IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 17, 2021, accepted December 1, 2021, date of publication December 6, 2021, date of current version December 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3133097

Security Analysis of LNMNT-LightWeight Crypto Hash Function for IoT

NUBILA NABEEL, MOHAMED HADI HABAEBI, (Senior Member, IEEE), AND M. D. RAFIQUL ISLAM, (Senior Member, IEEE)
IoT and Wireless Communication Protocol Laboratory, Department of Electrical and Computer Engineering, International Islamic University Malaysia (IIUM), Kuala Lumpur 53100, Malaysia

Corresponding author: Mohamed Hadi Habaebi (habaebi@iium.edu.my)

This work was supported by International Islamic University Malaysia, Publication-Research Initiative Grant Scheme P-RIGS18-003-0003.

ABSTRACT Conventional cryptographic techniques are inappropriate for resource-constrained applications in the Internet of Things (IoT) domain because of their high resources requirement. This paper introduces a new Lightweight (LWT) hash function termed Lightweight New Mersenne Number Transform (LNMNT) Hash function, suitable for many IoT applications. The proposed LWT hash function is evaluated in terms of randomness, confusion, diffusion, distribution of hash function, and different attacks. The randomness analysis is performed using the NIST test suit. The LNMNT LWT hash function has been benchmarked against other LWT hash functions in terms of execution time, cycles per byte, memory usage, and consumed energy. The analysis showed that the LNMNT LWT hash function has excellent randomness behavior and is highly sensitive to the slight change in the input message too. Moreover, it provides low execution time, memory usage, and power consumption against the other LWT hash functions.

INDEX TERMS IoT security, lightweight hash functions, new Mersenne number transform, NIST test suite.

I. INTRODUCTION

A. IoT OVERVIEW

Today's networking concept enables the interconnection of millions of devices that humans use daily and create a smart environment everywhere [1], [2]. In all sectors with human interaction, like health, military, transportation, industry, and cities, IoT can create a smart environment. This can be reached by making devices mutually connected via the Internet. Further, it makes our tools and equipment smart too. The Internet of Things architecture [3] is made up of four layers. The architecture is shown in Fig.1. The first layer is the device layer, which consists of sensors, actuators, RFIDs, and gateways and these are mostly resource-constrained devices in nature. Gateways collect the data from the IoT devices for further processing, while the network layer provides transport and networking capabilities for routing the data. The third layer is a middle layer that hides the complexity of lower layers from the higher layers and provides storage and further processing of data. The Application layer houses different IoT applications such as smart city, smart industry and smart health. IoT applications must be designed to provide security, privacy, transparency, and subject to keep ethical constraints [4].

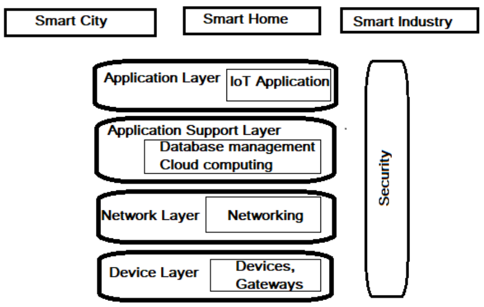


FIGURE 1. Simulation results for the network.

Some examples of theoretical and experimental attacks to highlight the weakness of more efficient but less secure designs of IoT applications are discussed in [5] and [6]. In [5] crypt-analysis on Photon using cube attack is conducted and found that it is possible to recover more bits of plaintext from the hash digest. In photon key-recovery-based attacks remains an open issue. In [6] we can find a comparative

The associate editor coordinating the review of this manuscript and approving it for publication was Chunsheng Zhu.

study about the different hash function based on speed, power requirements and security. Each LWT hash function is have its own advantages and disadvantages. None of the LWT hash function solves conflict between resource requirements and security. This leads to the requirement of design of new LWT hash function.

The security module provides necessary security features for each layer. The main cryptographic mechanisms are designed to ensure the following properties, such as data integrity, data confidentiality, authorized access, and access control. For this many crypto solutions like data encryption and generation of digital signatures are developed. Data encryption is done by using AES and DES etc. Data integrity is ensured by using cryptographic hash functions SHA and MD5 etc. The device layer of IoT consists of resource constrained devices. To implement a security mechanism in the device layer, it requires LWT cryptographic solutions [7], [8]. Memory utilization, power consumption, and processing speed are the most important elements to consider while building LWT crypto solution. Conventional cryptographic approaches, such as SHA and AES, for example, are impractical to use in such kinds of devices due to the high complexity of the techniques. RFID tags have a total gate count of 1000-10000, with only 200-2000 gates set for security. Most of the conventional cryptographic techniques need more than 10000 gate equivalents for their implementation [1] whereas all IoT devices should ensure privacy, confidentiality, availability, data integrity, and authorization.

B. LWT CRYPTOGRAPHY

The main challenges associated with conventional cryptographic techniques faced when implementing IoT applications are high memory requirement, high battery power consumption and large computational power. For encryption, LWT stream and block ciphers were introduced in the early on and for message authentication codes, pseudo random generators, and key derivation functions LWT hash functions have been developed. Cryptographic hash functions such as PHOTON [9], QUARK [10], Spongent [11] and Lesamanta-LWT [12] are defined as LWT hashing methods in ISO/IEC 29192-5 documentation, while PRESENT [13] and CLEFIA [14], [15] are used for block ciphers as per ISO/IEC 29192-2:2012 documentation. The LWT stream ciphers are TRIVIUM, GRAIN, CHACHA and ESPRESSO [16]. In this paper we have introduced a new adaptive LWT hash function.

PHOTON uses sponge construction architecture with AES like primitives as its basic building blocks. The sponge construction is a simple iterated construction for building a function f with variable length input and arbitrary output length. In sponge construction, a function f is repeatedly process in two phases, the Absorption and squeezing phase. The output from the absorption phase taken as the input to squeezing phase. Final out of sponge construction is taken from the squeezing phase. The sponge construction operates on a state of $b = br + cp$ bits. The value br is called the bitrate and the value cp the capacity. The message is divided into different

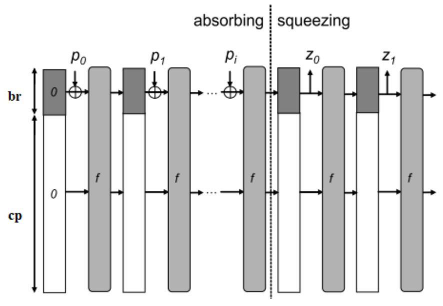


FIGURE 2. Sponge construction.

blocks $p_0, p_1, \ldots, p_n$ and takes the output as $z_1, z_2, \ldots z_m$. The sponge construction architecture diagram is shown in Fig.2. Different variants of PHOTON are available with 80, 128,160,224, and 256 bits hash lengths. To cope up with the LWT property, the internal state of the PHOTON is reduced by taking the 4 × 4 S box. An S-Box S is a $m \times n$ matrix which maps $S : \{0, 1\}^x \rightarrow \{0, 1\}^y$. that means it converts input message a^x to an output vector b^y in such a way that it hides the information about the plain text from the ciphertext.

