential relations between internal variables, which can be exploited to distinguish the cipher from a random permutation. To quantify a function's resistance to such attacks, two standard metrics are used: the maximum expected linear probability (MELP) and the maximum expected differential probability (MEDP). These measures respectively express the highest expected probability that a given linear or differential approximation holds, averaged over all possible keys.

In our setting, we study the security of the symbol L_{pq}, with p and q two prime numbers congruent to $3 \bmod 4$. By applying our analysis to linear approximations, we establish an analytical bound for the MELP associated with L_{pq}. This result quantifies the maximum probability with which a given linear approximation can hold, thus providing a direct measure of the resistance of L_{pq} to linear cryptanalysis.

Theorem 5.1. *Let p, q be two prime numbers $\equiv 3 \pmod 4$, and u, v such that $up + vq = 1$, then:*

$$MELP_{L_{pq}} = \max_{a,b,c} \mathbb{E}\left[\mathbb{P}_x\left[a \cdot x + b \cdot (L_{pq}(x)) = c\right]\right]$$

$$MELP_{L_{pq}} = \frac{1}{4} - \frac{p+q-1}{4pq}$$

Proof

We first show that the **MELP** reaches its maximum value when $a = 0$.
Let the function $f(x) = ax + b \cdot L_{pq}(x)$. We have:

$$f(x) = ax + b \cdot L_{pq}(x) = c \quad \Rightarrow \quad L_{pq}(x) = \frac{c - ax}{b} \in \{1, -1, vq - up, up - vq\}$$

This expression attains its maximum when $a = 0$.

In this case, we simply have:

$$b \cdot L_{pq}(x) = c \quad \Rightarrow \quad L_{pq}(x) = \frac{c}{b} \in \{1, -1, vq - up, up - vq\}$$

Since x is chosen uniformly at random from $U(Z_{pq})$, for any $\alpha \in \{1, -1, vq - up, up - vq\}$, we have:

$$|\{x \in U(Z_{pq}) \mid L_{pq}(x) = \alpha\}| = \frac{(p-1)(q-1)}{4} \qquad \text{(see 4.2)}$$

Therefore, the probability is:

$$\mathbb{P}_{x \in \mathbb{Z}_{pq}}\left[L_{pq}(x) = \alpha\right] = \frac{(p-1)(q-1)}{4pq} = \frac{1}{4} - \frac{p+q-1}{4pq}$$

□

Similarly, we analyze the differential resistance of the symbol L_{pq} by computing the MEDP, which measures the maximum probability that a given input difference produces a specific output difference. By adapting the structure of our MELP proof to the differential setting, we derive an exact expression for the MEDP of L_{pq}. This result complements our statistical analysis, allowing us to simultaneously characterize the security of L_{pq} against the two primary types of statistical attacks: linear and differential cryptanalysis.

Theorem 5.2. *Let p, q be two prime numbers such that $p \equiv q \equiv 3 \pmod 4$, and let u, v be integers such that $up + vq = 1$, then:*

$$\mathrm{MEDP}_{L_{pq}} = \max_{\Delta x, \Delta y} \mathbb{E}\left[\mathbb{P}_x\left[L_{pq}(x + \Delta x) - L_{pq}(x) = \Delta y\right]\right].$$

$$\mathrm{MEDP}_{L_{pq}} = \frac{1}{4} - \frac{p+q-1}{4pq}.$$

Proof

For all x, the value $L_{pq}(x)$ belongs to the set

$$V = \{-1, 1, vq - up, up - vq\}.$$

Since $L_{pq}(x)$ and $L_{pq}(x + \Delta x)$ are independent and uniformly distributed over V, the possible values of

$$\Delta y = L_{pq}(x + \Delta x) - L_{pq}(x)$$

are

$$V' = \{0, \pm 2, \pm 2up, \pm 2vq, \pm(u - v)(p + q)\}.$$

For each $\delta \in V'$, we define

$$N_\delta = \left|\{(a, b) \in V^2 \mid b - a = \delta\}\right|.$$

We then obtain:

- For $\delta = 0$, the possible pairs are $(1, 1), (-1, -1), (vq - up, vq - up), (up - vq, up - vq)$, hence $N_0 = 4$.
- For $\delta = \pm 2$, the pairs are $(-1, 1)$ and $(1, -1)$, hence $N_2 = N_{-2} = 1$.
- For $\delta = \pm 2up$:
$$\begin{cases} 2up : (-1, up - vq), (vq - up, 1), \\ -2up : (1, vq - up), (up - vq, -1), \end{cases}$$
hence $N_{2up} = N_{-2up} = 2$.
- For $\delta = \pm 2vq$:
$$\begin{cases} 2vq : (-1, vq - up), (up - vq, 1), \\ -2vq : (vq - up, -1), (1, up - vq), \end{cases}$$
hence $N_{2vq} = N_{-2vq} = 2$.
- For $\delta = \pm(u - v)(p + q)$, we have $(vq - up, up - vq)$ and $(up - vq, vq - up)$, hence $N_{(u-v)(p+q)} = N_{-(u-v)(p+q)} = 1$.

