applied sciences

MDPI

Article

SIMECK-T: An Ultra-Lightweight Encryption Scheme for Resource-Constrained Devices

Alin-Adrian Anton *, Petra Csereoka, Eugenia-Ana Capota and Răzvan-Dorel Cioargă

Computer and Information Technology Department, Faculty of Automation and Computing, Politehnica University Timișoara, 2nd Vasile Pârvan Ave., 300223 Timișoara, Romania; petra.csereoka@cs.upt.ro (P.C.); eugenia.capota@cs.upt.ro (E.-A.C.); razvan.cioarga@cs.upt.ro (R.-D.C.)
* Correspondence: alin.anton@upt.ro

Abstract: The Internet of Things produces vast amounts of data that require specialized algorithms in order to secure them. Lightweight cryptography requires ciphers designed to work on resource-constrained devices like sensors and smart things. A new encryption scheme is introduced based on a blend of the best-performing algorithms, SIMECK and TEA. A selection of software-oriented Addition–Rotation–XOR (ARX) block ciphers are augmented with a dynamic substitution security layer. The performance is compared against other lightweight approaches. The US National Institute of Standards and Technology (NIST) SP800-22 Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications and the German AIS.31 of the Federal Office for Information Security (BSI) are used to validate the output of the proposed encryption scheme. The law of iterated logarithm (LIL) for randomness is verified in all three forms. The total variance (TV), the Hellinger Distance (HD), and the root-mean-square deviation (RMSD) show values smaller than the required limit for 10.000 sequences of ciphertext. The performance evaluation is analyzed on a Raspberry PICO 2040. Several security metrics are compared against other ciphers, like χ^2 and encryption quality (EQ). The results show that SIMECK-T is a powerful and fast, software-oriented, lightweight cryptography solution.

Keywords: statistical test suite; lightweight cryptography; law of iterated logarithm; internet of things

check for updates

Academic Editor: Stefan Fischer

Received: 11 November 2024
Revised: 14 January 2025
Accepted: 15 January 2025
Published: 26 January 2025

Citation: Anton, A.-A.; Csereoka, P.; Capota, E.-A.; Cioargă, R.-D. SIMECK-T: An Ultra-Lightweight Encryption Scheme for Resource -Constrained Devices. *Appl. Sci.* **2025**, *15*, 1279. https://doi.org/10.3390/app15031279

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Internet of Things (IoT) is becoming a key technology in various economic sectors such as agriculture [1], healthcare [2] and energy [3]. The Industrial Internet of Things (IIoT) improves manufacturing and industrial processes. The data produced by smart objects is vast and raises many security challenges. One of these is lightweight cryptography, a type of encryption specifically designed for resource-constrained devices.

The vast amount of literature on lightweight ciphers covers solutions that try to work well both in hardware implementations and directly in software. Not a lot of effort has been put into developing secure and lightweight ciphers that are specifically designed for software implementations on devices with very limited resources.

Symmetric key cryptography uses stream and block ciphers to protect sensitive data. Stream ciphers work at the byte level or directly at the bit level, being as light as possible on resources. Block ciphers encrypt a block of data at a time and may require padding.

Although they are the lightest form of hardware-oriented encryption, stream ciphers mimic the one-time pad by using pseudorandom number generation. Advances in

parametrizable parallel pseudo-random number generators [4] result in the creation of parallel stream ciphers [5,6], but also in an increase in hardware production costs.

It is a rare case for software implementations of encryption algorithms to use bit- or byte-level operations. This is because a read or write operation works on a block of data, and this situation means that carefully selected block ciphers the desirable solution.

The introduction of the SPECK and SIMON family of block ciphers was a milestone for lightweight cryptography [7]. SPECK's design philosophy of prioritizing simplicity and efficiency influenced subsequent cryptographic work, making it a key reference point in the development of lightweight encryption algorithms. It demonstrated that strong encryption could be achieved with minimal resource consumption, which was crucial as the demand for security in constrained environments was growing rapidly.

SPECK was second best in a survey covering 19 algorithms with different approaches to secure the Internet of Things [8]. Dinu et al. have shown that Addition–Rotation–XOR (ARX) ciphers are the preferred approach to secure resource-constrained devices [8].

Another notorious family of block ciphers is RC5 [9] and RC6 [10]. RC5 is a block cipher that has a variable word size, a variable number of rounds, and a variable-length secret key. An interesting feature of RC5 is the heavy use of data-dependent rotations [11]. In other words, the rotational operator depends on the plaintext. In our proposed implementation of the encryption scheme, the number of rounds is increased based on the supplied password.

RC6 is an Advanced Encryption Standard (AES) candidate, using a 128-bit block. Such a block size is too large for resource-constrained devices. It is twice as slow as RC5 [9]. Improved RC6 (IRC6) [12] allows for a variable block size and relies on a layer of "Permuted XORed Bytes" (PXB) for the confusion/diffusion network.

ROADRUNNER [13] is a cipher designed specifically for the 8-bit ATtiny45 microprocessor. The rapid increase in IoT applications implies an improved computational capacity for IoT hardware, meaning that 8-bit microprocessors are obsoleted by devices with a larger word size.

The LED [14] block cipher is also hardware-oriented, offering a small silicon footprint. LED is a substitution box-oriented cipher that performs well with hard-coded lookup tables. The implementation of software on devices with very limited resources is not as fast.

Another 64-bit block cipher is PICCOLO [15], also very compact when implemented in hardware. It has two substitution box layers and is considered one of the competitive ultra-lightweight block ciphers for RFID applications. Because it uses a substitution–permutation network, it is not as fast when implemented in software.

PRESENT [16] uses a 4 by 4-bit substitution box, keeping in mind that the implementation of such a substitution box is typically much more compact than that of an 8-bit version. PRESENT is also optimized for small hardware implementations, making it less ideal for software running on microcontrollers.

CHASKEY [17] is an ARX cipher 2.1x faster than SIMON and SPECK, and up to 8.3× faster than AES, according to the Fair Evaluation of Lightweight Cryptographic Systems (FELICS) project [18]. The 12-round version is standardized as ISO/IEC 29192-6:2019. CHASKEY has a 128-bit block using 32-bit words for 32-bit microcontrollers. Although it is fast, the block size is too large for some applications. For example, when 1–2 bytes need to be encrypted, CHASKEY will always encrypt a block of 16 bytes.

The remainder of this paper is organized as follows. Section 1.1 "Motivation and strategy" explains the reasoning and the strategy applied to our work. A point-by-point enumeration of contributions is presented in Section 1.2 "Contributions".

Section 2 "Methodology" describes the methodology used to implement and test the cipher. Section 3 "Encryption Evaluation" presents the theoretical setup for evaluating the security of the encryption system. Section 4 "Results" contains the results obtained.