We can achieve the LWT property this way, but we won't be able to achieve an appropriate level of security. A scan-based side-channel attack is a serious issue in small block size, small key size, and less complex round(4 bit S box) operation based techniques. In [17] we can see an attack in 128-bit AES key on an Arduino device using differential power analysis and correlation power analysis. The block length of a block cipher is an important parameter that directly affects security. AES has limited flexibility to support variable-length parameters like key size and block size, even though it offers three options for the number of rounds, namely: 10, 12, and 14. These parameters directly affect the security as well as the resource requirements of IoT devices.

QUARK uses the sponge construction method also. The permutation is implemented by non-linear Boolean functions and a linear Boolean function. The three classifications of QUARK hash family are uQUARK, dQUARK, and tQUARK. Permutation of QUARK uses two Nonlinear Feedback Shift Registers (NFSR), one Linear Feedback Shift Register (LFSR), and three Boolean functions f, g and h.

SPONGENT is a LWT hashing technique, and it also uses sponge construction. However, it uses a 4 × 4 bits S box with PRESENT permutation. Besides security and efficiency, the design of the S box mainly considers the simplicity of implementation. SPONGENT is available in two forms with 128 bit and 88 bit hash digest.

Lesamnta-LWT uses AES as its building method and it uses Merkle Damgard construction so that it requires more memory registers than sponge construction methods. This LWT hash function provides a 256-bit hash digest. Developers claim to have a security level of 2^{120} to collision, preimage, and second preimage resistance [12].

GLUON [18] is a family of a LWT hash functions based on sponge construction. This new family is based on Feedback

TABLE 1. List of symbols and abbreviations.

Symbols	Abbreviations
x(n)	set of n elements
X[k]	NMNT transform output of x(n)
mnp	Mersenne number prime
p	prime number
$\beta(nk)$	NMNT parameters
$\beta_1(nk)$	NMNT parameters
$\beta_2(nk)$	NMNT parameters
α_1	NMNT parameters
α_2	NMNT parameters
N_{max}	Maximum transform length

with Carry Shift Register (FCSR). FCSR is a promising alternative to LFSR for the design of stream ciphers. The FCSR consists of binary main register and carry register but contrary to LFSRs. LFSR performs XOR but FCSR performs addition with carry operation. FCSR can help to solve the problem which is always raised when using LFSR where in LFSR based stream ciphers, filtering or combining Boolean functions must be used to break the linearity of LFSR. With FCSR based stream ciphers, this issue is directly solved by the intrinsic non-linearity of the FCSR. The transition function of an LFSR is linear while it is quadratic for an FCSR. The main advantage of this quadratic transition function is the intrinsic resistance to algebraic attacks and to correlation attacks, which are the main weaknesses of LFSR based stream ciphers. However, the implementation of an FCSR costs more than the one of an LFSR. Like LFSR, FCSR is also not suitable to use directly for cryptography. It requires some filters to modify. The hardware size of such implementation is heavier than that of the basic building block used in QUARK and PHOTON.

SPN-Hash [19] is a new family of hash functions that gives the variable hash length of 128, 256, and 512 bits. It is constructed as resistance to the collision as well as resistance to common attacks against hash functions. The internal permutation is implemented as a Substitution Permutation Network (SPN). It uses AES-based internal permutations with fixed key sizes. One round of an SPN structure consists of three layers, they are Key addition, substitution, and linear transformation. The substitution layer is made up of S boxes implemented in parallel. SPN structure has good confusion and diffusion properties.

C. DESIGN COMPARISON OF DIFFERENT LWT HASH FUNCTIONS

The comparison of different LWT Hash functions in terms of design can be seen in Table 2 [7]. All LWT hash functions are created by weakening existing techniques, such as reducing the size of the S box, the key and block size, and the number of rounds. We can reduce resource requirements by using these strategies, but security becomes a challenge. the questions arising are how to minimize the number of blocks without compromising security, how to create a lighter S box or alternative approach without compromising security, and how to reduce block and key size without compromising security features.

II. CHALLENGES IN LWT HASH FUNCTIONS

A balancing of resource requirements and security is a challenging problem in LWT cryptographic design. The majority of LWT hashing approaches attempt to simplify existing cryptographic techniques. This is a bad practice because "lightweight" does not imply "weak" or "light" cryptography. LWT designs require the same or better level of security features than conventional cryptography.

Most of the LWT hash function family known so far managed to run on resource-constrained devices, but do have some issues in terms of security level achieved and throughput. Consequently, it is necessary to develop new approaches to hash function algorithm design that can prevent attacks effectively in comparison to existing algorithms as they are not sufficient to meet the requirement of the latest technologies and security concerns of IoT applications. These issues are always considered in the greater perspective of the ICT rather than the specific application domain of the IoT.

There are more than fifty symmetric LWT cryptographic algorithms proposed by various sectors. However, the design focus is only on how to reduce cost and enhance hardware and software performance. Furthermore, many of them do not properly consider mitigating security attacks [8]. The ideal LWT cryptographic technique should reduce the conflict and strike a balance between cost, performance, and security. To reach these three often contradicting properties all together is a challenging design problem. Security problems arise often when key or block sizes are reduced, and when algorithm design is simplified.

Making a hash function LWT by reducing the block size can be a serious concern against security because there is an increase in the probability of guessing the random IV by the attackers (e.g., the probability of recovering the plaintext block by block by using chosen plaintext attack increases). LWT does not mean weak cryptography, as we need more from less availability. Hence, weakening conventional cryptography is not a good practice to achieve the design goal of LWT cryptography and an alternative design approach should be considered. There are many possibilities of side-channel attacks, and it is vital to consider countermeasures at the implementation level.

Two main principles for secure cryptographic systems are diffusion and confusion property. Confusion tries to make complications between plaintext and ciphertext. Diffusion is the process in which the plaintext is rearranged into ciphertext. The strength of diffusion is measured by how the plaintext is rearranged into ciphertext. A small change in the plaintext should have a significant change in the ciphertext (e.g., the avalanche effect). The crypto system should have a good avalanche effect in such a way that half of the ciphertext should change for a single change in the plaintext. Currently, the AES algorithm uses Mix Column Transformation in its diffusion layer. Mix Column transformation is powerful for

TABLE 2. Comparison of different LWT hash functions in terms of design.

LWT hash functions	Techniques used and Contributions	Drawbacks
PHOTON	It uses Sponge construction. The s/w and h/w implementation is possible. It is Available in 5 different Hash length.	It uses mix column transform for diffusion property. These transformations are effective in diffusion, but they only support blocks of a fixed length. As a result, it takes a long time to hash a long data.
QUARK	It uses sponge construction. U-Quark provides 128bit preimage resistance and 64-bit collision resistance. D-Quark provides 160-bit preimage resistance and 80-bit collision resistance. T- Quark provides 224-bit preimage resistance and 112-bit collision resistance. It provides Good preimage and collision resistance.	In QK, Permutation is constructed by using two NFSR and one LFSR. The transformation function of LFSR is linear. So for breaking the linearity in LFSR, filters, or Boolean functions are needed. This in- curs extra cost and Quark is only optimized for the hardware. It takes a long time to hash the longer message.
SPONGENT	It uses sponge construction. It uses PRESENT permutation	It uses bit permutation for the linear diffusion layer. It also uses LFSR for the diffusion layer. However, the performance is somewhat similar to the QUARK LWT hash family and the same drawbacks persist.
GLUON	It uses sponge construction. Based on particular FCSR. Soft- ware and hardware performances of GLUON are worse than the ones of PHOTON, they are comparable when targeting hardware to the parallelized versions of Quark.	Hardware size of such implementation is quite heavier than that of the basic building block used in Quark and PHOTON.
Lesamnta-LWT	It uses Merkle-Damgard construction. It uses an AES primitives taking a 256-bit plaintext and a 128-bit key.	It uses mix-column transformation. The main problem with the MDS matrix is that it is fixed.

diffusion property. However, this transformation is fixed in terms of their length such that we cannot make a significant change in the algorithm to make it completely adequate for IoT applications. As a result, an alternate technique to meet the needs of resource-constrained IoT devices is the usage of parameter-based transformation, which permits changeable block length and key size by adjusting the transform length.