Since x is sampled from $\mathbb{Z}_{pq}$ and non-units do not contribute, we multiply by the proportion of units $\frac{(p-1)(q-1)}{pq}$. Thus:

$$\mathbb{P}[L_{pq}(x + \Delta x) - L_{pq}(x) = \delta] = \frac{(p - 1)(q - 1)}{pq} \cdot \frac{N_\delta}{16}.$$

The maximum value is reached for $\Delta y = 0$ since $N_0 = 4$. Therefore:

$$\mathrm{MEDP}(L_{pq}) = \frac{(p - 1)(q - 1)}{pq} \cdot \frac{4}{16} = \frac{(p - 1)(q - 1)}{4pq} = \frac{1}{4} - \frac{p + q - 1}{4pq}.$$

□

6. Security of the S-Box Function

This section establishes explicit formulas for the linearity probability (MELP) and the differential probability (MEDP) of $f(x) = x^d L_{pq}(x)$, under the assumptions $p, q \equiv 3 \pmod 4$ and $\gcd(d, (p - 1)(q - 1)) = 1$. Leveraging the Chinese Remainder Theorem (CRT) factorization of $(\mathbb{Z}/pq\mathbb{Z})^\times$ and a zero-counting argument via

the Schwartz–Zippel lemma over $\mathbb{F}_p^\times$ and $\mathbb{F}_q^\times$, we obtain

$$\mathrm{MELP}_f = \frac{4d^2}{(p-1)(q-1)} \quad \text{and} \quad \mathrm{MEDP}_f = \frac{(4d-2)^2}{(p-1)(q-1)}.$$

These formulas make explicit the dependence on d and on the domain size, and provide parameter-selection guidelines to calibrate pseudo-linearity and differential resistance.

Lemma 6.1 (Schwartz–Zippel [12],[16])**.** *Let $\mathbb{F}$ be a field and let $P \in \mathbb{F}[X_1, \ldots, X_n]$ be a nonzero polynomial of degree $\leq d$. Let $S \subseteq \mathbb{F}$ be a finite subset and let $v_1, \ldots, v_n \in S$ be chosen uniformly and independently. Then:*

$$\mathbb{P}\big[P(v_1, \ldots, v_n) = 0\big] \leq \frac{d}{|S|}.$$

Lemma 6.2. *Let p, q be two distinct primes, $U = U(\mathbb{Z}/pq\mathbb{Z})$. Let $g \in \mathbb{Z}/pq\mathbb{Z}[x_1, \ldots, x_m]$ be **nonzero**. Denote by g_p, g_q its reductions in $\mathbb{F}_p[x_1, \ldots, x_m]$ and $\mathbb{F}_q[x_1, \ldots, x_m]$, with total degrees $d_p = \deg g_p$ and $d_q = \deg g_q$ (with $d_p, d_q \leq \deg g$). If $x = (x_1, \ldots, x_m)$ is drawn uniformly and independently from U^m, then*

$$\mathbb{P}\big[g(x) \equiv 0 \pmod{pq}\big] \leq \begin{cases} \dfrac{d_p}{p-1} \cdot \dfrac{d_q}{q-1}, & \text{if } g_p \not\equiv 0 \text{ and } g_q \not\equiv 0, \\ \dfrac{d_q}{q-1}, & \text{if } g_p \equiv 0 \text{ and } g_q \not\equiv 0, \\ \dfrac{d_p}{p-1}, & \text{if } g_q \equiv 0 \text{ and } g_p \not\equiv 0. \end{cases}$$

Proof
By the Chinese Remainder Theorem,

$$\Pi : U = \xrightarrow{\sim} \mathbb{F}_p^\times \times \mathbb{F}_q^\times.$$

In particular, for $m \geq 1$,

$$\Pi^m : U^m \longrightarrow (\mathbb{F}_p^\times)^m \times (\mathbb{F}_q^\times)^m, \qquad x \longmapsto (u, v) = (\pi_p(x), \pi_q(x)).$$

Drawing x independently and identically distributed from U^m, we obtain by the above isomorphism (coordinate-wise). Since x is drawn uniformly and independently from U^m, then, by the isomorphism above, u and v *are as well* over $(\mathbb{F}_p^\times)^m$ and $(\mathbb{F}_q^\times)^m$.