In Section 5 “Discussion”, the implications of the findings are discussed and further analyzed, and Section 6 “Conclusions” contains the final observations that are drawn with a perspective on future improvements.

1.1. Motivation and Strategy

The scope of the research is to investigate encryption algorithms that can be downgraded to a reduced number of rounds, without compromising their security level. To preserve the cipher’s strength, the reduced-round algorithms are enhanced by inserting new layers of security, and by twining.

SPECK-R [19] is a cipher that combines the round function of SPECK [7], with a key scheduling algorithm to create RC4 substitution boxes [20]. Our public implementation of SPECK-R [21], an algorithm proposed by Sleem et al. in [19], is an improved version using the RC4D key schedule algorithm [22], and a round key of 26 words [23].

Because SPECK-R provides the same security level as SPECK [7], using only seven rounds [19], we apply the same technique for a selection of high-speed, software-oriented, ARX block ciphers. The selected ciphers are fortified with dynamic substitution boxes, an additional security layer that has been used to convert SPECK into SPECK-R [19]. The modified ciphers are based on our previous work with SPECK-R in [21,23].

The output of the augmented ciphers in our selection, using reduced-round versions, is verified to be indistinguishable from random data. This is achieved using NIST SP800-22 [24] and AIS.31 [25,26] standards, and using the law of the iterated logarithm (LIL) [27,28].

The best performing ciphers that we obtain are combined, in a manner not involving any lookup tables.

1.2. Contributions

The proposed SIMECK-T encryption scheme is a combination of SIMECK [29] with internal TEA rounds [30]. The algorithm produces random output at high speeds, making it suitable for low-powered devices.

Without having any lookup tables, the proposed encryption scheme is immune to side-channel attacks.

Performance discontinuities that occur when substitution boxes are dynamically interchanged in SPECK-R [19] are completely avoided.

The proposed algorithm was evaluated using Hash XOR Permutation (HXP) metrics, an encryption scheme proposed by Oladipupo et al. [31], and it was shown that ultra-lightweight encryption schemes can be produced without compromising security levels and without having to rely on substitution boxes. The same metrics have been used by Faragallah et al. for the evaluation of IRC6 [12].

TErrestrial Trunked RAdio (TETRA) TEA3 [32] was investigated for potential use in a resource-constrained environment with reduced rounds. The security level of TEA3 was shown to be unable to be preserved by reducing the number of rounds due to performance issues. The cipher uses a single substitution box as lightweight DES [33] does, but it was shown to be very slow.

The augmented version of SIMECK [29], named SIMECK-R, is shown to be a viable encryption scheme comparable to SPECK-R [19], using the same number of rounds.

The implementation of SIMECK-T is very fast, produces high-quality random output, and is available as free software [34].

1.3. Related Work

The related work consists mostly of ARX block ciphers specifically designed for software implementations. These are considered a new paradigm in lightweight cryptog-

raphy [8]. A detailed survey in [35] presents TEA [30] as the third most energy efficient algorithm [10], third in the least number of cycles [36], first in least cycles, least cycles per byte and combined metric [37], first in code size and second in the Random Access Memory (RAM) requirements [37], third in the cycles/block metric [38], third in the code size metric [38] and, most importantly, the first in speed out of 13 software-oriented ciphers [39].

A good balance of speed, lightness and security is obtained using encryption primitives designed for a 64-bit block size and a key size of 96 or 128 bits.

A more recent survey on lightweight cryptography [40] lists HIGHT [41], SPECK [7], LEA [42], SPARX [43] and CHAM [44] as new software-oriented lightweight block ciphers, where SPECK is considered the fastest, having 0 RAM requirements and only 186 bytes read-only memory (ROM) requirements. SPECK was designed for software implementations, and SIMON [7] was designed to be more efficient in hardware.

SPECK-R is a SPECK variant with only seven rounds where RC4 [20] substitution boxes are used in order to maintain similar security levels to the full round version of the cipher [19]. Although IoT devices have limited memory and may be slow to access it, another approach to lightweight cipher design that has a 4 by 4 substitution box is proposed by Prathiba et al. [45].

TErrestrial Trunked RAdio (TETRA) encryption ciphers like TEA2 and TEA3 developed by the European Telecommunications Standards Institute (ETSI) were made available to the public [46] due to surveillance and backdoor suspicions [32]. These ciphers have been used by important agencies in Europe, but also abroad. Critical infrastructure protection depends on them, but they were available until recently only under a non-disclosure agreement (NDA). The ciphers themselves have been long known to be dependable and are considered to this day unbreakable from a cryptography point of view. However, the implementation vulnerabilities of some versions once again prove that closed-source software is not to be trusted.

HIGHT [41] uses 8-bit instructions, and attacks have been published for all rounds [47,48].

Although oriented on 32-bit instructions, LEA [42] is a 128-bit block cipher, having 24 rounds for a 128-bit key. This makes it inefficient when less than 16 bytes need to be encrypted, as is frequently the case with IoT.

CHAM [44] is very slow using 80 rounds. The revised versions use a minimum of 88 rounds, and the authors claim that the speed in the software is comparable to that of SPECK [49]. But SPECK has 26 rounds for a 128-bit key, and our benchmarks show that SPECK is faster.

SPARX [43] is an ARX cipher with provable limits, designed for 16-bit instructions. It is based on SPECK. The key schedule takes a 128 bit master key and turns it into 96-bit subkeys. Because it is designed for 16-bit microcontrollers, it is not efficient on 32-bit devices.

At the same time that the survey was made available [35], the SIMECK cipher was published [29]. SIMECK combined the best of SPECK and SIMON from the National Security Agency (NSA) [7] into a single, fast lightweight cipher, using only ARX instructions, relying on the AND operator for non-linearity. SIMECK was designed with high performance parallelization in mind [50].

TEA [30] is notorious for its tinyness, having simplicity and memorability as design goals. TEA has been used in parallel implementations for RFID [51] applications or single instruction multiple data (SIMD) environments like graphic processing units (GPUs) [52].

XTEA [53] is a corrected form of TEA, with a larger number of rounds. It is also notably slower. XXTEA [54] is the corrected version of XTEA (also called block TEA or BTEA), which is also slower than TEA.

CHACHA [55] is a 256-bit block cipher that aims to have better diffusion per round compared to SALSA [55]. It is a lightweight cipher, but the block size is not very efficient on resource-constrained devices, where a 64-bit block size is preferable.

THREEFISH [56] is part of the Skein hash function and is a large, flexible cipher. Its ARX operations work on 64-bit words, and it can have a block size of 256, 512, or 1024 bits. The block size is not very advantageous, and the 64-bit instruction set makes it less than ideal for resource-constrained devices.

