INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2023, VOL. 69, NO. 2, PP. 253–259
Manuscript received February 26, 2023; revised April, 2023. DOI: 10.24425/ijet.2023.144358

# Energy Saving Chaotic Sequence Based Encryption, Authentication and Hashing for M2M Communication of IoT Devices

Bartosz Kościug, and Piotr Bilski

***Abstract*—In this paper, the secure low-power Internet of Things (IoT) transmission methods for encryption and digital signature are presented. The main goal was to develop energy-efficient method to provide IoT devices with data confidentiality, integrity, and authenticity. The cryptograph energy efficient and security algorithms modifications for IoT domain were made. The novelty in our solution is the usage of encryption method popular in the image processing in the domain of the Internet of Things. Proposed modification improves immunity for the brute-force and plain-text attacks. Furthermore, we propose the modifications for hash calculation method to transform it into digital signature calculation method that is very sensitive to input parameters. The results indicate low energy consumption of both methods, however it varies significantly depending on the architecture of the devices.**

***Keywords*—Power optimization; Security protocols; Application layer protocols; Machine to machine communication; IoT**

## I. Introduction

CYBER security has been increasingly studied in recent years regarding methods of providing information systems with data confidentiality, authenticity, and integrity. In the modern world, inventions in the field of algorithms such as encryption, hashing, or digital signature help to protect data privacy in the IP networks.

The idea of secure transmission protocols is the same for the Internet of Things networks with different challenges. Especially, in environments that connect small low-power sensor devices into networks - LPWAN (Low-Power Wide-Area Network). These devices in many cases are used to perform critical tasks such as monitoring and controlling industrial processes or healthcare systems, transmit sensitive information such as personal, financial, and medical data. They are powered by small batteries and operate with low throughput on long distances (see Fig. 1). Therefore any method that offers secure transmission of data in an IoT environment must minimize the consumed energy. One should also ensure that the introduced protocol has as small payload overhead over the data to transmit as possible. In practice, this stipulates that developed algorithms utilize a negligible amount of CPU processing time. LPWANs connect together a vast variety of the IoT devices. This includes small units, such as PIC18F25K50 microcontroller (with an 8-bit processor and 2kB of RAM) as well as large and complex systems (such as ESP-WROOM-32D with a 32-bit processor and 520KB of RAM) - see Fig. 2. Therefore there is the need to level down the usage of dynamic and static memory during the design and implementation of data transmission protocols for IoT systems.

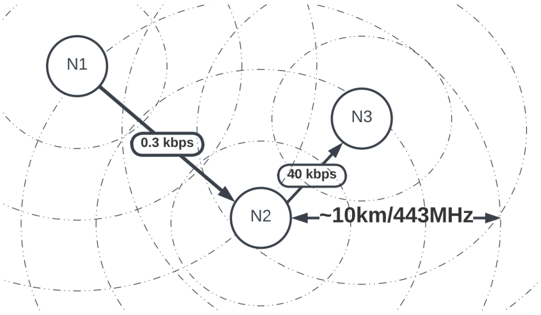

Fig. 1. Example of LPWAN. N1, N2 and N3 are LoRaWAN nodes

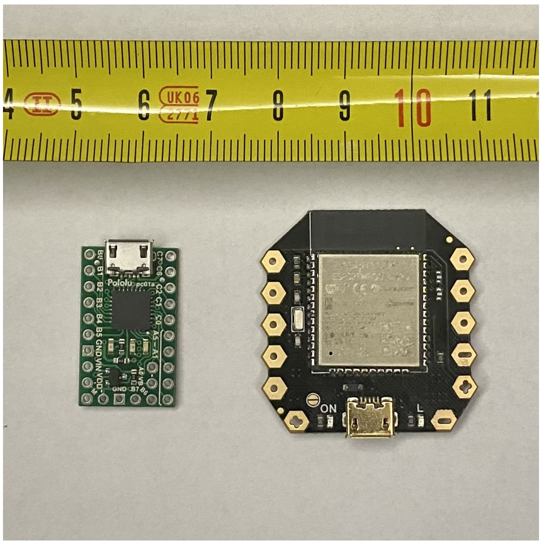

Fig. 2. PIC18F25K50 and ESP-WROOM-32D

Authors are The Faculty of Electronics and Information Technology on Warsaw University of Technology, Warsaw, Poland (e-mail: bartosz.kosciug.dokt@pw.edu.pl, piotr.bilski@pw.edu.pl).