In this article, a new LWT hash function, LNMNT, that uses the New Mersenne Number transform (NMNT) in its diffusion layer, is described. The proposed LNMNT design builds on the sponge construction method to reduce the internal memory size as much as possible. Further, the article also includes the implementation details and the NIST test result of the new LWT hash function. LNMNT is not only LWT but it also achieves excellent diffusion and confusion property, making it quite attractive to a variety of IoT applications.

Since device layer of IoT applications consists of resource constrained devices, LWT crypto techniques are the only solutions for the security issues of these devices. The main contribution of this paper as follows.

1) The paper introduced a LWT crypto hash function LNMNT for IoT applications.
2) The proposed LWT hash function has been compared with existing LWT hash functions in terms of execution time, cycles per byte, memory usage, power consumption and collision resistance.
3) The proposed LWT hash function is evaluated in terms of randomness, confusion, diffusion, hash function distribution and different attacks. Further, the results were compared against other results for LWT hash functions.
4) The LNMNT LWT hash function has been further implemented in real Zigbee motes Z1 from Zolertia and tested in the lab to evaluate its capabilities.

III. NEW APPROACH TO LWT HASH

As previously stated, a new design approach is required for the construction of a LWT hash function, where simplification of existing techniques is not recommended. As a result, NMNT is introduced in the design of the LWT hash function. Table 1 shows the list of symbols and abbreviations used below.

A. NEW MERSENNE NUMBER TRANSFORM

The main motivation of this research is the adoption of the NMNT functions as a building block for the proposed LWT hash function [20]. Number Theoretic Transforms (NTT) are mainly adopted to improve the diffusion property of cryptographic techniques. The NMNT is an important NTT introduced by Holt and Boussakta, which can be used for fast error free calculation of convolutions and correlations [21].

Mersenne number is a number which is one less than a power of two. It can be represented as $M = 2^p - 1$ where p is an integer and M is the Mersenne number. An example for Mersenne number includes 1,3,7,15,31 etc. If M is prime then it is called Mersenne Prime Numbers.

NMNT can be defined as the modulo of a Mersenne number, where the arithmetic operations are simple equivalents to one's complement operation. The one-dimensional NMNT $X(k)$ of an integer sequence $x(n)$ with length N is defined in (1).

$$X(k) = \langle \sum_{n} x(n)\beta(nk) \rangle_{mnp}. \tag{1}$$

where mnp is the Mersenne prime in the form of $mnp = 2^{p-1}$, p is the prime number, $n = 0$ to $(N-1)$ and $k = 0$ to $N-1$

$$\beta(nk) = \beta_1(nk) + \beta_2(nk). \tag{2}$$
$$\beta_1(nk) = \langle Rel(\alpha_1 + j\alpha_2)^{nk}\rangle_{mnp}. \tag{3}$$
$$\beta_2(nk) = \langle Img(\alpha_1 + j\alpha_2)^{nk}\rangle_{mnp}. \tag{4}$$

The value of $\beta(nk)$ is in (2). $\beta_1(nk)$ and $\beta_2(nk)$ are showed in (3) and (4) respectively. Here $Rel()$ and $Img()$ indicates real and imaginary parts. The values of α_1 and α_2 can be obtained using (5) and (6).

$$\alpha_1(nk) = \pm\langle 2^q\rangle_{mnp}. \tag{5}$$
$$\alpha_2(nk) = \pm\langle -3^q\rangle_{mnp}. \tag{6}$$

where, $q = 2^{p-2}$

The value of NMNT parameters α_1, α_2 and $\beta(nk)$ depends upon the transform length. For the transform length less than $N = 2^{p+1}$ the value of $\beta_1(nk)$ and $\beta_2(nk)$ can be calculated as shown in (7) and (8).

$$\beta_1(nk) = \langle Rel((\alpha_1 + j\alpha_2)^i)^{nk}\rangle_{mnp}. \tag{7}$$
$$\beta_2(nk) = \langle Img((\alpha_1 + j\alpha_2)^i)^{nk}\rangle_{mnp}. \tag{8}$$

where i is an integer power of 2. The (7) and (8) is for the transform length N/i.

B. PROPERTIES OF NMNT

This section lists the motivation to select NMNT in the design of LWT hash.

1) We know that a LWT cryptosystem does not mean to be a weak cryptosystem. So that we should need an alternative method instead of weakening existing approaches. Moreover, we know that AES is hard to become LWT compactable due to its fixed transformation. The proposed technique uses NMNT, which provides variable transform length (power of two). This variable transform length can be utilized to create a hash function that has a high diffusion rate during the hashing process. NMNT has good diffusion and avalanche characteristics [22]. So that by designing the LWT hash function with NMNT, a good diffusion property is achieved.
2) NMNT provides variable transform length (power of two) so that we can design a hash function with variable block size according to its security requirements. Hence, there is no need to adjust the number of rounds; all you have to do is change the NMNT settings to make it variable length.
3) NMNT also provides the property of cyclic convolutions used for fast calculation of error-free convolution.
4) The key feature of NMNT is that it provides an extremely long transform length for a proper Mersenne Number prime (power of two), so that the LWT hash function can be designed with variable length hash digests. The proposed system gives the hash digest size of 80,128,160,224 and more and it is very flexible to change the length of the hash digest.

TABLE 3. Relationship between mnp, N and transform parameters.

Prime Number(p)	Mersenne Prime(mnp)	N_{max}	N	α_1 , α_2
5	31	64	32	$5, 10$
7	127	256	128	$5, 22$
13	8191	16384	8192	$5, 5381$
17	131071	262144	131072	$5, 94889$
19	524287	1048576	525288	$5, 46561$

The NMNT transform is mainly based on the NMNT parameters, which include α_1, α_2, β_1, β_2 and transform length. The value of each parameter has a direct impact on the length of the transform. We must first set the transform parameters before we can calculate NMNT. For calculating NMNT first we must set the transform parameters. To begin, select a prime number p from which to calculate the Mersenne prime mnp. Let's choose $p = 7$. Then $mnp = 2^7 - 1 = 127$ is the corresponding value. $N_{max} = 2^{p+1}$ is the maximum transform length. As a result, the maximum transform length for $p = 7$ is $2^{7+1} = 256$. The parameters can then be calculated. For $N_{max} = 256$(maximum transform length) α_1, and α_2 can be calculated as shown in (9), (10) and (11).

$$q = 2^{7-2} = 32. \tag{9}$$
$$\alpha_1 = \pm\langle 2^{32}\rangle = \pm 16. \tag{10}$$
$$\alpha_2 = \pm\langle -3^{32}\rangle = \pm 39. \tag{11}$$