Lara et al. [57] propose a stream cipher for IoT whose output has been validated to be random using the NIST Statistical Test Suite (STS) [24] and other randomness testing tools. SIMECK-T has been validated to produce random output, using the NIST SP800-22 [24] and AIS.31 [26] standards, and the law of the iterated logarithm (LIL) [27,28].

Oladipupo et al. [31] proposed Hash XOR Permutation (HXP), a lightweight cryptosystem for the encryption of images with cloud storage and the Internet of Things in mind. HXP is hash-based, using a Secure Hash Algorithm (SHA) [58] with a truncated output, which is not optimal. The algorithm was evaluated using various security metrics, showing that security can be preserved without substitution boxes. The same metrics were applied by Faragallah et al. for the evaluation of IRC6 [12].

The SIMECK-T cryptographic scheme will be verified against the security evaluation metrics applied to HXP [31].

As explained in Section 1.1, high-speed software-oriented ARX block ciphers are selected from the related work and modified to produce the same level of security using a reduced number of rounds. We opt for 64-bit block ciphers with a key size of 128 bits, in order to maintain a good balance between speed, lightness, and security. NSA-based SPECK and SIMON [7], and SIMECK [29], on one hand, and TEA [30], XTEA [53] and XXTEA [54] primitives, on the other, will be combined with RC4D [22] substitution boxes as demonstrated by the original SPECK-R [19] which uses RC4 [20]. SIMON is more appropriate for hardware implementations but is included for completeness.

2. Methodology

This section introduces the theoretical model of the encryption system, followed by implementation details. The evaluation methodology for assessing the cipher strength and the various statistical tests used for evaluating the security of the encryption system are described.

2.1. Theoretical Model

The SIMECK round function is described by Equations (1) and (2).

$$\text{SIMECK}(L_i, R_i) = (R_i \oplus f(L_i) \oplus K, L_i) \tag{1}$$

where

$$f(L_i) = (L_i \odot (L_i \lll 5)) \oplus (L_i \lll 1) \tag{2}$$

The TEA round function is defined in Equation (3):

$$TEA_{K_i}(L_i, R_i) = (R_i + ((L_i \ll 4 \oplus L_i \gg 5) + L_i) \oplus (K_i + \Delta), L_i) \tag{3}$$

where

- L_i is the left part of the input.
- R_i is the right part of the input.
- K_i is the round key.
- Δ is the key schedule constant ($\Delta = 0x9e3779b9$).

In order to apply 3 rounds of SIMECK each including 5 cycles of TEA, Equation (4) defines $SIMECK_3$:

$$\text{SIMECK}_3(L_i, R_i) = \text{SIMECK}_2(\text{SIMECK}_1(\text{SIMECK}_0(\text{TEA}_5(L_i, R_i)))) \quad (4)$$

where each SIMECK function call includes the TEA function in Equation (6):

$$\text{SIMECK}_j(L_i, R_i) = \text{SIMECK}_{\text{TEA}_5}(L_i, R_i) \quad (5)$$

for each round j from 0 to 2.

$$\text{TEA}_5(L_i, R_i) = \text{TEA}_4(\text{TEA}_3(\text{TEA}_2(\text{TEA}_1(\text{TEA}_0(L_i, R_i))))) \quad (6)$$

2.2. Implementation

As with SPECK-R, SIMECK-T is implemented to operate in block counter mode [59]. An initialization vector (IV) is obtained by initializing the Rand1 pseudorandom generator from [60] using the file size, the filename of the output file and the modification time of the source file as the seed. The Fowler-No-Vo 1a hash on 64 bits is used to convert the filename and the file modification time string into numbers [61]. The generator is queried a number of *psum* + *pmul* times, where these two values depend on the derived password obtained using a password-based key derivation scheme. This makes it improbable for files with the same filename and the same file size to make use of the same pair of (*IV*, *password*) while using the same modification time attribute. The IV is used as a starting value to set the cipher in counter mode—line 4 in the Algorithm 1.

SIMECK-T from Algorithm 1 uses LROT32 and ROUND64 functions from SIMECK [62]. Lines 11–12 depend on the hash of the password. The constants *psum* and *pmul* are produced based on the sum and product of the bytes in the derived key, respectively, on the XOR of its byte values. This means that the SIMECK-T algorithm is tweaked so that an attacker cannot know upfront the number of external SIMECK rounds and the number of internal TEA rounds.

In order to validate the proposed encryption algorithm against other suitable ciphers for resource-constrained devices, a selection of algorithms that are well known for their simplicity and efficiency are modified using the strategy already proven by the authors of SPECK-R [19]. The encryption primitives are combined with dynamic substitution boxes which are designed to retain the same cryptographic properties for the cipher but without having to use the full number of rounds.

SPECK-R authors use RC4 as a key scheduling algorithm (KSA) to produce 3 substitution boxes [19]. This algorithm is known to contain vulnerabilities as described in [63–66]. In order to mitigate for that, it was replaced by a secure version called RC4D [20,22]. Three substitution boxes are used exactly as shown in [19], but where applicable, the full expanded key is used because the expanded keys are important for the SPECK and SIMON ciphers [7]. The 7-round SPECK-R cipher uses a round key truncated to 7 32-bit words despite the fact that the SPECK cipher on 64 bits using a 96-bit key size requires a 26-word round key. Consequently, in our implementation [21], even though there are only 7 rounds of SPECK-R, a full 26-word round key is used by cycling the full length of the expanded key.

Algorithm 1: The proposed SIMECK-T algorithm [34]

Input: master_key[4], plaintext[2]
Output: ciphertext[2]

Initialize:
simeckrounds ← 3
tearounds ← 5
static cnt ← IV
plain[2] ← split_uint64_to_uint32(cnt) ; `// block counter mode encrypts cnt`
keys[4] ← {master_key[0], master_key[1], master_key[2], master_key[3]}
ciphertext[0] ← plain[0]
ciphertext[1] ← plain[1]
constant ← 0xFFFFFFFC
sequence ← 0x938BCA3083F
simeckrounds ← simeckrounds + psum ; `// depends on password`
tearounds ← tearounds + pmul ; `// depends on password`
for *idx* ← *0* **to** *simeckrounds - 1* **do**
    ROUND64(keys[0], ciphertext[1], ciphertext[0], temp);
    constant ← constant ∧ 0xFFFFFFFC;
    constant ← constant ∨ (sequence ∧ 1);
    sequence ← sequence » 1;
    ROUND64(constant, keys[1], keys[0], temp);
    Rotate keys left ; `// keys[1]` → `keys[2], keys[2]` → `keys[3], keys[3]` → `keys[1]`
    y ← ciphertext[0];
    z ← ciphertext[1];
    sum ← 0;
    delta ← 0x9e3779b9 ; `// a key schedule constant`
    for *tearounds* ← *tearounds - 1* **to** *0* **do**
        sum ← sum + delta;
        y ← y + ((z « 4) + keys[0]) ⊕ (z + sum) ⊕ ((z » 5) + keys[1]);
        z ← z + ((y « 4) + keys[2]) ⊕ (y + sum) ⊕ ((y » 5) + keys[3]);
    ciphertext[0] ← y;
    ciphertext[1] ← z;
ciphertext[0] ← ciphertext[0] ⊕ plaintext[0] ; `// block counter mode`
ciphertext[1] ← ciphertext[1] ⊕ plaintext[1] ; `// block counter mode`
cnt ← cnt + 1 ; `// block counter mode`

The TEA [30], XTEA [53], XXTEA [67], SPECK [7], SIMON [7], and SIMECK [29] ciphers are used to encrypt a 1 GB file that contains only bits of zero. The selection of these two families of ciphers, TEA and SPECK, where SIMECK is also accounted for to be part of the SPECK family by incorporating SIMON, is not arbitrary.