Data confidentiality is the first principle of the IoT system's security. Multiple algorithms currently provide data privacy (RSA, DES, AES, etc.). An efficient encryption system is expected to be cost-efficient and compact [1].

The RSA is still the standard of the public key infrastructure in the Internet, operating on two keys: private and public. Usually the former is mused to encrypt data, while the latter aims at decryption. This system requires the infrastructure to manage private keys, distribute them through nodes, and store them safely. It also has been proven that RSA is resilient to attack using the quantum computer.

On the other hand, AES and DES are symmetric systems, so only one key is needed for decryption and encryption. Storing variables and constants used by the algorithms consumes the memory of the device. For example, in the AES at least 128 bits (16 bytes) must be allocated in memory for node-to-node communication. The encryption requires the utilization of the Initialization Vector (IV) to ensure that the same input data processed multiple times with the same secret key does not result in an identical cipher. The IV generation requires additional 16 bytes. Devices share the same key, but this increases the probability of compromising the communication scheme (therefore to distribute it over the network the additional scheme must be used). This way the secure connection of one thousand devices in the machine-to-machine mode will require the allocation of 3200-bits of memory only for key handling. This is about 160% of RAM in PIC18F25K50, which drastically decreases the resource availability for microcontroller tasks.

Data confidentiality and data integrity are the core security paradigms. It is strongly recommended to introduce data integrity validation mechanisms, such as hash functions. Popular hashing methods such as MD5, SHA1/2, and Keccack are characterized by high computation load, as they were not designed for IoT devices. For example, the MD5 running on the Mica2 MOTE development board equipped with the 8-bit ATmega128 computer requires 36360 CPU clock cycles to calculate the hash value [2]. The calculated hash values from MD5 and SHA families are generally long - more than 100 bits. In particular, the MD5 hash value is 128-bit long, SHA-2 uses 256/512-bits, while SHA-3 utilizes up to 1600-bits [3]. This further decreases the amount of memory accessible for the programmable tasks of IoT devices.

Hash functions are also used for data authenticity. They ensure that the information exchange is executed between two valid devices (eliminating all penetration attempts by external nodes). Digital signatures such as RSA, ECDSA, DSA or Shnnor combine the data integrity and data authenticity validation functionalities. The performance of such algorithms has also a high impact on the accessible resources of IoT devices. For instance, to provide the minimum level of security RSA requires at least a 2048-bit key [4], while the ECDSA's key requires at least 224-bits.

In the following paper, the simple and energy-efficient methods to provide IoT machine-to-machine communication with data confidentiality, integrity, and authenticity are presented. In the II section, we described the selection and proposed modifications of chosen cryptographical algorithms. In the III section, the readers will find out about the performance results of examined algorithms. Finally, in the last section IV, the authors conclude the work. The novelty of the presented solution regarding confidentiality is that we modified and simplify the usage of the chaotic neural network encryption. By introducing the feedback factor with cipher xor type, the control of the minimum length of the plain text, and the changes of initial encryption parameters per each transmission. That as a result makes it difficult for attackers to apply brute-force and plain-texts attacks. Regarding data integrity and authenticity, we introduced the modification for the one-way hash function. Which now takes additional input parameters that work in the form of a message signing key and could be easily changed in each transmission.

## II. Proposed Algorithms

### A. Encryption

Considering the constraints of IoT devices, the appropriate cryptographic algorithm for the IoT devices should utilize minimal possible hardware resources.

The interesting approach of low-power data encryption based on the chaotic sequence and Artificial Neural Network (ANN) was initially presented in [5]. Later [6], [7] it was stated that compared to popular algorithms, such as DES, RSA or AES, the Chaotic ANN (CNN) are worth considering as a tool for data encryption regarding their low-computation overhead.

In [5] the symmetric stream cipher based on CNN was proposed. The latter is the neural network with weights and biases determined by the chaotic sequence [6]. The coding of neuron parameters is calculated with the usage of so-called chaotic maps. The idea is to incorporate the nondeterministic characteristics of chaotic sequence into the ANN output function. In the result, the CNNs have a strong sensitivity to the initial conditions (input values). Such a configuration of ANN appears to behave randomly, where the randomness results from its structure [6].