The α_1 and α_2 pair can be written as $(16, 39)$, $(16, -39)$, $(-16, 39)$ and $(-16, -39)$. These values are related to maximum transform length. It will vary based on the transform length. For the pair $(16, -39)$, the value of α_1 and α_2 for the transform length 128, can be calculated as in (12) and (13).

$$\langle Rel(\alpha_1 + j\alpha_2)^i\rangle_{mnp} = \langle Rel(16 - j39)^2\rangle_{127} = 5. \tag{12}$$
$$\langle Img(\alpha_1 + j\alpha_2)^i\rangle_{mnp} = \langle Img(16 - j39)^2\rangle_{127} = 22. \tag{13}$$

Here $i = N_{max}/required transform length(N) = 256/128 = 2$. Table 3 gives the value of Mersenne prime and transform parameters. Table 4 gives the value of different transform parameters for different transform lengths with mnp 127. The same procedure can be used for the calculation of transform parameters of various mnp. Next we can calculate β_1(nk) and β_2(nk) as (14) and (15)

$$\beta_1(nk) = \langle Rel(5 + j22)^{nk}\rangle_{127}. \tag{14}$$
$$\beta_2(nk) = \langle Img(5 + j22)^{nk}\rangle_{127}. \tag{15}$$

Let us consider one numerical example from the NMNT calculation. Let $x(4) = [74, 65, 110, 91]$. First, we choose the mnp in such a way that it should be greater than any data in $x(4)$. Here $mnp = 127$. The input array contains 4 elements, so the transform length is 4. We can take the value of parameters as $\alpha_1 = 70$ and $\alpha_2 = 16$ from the Table 4. Next, we calculate the value of $\beta(nk)$using (2). $\beta(nk) = [1, 181, 118, 97, 1, 118, 181]$. Then using (1) we can calculate NMNT for the input as $x[4] = [74, 65, 110, 91]$.

TABLE 4. Transform parametes for mnp = 127.

N(transform length)	α_1	α_2
256	± 16	± 39
128	5	22
64	49	-34
32	102	-30
16	106	-24
8	-8	-8
4	70	16

The first value is

$$\begin{aligned}X[0] &= (x[0]\times\beta(0\times 0)+x[1]\times\beta(1\times 0)+x[2]\times\beta(2\times 0)\\ &\quad +x[3]\times\beta(3\times 0))mod127\\ &= (74\times 1+65\times 1+110\times 1+91\times 1)mod127\\ &= 340mod127\\ &= 86\end{aligned}$$

The second value is

$$\begin{aligned}X[1] &= (x[0]\times\beta(0\times 1)+x[1]\times\beta(1\times 1)+x[2]\times\beta(2\times 1)\\ &\quad +x[3]\times\beta(3\times 1))mod127\\ &= (74\times 1+65\times 181+110\times 118+91\times 97)mod127\\ &= 33646mod127\\ &= 118\end{aligned}$$

The third value is

$$\begin{aligned}X[2] &= (x[0]\times\beta(0\times 2)+x[1]\times\beta(1\times 2)+x[2]\times\beta(2\times 2)\\ &\quad +x[3]\times\beta(3\times 2))mod127\\ &= (74\times 1+65\times 118+110\times 1+91\times 181)mod127\\ &= 24325mod127\\ &= 68\end{aligned}$$

The fourth value is

$$\begin{aligned}X[3] &= (x[0]\times\beta(0\times 3)+x[1]\times\beta(1\times 3)+x[2]\times\beta(2\times 3)\\ &\quad +x[3]\times\beta(3\times 3))mod127\\ &= (74\times 1+65\times 97+110\times 118+91\times 181)mod127\\ &= 35830mod127\\ &= 16\end{aligned}$$

Finally NMNT output is given by $X[3] = [86, 118, 68, 16]$.

C. IMPLEMENTATION OF LNMNT

Direct implementation of NMNT using (1) requires a lot of multiplications. We aim to reduce the number of multiplication and hence make it suitable for LWT applications. The use of the radix algorithm [23] for the implementation of NMNT will achieve the fast realization of NMNT. The radix-4 algorithm decomposes the input sequence $x(n)$ into four equal sequences of length $N/4$, where N is the transform length. The DIF Radix-4 algorithm is used to implement the NMNT layer in the proposed hash function. The NMNT transform

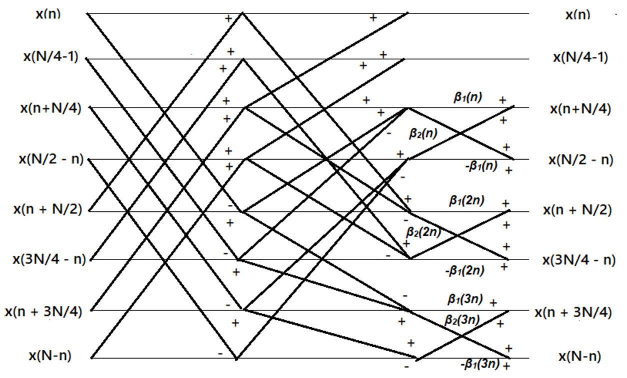


FIGURE 3. Butterfly diagram for Radix-4 NMNT.

for $x(n)$ can be represented as the sum of four output samples using the Radix-4 algorithm, as shown in (16).

$$X(k) = \langle X_0(k) + X_1(k) + X_2(k) + X_3(k)\rangle_{mnp}. \tag{16}$$

where $X_i(k)$ is represented by using (1), which is shown in (17).

$$X_i(k) = \langle \sum_{0}^{N/4-1} x(4n+i)\beta(4n+i)\rangle_{mnp}. \tag{17}$$

where $0 \leq i \leq 3$ Consider $X(k)$ as four samples, ie, $X(4k)$, $X(4k+1)$, $X(4k+2)$ and $X(4k+3)$. If we consider each term separately we will have the 4 points of Radix-4 DIF algorithm, which is shown in (18) - (21).

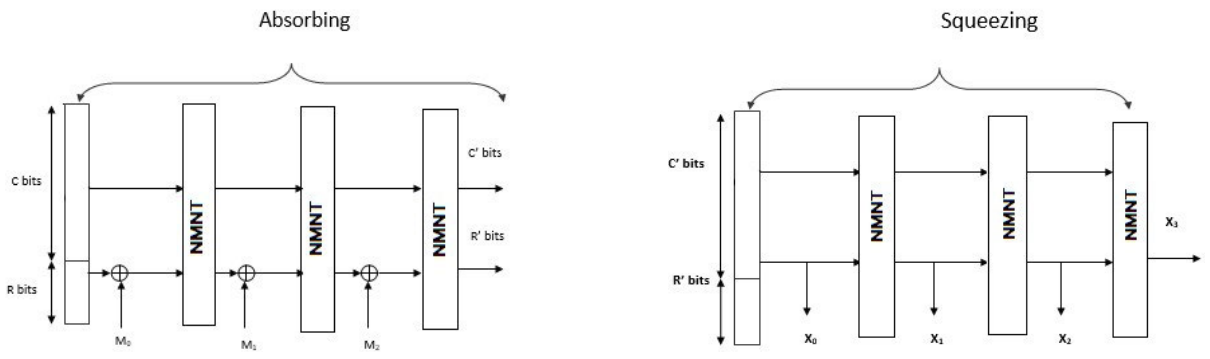
$$\begin{aligned}x_{4n} &= \langle \sum_{n=0}^{N/4-1} x(n) + x(n+N/4)\\ &\quad + x(n+N/2) + x(n+3n/4)\rangle_{mnp}. \end{aligned}\tag{18}$$