The tinyness of TEA is unequivocally unique in the history of cryptography. This family of ciphers has received widespread public scrutiny and adoption in many industries. TEA emerged as the fastest cipher in literature surveys [35]. A more recent survey presents SPECK as the fastest cipher [40]. The public scrutiny and adoption of SPECK as a lightweight cryptography solution is also taken into account. The US National Security Agency's family of ciphers was not only a milestone in modern cryptography, the reputed agency's algorithms, SPECK and SIMON, were also a heads-up call for derivative work to appear, reviving the design and use of ARX ciphers.

Derived by enhancing these ciphers using the dynamic substitution layer of SPECK-R [19] with RC4D [22], the ciphers called SIMON-R, SIMECK-R, TEA-R, XTEA-R, and XXTEA-R are obtained. These are also used to encrypt the 1 GB file containing bits set to zero.

NIST recommends a password-based key derivation function (PBKDF). Memory hard techniques such as balloon hashing [68] and Argon2 [69] are incompatible with devices with limited resources.

Algorithm 2 uses PBKDF2 [70] with 65000 iterations to produce a derived key based on the given password. The counter-mode initialization vector (IV) is computed in a way that prevents the same pair (IV, password) to be used twice. The file modification time string of the input file is converted into a Fowler-No-Vo (FNV) hash [61], and this is used as a seed for a pseudorandom number generator [60]. Since *psum* and *pmul* are checksums based on the derived key, these two variables are used, along with the FNV hash of the filename (without the extension), to obtain a unique IV on lines 12–13.

Algorithm 2: Derived Key Processing with PBKDF2

Input: Password *passwd*

Output: Processed values *psum* and *pmul*

```
1  PBKDF2(passwd, derived_key, 65000) ;          // Derive the key with PBKDF2
2  psum ← 0 ;
3  pmul ← 1 ;
4  for i ← 0 to 3 do
5  |   psum ← psum + derived_key[i] ;      // Addition with overflow by design
6  |   pmul ← pmul ⊕ derived_key[i] ;           // Multiplication with overflow
7  psum ← psum mod 10 ;
8  pmul ← pmul mod 5 ;
9  IV ← fnv1a_hash(file_modification_time_str) ;
10 IV ← IV + fsize + psum + pmul + fnv1a_hash(filename) ;
11 srand(IV) ;
12 for i ← 0 to psum + pmul − 1 do
13 |   IV ← rand () ;
```

A Meyer-Schilling detection code (MDC) using SIMECK-T is shown in Algorithm 3. This code is used to build the PBKDF2 function in Algorithm 4.

In resource-constrained environments, it is not always easy to access file modification times and sometimes filenames are not available at all. If this is the case, the IV should simply be a random value, unrelated to the password. If this is manageable, the derived key can also be configured to a random value, without the need for PBKDF2 [70].

Algorithm 3: MDC-2 Hash Function using SIMECK-T Cipher for Encryption

Input: Data array *data* of length *len*, key array *key*
Output: Hash array *hash* with two 32-bit values

$\mathsf{Pt} \leftarrow \{0,0\}$; // Initialize plaintext block
$\mathsf{Ct} \leftarrow \{0,0\}$; // Initialize ciphertext block
$\mathsf{K1}, \mathsf{K2} \leftarrow 0, 0$; // Initialize hash values
for $i \leftarrow 0$ **to** $len - 1$ **do**
  $\mathsf{Pt}[i \bmod 2] \leftarrow \mathsf{Pt}[i \bmod 2] \oplus \left(\text{uint32}(data[i]) \ll ((i \bmod 2) \times 8)\right)$;
  if $(i+1) \bmod 2 = 0$ **then**
    simeckTeaECB(key, Pt, Ct) ; // Encrypt Pt using SIMECK-T
    $\mathsf{K1} \leftarrow \mathsf{K1} \oplus \mathsf{Ct}[0]$;
    $\mathsf{K2} \leftarrow \mathsf{K2} \oplus \mathsf{Ct}[1]$;
    $\mathsf{Pt} \leftarrow \{0,0\}$; // Reset plaintext block
if $len \bmod 2 \neq 0$ **then**
  simeckTeaECB(key, Pt, Ct) ;
  $\mathsf{K1} \leftarrow \mathsf{K1} \oplus \mathsf{Ct}[0]$;
  $\mathsf{K2} \leftarrow \mathsf{K2} \oplus \mathsf{Ct}[1]$;
for $j \leftarrow 0$ **to** 15 **do**
  $\mathsf{K1} \leftarrow \mathsf{K1} + \left((\mathsf{K2} \wedge 3) \times 0x9e3779b9\right) \oplus \left((\mathsf{K2} \gg 5) + 0x9e3779b9\right)$;
  $\mathsf{K2} \leftarrow \mathsf{K2} + \left((\mathsf{K1} \wedge 3) \times 0x9e3779b9\right) \oplus \left((\mathsf{K1} \gg 5) + 0x9e3779b9\right)$;
$hash[0] \leftarrow \mathsf{K1}$;
$hash[1] \leftarrow \mathsf{K2}$;

Algorithm 4: PBKDF2 Key Derivation using MDC-2 based on SIMECK-T

Input: Password *password* with length *password_len*, Salt array *salt* with length *salt_len*, Number of iterations *iterations*
Output: Derived key array *key*

result $\leftarrow \{0,0\}$; // Initialize result hash values
temp $\leftarrow \{0,0\}$; // Temporary hash storage
temp_buffer $\leftarrow$ array of size 4 + salt_len + 4 ; // Buffer for data preparation
for $i \leftarrow 1$ **to** *iterations* **do**
  // Prepare data for hash computation
  memcpy(temp_buffer, &i, sizeof(uint32_t)) ; // Copy block index
  memcpy(temp_buffer + sizeof(uint32_t), salt, salt_len) ; // Copy salt
  memcpy(temp_buffer + sizeof(uint32_t) + salt_len, &i, sizeof(uint32_t)) ; // Copy block index again
  // Perform hash computation
  MDC2_Hash(temp_buffer, sizeof(TempBuffer), temp, (const uint32_t*)password) ;
  // XOR result with temporary hash
  result [0] $\leftarrow$ result [0] $\oplus$ temp [0] ;
  result [1] $\leftarrow$ result [1] $\oplus$ temp [1] ;
// Copy the result to the derived key
memcpy(key, result, 2 $\times$ sizeof(uint32_t)) ;

3. Encryption Evaluation

This section thoroughly presents the theoretical setup to evaluate the security of the encryption system. The randomness of the encrypted output produced by the proposed algorithm is assessed using standardized statistical tests. The cipher strength is also evaluated using cryptanalysis and statistical evaluation metrics.