The proposed chaotic encryption algorithm is depicted in Algorithm 1. During the encryption, the $while$ loop is executed, in every iteration, the subsequent elements of the chaotic sequence are calculated. The number of iterations is determined by the length of the byte arrays $data$ that hold the secret values (plaintext) to be encrypted. In most works devoted to the chaotic encryption problem, the authors use the logistic map (see 1) to obtain the next elements of the chaotic map. In the proposed algorithm, the tent map (see 2) is used instead due to its better performance regarding speed and entropy values compared to other examined chaotic maps (see I).

$$x(n+1) = \mu{\cdot}x(n){\cdot}(1 - x(n)) \tag{1}$$

$$x(n+1) = \begin{cases} \mu{\cdot}x(n), & x(n) < \frac{1}{2} \\ \mu{\cdot}(1 - x(n)), & x(n) \geq \frac{1}{2} \end{cases} \tag{2}$$

The elements of the chaotic sequence are real floating point numbers. After the calculation of the next chaotic sequence element, the least significant byte of its mantissa is retrieved.

In [8] it was shown that the least significant 32 bits of the logistic map's mantissa elements are actually pseudo-random. In the proposed solution the least significant 8 bits of the mantissa are used to encrypt the data stream. They are next stored in the $seqElemLSB$ variable. The encryption is done through iteration over each byte of the element of the plain text array $data$ and by applying the XOR operation with each byte of the least significant byte of the $seqElemLSB$. The idea of the XOR-based encryption is motivated by [9], where it was proposed to simplify the encryption procedure. In that solution, the combination of XOR and addition replaces the ANN.

As stated in [10], the XOR operation is vulnerable to cryptographic attacks such as brute force and plain text. In which the attacker tries to find out the cipher key used in the encryption process to reveal message content. According to [11], to resist such a threat the length of the encryption key should be increased to at least 64 bits. For that purpose, the *validateDataLength* method (see Algorithm 1) is used to add extra padding if the data length is less than 16 elements/characters (1 byte multiplied by 16 equals 64 bits).

In the plain-text attack, the intruder knows or tries to guess the data intercepted from the transmission. He then tries to retrieve the encryption key mostly by performing the XOR operation on the data. In the proposed solution we change the encryption key for each transmission. The new value of the first element of the chaotic sequence is calculated after data has been encrypted. The new value is set by calculating $k$ elements of the chaotic sequence beginning from its last element that was used for encryption. For example, while processing 16 bytes-long data, all 16 elements of a chaotic sequence are calculated. Next, $16 + k$ elements are computed and a new value of $x(0)$ is set up. The $k$, $x(0)$ and $\mu$ are the secret values, shared between devices that wish to communicate. This way even if the attacker intercepts the data and discovers the key, the next transmission will be done in different conditions. To make the job more difficult for the plain-text attack scheme, the proposed algorithm introduces the *rec* variable. It works as a feedback factor. To decrypt the byte $k$ every previous element of the chaotic sequence must be known. Knowledge of its single element is useless, as presented in [5].

**Algorithm 1** Proposed chaotic encryption

```
Precondition: $x(0) \in (0,1), \mu \in [1.99, 2)$, $k \in \mathbb{N}$
  function ENCRYPT(data[], x, $\mu$, k)
      $i \leftarrow 1$
      $rec \leftarrow getLSB(0)$
      $validateDataLenght(data)$        ▷ If less than 16 add padding
      while $i \neq length(data)$ do
          if $x(i) < 0.5$ then
              $x(i)=\mu{\cdot}x(i-1)$
          else
              $x(i)=\mu{\cdot}(1-x(i-1))$
          end if
          $seqElemLSB=getLSB(x(i))$        ▷ Receive the 8 least significant bits from Mantissa
          $seqElemLSB=seqElemLSB \oplus rec$
          for $k \leftarrow 1$ to 7 do
              $data[i][k]'=data[i][k] \oplus seqElemLSB[k]$
          end for
          $rec=seqElemLSB$
          $i=i+1$
      end while
      while $k \neq 0$ do
          if $x(i) < 0.5$ then
              $x(i)=\mu{\cdot}x(i-1)$
          else
              $x(i)=\mu{\cdot}(1-x(i-1))$
          end if
          $i=i+1$
          $k=k-1$
      end while
      $x(0)=x(i)$
      return x(0), data'   ▷ New initial value for next rounds of encryption and encrypted byte array
```

### *B. Data integrity and authenticity*

To achieve data integrity and authenticity modification of the LOCHA - *A Light-weight One-way Cryptographic Hash Algorithm* [2] is proposed to transform it into a digital signature. Two factors were taken into account while selecting the algorithm.