Let g_p and g_q be the reductions of g in $\mathbb{F}_p[X_1, \ldots, X_m]$ and $\mathbb{F}_q[X_1, \ldots, X_m]$; we have

$$g(x) \equiv 0 \pmod{pq} \iff g_p(u) = 0 \text{ in } \mathbb{F}_p \text{ and } g_q(v) = 0 \text{ in } \mathbb{F}_q.$$

Since u and v are drawn independently, the events $\{g_p(u) = 0\}$ and $\{g_q(v) = 0\}$ are independent. Thus

$$\mathbb{P}\big[g(x) \equiv 0 \pmod{pq}\big] = \mathbb{P}\big[g_p(u) = 0\big] \cdot \mathbb{P}\big[g_q(v) = 0\big]. \tag{$*$}$$

If $g_p = 0$, then $\mathbb{P}\big[g_p(u) = 0\big] = 1$ (resp. for g_q).

Applying Schwartz–Zippel on $(\mathbb{F}_p^\times)^m$ and $(\mathbb{F}_q^\times)^m$ of respective sizes $p-1$ and $q-1$, we obtain:

- if $g_p \neq 0$ and $g_q \neq 0$, then
$$\mathbb{P}\big[g(x) \equiv 0 \pmod{pq}\big] \leq \frac{d_p}{p-1} \cdot \frac{d_q}{q-1};$$

- if $g_p = 0$ and $g_q \neq 0$, then

$$\mathbb{P}\big[g(x) \equiv 0 \pmod{pq}\big] \leq \frac{d_q}{q-1};$$

- if $g_p \neq 0$ and $g_q = 0$, then

$$\mathbb{P}\big[g(x) \equiv 0 \pmod{pq}\big] \leq \frac{d_p}{p-1}.$$

□

Theorem 6.3. *Let p, q be two primes $\equiv 3 \pmod 4$, and let u, v be such that $up + vq = 1$. Let $f(x) = x^d \cdot L_{pq}(x)$ with $\gcd(d, (p-1)(q-1)) = 1$. Then:*

$$\textit{MELP}_f = \max_{a,b,c} \mathbb{E}\left[\mathbb{P}_x\left[a \cdot x + b \cdot \left(x^d \cdot L_{pq}(x)\right) = c\right]\right]$$

$$\textit{MELP}_f = \frac{4d^2}{(p-1)(q-1)}$$

Proof
We have

$$f(x) = x^d \cdot L_{pq}(x) \;=\; x^d \cdot x^{\frac{(p-1)(q-1)}{4}} \;=\; x^{\,d+\frac{(p-1)(q-1)}{4}} \quad \text{(see 4.3)}.$$

$$\begin{aligned}
a \cdot x \;+\; b \cdot f(x) = c \;&\Longleftrightarrow\; a \cdot x \;+\; b \cdot x^{\,d+\frac{(p-1)(q-1)}{4}} \;=\; c \\
&\Longleftrightarrow\; b \cdot x^{\,d+\frac{(p-1)(q-1)}{4}} \;=\; c - a \cdot x \\
&\Longrightarrow\; b^2 x^{\,2d+\frac{(p-1)(q-1)}{2}} \;=\; c^2 - 2acx + a^2x^2 \quad \text{(by 4.2)} \\
&\Longleftrightarrow\; b^2x^{2d} \;-\; c^2 \;-\; a^2x^2 \;+\; 2acx \;=\; 0\,.
\end{aligned}$$

For all a, b, c, this is a polynomial in x of degree $2d$. By Lemma 6.2,

$$\mathrm{MELP}_f \leq \frac{4d^2}{(p-1)(q-1)}.$$

□

Theorem 6.4. *Let p, q be two primes $\equiv 3 \pmod 4$, and let u, v satisfy $up + vq = 1$. Let $f(x) = x^d \cdot L_{pq}(x)$ with $\gcd(d, (p-1)(q-1)) = 1$. Then:*

$$\mathrm{MEDP}_f = \max_{\Delta x, \Delta y} \mathbb{E}[\mathbb{P}_x\left[f(x + \Delta x) - f(x) = \Delta y\right]].$$