3.1. Randomness of the Output

In order to compare the new ciphers with different numbers of rounds, the National Institute of Standards and Technology (NIST) Statistical Test Suite [24,71] and the AIS.31 [25,26,72] standard tests are used. The best cipher is selected and compared with the proposed SIMECK-T algorithm and against the TErrestrial Trunked RAdio (TETRA) TEA3 cipher [32], which produces random output when only four rounds are used.

The law of the iterated logarithm (LIL) is also used because it was proven to detect nonrandom characteristics when standard tests fail to do so [27]. LIL successfully detects the randomness vulnerability in Debian GNU/Linux distributions that plagued many software tools and services with cryptographic primitives [28].

A cipher is considered strong as long as the output is indistinguishable from true random data [73]. Therefore, stating how well cryptographic augmentations on reduced-round ciphers manage to produce random output is critical when selecting primitives for creating new ciphers.

3.2. Cipher Strength

An ideal cipher is unbreakable by other means than brute force. For a decent key size, this is an unfeasible attack that can take longer than the age of the universe. Password-based key derivation functions, as specified by NIST SP 800-132 [70], can be used to defend against weak passwords. Memory hard functions (MHFs) such as Argon2 also protect against this kind of attack [69].

When a reduced-round version of a cipher can be approximated with a linear function, or when a trail can be detected throughout the randomness of the output in relation to the input with a certain bias, attacks that can break the cipher can be created. When these vulnerabilities escalate to a large number of rounds, up to the full round cipher, the algorithm is considered fully broken.

Anything that can help an attacker distinguish the specific cryptographic algorithm from true random data is a characteristic of that cipher. Consequently, it is important that a cipher does not have any.

3.2.1. The Avalanche Effect

The avalanche effect is a property of secure block ciphers, which requires that for any bit that is flipped in the input, half of the output bits should be changed. The strict avalanche effect requires that exactly 50% of the output bits be flipped, but this is hardly met by any of the available ciphers, as a probabilistic feature.

Equation (7) defines the avalanche effect. For 64-bit ciphers, this criterion requires that on average 32 bits are flipped in the output whenever a single random bit is changed in the input.

$$AE(\%) = \frac{\#\,of\ changed\ bits\ in\ ciphertext}{total\ \#\,of\ bits\ in\ ciphertext} \times 100 \tag{7}$$

3.2.2. Linear Cryptanalysis

Listing 1 shows the bias calculation function for finding linear approximations of SIMECK-T.

Algorithm 5 was proposed by Mitsuru Matsui [74] in 1994 to find statistical biases over a large number of plaintext and encrypted block pairs such that possible linear approximations of the encryption function can be detected.

Algorithm 5: Mitsuru Matsui's [74] first algorithm for guessing $c \cdot k$ according to Pascal Junod [75]

Input: An oracle Ω, a data complexity ν (number of pairs), a, b, c, ϵ (bias)
Output: A guess about $c \cdot k$

Initialize a counter $\hat{m}$ to 0;
for $i \leftarrow 1$ ***to*** ν **do**
  Generate a random plaintext x_i;
  Ask x_i from Ω and get $y_i = f_k(x_i)$;
  if $a \cdot x_i \oplus b \cdot y_i = 0$ **then**
    $\hat{m} = \hat{m} + 1$;
  end
end
if $\epsilon > 0$ **then**
  if $\hat{m} > \frac{\nu}{2}$ **then**
    Output "$c \cdot k = 0$";
  end
  else
    Output "$c \cdot k = 1$";
  end
end
else
  if $\hat{m} > \frac{\nu}{2}$ **then**
    Output "$c \cdot k = 1$";
  end
  else
    Output "$c \cdot k = 0$";
  end
end

Initially used on the Data Encryption Standard (DES) algorithm, Matsui broke the 8-round DES with a number of 2^{21} pairs and the full-round DES with 2^{47} pairs. The higher the number of pairs, the better the accuracy of the algorithm.

Using a linear approximation attack on SIMECK requires a minimum of 2^{24} pairs for the 32/64 version [76], while SIMECK with a 128-bit key requires at least 2^{60} [76,77].

A linear attack on TEA requires a minimum of 2^{62} pairs [78,79].

Listing 1. Bias calculation function for linear approximations of SIMECK-T.

```
long int calculate_bias(uint32_t *plain0, uint32_t *
    plain1,
uint32_t *cipher0, uint32_t *cipher1, long int
    num_pairs,
int guess_key) {
    long int count = 0;
    for (uint32_t i = 0; i < num_pairs; i++) {
        int p_pos = rand() % 32;
        int c_pos = rand() % 32;
        int k_pos = rand() % 8;

        int p_bit = (plain0[i] >> p_pos) & 1;
        int c_bit = (cipher1[i] >> c_pos) & 1;
        int key_guess_bit = (guess_key >> k_pos) & 1;

        if ((p_bit ^ c_bit) == key_guess_bit) {
            count++;
        }
    }
    return count - (num_pairs / 2);
}
```

The largest experimental linear attack we attempted against the proposed SIMECK-T cipher has 2^{32} pairs, and the source code for the tool is available at [34] and requires 60 GB of free hard disk space.

3.2.3. Differential Cryptanalysis

Algorithm 6 shows the corresponding distinguisher used to identify differential trails. For such a distinguisher to be effective, a large number of randomly sampled pairs should be available.

Algorithm 6: Generic Distinguisher for Permutations based on Pascal Junod [75]

Input: An oracle Ω implementing an unknown permutation U on $\mathbb{B}$, a complexity v, a differential (a, b), an acceptance function accept.

Output: A decision bit.

Initialize $u \leftarrow 0$;
for $i \leftarrow 1$ **to** v **do**
    Pick x uniformly at random;
    Query Ω for $U(x)$ and $U(x+a)$;
    if $U(x+a) = U(x) + b$ **then**
        $u \leftarrow u + 1$;
return accept(u)

Listing 2 constructs the hash table to count difference pairs in SIMECK-T.