- The number of CPU cycles required to perform the hashing operation on the input data. It is also the amount of energy consumed by the hashing algorithm.
- The size of the obtained hash value.

There are multiple hashing algorithms. Based on their comparison [2], [12], [13] LOCHA was considered the best candidate for a hash algorithm that consumes a minimum amount of energy. The algorithm demands only 2952 clock cycles and 9.4464 µJ of energy estimated for the Mica2 MOTE sensor board equipped with ATMega128 computer [2]. The LOCHA generates a relatively small fixed-size 24-byte hash digest.

The idea behind the LOCHA hashing procedure is that the values of the ASCII characters are mapped from the byte array to the prime numbers, then subsequent conversions are calculated and finally mapped to hex values. As indicated in [2], the algorithm's advantages are preimage resistance, collision resistance, and second preimage resistance.

The modified version of LOCHA is presented in Algorithm 2. The main difference in comparison to its original version is that values for sTable1 and sTable2 arrays are not constant. These two arrays are used to keep values for prime numbers used in the hash calculation process. In presented version the needed values are calculated at the begin of each algorithm run (see Algorithm 2 $calcPrimesTab$ - bond font). This way our solution of LOCHA can now be parameterized. In other words, each round of the algorithm run can generate different values of hash/digital signature - if we want it.

The arguments of the $calcPrimesTab$ function (see Algorithm 3) allow for the generation of custom sets of prime

**Algorithm 2** Proposed chaotic digital signature calculation

```
Precondition: $x1, x2 \in (-1,1), p1, p2 \in \mathbb{N}$
  function LOCHASIGNATURE(data[], x1, x2, p1, p2)
      $cf1 \leftarrow 1$
      $cf2p1 \leftarrow 0$
      $cf2p2 \leftarrow 0$
      $cf2p3 \leftarrow 7$
      $acf \leftarrow 0$
      $subHex[3] \leftarrow [0,0,0]$
      $hexDecimal \leftarrow 0$
      sTable1 ← calcPrimesTab(x1, p1, 96)
      sTable2 ← calcPrimesTab(x2, p2, 63)
      for $i \leftarrow 0$ to 7 do
          for $j \leftarrow 0$ to 7 do
              ▷ ——— Caclulate conversion factors ———
              if $data[8{\cdot}i + j] \neq 0$ then
                  $pk{=}castToInt(data[8{\cdot}i + j] - 31)$
              else
                  $pk{=}1$
              end if
              $cf1{=}$sTable1[pk]$\cdot cf1$
              if $cf1 > 65535$ then
                  $cf1{=}cf1 \pmod{65535}$
              end if
              $cf2p1{=}cf1 \pmod{67}$
              if $data[1][7] == 1$ then
                  $cf2p2{=}cf2p1{-}67$
              else
                  $cf2p2{=}cf2p1$
              end if
              $cf2p3{=}(cf2p3 + (cf2p1 + cf1) \pmod{256})$
              ▷ ——— Caclulate after conversion factor ———
              $acf{=}(cf1 \pmod{cf2p3}){+}cf1$
              $acf{=}acf{+}$sTable2[cf2p2 - 1]
              $acf{=}acf \pmod{255}$
              $acf{=}acf{+}int(data[j{\cdot}8] \pmod{127})$
              $hexDecimal{=}hexToDec(subHex)$
              $acf{=}acf{+}int(data[j{\cdot}8{+}2]){+}hexDecimal$
              ▷ ——— Caclulate part of the hash value ———
              $subHex = intToHex(afc)$
              $hash[j{\cdot}3]{=}subHex[0]$
              $hash[j{\cdot}3{+}1]{=}subHex[1]$
              $hash[j{\cdot}3{+}2]{=}subHex[2]$
              $swap(subHex[1], subHex[2])$
              $subHex[0]{=}0$
          end for
      end for
       return hash
```

numbers. This way hash values calculated with different parameters will be highly different from each other.

Let's assume devices A and B have the same values for parameters $x1$, $x2$, $p1$ and $p2$. Device A sends the signature with the message to device B. Then device B calculates the signature on the received message and compares it with the received signature from device B. If signatures are the same, Device A is authenticated, because secret parameters can be shared with it. This way both devices are synchronized to communicate with each other.

In Algorithm 3 we decided to use the so-called primes gap pattern that appears in Logistic Map [14]. That allows for generating multiple prime numbers in a short period of time (see Fig. 4).