$$\mathrm{MEDP}_f = \frac{(4d-2)^2}{(p-1)(q-1)}$$

Proof

We have
$$\begin{aligned}
f(x+\Delta x)-f(x)=\Delta y &\iff (x+\Delta x)^{d+\frac{(p-1)(q-1)}{4}} - x^{d+\frac{(p-1)(q-1)}{4}} = \Delta y \\
&\iff (x+\Delta x)^{d+\frac{(p-1)(q-1)}{4}} = \Delta y + x^{d+\frac{(p-1)(q-1)}{4}} \\
&\Rightarrow (x+\Delta x)^{2d+\frac{(p-1)(q-1)}{2}} = (\Delta y)^2 + 2(\Delta y)x^{d+\frac{(p-1)(q-1)}{4}} + x^{2d+\frac{(p-1)(q-1)}{2}} \\
&\iff (x+\Delta x)^{2d} = (\Delta y)^2 + 2(\Delta y)x^{d+\frac{(p-1)(q-1)}{4}} + x^{2d} \quad \text{(by 4.2)} \\
&\iff (x+\Delta x)^{2d} - (\Delta y)^2 - x^{2d} = 2(\Delta y)x^{d+\frac{(p-1)(q-1)}{4}} \\
&\Rightarrow \left((x+\Delta x)^{2d} - (\Delta y)^2 - x^{2d}\right)^2 = 4(\Delta y)^2 x^{2d+\frac{(p-1)(q-1)}{2}} \\
&\iff ((x+\Delta x)^{2d} - (\Delta y)^2 - x^{2d})^2 - 4(\Delta y)^2 x^{2d} = 0 \quad \text{(by 4.2)}.
\end{aligned}$$

We now show that the degree in x of the polynomial is $4d-2$.

$$(x+\Delta x)^{2d} = \sum_{k=0}^{2d} \binom{2d}{k} x^{2d-k}(\Delta x)^k = x^{2d} + \sum_{k=1}^{2d} \binom{2d}{k} x^{2d-k}(\Delta x)^k.$$

Thus,
$$\begin{aligned}
(x+\Delta x)^{2d} - x^{2d} - \Delta y &= \sum_{k=1}^{2d} \binom{2d}{k} x^{2d-k}(\Delta x)^k - \Delta y \\
&= 2d\,(\Delta x)\,x^{2d-1} + \sum_{k=2}^{2d} \binom{2d}{k} (\Delta x)^k x^{2d-k} - \Delta y\,.
\end{aligned}$$

Hence, $(x+\Delta x)^{2d} - (\Delta y)^2 - x^{2d}$ has degree $2d-1$ in x, squaring it yields a polynomial of degree $4d-2$. By Lemma 6.2,
$$\mathrm{MEDP}_f \le \frac{(4d-2)^2}{(p-1)(q-1)}\,.$$

□

7. Comparative Analysis of the Classical and Generalized Models

To make the benefit of our approach explicit, Table 1 contrasts the classical construction with our CRT-based generalization at two levels.

Symbol level.
MEDP/MELP move from $\frac{1}{2} - \frac{1}{2p}$ to $\frac{1}{4} - \frac{p+q-1}{4pq}$: the central term is halved and a $1/(pq)$ correction appears.

S-box level.
The bounds shift from $\mathrm{MEDP}_f \le \frac{4\alpha-2}{p}$ *and* $\mathrm{MELP}_f \le \frac{2\alpha}{p}$ *to* $\mathrm{MEDP}_f \le \frac{(4d-2)^2}{(p-1)(q-1)}$ *and* $\mathrm{MELP}_f \le \frac{4d^2}{(p-1)(q-1)}$.

Consequences.
(i) dual-modulus balancing: worst-case bias improves from order $1/p$ to order $1/(pq)$
(ii) stronger diffusion via cross-modulus interaction

(iii) higher nonlinearity through the tunable parameter d
(iv) tighter analytic bounds, hence more predictable guarantees. These shifts explain the generalized design's systematic advantage at both levels.

Table 1. Comparison of MEDP/MELP at symbol level and S-box level

	Classical (over $\mathbb{F}_p$)	**Generalized (mod pq)**
Symbol level (Legendre vs. L_{pq})		
Domain	$\mathbb{F}_p$	$\mathbb{F}_p \times \mathbb{F}_q$
Symbol	$\left(\frac{x}{p}\right)$	$L_{pq}(x)$
MEDP (symbol)	$\frac{1}{2} - \frac{1}{2p}$	$\frac{1}{4} - \frac{p+q-1}{4pq}$
MELP (symbol)	$\frac{1}{2} - \frac{1}{2p}$	$\frac{1}{4} - \frac{p+q-1}{4pq}$
S-box level		
Construction	From quadratic residues mod p	CRT combination of residues mod p and q
MEDP$_f$ (S-box)	$\frac{4\alpha - 2}{p}$	$\frac{(4d-2)^2}{(p-1)(q-1)}$
MELP$_f$ (S-box)	$\frac{2\alpha}{p}$	$\frac{4d^2}{(p-1)(q-1)}$
Qualitative	Single-modulus balance; residual bias $\sim 1/p$	Dual-modulus balance; reduced bias $\sim 1/(pq)$

8. A Step-by-Step Example

8.1. Inputs

The hash function takes as main parameters:

- two prime numbers p, q such that $p \equiv q \equiv 3 \pmod 4$;
- the size of the internal state m and the number of words absorbed per round r ;
- the number of rounds n ;
- the parameter constants λ and d ;
- the desired output length (by default 256 bits) ;
- a message $M \in \{0,1\}^\star$, padded to comply with the sponge interface.