Listing 2. Hash table algorithm for counting difference pairs during differential cryptanalysis of SIMECK-T.

```
// Hash table size
#define TABLE_SIZE 10000007 // Prime number for hash table size
DifferenceEntry *hash_table;

// Hash function for the differences
unsigned int hash_function(uint32_t diff1, uint32_t diff2) {
    return (diff1 * 31 + diff2) % TABLE_SIZE;
}

// Insert or increment the counter
void insert_or_increment(uint32_t diff1, uint32_t diff2) {
    unsigned int index = hash_function(diff1, diff2);
    while (hash_table[index].count > 0) {
        if (hash_table[index].diff1 == diff1
        && hash_table[index].diff2 == diff2) {
            hash_table[index].count++;
            return;
        }
        index = (index + 1) % TABLE_SIZE;
    }
    hash_table[index].diff1 = diff1;
    hash_table[index].diff2 = diff2;
    hash_table[index].count = 1;
}
```

XOR-Based Differences

Differential cryptanalysis is a method used to analyze the security of symmetric key ciphers, particularly block ciphers. This technique focuses on how differences in plaintext input can affect differences in the resulting ciphertext output. By examining the relationship between these differences, cryptanalysts can identify vulnerabilities in the encryption algorithm.

In differential cryptanalysis, the difference between two plaintexts P_1 and P_2 is often calculated using the following XOR operation:

$$\Delta P = P_1 \oplus P_2.$$

A differential pair is defined as a pair of plaintexts together with their corresponding ciphertexts:

$$(P_1, C_1) \quad \text{and} \quad (P_2, C_2),$$

where C_1 and C_2 are the ciphertexts obtained after encrypting P_1 and P_2 using a secret key.

The difference between the ciphertexts is computed as follows:

$$\Delta C = C_1 \oplus C_2.$$

The key focus of differential cryptanalysis is to understand how differences propagate through the encryption process. Observing how an input difference ΔP leads to an output difference ΔC, cryptanalysts can identify patterns that can indicate weaknesses in the cipher.

Cryptanalysts study the frequency distribution of output differences for a given input difference. If certain output differences occur more frequently than expected, it may reveal vulnerabilities in the cipher's structure.

A differential characteristic describes the probability that a specific input difference ΔP will produce a specific output difference ΔC. Analyzing these characteristics helps assess the security of the cipher.

Rotational Cryptanalysis

Rotational differences capture the effect of cyclic shifts in the position of the bits. Kovratovich et al. [80] introduces a rotational cryptanalysis of the cipher THREEFISH [56].

When RC5 [9] was published, the authors stated that 100 million trials with random pairs of keys and text were used to check whether the data-dependent rotation amounts were related to every bit in the text. The test verified whether flipping a bit in the text determined that some intermediate rotation was a rotation of a different amount [9]. In our implementation of differential cryptanalysis, the largest number of random pairs used for finding trails is 10^7.

The result of rotating a value v_1 left by bits k, denoted as v_1 ROL k, shifts all bits k to the left and wraps any overflow bits around to the right end.

The result of rotating a value v_2 right by bits k, denoted as v_2 ROR k, shifts all bits k to the right and wraps any overflow bits around to the left end.

The rotational difference between v_1 and v_2 can be defined as follows:

$$\Delta v_{\text{rot}} = (v_1 \text{ ROL } k) - (v_2 \text{ ROR } k)$$

or, in code-like notation,

$$\text{result}[0] = (v_1[0] \text{ ROL } k) - (v_2[0] \text{ ROR } k)$$

This rotation-based differential provides insight into how the bit positions shift through the rounds of a cipher. The primary goal of rotational difference cryptanalysis is to understand how small changes in rotation impact the propagation of differences.

Additive Cryptanalysis

Differential cryptanalysis based on additive operator introduces an approach for analyzing block ciphers by examining how differences defined by addition propagate through the cipher. Unlike traditional XOR-based differentials, additive differentials use integer addition or subtraction to define and track differences, taking into account modular arithmetic properties. This method is particularly useful in ciphers that involve modular addition or rely heavily on arithmetic operations.

In additive differential cryptanalysis, the difference between two values v_1 and v_2 is defined as follows:

$$\Delta v_{\text{add}} = (v_1 - v_2) \bmod 2^{32}$$

This expression, equivalent to computing

$$\text{result}[0] = (v_1[0] - v_2[0]) \wedge 0xFFFFFFFF$$

in practical implementations, ensures that the result wraps around modulo 2^{32} (32-bit unsigned integer arithmetic). This definition captures the signed difference between two values under modular conditions. Additive differentials are particularly relevant for analyzing ciphers where addition is a primary operation, such as in Feistel networks or ARX (Addition–Rotation–XOR) cipher designs.

Additive differentials introduce unique properties compared to XOR-based differentials. Unlike XOR, which captures bitwise differences, additive differentials focus on the cumulative effect of subtracting values under a modulus. This accumulation highlights specific propagation patterns in ciphers structured around integer addition.

When defined with modular subtraction (mod 2^{32}), additive differences exhibit wrap-around effects as the values reach the boundary of 2^{32}. This wrap-around affects the propagation of differences, particularly in ARX-based designs where modular addition is central to the cipher.

OR-Based Cryptanalysis

The effect of the OR operator on pairs of bits introduces a distinct set of properties that affect propagation patterns.

For two binary values X_1 and X_2, we define the OR difference as follows:

$$\Delta X_{\text{OR}} = X_1 \mid X_2$$

where $\mid$ represents the bitwise OR operator and ΔX_{OR} provides a "union" of bits where at least one bit is set.

Unlike XOR, where each bit can cancel out, OR-based differences tend to accumulate bits. This affects the diffusion process of the cipher. Once a bit is set to 1 in an OR-based difference, it remains set through all operations unless explicitly reset by the cipher's structure. OR-based differences are harder to "undo", meaning that patterns persist more strongly across rounds but may lack some of the subtlety XOR offers in canceling bits.

AND-Based Cryptanalysis

Unlike XOR, which captures bitwise differences, the AND operator focuses on the overlap of bit positions set to 1 in both inputs. Thus, the AND-based difference between X_1 and X_2 is given by the following:

$$\Delta X_{\text{AND}} = X_1 \wedge X_2$$

where $\wedge$ denotes the bitwise AND operator.

AND-based differences only retain bits set to 1 in both values. This property results in an accumulation effect similar to OR-based differences, but only for bits that are jointly 1 in both inputs. Only bits that are consistently 1 across both inputs remain set in ΔX_{AND}. Differences in zeroed bits (0s) are "ignored", meaning the propagation will depend strongly on consistently shared bits. Since AND-based differences do not have cancellation properties, differences in shared bits tend to persist across rounds, accumulating in predictable patterns.