**Algorithm 3** Proposed chaotic prime numbers generation

```
Precondition: $x \in (-1,1), pow \in \mathbb{N}$
  function CALCPRIMESTAB(x, pow, n)
      $isPrime \leftarrow 0$
      $p \leftarrow 0$
      $counter \leftarrow 0$
      $prime \leftarrow 0$
      $coeff \leftarrow 10^{pow}$
      $table[n]$
      while $counter < n$ do
          $prime{=}castToInt(x \times coeff)$
          $prime{=}(1024{+}(prime \pmod{1024}))$
  $\pmod{1024}$
          if $prime > 1$ then
              for $i \leftarrow 2$ to $\sqrt{prime} + 1$ do
                  if $prime \pmod{i} == 0$ then
                      $isPrime{=}1$
                  else
                      $isPrime{=}0$
                  end if
              end for
          end if
          if $isPrime == 0 \wedge isPrime < 1000$ then
              $table[counter] = prime$
              $counter{=}couter + 1$
          end if
          $x \leftarrow 1 - x^2{\cdot}1.5437$
      end while
       return table
```

## III. Experiments

The goal of this section is to present the experiments regarding the proposed algorithm's performance in the following scope:

- Confirmation that the proposed key-generation method (using pseudo-random values) will pass the NIST (National Institute of Standards and Technology of the U.S. Government) test suite (section A).
- Verification of the computational complexity for the prime number generation operation (section B). The solution will be accepted, if at most $O(n)$ or $O(n \log n)$ time complexity is obtained.
- Validation of the sensitivity of the proposed digital signature calculation (section C). A high sensitivity regarding input parameters is expected.
- The energy consumption of proposed encryption and digital signature calculation methods (section D). This is important to estimate the run-out time of the battery that powers IoT devices. The energy consumption measurement strategy is as follows:

1) Program the device in a way that it is possible to capture the algorithm duration.
2) Capture the current and voltage values in the active mode of the device using TC66 measurer. The setup of the circuit used for measuring voltage and current values is presented in 3.
3) Estimate the energy consumption with 3.

$$E = \Delta T \cdot I \cdot U \tag{3}$$

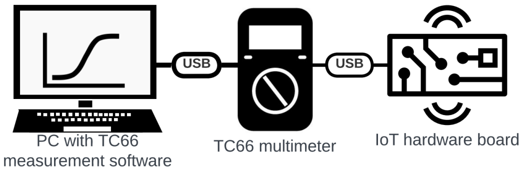

Fig. 3. Scheme of the measurement circuit

## A. PRNG with the tent map

The decision on the tent map utilization was dictated by the results of the tent map in comparison with other maps such as logistic, cubic, sine, etc. (see Tab. I). The average generation time of chaotic sequence elements and entropy were examined. Entropy is well-known in cyber security, as a measure of data randomness. The elements of the sequence are used in the process of creating the cipher key, therefore we want the entropy value to be as high as possible. Results of evaluating 1 million samples for each evaluated chaotic map are depicted in I.

TABLE I
RESULTS OF TENT MAP COMPARISON

| Map | Avg. Gen. time [$\mu s$] | Entropy | Ref. | Choice |
|---|---|---|---|---|
| Logistic | 0.251 | 19.577 | [15] | No |
| Tent | 0.241 | 19.685 | [15] | Yes |
| Cubic | 0.344 | 19.609 | [15] | No |
| Sine | 0.360 | 19.614 | [15] | No |
| Digital Cosine | 0.373 | 19.472 | [16] | No |
| Arcsine | 0.433 | 19.671 | [17] | No |
| Amplified | 0.765 | 19.577 | [18] | No |

The tent map in the proposed algorithm acts as a key generator tool. Ideally, the encryption key should be as random as possible. To examine the pseudo-random generation properties of the proposed key construction method the statistical tests suite - NIST was used. It includes a set of tests that could be applied to the binary sequences [19]. Results presented in II show the quality of the generated bytes sequences by the tent map. Each byte of the tested sequence was the least significant byte of the chaotic sequence element mantissa (as we stated in the section II). The ”test passed” status is reached when its p-value is greater than 0.05 ($p - value > 0.05$).