Example of parameters.

$$p = 239, \quad q = 271, \quad m = 8, \quad r = 4, \quad n = 6, \quad \lambda = 12, \quad d = 5.$$

Message.

$$M = \texttt{"crypto"}.$$

8.2. Internal steps

The construction relies on an arithmetic-and-linear permutation applied n times :

- **MDS matrices.** For each branch mod p and mod q, we build a matrix $M \in \mathbb{F}_p^{m\times m}$ (resp. $\mathbb{F}_q^{m\times m}$) from a primitive root of order $p-1$ (resp. $q-1$). These matrices ensure diffusion among the m words of the state.

For $p = 239$:

$$M_1 = \begin{bmatrix} 19 & 145 & 109 & 165 & 52 & 84 & 182 & 140 \\ 31 & 4 & 109 & 26 & 36 & 101 & 230 & 184 \\ 150 & 182 & 223 & 116 & 34 & 196 & 129 & 178 \\ 36 & 148 & 225 & 196 & 51 & 168 & 88 & 193 \\ 82 & 58 & 153 & 44 & 194 & 11 & 161 & 101 \\ 7 & 148 & 73 & 88 & 38 & 74 & 229 & 200 \\ 215 & 88 & 199 & 91 & 211 & 108 & 146 & 27 \\ 35 & 67 & 163 & 113 & 61 & 89 & 3 & 102 \end{bmatrix}$$

For $q = 271$:

$$M_2 = \begin{bmatrix} 30 & 110 & 13 & 250 & 174 & 113 & 232 & 111 \\ 78 & 45 & 198 & 121 & 52 & 251 & 120 & 87 \\ 171 & 163 & 92 & 268 & 83 & 127 & 110 & 21 \\ 88 & 42 & 165 & 193 & 128 & 17 & 121 & 2 \\ 60 & 37 & 68 & 123 & 270 & 83 & 210 & 72 \\ 263 & 121 & 160 & 182 & 185 & 5 & 256 & 72 \\ 263 & 53 & 244 & 3 & 244 & 191 & 178 & 118 \\ 17 & 235 & 232 & 205 & 210 & 28 & 196 & 268 \end{bmatrix}$$

- **Round constants.** They are generated deterministically by SHAKE256 from the seed

  $$\texttt{Grendel-Modified-}(p, q, m, \lambda).$$

  We obtain $n \times m$ constants; each row below corresponds to the m constants injected in the corresponding round:

42701	30871	61978	32920	21679	36767	11808	46089
46200	51515	14583	23964	11334	6618	11730	41137
63228	51244	41263	13936	42301	26721	1811	32943
15408	64456	60287	17460	16573	12913	36389	45306
27528	28637	10254	38566	42113	4524	24758	51088
25880	37303	44133	30950	28689	27983	41900	35303

- **Permutation (per round).** At each round:
  1. branch-wise S-box ;
  2. linear mixing via M_1 and M_2 ;
  3. nonlinear mixing with exponents coprime to $p - 1$ and $q - 1$;
  4. addition of the round constants.

  The two branches are recombined via the (CRT).

8.3. Outputs

After complete absorption of the message and the squeezing phase, the final digest is obtained.

Hexadecimal output (256 bits):

```
aa675b2bb1f87f52b95b3700979d631927c954e8dfdabab6242371d9f1f30508
```

9. Experimental Results

Objective. We empirically evaluate the statistical quality and diffusion of GRENDEL_MODIFIED_HASH. The tests target: (i) **output uniformity**, (ii) **strict avalanche** (sensitivity of each output bit to flipping a single input

bit), and (iii) **near-independence** of output bits. We also measure a **Goodness-of-Fit** (GOF) of Hamming distances to the binomial law, as well as a **random-avalanche** test (uniform flips).

Parameters and protocol. Unless otherwise stated: `MSG_LEN` $= 16$ bytes (128 bits), output $= 256$ bits.