3.3. Statistical Evaluation Metrics

Oladipupo et al. proposed a series of metrics to evaluate the security of the Hash XOR Permutation (HXP) lightweight cryptosystem for images [31]. The metrics have been used by Faragallah et al. in Improved-RC6 (IRC6) [12], to evaluate the security of IRC6, where diffusion/confusion is obtained through a "Permuted XORed Bytes" (PXB) layer. A

random file of any kind can be seen as a grayscale image with pixel values of 0.255 and a size of $M \times N$. When $M = N = 32768$, a 1 GB file is obtained.

In order to investigate the security of the proposed SIMECK-T algorithm, the metrics from the work of Oladipupo are applied to an encrypted file of 1 GB [31]. The following paragraphs describe the metrics involved and their significance.

3.3.1. Chi-Squared

In a random (securely encrypted) file, each byte should have equal probability, with a uniform distribution close to 1/256 for each value. The test was introduced by Vaudenay [73] as a complementary method for linear and differential cryptanalysis.

The critical chi-squared value for a given degree of freedom (df) and significance level (α) is obtained from statistical software or tables. In our case, n = 256 is the number of values that a byte can take. The critical chi-squared value for $df = n - 1 = 255$ and $\alpha = 0.05$ is approximately 293.248. This means that if the calculated chi-squared statistic is greater than 293.248, we reject the null hypothesis at the 0.05 significance level.

Equation (8) is used to validate the uniformity of histograms. Histograms are graphical representations of the distribution of byte levels in pixels of images [31,81]. If the cipher image has a uniform histogram, it indicates a secure encryption scheme [31,82].

$$\chi^2 = \sum_{i=1}^{256} \frac{(O_i - E)^2}{E} \tag{8}$$

where i is a pixel value and O_i is the observed frequency of each pixel value. For a grayscale $M \times N$ image, the expected frequency is $E = \frac{MxN}{256}$. If the result of this equation is greater than 0.05, the histogram is uniform; otherwise, the histogram is not uniform.

For an ideal random file (256 bins), the chi-squared score would be close to 255 but, due to the small file deviations, this may vary.

3.3.2. Encryption Quality

Equation (9) represents the encryption quality (EQ) of the cryptosystem.

$$EQ = \sum_{L=0}^{255} \frac{(f_L(C) - f_L(P))^2}{256} \tag{9}$$

where P is the plain image and C is the cipher image, both of equal size. L gives the range of values for each byte in P and C. Functions $f_L(C)$ and $f_L(P)$ represent the occurrence of each byte value, L, in P and C.

3.3.3. Mean Square Error

Equation (10) gives the mean square error (MSE). This is the distance between the input, P and the output, C. A high value of MSE means better encryption quality.

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (P(i,j) - C(i,j))^2 \tag{10}$$

where P is the plain image and C is the cipher image, both of size $M \times N$. (i,j) represents the grid location of pixels $P(i,j)$ and $C(i,j)$.

Equation (11) represents the peak signal-to-noise ratio (PSNR) and is used to measure image fidelity. It compares the cipher image (C) and the plain image (P) in terms of quality.

3.3.4. Peak Signal-to-Noise Ratio

$$PSNR = 10\log_2\left(\frac{I^2{}_{\max}}{MSE}\right) \tag{11}$$

where I_{max} is the highest pixel value.

3.3.5. Structural Similarity Index Measure

Equation (12) gives the measure of the structural similarity index (SSIM) between two images, P and C. These values range from −1 to +1. High values imply a high similarity between the two images, while low values indicate a low similarity.

$$\text{SSIM}(P,C) = \frac{(2\mu_p\mu_c + D_1)(2\sigma_{pc} + D_2)}{\left(\mu_p^2 + \mu_c^2 + D_1\right)\left(\sigma_p^2 + \sigma_c^2 + D_2\right)} \tag{12}$$

where μ_p gives the luminance, which is the average of all the pixel values, and σ_p is the standard deviation of all the pixel values. D_1 and D_2 represent constant to steady division with the feeble denominator. If the SSIM between the plain image (P) and the cipher image (C) is low and close to −1, the system is secure.

3.3.6. Normalized Cross-Correlation

Equation (13) is the normalized cross-correlation (NCC). This metric also ranges from −1 to +1, and it measures the similarity between two images P and C. Values close to −1 indicate there is a strong correlation between P and C, while values close to +1 indicate a weak, or no, correlation between the two images. A good cipher is represented by values close to +1.

$$NCC = \frac{1}{M \times N} \times \frac{\sum_{i,j}^{M,N}\left(P_{i,j} - \mu_p\right) \times \left(C_{i,j} - \mu_c\right)}{\sqrt{\text{var}_P \times \text{var}_C}} \tag{13}$$

where M and N give the height and width of images P and C (they should have the same size) and $P_{i,j}$ is the pixel of image P at row i and column j. μ_p and μ_c are the means of the pixels in images P and C, respectively, while var_P and var_C represent the variances of images P and C, respectively.

3.3.7. Mean Absolute Error

Equation (14) defines the mean absolute error (MAE). If the value of MAE is greater than 75, the image cryptosystem is considered secure. The higher the value of MAE, the more secure the cryptosystem.

$$MAE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}|P(i,j) - C(i,j)| \tag{14}$$

where P(i, j) and C(i, j) are pixels at row i and column j of images P and C, respectively.

3.3.8. Normalized Absolute Error

Equation (15) represents the normalized absolute error (NAE). This is used to compare the quality of an image (C) to that of a reference image (P). A high value indicates a low image quality, while a low value means a high image quality.

$$\frac{\sum_{i,j=1}^{M,N}(|P(i,j) - C(i,j)|)}{\sum_{i,j=1}^{M,N}P(i,j)} \tag{15}$$

3.3.9. Maximum Difference

Equation (16) gives the maximum difference (MD), between two pixels at the grid location (i, j), of the plain image P and the cipher image C. A high value implies that the cryptosystem is more secure than a low value.

$$MD = MAX(P(i,j) - C(i,j)) \tag{16}$$

3.3.10. Average Difference

Equation (17) defines the average difference (AD). Given the random output of the cipher, a high value means that the cryptosystem is not secure, while a lower value indicates a secure cryptosystem.

$$AD = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(P(i,j) - C(i,j)) \tag{17}$$

3.3.11. Structural Content

Equation (18) defines the formula for calculating the structural content (SC) between the original image, P, and the processed image, Pm. A value of 1 indicates a high quality of the processed image, while a higher value implies a low quality of the processed image. This equation is a measure of the degradation of the processed image when compared to an ideal image, P. For a secure cryptosystem, the value of SC should be high.

$$SC = \frac{\sum_{i,j}^{M,N}\left(P_{i,j}\right)^2}{\sum_{i,j}^{M,N}\left(Pm_{i,j}\right)^2} \tag{18}$$

3.3.12. Two-Dimensional Correlation Coefficient

Equation (19) represents the two-dimensional correlation coefficient (2DCC). This coefficient can be used to calculate the difference or similarity between two images. A low value (or zero) means that the two images are different, while a high value implies that the two images are similar.

$$2DCC = \frac{\sum_{i,j}^{M,N}\left(P_{i,j} - \mu_p\right) \times \left(C_{i,j} - \mu_c\right)}{\sqrt{\left(\left(\sum_{i,j}^{M,N}\left(P_{i,j} - \mu_p\right)^2\right) \times \left(\sum_{i,j}^{M,N}\left(C_{i,j} - \mu_c\right)^2\right)\right)}} \tag{19}$$

where P(i, j) and C(i, j) are pixels at row i and column j of images P and C, respectively, while μ_p and μ_c are the averages of the pixels in P and C, respectively.