$$\begin{aligned}x_{4n+1} &= \langle \sum_{n=0}^{N/4-1} (x(n) + x(N/4-n) - x(n+N/2)\\ &\quad - x(3N/4-n) + x(n+3N/4))\beta_1(n)\\ &\quad + (x(n+3N/4) + x(N/2-n) - x(n+N/4)\\ &\quad - x(n-N))\beta_2(n)\rangle_{mnp}\end{aligned}\tag{19}$$

$$\begin{aligned}x_{4n+2} &= \langle \sum_{n=0}^{N/4-1} (x(n) - x(n+N/4) + x(n+N/2)\\ &\quad - x(n+3N/4) + x(n+3N/4))\beta_1(2n)\\ &\quad + (x(3N/4-n) - x(N/2-n) + x(N/4-n)\\ &\quad - x(n-N))\beta_2(2n)\rangle_{mnp}\end{aligned}\tag{20}$$

$$\begin{aligned}x_{4n+3} &= \langle \sum_{n=0}^{N/4-1} (x(n) - x(N/4-n) - x(n+N/2)\\ &\quad - x(3N/4-n) - x(n+3N/4))\beta_1(3n)\\ &\quad - (x(n+3N/4) - x(N/2-n) - x(n+N/4)\\ &\quad + x(N-n))\beta_2(3n)\rangle_{mnp}\end{aligned}\tag{21}$$

The butterfly diagram for NMNt radix-4 algorithm is shown in Fig.3.



(a) NMNT LWT Hash Sponge construction Absorbing phase

(b) NMNT LWT Hash Sponge construction Squeezing phase

FIGURE 4. **LNMNT sponge construction.**

IV. DESIGN CHOICES

The newly proposed hash algorithm in this research must also be able to run with minimal resources and overcome security and throughput issues. Because it uses sponge construction, the proposed hash function has a small internal memory footprint. The proposed hash function is described in detail in this section and hash digests of 80, 128, 160, and 224 bits are produced by the proposed hash function LNMNT.

A. SPONGE CONSTRUCTION OF LNMNT LWT HASH

In cryptography, sponge construction [24], [25] is a class of algorithms made up of internal states that take any length of input and produce any length of the output. LNMNT LWT [26] Hash's sponge construction is divided into two phases: absorption and squeezing. The message is iteratively divided into different blocks during the absorption phase. For designing variable length hash function we can change the block length. After that, the padding of the Message block is done to make it a power of two. Fig. 4a and 4b depict the absorption and squeezing phases. An algorithmic representation of LNMNT is shown below.

The internal state S is initialized with a secret key and is divided it into C bit capacity and R bit bitrate. Each block of the message passes through the NMNT block and gives the NMNT of each message block. Then the message is divided into any number of blocks. This is followed by bitwise XOR operation of the bitrate R with the massage block M_0. The output from this stage and capacity C bits are passed through NMNT and the process repeats again for the entire message blocks. Initialize the absorption phase using the secret values p (prime number), mnp (Mersenne number prime), and round constant RK. These values are used to calculate NMNT. Then calculate the value of q, α_1, and α_2 using (5) and (6). NMNT parameters α_1, α_2 and $\beta(nk)$ depend upon the transform length. Here the message is divided into different block sizes (power of 2), and each block returns a different transform length (power of 2). Hence, the NMNT parameters depend upon the block size, its values are different in each block.

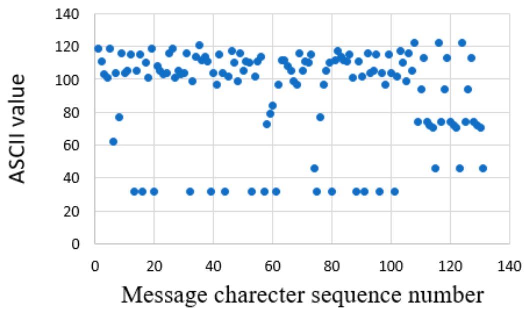
Next is the squeezing phase. The output from the absorption phase is passed to the squeezing phase, where the required output can be taken from the squeezing phase. Again, we also apply NMNT repeatedly on the output of LNMNT LWT Hash. We can take the output as X_0, X_1, X_2 and X_3, where the size of output decides the number of steps in the squeezing phase.

Algorithm 1 Algorithm for LNMNT LWT Hash

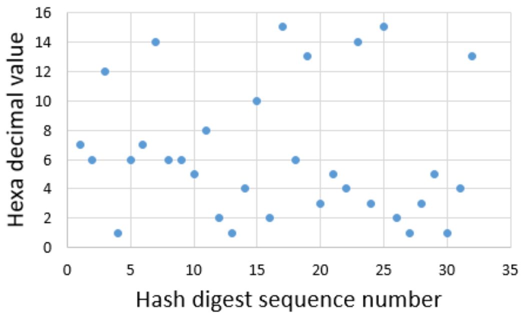
```
Input: an arbitrary length message:M
Output: 128 bit LNMNT hash digest
    Initialization: state variable with one secret value and keys
 1: Read the message to hash $M$
 2: Divide $M$ into variable length number of blocks, $m_0$, $m_1$, $m_2, \ldots, m_n$
 3: Then divide the state variable value into $c$ and $r$ number of bits. $b = c + r$
 4: c bits go into absorption phase without any change and the r bit Xored with first block of data m0, then goes into absorption phase.
 5: The Absorption phase is designed using NMNT. There are many secret values are initialized inside the Absorption phase for NMNT calculation. These values are not visible to the outside world. So that no one can reverse the Absorption phase by reversing the NMNT operation.

 6: The output from NMNT again Xor with next message block and goes into next absorption phase. This process repeats for all message blocks.
 7: Next is the squeezing phase Repeatedly perform NMNT operation on the output from the Absorption phase until we reach the required output length.
```

V. STATISTICAL ANALYSIS AND PERFORMANCE EVALUATION

In this section, we discuss the Uniform distribution of hash value, Diffusion and Confusion. For the evaluation, we select a random message M and its LNMNT hash digest H is given below.

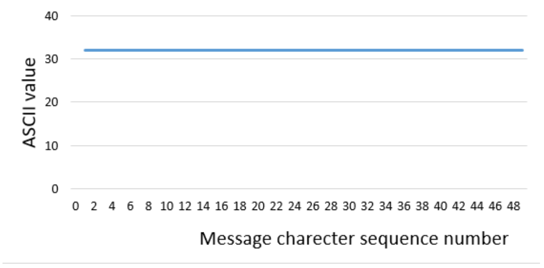


(a) Distribution of message M

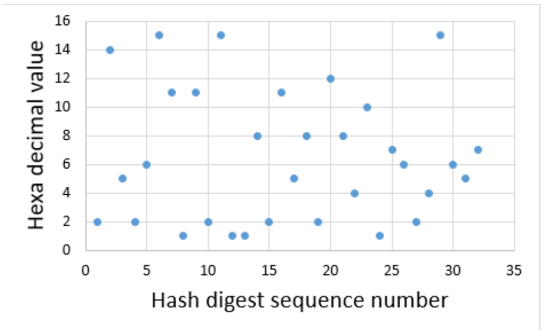


(b) Distribution of LNMNT hash digest

FIGURE 5. Plot of message M and it's LNMNT LWT hash digest.