- **Uniformity (UNI).** 10,000 independent outputs. For each bit j, we estimate the bias $\widehat{p}_j - 0.5$ and the two-sided 95% CI:
$$\widehat{p}_j \pm 1.96\,\sqrt{\widehat{p}_j(1-\widehat{p}_j)/n}, \qquad n = 10{,}000,$$
i.e., ± 0.0098 at $p = 0.5$.
- **SAC.** A $256 \times m$ matrix with m tested input bits (typically $m = 32$). For each input i, we draw 10,000 pairs $(m,\, m \oplus e_i)$ and estimate $s_{j,i}$ (the flip rate of output bit j). We report: global mean, row/column means, worst deviations (cell/row/column), and per-cell CIs.
- **BIC.** We sample $m = 128$ input bits, use $n = 2{,}000$ trials, and evaluate 32,640 pairs (j, k) of output bits per trial (about 4.18 million correlations in total). For each $(i; j, k)$, we compute Pearson's correlation $r_{i;j,k}$ between flip vectors. We publish the absolute mean $\overline{|r|}$ and the absolute maximum.
- **GOF (Hamming). Over** 10,000 **trials, we compare independent outputs. Theoretical targets (256 bits): mean** $= 128$**, standard deviation** $= 8$**. We report** $(\bar{H}, s_H)$ **and a** χ^2 **(or** z**) statistic against** $\mathrm{Bin}(256, 0.5)$**.**
- **Random avalanche.** Same as GOF, but between $H(m)$ and $H(m \oplus r)$ with $r \sim \mathrm{Bernoulli}(0.5)$.

Statistical expectations (quick read).

- **UNI.** Over 256 bits and 10,000 trials, about $5\% \times 256 \approx 13$ bits outside the CI are expected (multiple-comparisons effect). A worst bias between 0.01 and 0.02 is compatible with randomness.
- **SAC.** Row/column means near 0.5 (typically $[0.49,\, 0.51]$ at $n = 10{,}000$). The worst cell may deviate by 0.015–0.02 given the total number of cells (e.g., $256 \times 32 = 8{,}192$).
- **BIC.** At $n = 2{,}000$, the natural scale of sampling correlations is $O(n^{-1/2}) \approx 0.022$; the expectation for $\overline{|r|}$ is $\sqrt{2/\pi}/\sqrt{n} \approx 0.0179$. With millions of pairs, a maximum around 0.10–0.12 is common (extreme-value effect).
- **GOF/Random avalanche.** $(\bar{H}, s_H)$ close to $(128, 8)$; a moderate, non-significant χ^2/z confirms conformity.

Reproducibility. We publish the seeds (`seed`), the sampled input-bit subsets (SAC/BIC), the sample sizes (`TRIALS`, `PAIRS`), and the code versions. Cryptographic parameters $(p, q, m\_state, r\_words, n\_rounds, \lambda, d)$ and the implementation are fixed for the entire campaign. The scripts produce tables with CIs and summaries to enable exact replication.

How to read the tables 2. Each block (UNI, SAC, BIC, GOF, Avalanche) states the output size, the number of trials, the parameters (seeds, subsets, pairs), and the results (biases, means/variances, CIs, correlations, extremes, χ^2/z). Occasional breaches of per-bit CIs are expected when many tests run in parallel and do not, by themselves, constitute an anomaly.

10. Application to blockchain

10.1. Digital signature

A digital signature serves as a highly secure method for confirming the genuineness and unaltered state of electronic messages and files. In this groundbreaking presentation regarding blockchain signatures, the initial step involves

Table 2. Statistical tests for GRENDEL_MODIFIED_HASH (256-bit output)..

Block / Metric	Size	Trials	Parameters	Result
Uniformity (UNI)				
samples, Output bits	256	10 000	MSG_LEN=16;out_bits=256;TRIALS=10 000; seed=42	
Worst per-bit bias	256	10 000	–	0.0157 ($IC_{95} \approx 0.0098$)
Bits outside the CI_{95}	256	10 000	–	13
χ^2_z (bytes.)	256	10 000	–	-0.096
Strict Avalanche Criterion (SAC)				
Config / Inputs	$128 \rightarrow 256$	10 000	BITS_TO_TEST=32; seed=42	
global mean	256	10 000	MSG_LEN=16;out_bits=256;TRIALS=10 000	0.500001
row mean (min..max)	256	10 000	–	0.499 .. 0.501
column mean (min..max)	256	10 000	–	0.498 .. 0.503
worst cell deviation	256	10 000	–	0.020
worst row deviation	256	10 000	–	0.001
worst column deviation	256	10 000	–	0.003
IC_{95} Per-cell	256	10 000	–	$\approx 0.5 \pm 0.010$
Bit Independence Criterion (BIC)				
Config / Inputs	$128 \rightarrow 256$	2 000	BITS_TO_TEST=128; PAIRS=32 640; seed=4	
Evaluated pairs	256	2 000	(= 32 640 × 128)	4 177 920
Mean absolute correlation	256	2 000	–	0.017839
Maximum absolute correlation	256	2 000	–	0.114233 (i=79, j=25, k=222)
Goodness-of-Fit (GOF) Hamming				
Output bits	256	10 000	MSG_LEN=16;out_bits=256;TRIALS=10 000; seed=42	
Mean Hamming distance	256	10 000	–	128.0648
Hamming standard deviation	256	10 000	–	8.2127
χ^2, ddl, z_{norm}	256	10 000	–	61.4768, 50, $+1.1477$
Avalanche (bit-flip aléatoire)				
Output bits	256	10 000	MSG_LEN=16;out_bits=256;TRIALS=10 000; seed=42	
Mean Hamming distance	256	10 000	–	128.0405 (norm. 0.500158)
Hamming standard deviation	256	10 000	–	8.0185 (norm. 0.031323)
Min / Max Hamming distance	256	10 000	–	99 / 159