TABLE II
RESULTS OF NIST TEST FOR TENT MAP

| Statistical test | P-value | Passing rate |
|---|---|---|
| Frequency | 0.55 | 0.98 |
| Block frequency | 0.85 | 1.0 |
| CumulativeSums | 0.49 | 0.98 |
| Runs | 0.08 | 0.99 |
| LongestRun | 0.13 | 1.0 |
| Rank | 0.94 | 0.99 |
| FFT | 0.28 | 0.99 |
| NonOverlappingTemplate | 0.20 | 0.97 |
| OverlappingTemplate | 0.22 | 0.99 |
| Universal | 0.81 | 1.0 |
| ApproximateEntropy | 0.45 | 1.0 |
| Serial | 0.59 | 0.99 |
| LinearComplexity | 0.20 | 0.97 |

## B. Primes generation complexity

To validate the proposed method for prime number generation (see 3), the time complexity has been examined. Figure 4 shows the obtained results. The proposed method of prime numbers generation (despite some noise samples) can be classified as having linear time complexity. The orange curve represents the ”raw” results of captured time generation. The blue curve represents the first-level polynomial approximation. These results are satisfactory, producing 1 million prime numbers below 5 seconds.

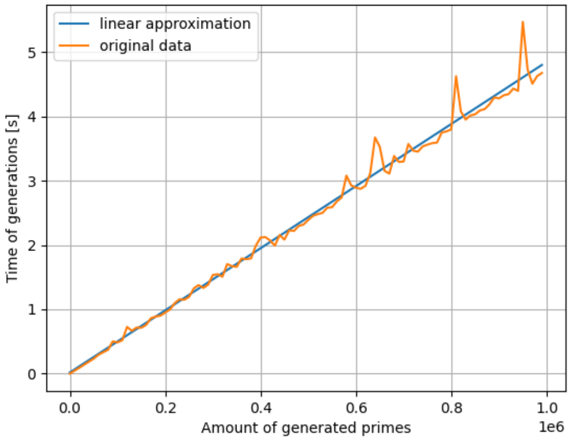

Fig. 4. Time complexity of Algorithm 3

## C. Signature sensivity

Table III shows the sensitivity of the generated signature in relation to input parameters. While comparing the first sample with the subsequent ones, significant changes in the output value of the signature are observed. The level of security in such a signature system is high because of the low probability of sharing 4 variables (128 bits) by two non-familiar devices.

That means if an attacker wants to compromise such a digital signature system, he or she needs to know those variables.

TABLE III
DIGITAL SIGNATURE INPUT PARAM. SENSITIVITY OF 'HELLO WORLD!'

| x1 | x2 | pow1 | pow2 | signature value |
|---|---|---|---|---|
| 0.7 | 0.7 | 6 | 10 | 0f9268157133072109108118 |
| 0.7 | 0.71 | 6 | 10 | 11e1ed1550ed1430f9113085 |
| 0.71 | 0.7 | 6 | 10 | 1a91f30d204f1250cf15c128 |
| 0.7 | 0.7 | 7 | 10 | 0d219d1081391851410b6098 |
| 0.7 | 0.7 | 6 | 11 | 15c25f1950b207d18c0ff171 |
| 0.7000001 | 0.7 | 6 | 10 | 1771fd15b0d406c0e507d0f6 |

## D. Energy consumption measurements

The proposed algorithm was tested regarding energy consumption by different microcontrollers from the top five popular brands. The list of examined devices with their hardware resources is presented in Table IV. The selected boards are of different architectures, covering a wide range of accessible solutions.

TABLE IV
EXAMINED DEVICE HARDWARE PARAMETERS

| Device | Arch. | $f$ [MHz] | RAM [kB] | Flash [kB] |
|---|---|---|---|---|
| ESP-WROOM-32 | 32-bit | 240 | 520 | 16 |
| ATMEGA32U4 | 32-bit | 16 | 2.5 | 32 |
| MSP-EXP430F5529LP | 16-bit | 8 | 8 | 128 |
| PIC18F25K50 | 8-bit | 24 | 2 | 32 |
| STM8S207K8T6C | 8-bit | 24 | 6 | 64 |

Table V shows the result of the energy consumption for the calculation of the 97 prime numbers array - sTable1 (see Algorithm 2 $calcPrimesTab$). This process is the first part of the digital signature calculation. In comparison to hash calculation (see Table VI) or encryption (see Table VIII), sTable's calculation is much more energy consuming. In the proposed scheme $sTable1$ and $sTable2$ arrays are calculated for each signature (see 2). However, to reduce the energy utilization the calculations for sTable1 and sTable2 could be done - just per communication session, before hash calculation part.