(a) Distribution of blank space message



(b) Distribution of LNMNT hash digest

FIGURE 6. Plot of blank message and it's LNMNT LWT hash digest.

M: This is a new lightweight crypto hash function designed for Internet of Things (IoT) applications. The main goal of this hash function is to ensure secure communication between devices with limited resources.

H: 76C167E6658214A2F6D354E3F213514D

A. UNIFORM DISTRIBUTION OF LNMNT LWT HASH

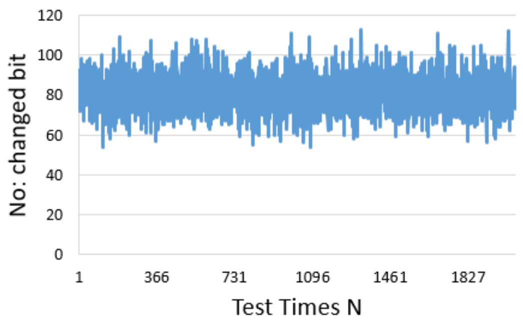
The important property of a hash function that is directly related to the security of the hash function is the uniform distribution of the hash digest. For this, we can consider a message m and plot the message m in a two-dimensional graph. As depicted in Fig. 5a, the plain text spreads in the range of [32,126], which is the range of ASCII values. Fig. 5b shows the hexadecimal value of the LNMNT hash digest. we can see that the LNMNT hash digest spreads uniformly and randomly and also hides the statistical information of the plain text.

Moreover, we evaluate the case of ''blank space message'', which is shown in Fig. 6a and 6b. We can see that even in this condition also the distribution of hash digest is uniform. So that we can prove that the proposed hash function has a uniform distribution on hash digest and it effectively hides the information about the plain message.

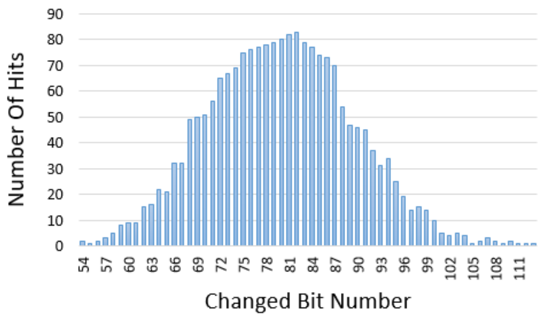
B. DIFFUSION AND CONFUSION ANALYSIS

Diffusion and confusion are considered to be important factors in the design of hash functions. Confusion implies the relationship between the message and the hash digest, which measures the difficulty of predicting the hash digest. Diffusion defines the dependency of hash value on plain text. For a good diffusion, a single bit change in the plain text should result in a 50 percent change in the hash digest. The NMNT is sensitive to slight variations of input. It also supports variable block length and has good diffusion properties. The main motivation for using NMNT in the design of the LNMNT LWT hash is due to these properties. The following statistical analysis is done to measure the diffusion and confusion property of the LNMNT LWT hash function. In this statistical experiment, first, a random message is selected and its LNMNT hash value is calculated. Then, randomly toggle a single bit in the message. The LNMNT hash of the toggled message is then calculated and the two hash values are compared bit by bit to find the number of changed bits. This statistical experiment is repeated N times and the Mean number of bits changed and the standard deviation is calculated. Table 5 shows the result of statistical analysis of diffusion and confusion [27], [28].

From Table 5 results, we can say that the proposed LNMNT LWT hash function generated the mean number of bits changed b^{-} and the mean changed bit probability P values are more than 50 percent. So that the proposed hash function has a good diffusion and confusion property. As shown in Fig. 7a, the value of the number of changed bits (b_i) is evenly distributed, and the histogram of b_i has a normal distribution, as shown in Fig. 7b. The statistical confusion and



(a) Plot of number of changed bit



(b) Histogram of bi

FIGURE 7. Diffusion and confusion analysis.

TABLE 5. Statistical analysis of LNMNT.

parameters	N=256	N=512	N=1024	N=2048	Mean
Mean number of bits changed (b^{-})	81.68	81.49	81.35	81.1	81.41
Mean changed probability(p)	63.81	63.66	63.55	63.35	63.59
Standard variation of b^{-} (Δb)	7.15	7.15	7.21	7.48	7.25
Standard variation of p (Δp)	4.1	4.2	4.33	4.41	4.26

TABLE 6. NIST test result.

Tests	p-values
Block Frequency	0.68
Cumulative sums	0.56
FFt	0.38
Runs	0.70
Longest run of ones	0.47
Rank	0.72
Discrete Fourier transform	0.38
Non overlapping template matching	0.85
Overlapping template matching	0.63
Universal statistical	0.45
Approximate entropy	0.93
Random excursions	0.53
Random excursions variant	0.43
Serial test	0.70
Linear complexity	0.42

diffusion results ensure that the proposed LWT hash function resists against any kind of differential and linear attacks.

C. COLLISION RESISTANCE

Collision resistance in cryptographic hash functions refers to the difficulty of finding two plaintexts that hash to the same hash value. That means it should be hard to find two different messages xa and xb such that $H(xa) = H(xb)$. The proposed hash function applies NMNT in every message block. Because this NMNT has a good diffusion property, it has a good avalanche effect.

For the proposed hash function, we ran a collision resistance test. The LNMNT LWT hash value for a randomly chosen message is generated for this purpose. Then a part of the randomly chosen message is randomly modified and its LNMNT hash value calculated. The two LNMNT hash values are then compared, and the number of identical ASCII values in the same position is counted. This experiment repeated 5000 times, as depicted in Fig.8.

There are no equal characters in 4435 cases out of 5000, and there are 552 cases with one equal character at the same location. There are about 12 cases of two identical characters at the same location, and the remaining cases of two identical

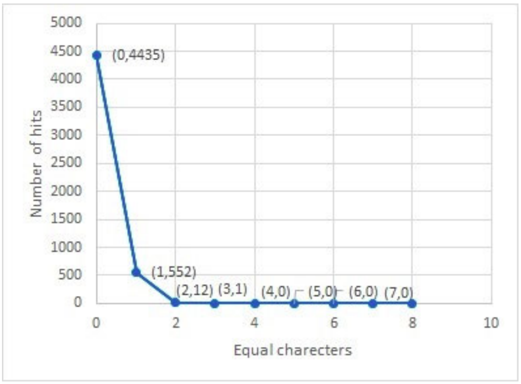


FIGURE 8. Distribution of number of hits.

characters at the same location are all zero. There can be a maximum of three identical characters in the same location. As the result, we can say that collision of our proposed hash function is very low.

D. RANDOMNESS TEST

In cryptography, statistical tests are used to ensure that the output is random [29]. The randomness property is strongly

TABLE 7. Performance comparison of LNMNT with other LWT hash functions.