the computation of a hash value for the specified content utilizing the cutting-edge Grendel hashing algorithm. This resulting hash serves as a distinctive digital representation of the document, subsequently undergoing computing a digital signature with the private key of the signer. Following this encryption process, the digital signature is affixed to the original message or file. Upon the recipient's reception of the message, utilization of the signer's public key becomes necessary to verify the signature and authenticate its alignment with the hash of the content received, as depicted in Figure 3. This crucial authentication process guarantees that the document has remained unaltered since its signing and confirms the true identity of the sender. The foundation of digital signature security is firmly rooted in robust cryptographic principles, notably the inclusion of dependable hash functions, which play a pivotal role in upholding the integrity and authenticity of electronic communications.

Blockchain clarifications. We clarify the *threat model*: our proposal targets cryptographic robustness (preimage, second-preimage, and collision resistance) and SUF-CMA security of the signature with **context binding** (tag, chainID, epoch, header/Merkle root), without claiming protection against consensus-level attacks (51 %, selfish mining), which belong to the protocol layer [2]. The *signature workflow* is completed: key derivation via a KDF,

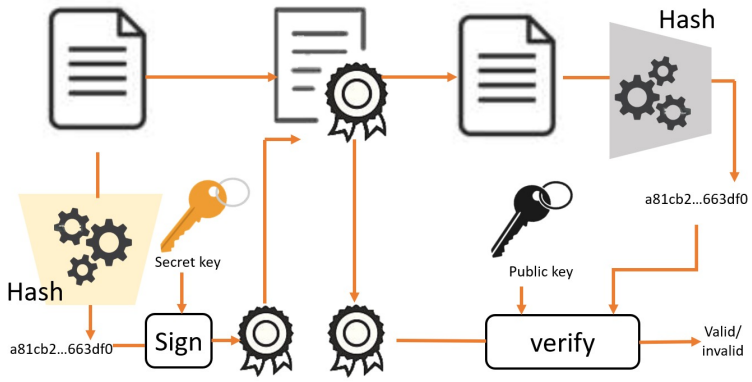

Figure 3. Our Grendel signature recommended process

deterministic nonce in the style of RFC 6979, **domain separation** in Grendel Modified Hash, and signing of the block **header** to limit propagation latency. On the system side, the storage footprint remains unchanged and verification is parallelizable, which bounds the impact on block propagation.

10.2. Signing a block

As illustrated in Figure 4, the blockchain, originally conceptualized as a sequential series of blocks, sets itself apart by its systematic inclusion of transactional information, organized within consecutive blocks. Each block contains a link to the previous one through a cryptographic hash function, consequently upholding the overall integrity of the entire chain. It is of paramount importance to emphasize a specific strategy that involves employing a digital signature based on the Grendel hash algorithm. This particular signature mechanism, which is deeply rooted in a distinct hashing algorithm, enhances the security measures by ensuring both the genuineness and unchangeable nature of the data. Consequently, the integration of this particular method reinforces the resilience of the blockchain infrastructure by not only guaranteeing the confidentiality and integrity of the transactions, but also ensuring non-repudiation. This multifaceted approach underscores the significance of employing advanced cryptographic techniques to fortify the security and reliability of the blockchain system.