TABLE V
STABLE N=97 CALC. ENERGY PER DEVICE

| Device | Energy of STable values calc. [J] |
|---|---|
| ESP-WROOM-32 | 0.0263 |
| ATMEGA32U4 | 0.8213 |
| MSP-EXP430F5529LP | 0.0003 |
| PIC18F25K50 | 3.0625 |
| STM8S207K8T6C | 0.2011 |

Table VI shows the results of the second part of digital signature calculation (without calculation of sTables). The lowest amount of energy needed to calculate hash value was noticed for ESP-WROOM-32 microcontroller, while the most energy-demanding was PIC18F25K50. In comparison to [2] higher levels of energy needed to calculate the hashes were observed. In [2] the amount of energy needed by the algorithm was estimated only theoretically - based on the ATmega128 datasheet, while this section shows the real-world outcomes. These are satisfactory, as, for instance, the energy needed for the hash calculation on ATmega32U4 is lesser than for SHA1 hash. We calculated that our hash calculation algorithm requires 47% less energy in comparison to SHA1 algorithm. Which is assumed to be one of the most power-safety hash algorithms implemented for IP networks.

TABLE VI
HASH CALCULATION ENERGY PER DEVICE

| Device | Energy of digital signature [uJ] |
|---|---|
| ESP-WROOM-32 | 6.144 |
| ATMEGA32U4 | 517.017 |
| MSP-EXP430F5529LP | 3946.875 |
| PIC18F25K50 | 16116.408 |
| STM8S207K8T6C | 60093.601 |

By comparing Table VII with Table VIII it can be stated that the proposed version of the simplified chaotic encryption is more energy efficient. The reason behind this is the amount and type of operations needed to implement the encryption method proposed in [5]. Instead of just calculating XOR to encrypt the data, CPU needs to calculate the weights and biases of CNN. This requires many conditional statements ($if$, $else$) in the code and mathematical operations such as addition, substitution, and multiplication. This extends the encryption duration.

TABLE VII
CHAOTIC NEURAL NETWORK ENCRYPTION ENERGY PER DEVICE

| Device | Energy of encryption calculation [uJ] |
|---|---|
| ESP-WROOM-32 | 41.164 |
| ATMEGA32U4 | 3816.345 |
| MSP-EXP430F5529LP | 29133.710 |
| PIC18F25K50 | 118962.648 |
| STM8S207K8T6C | 474310.210 |

TABLE VIII
XOR ENCRYPTION ENERGY PER DEVICE

| Device | Energy of encryption calculation [uJ] |
|---|---|
| ESP-WROOM-32 | 0.410 |
| ATMEGA32U4 | 166.810 |
| MSP-EXP430F5529LP | 1273.412 |
| PIC18F25K50 | 5199.768 |
| STM8S207K8T6C | 17169.601 |

In each examined algorithm the pattern is the same. The less powerful microcontrollers utilize more energy for both encryption and digital signature. This could be assumed as the strict drawback of smaller devices. Their sleep mode energy consumption is also important during long-time operation. In the active mode, the ESP-WROOM-32 works for a longer time period than PIC18F25K50. However, based on Table IX it can be said that in the active-sleep mode, the PIC18F25K50 has a longer duration. Here the fractions of the active-sleep mode are important. It is not true that smaller devices drain less energy from batteries. For instance, 8-bit STM8S207K8T6C in sleep mode consumes about 323 times more energy than 8-bit PIC18F25K50.

TABLE IX
ENERGY CONSUMPTION DURING SLEEP MODE

| Device | Energy of sleep mode [uJ] |
|---|---|
| PIC18F25K50 | 0.102 |
| MSP-EXP430F5529LP | 0.513 |
| ATMEGA32U4 | 0.768 |
| STM8S207K8T6C | 33.215 |
| ESP-WROOM-32 | 51.201 |

## IV. CONCLUSIONS

In this paper, two lightweight methods for providing security to IoT programmable devices were presented. Their confidentiality, integrity, and authenticity were examined. Considered algorithms can be used during machine-to-machine end-to-end communication and subsequently provide different devices with low computation and memory overhead. Both proposed methods are strongly sensitive to initial parameters. This security feature could be used in future research regarding IoT power-safety communication protocol, for instance by implementing the initial parameters agreement method with the usage of the Diffie–Hellman algorithm or similar. In future research, it will be important to examine the energy overhead of the proposed methods in real-world scenarios. We assume that the scope of such tests would be to set up the Sigfox and Lora networks and investigate the run-out of batteries time in different weather and environmental conditions.