LWT Hash Function	Construction method	Collision resistance	Hash output size(bits)	Cycles per byte	Power(μW)
LMNT Hash 80	Sponge construction	2^{50}	80	51180	5.52
LMNT Hash 128		2^{80}	128	52042	6.57
LMNT Hash 160		2^{100}	160	52742	6.68
LMNT Hash 224		2^{140}	224	55260	6.82
Photon 80	Sponge construction	2^{40}	80	51880	6.59
Photon 128		2^{64}	128	121593	7.29
Photon 160		2^{80}	160	137806	7.99
Photon 224		2^{112}	224	138964	8.01
Photon 256		2^{128}	256	162125	8.8
U Quark	Sponge Construction	2^{64}	128	151992	8.44
D Quark		2^{80}	160	170231	8.90
T Quark		2^{112}	224	173705	8.95
spongent−88	Sponge Construction	2^{40}	88	66324	5.87
spongent−128		2^{64}	128	121992	7.28
spongent−160		2^{80}	160	159700	8.82
spongent−224		2^{112}	224	153439	8.94
spongent−256		2^{128}	256	177324	9.11
Lesamnta-LWT	Merkle-Damgard	2^{128}	256	146925	−

related to the security of cryptographic algorithms [30]. The NIST statistical test was used to determine the randomness of the LNMNT LWT hash. The outcome is shown in Table 6.

Our proposed system ran many times for different inputs to accomplish the NIST test and generate a binary output file of about 10MB. If the P-value is greater than 0.01, the sequence is said to be random, i.e. ($P - value > 0.01$). It is clear from Table 6 that all of the frequency tests have a P-value is greater than 0.01. So that our LNMT hash function has enough randomness.

E. COLLISION ATTACK

The cryptographic hash function should be resistance to collision attack, second pre-image attack and pre-image attack. In the previous section we analyzed the collision resistance of proposed LWT hash function and obtained that our LNMT LWT hash provides good collision resistance. Next attack is second preimage attack. In second pre-image attack we are trying to find m_2 for a given m_1, such that hash(m_1) = hash(m_2). Collision resistance implies second pre-image resistance but not the third one pre-image resistance. In pre-image resistance it is computationally difficult to find any input message that hashes to given hash digest. LNMNT hash function is designed to support variable parameters. So that it computationally difficult to invert the hash digest.

F. BIRTHDAY ATTACK

In a birthday attack, the attacker tries to find two different input messages (m, m') that have the same hash values $h(m) = h(m') = h'$ within fewer than $2^{n/2}$ trials, where n is the hash digest length. We have seen from the previous sections that our proposed hash function has a good avalanche effect and is resistant to collision attacks. So that the LMNT LWT hash design is sufficient to withstand birthday attacks.

G. MEET IN THE MIDDLE ATTACK

The meet-in-the-middle attack tries to find a collision in the intermediate hash chaining values instead of the final hash values. A collision can be found if there is a match between two intermediate hash chaining values Because we used variable block length and secret values like NMNT parameters and round constant RK, the proposed hash function is resistant to meet-in-the-middle attacks. As a result, finding the collision in the intermediate hash chaining values is extremely difficult.

VI. COMPARISON WITH OTHER LWT HASH FUNCTIONS

The LNMNT hash function was implemented in C, and test runs for Contiki OS was performed using the COOJA simulator [31], [32]. In the cooja simulator, we simulated a wireless network platform using Z1 (Zolertia) motes and ran the LMNT LWT Hash. The benchmarks were run on an Intel Core 3 processor with a clock speed of 1.7 GHz and 64-bit architecture. For four different hash digests, namely, LNMNT80, LNMNT128, LNMNT160, and LNMNT224, the proposed LWT hash function is evaluated. According to the result, the memory usage for LNMNT128 is 4256 bytes Photon 128, u-QUARK (hash digest 128 bit), and Lesamnta(hash digest 128 bit) each consume 6728, 5646, and 6708 bytes of memory, respectively. When compared to other LWT hash functions with the same hash digest size, we can say that LNMNT uses less memory.

Similarly, the LNMNT128 LWT Hash takes 1.3 seconds to execute and consumes 6.57 μ of power. For a hash length of 128 bits, the proposed system uses 54042 cycles per byte.

Table 7 lists additional benchmark LWT Hash functions. As can be seen, our proposed LWT hash function outperforms other LWT hash functions in terms of cycles per byte, execution speed, collision resistance, and power consumption.

The main features of our proposed hash function are that it supports variable block size and that NMNT is a fast algorithm for calculating transforms, allowing us to hash long messages in a few rounds and very quickly without sacrificing resource efficiency or security. Other LWT hash functions require more rounds in order to produce a different hash digest.

Furthermore, other LWT hash functions are not adaptable to different hash digest lengths. However, with LNMNT, we can change the transform length and create variable-length hash digests without increasing the number of rounds. This is the reason why the LNMNT LWT hash function shows only a slight difference in power consumption and cycles per byte for different hash digest sizes. Other LWT hash functions have a large difference in cycles per byte and power consumption for different hash digest sizes when compared to LNMNT, which makes our proposed hash function appealing. Moreover, LNMNT LWT hash function provides good collision resistance than other LWT hash functions proving that the proposed system is more secure against collision attack than other LWT hash functions.

VII. CONCLUSION

The LWT hash function based on the NMNT is introduced as a LWT solution for IoT applications. The NMNT has a good diffusion property and a fast computation algorithm. The proposed hash function generates a transform with a variable length (powers of two). These characteristics make the New LWT Hash Function design suitable for a wide range of IoT applications. The proposed hash function passed the NIST test suite for randomness, confusion, diffusion, and attack types. Cooja Simulator was used to assess its computational complexity and energy consumption.

REFERENCES

[1] H. Zhang and L. Zhu, "Internet of Things: Key technology, architecture and challenging problems," in *Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng.*, Jun. 2011, pp. 507–512.

[2] Y. Ashibani and Q. H. Mahmoud, "An efficient and secure scheme for smart home communication using identity-based signcryption," in *Proc. 36th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2017, pp. 1–7.

[3] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2018.

[4] N. Kshetri, "Can blockchain strengthen the Internet of Things?" *IT Prof.*, vol. 19, no. 4, pp. 68–72, 2017.

[5] C. Lu, Y. Lin, S. Jen, and J. Yang, "Cryptanalysis on PHOTON hash function using cube attack," in *Proc. Int. Conf. Inf. Secur. Intell. Control*, 2012, pp. 278–281, doi: 10.1109/ISIC.2012.6449760.

[6] D. N. Gupta and R. Kumar, "Sponge based lightweight cryptographic hash functions for IoT applications," in *Proc. Int. Conf. Intell. Technol. (CONIT)*, Jun. 2021, pp. 1–5, doi: 10.1109/CONIT51480.2021.9498572.

[7] N. Nabeel, M. H. Habaebi, N. Arfah, and C. Mustapha, "IoT light weight (LWT) crypto functions," *Int. J. Interact. Mobile Technol.*, vol. 13, no. 4, pp. 117–129, 2019.

[8] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities," *IEEE Access*, vol. 9, pp. 28177–28193, 2021, doi: 10.1109/ACCESS.2021.3052867.

[9] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON lightweight hash functions family," in *Crypto* (Lecture Notes in Computer Science), vol. 6841. Berlin, Germany: Springer, 2000, pp. 222–239.

[10] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "QUARK: A lightweight hash," *J. Cryptol.*, vol. 26, no. 2, pp. 313–339, 2013.