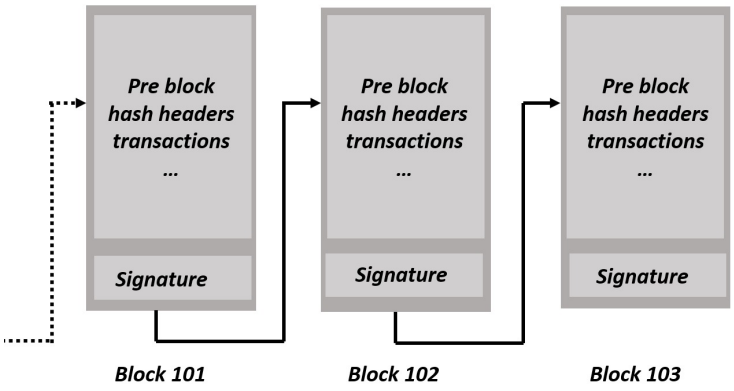

Figure 4. Our Grendel recommended signing process to blocks

11. Conclusion

In conclusion, the study at hand provides a comprehensive overview of a pioneering sponge-based function that has been developed with the specific goal of operating independently of hashing libraries. Through a careful examination of the Keccak techniques utilized in the SHA-3 algorithm and the integration of our unique Grendel-based permutation strategy, we have effectively deployed this function within a blockchain signature framework. The innovative combination of the Grendel permutation and Legendre symbols represents a significant advancement in cryptographic protocols, with a primary focus on enhancing both the security and efficiency of such systems. Our research outcomes strongly indicate that this novel approach holds considerable promise for driving future advancements in blockchain technology, particularly in the realms of secure digital signatures and the validation processes for blockchain blocks. Our findings from result 1 to result 4, were our main theoretical contributions here, in addition to the introduction of an extended symbol instead of the Legendre one. As for our experimental contribution, it has been presented through the novel idea to integrate our Grendel hash developed process into the blockchain digital signature and which showed a promising potential that could reinforce the security of blockchain technology.

Acknowledgement

We would like to thank all editors of the SOIC journal and their anonymous referees as well as members of the organizating committee of CSAIA'2024 for their valuable advice and support during the submission period of this project. We are especially grateful to our professors for their guidance and insightful advice. Lastly, we deeply appreciate our families for their unwavering support.

REFERENCES

1. Abdelalim, S., Lkoiaza, A., Cherkaoui, A., Elmouki, I., and Abghour, N. A Python Programming Initiative for Hash Construction through the Example of SHA-2. *Finite Abelian Groups, Elliptic Curves, Blockchain With Hashing and Graphs* (2025), 264–278. .
2. Abdelalim, S., Cherkaoui, A., Lkoiaza, A., Elmouki, I., and Abghour, N. Advancing Blockchain Security Using Graph Theory: A Python Programming Perspective. *Finite Abelian Groups, Elliptic Curves, Blockchain With Hashing and Graphs* (2025), 279–293. .
3. Albrecht, M.R., Grassi, L., Rechberger, C., Roy, A., Tiessen, T. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: Cheon, J.H., Takagi, T. (eds.) *Advances in Cryptology – ASIACRYPT 2016*, LNCS 10031, pp. 191–219, Springer (2016).
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G. The KECCAK SHA-3 submission, Version 3 (Jan. 2011).
5. Gupta, S., Sadoghi, M. Blockchain transaction processing. *arXiv*:2107.11592 (2021).
6. Kamal, Z.A., Fareed, R. A proposed hash algorithm to use for blockchain-based transaction flow system. *Periodicals of Engineering and Natural Sciences* 9(4), 657–673 (2021).
7. Mattoussi, F., Roca, V., Sayadi, B. Complexity comparison of the use of Vandermonde versus Hankel matrices to build systematic MDS Reed–Solomon codes. In: *2012 IEEE 13th Int. Workshop on SPAWC*, pp. 344–348. IEEE (2012).
8. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system (2008).
9. Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press (2016).
10. Parmar, M., Kaur, H.J. Comparative analysis of secured hash algorithms for blockchain technology and Internet of Things. *International Journal of Advanced Computer Science and Applications* 12(3) (2021).
11. Sauer, J.F., Szepieniec, A. SoK: Gröbner basis algorithms for arithmetization-oriented ciphers. *Cryptology ePrint Archive*, Report 2021/870 (2021).
12. Schwartz, J.T. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM* 27(4), 701–717 (1980).
13. Szepieniec, A. On the use of the Legendre symbol in symmetric cipher design. *Cryptology ePrint Archive* (2021).
14. Szepieniec, A., Ashur, T., Dhooghe, S. Rescue-Prime: A standard specification (SoK). *IACR Cryptology ePrint Archive* 2020/1143 (2020).
15. Zhang, R., Xue, R., Liu, L. Security and privacy on blockchain. *ACM Computing Surveys (CSUR)* 52(3), 1–34 (2019).
16. Zippel, R. Probabilistic algorithms for sparse polynomials. In: Ng, E.W. (ed.), *Symbolic and Algebraic Computation, EUROSAM '79*, LNCS 72, pp. 216–226. Springer (1979).