## REFERENCES

[1] A. Baalaaji and R. Bevi, "Design of a novel chaotic neural network based encryption system for security applications," *Journal of the Chinese Institute of Engineers*, vol. 44, no. 5, 2021. [Online]. Available: https://doi.org/10.1080/02533839.2021.1919558

[2] A. Roy Chowdhurya, T. Chatterjeeb, and S. DasBit, "Locha: A light-weight one-way cryptographic hash algorithm for wireless sensor network," *Procedia Computer Science*, vol. 32, 2014. [Online]. Available: https://doi.org/10.1016/j.procs.2014.05.453

[3] A. Maetouq, S. Mohd Daud, N. Azurati Ahmad, N. Maarop, N. N. Amir Sjarif, and H. Abas, "Comparison of hash function algorithms against attacks: A review," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 8, 2018. [Online]. Available: https://dx.doi.org/10.14569/IJACSA.2018.090813

[4] D. Berendsen, "A comparative study on signature schemes for iot devices," *Tu Delft*, 2021. [Online]. Available: http://resolver.tudelft.nl/uuid:5d7ffd26-6576-4316-bc4f-81858be48018

[5] H. Kaur and T. S. Panag, "Cryptography using chaotic neural network," *International Journal of Information Technology and Knowledge Management July-December*, vol. 4, no. 2, 2011.

[6] L. Chen, Q. Zhang, J. Ma, and K. Li, "Research on neural network chaotic encryption algorithm in wireless network security communication," *EURASIP Journal on Wireless Communications and Networking*, no. 1, 2019. [Online]. Available: https://doi.org/10.1186/s13638-019-1476-3

[7] M. Wang, X. Wang, Y. Zhang, S. Zhou, T. Zhao, and N. Yao, "A novel chaotic system and its application in a color image cryptosystem," *Optics and Lasers in Engineering*, no. 121, 2019. [Online]. Available: https://doi.org/10.1016/j.optlaseng.2019.05.013

[8] F. Michael, D. David, and N. Christophe, "A fast chaos-based pseudo-random bit generator using binary64 floating-point arithmetics," *Scholarly Journal*, 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01024689

[9] L. Chengqing, L. Shujun, Z. D, and C. GR, "Chosen-plaintext cryptanalysis of a clipped-neural-network-based chaotic cipher," pp. 630–636, 01 2005.

[10] C. Li, S. Li, D. Zhang, , and G. Chen, "Cryptanalysis of a chaotic neural network based multimedia encryption scheme," *Lecture Notes in Computer Science*, 2004. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30543-9_53

[11] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C, 20th Anniversary Edition*. Wiley, 2015.

[12] Z. Al-Odat, E. Al-Qtiemat, and S. Khan, "An efficient lightweight cryptography hash function for big data and iot applications," 10 2020, pp. 66–71.

[13] A. Alfrhan, T. Moulahi, and A.-a. Abdulatif, "Comparative study on hash functions for lightweight blockchain in internet of things (iot)," *Blockchain: Research and Applications*, vol. 2, p. 100036, 11 2021.

[14] L. Wang, "Describe prime number gaps pattern by logistic mapping," 06 2013.

[15] N. Pareek, V. Patidar, and K. Sud, "Cryptography using multiple one dimensional chaotic maps," *Communications in Nonlinear Science and Numerical Simulation*, vol. 10, no. 7, 2005. [Online]. Available: https://doi.org/10.1016/j.cnsns.2004.03.006

[16] M. Alawida, A. Samsudin, J. Teh, and W. Alshoura, "Digital cosine chaotic map for cryptographic applications," *IEEE Access*, vol. 7, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2947561

[17] R. Boriga, A. Dsscslescu, and A. Diaconu, "A new one-dimensional chaotic map and its use in a novel real-time image encryption scheme," *Hindawi*, 2014. [Online]. Available: https://doi.org/10.1155/2014/409586

[18] A. Mansouria and X. Wang, "A novel one-dimensional chaotic map generator and its application in a new index representation-based image encryption scheme," *Information Sciences*, vol. 563, 2021. [Online]. Available: https://doi.org/10.1016/j.ins.2021.02.022

[19] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *NIST Special Publication 800-22, Gaithersburg, MD, US,*, vol. 800, p. 163, 05 2001.