[11] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and B. Verbauwhede, "SPONGENT: The design space of lightweight cryptographic hashing," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2041–2053, Aug. 2012, doi: 10.1109/TC.2012.196.

[12] A. Akhimullah and S. Hirose, "Lightweight hashing using lesamnta-LW compression function mode and MDP domain extension," in *Proc. 4th Int. Symp. Comput. Netw. (CANDAR)*, Nov. 2016, pp. 590–596.

[13] L. Yang, M. Wang, and S. Qiao, "Side channel cube attack on PRESENT," in *Proc. Int. Conf. Cryptol. Netw. Secur.* Berlin, Germany: Springer, 2009, pp. 379–391.

[14] T. Akishita and H. Hiwatari, "Very compact hardware implementations of the blockcipher CLEFIA," in *Proc. Int. Workshop Sel. Areas Cryptogr.* Berlin, Germany: Springer, 2011, pp. 278–292.

[15] C. Tezcan, "The improbable differential attack: Cryptanalysis of reduced round CLEFIA," in *Proc. Int. Conf. Cryptol.* Berlin, Germany: Springer, 2010, pp. 197–209.

[16] M. A. Philip and Vaithiyanathan, "A survey on LWT ciphers for IoT devices," in *Proc. Int. Conf. Technol. Adv. Power Energy (TAP Energy)*, 2017, pp. 1–4, doi: 10.1109/TAPENERGY.2017.8397271.

[17] O. Lo, W. J. Buchanan, and D. Carson, "Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)," *J. Cyber Secur. Technol.*, vol. 1, no. 2, pp. 88–107, Apr. 2017.

[18] P. T. Berger, J. D'Hayer, and K. Marquet, "The GLUON family: A lightweight hash function family based on FCSRs," in *Progress in Cryptology* (Lecture Notes in Computer Science), vol. 7374. Springer, 2012, pp. 306–323. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-31410-0_19/

[19] L. Dalmasso, F. Bruguier, P. Benoit, and L. Torres, "Evaluation of SPN-based lightweight crypto-ciphers," *IEEE Access*, vol. 7, pp. 10559–10567, 2019.

[20] N. U. Tyne, "New Mersenne number transform diffusion power analysis," *Amer. J. Eng. Appl. Sci.*, vol. 4, no. 4, pp. 461–469, 2011.

[21] N. Rutter, S. Boussakta, and A. Bystrov, "Assessment of the one-dimensional generalized new Mersenne number transform for security systems," in *Proc. IEEE Veh. Technol. Conf.*, Jun. 2013, pp. 1–5.

[22] S. Boussakta and S. Member, "Evaluation of one-dimensional NMNT for security applications," in *Proc. CSNDSP*, Jul. 2010, pp. 715–720.

[23] S. Boussakta and M. Aziz, "RADIX-4 ALGORITHM for the new Mersenne number transform," in *Proc. 16th World Comput. Congr.*, Aug. 2000, pp. 23–25.

[24] Y. Li and G. Ge, "Cryptographic and parallel hash function based on cross coupled map lattices suitable for multimedia communication security," *Multimedia Tools Appl.*, vol. 78, pp. 17973–17994, Apr. 2019. [Online]. Available: https://doi-org.ezlib.iium.edu.my/10.1007/s11042-018-7122-y

[25] M. Borowski, "The sponge construction as a source of secure cryptographic primitives," in *Proc. Mil. Commun. Inf. Syst. Conf.*, Oct. 2013, pp. 1–5.

[26] N. Nabeel, M. H. Habaebi, and M. R. Islam, "LNMNT-New Mersenne number based lightweight crypto hash function for IoT," in *Proc. 8th Int. Conf. Comput. Commun. Eng. (ICCCE)*, Jun. 2021, pp. 68–71, doi: 10.1109/ICCCE50029.2021.9467180.

[27] A. Maetouq and S. M. Daud, "HMNT: Hash function based on new Mersenne number transform," *IEEE Access*, vol. 8, pp. 80395–80407, 2020, doi: 10.1109/ACCESS.2020.2989820.

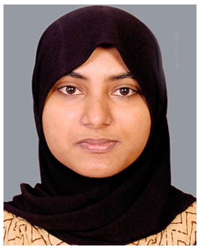
[28] N. Hidayah Lot, N. A. Nik Abdullah, and H. Abdul Rani, "Statistical analysis on KATAN block cipher," in *Proc. Int. Conf. Res. Innov. Inf. Syst.*, Nov. 2011, pp. 1–6.

[29] C. Georgescu, A. Nita, and A. Toma, "A view on NIST randomness tests (in) dependence," in *Proc. ECAI*, Jun./Jul. 2017, pp. 9–12.

[30] A. Rukhin, J. Soto, and J. Nechvatal, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 800-22 Rev 1a, Jul. 2021.

[31] A. Velinov and A. Mileva, "Running and testing applications for contiki OS using cooja simulator," in *Proc. Int. Conf. Inf. Technol. Dev. Educ.*, 2016, pp. 279–285.

[32] T. R. Sheltami, E. Q. Shahra, and E. M. Shakshuki, "Perfomance comparison of three localization protocols in WSN using cooja," *J. Ambient Intell. Hum. Comput.*, vol. 8, no. 3, pp. 373–382, Jun. 2017.



NUBILA NABEEL received the Bachelor of Technology degree in computer science engineering from the MES College of Engineering, Calicut University, Kerala, India, in 2012, and the Master of Technology degree in communication and network technology from Mahatma Gandhi University, Kerala, in 2015. She is currently pursuing the Ph.D. degree in computer Engineering with the Department of Electrical and computer Engineering, International Islamic University Malaysia (IIUM). Her current research interests include the IoT, cryptography, and block chain technology.

MOHAMED HADI HABAEBI (Senior Member, IEEE) received the B.Sc. degree from the Civil Aviation and Meteorology High Institute, Libya, in 1991, the M.Sc. degree in electrical engineering from University Technology Malaysia, in 1994, and the Ph.D. degree in computer and communication system engineering from University Putra Malaysia, in 2001. He is currently a full-time Professor and the Post Graduate Academic Advisor with the Department of Electrical and Computer Engineering, International Islamic University Malaysia, where he heads the research works on the Internet of Things. He has supervised many M.Sc. and Ph.D. students. He has published more than 120 articles and papers, and sits on the Editorial Boards of many international journals. He is actively publishing in M2M communication protocols, wireless sensor and actuator networks, cognitive radio, small antenna systems, and radio propagation, and wireless communications and network performance evaluation. He is also an Active Member of the IEEE and an Active Reviewer of many international journals.

M. D. RAFIQUL ISLAM (Senior Member, IEEE) received the Bachelor of Science degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology (BUET), Dhaka, in 1987, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Technology Malaysia, in 1996 and 2000, respectively. He is currently working as a Professor with the Department of Electrical and Computer Engineering, Faculty of Engineering, International Islamic University Malaysia. He has supervised more than 50 M.Sc. and Ph.D. students and has published more than 200 research papers in international journals and conferences. His research interests include wireless channel modeling, radio link design, RF propagation measurement and modeling in tropical and desert, RF design, smart antennas and array antennas design, and FSO propagation and modeling. He is a Life Fellow of Institute of Engineers Bangladesh and a member of IET.