3.3.13. Next Pixel Change Rate

Equations (20) and (21) define the net pixel change rate (NPCR) [31,83] and the unified average change intensity (UACI) [31,84], respectively. These formulas are used to measure the resistance of a cipher to cryptanalysis attacks between two images P and C. The ideal values for the two metrics are 99.6093% for NPCR and 33.4635% for UACI, respectively [31,85].

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M \times N} \tag{20}$$

$$UACI = \frac{\sum_{i,j} \frac{E(i,j)}{255}}{M \times N} \tag{21}$$

where $D(i,j) = 0$ if $P(i,j) = C(i,j)$; otherwise, $D(i,j) = 1$ and $E(i,j) = abs(P(i,j) - C(i,j))$.

3.3.14. Correlation Coefficient

Equation (22) defines the correlation coefficient of adjacent pixels in an image [31,86]:

$$\begin{cases} \mathrm{E}(x) = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad D(x) = \frac{1}{n}\sum_{i=1}^{n}(xi - E(x))^2 \\ \gamma x,y = \frac{\mathrm{cov}(x,y)}{(\sqrt{D(x)})(\sqrt{D(y)}}, D(x) \neq 0 \text{ and } D(y) \neq 0 \\ \mathrm{cov}(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - E(x))(y_i - E(y)) \end{cases} \tag{22}$$

where x_i represents the grayscale value of a pixel, n is the number of pairs (x_i, y_i), and E(x) and E(y) give the mean values of x_i and y_i, respectively. The result in the case of a secure encryption scheme should be close to 0 [31,87].

3.3.15. Entropy

Equation (23) measures the entropy of an image cryptosystem.

$$H(m) = \sum_{i=0}^{2^N-1} P(m_i)\log_2\left(\frac{1}{P(m_i)}\right) \tag{23}$$

where m is the image, 2^N is the pixel sample space, m_i represents pixel i of image m and $P(m_i)$ is the probability of m_i. The entropy H(m) of an image encrypted with a 2^N pixel sample space is N. Therefore, the cryptosystem is resistant to entropy attacks if the result of the entropy analysis H(m) is close to N.

3.4. Performance Evaluation on Raspberry PICO

The performance evaluation of the proposed SIMECK-T encryption scheme was implemented on a Raspberry PICO with a 2040 processor shown in Figure 1.

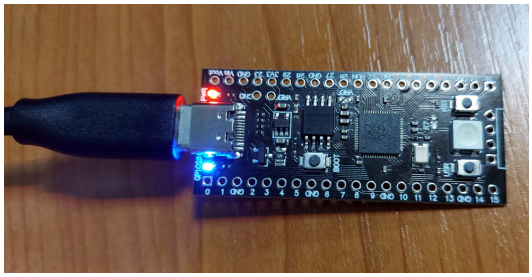


Figure 1. Raspberry PICO 2040 used for benchmarking SIMECK-T operations.

The code was compiled for the ARM architecture and installed bare metal on the device. The groups of measurements are performed with microsecond precision and the average is presented in the corresponding results section.

4. Results

4.1. Randomness of the Output

4.1.1. Analysis with NIST Statistical Test Suite

Table 1 shows the total number of failed NIST randomness tests for full-round ciphers. Lower values are better and are shown with a green color. The statistical test suite [71] comprises 188 tests and higher values are bad and are shown in red. The SPECK and TEA families are winners, with TEA failing 0 tests in 6 rounds.

Table 1. NIST Statistical Test Suite—number of failed randomness tests for full round ciphers (lower is better).

SIMECK	SIMON	SPECK	TEA	XTEA	XXTEA	Rounds
187	186	188	187	163	169	1
186	186	187	187	164	163	2
175	165	187	164	158	161	3
170	165	23	22	3	20	4
170	161	92	1	2	4	5
170	161	7	0	3	0	6
168	159	0	1	1	1	7
161	159	0	0	0	1	8
163	6	1	0	0	0	9
161	6	2	1	2	3	10
162	0	0	0	1	0	11
160	0	3	0	0	1	12
61	0	1	2	0	0	13
11	0	0	2	0	0	14

Table 2 contains the results for reduced-round versions of the same ciphers modified using the substitution boxes from RC4D [22] and the same approach as demonstrated by the SPECK-R cipher [19]. SIMECK-R is an absolute winner, with XTEA-R coming in second place.

Table 2. NIST Statistical Test Suite—the number of failed randomness tests for reduced round ciphers (lower is better).

XXTEA-R	TEA-R	SIMECK-R	SIMON-R	SPECK-R	XTEA-R	Rounds
167	172	0	160	161	163	1
165	165	1	161	158	53	2
10	12	1	159	107	0	3
3	2	0	158	1	3	4
1	0	0	2	2	0	5

4.1.2. Analysis with AIS.31 Statistical Test Suite

Table 3 contains the total number of failed AIS.31 [26] tests for the full-round ciphers. The test suite from [72] contains 11 tests, and in order to count the failures, it is considered that any partially failed test showing nongreen output from the test suite is a failure. XTEA and XXTEA are winners after only four rounds, and TEA follows in round 5.

Table 4 shows the results for the reduced round ciphers using the AIS.31 statistical tests. SIMECK-R is an outstanding winner, failing 0 tests with only 1 round of encryption enabled and keeping it low throughout the rest of the rounds.

It is very important to point out that for each test the complete number of rounds has been analyzed using both statistical test suites. For example, the full-round SIMECK cipher is 44 rounds, SIMON is 42 rounds, SPECK is 26 rounds, and the TEA family is 32 rounds. All have been carefully analyzed both in original form and with the substitution boxes inserted for SIMECK-R, SIMON-R, SPECK-R, TEA-R, XTEA-R, and XXTEA-R starting from round 1 until the full version. It is quite obvious that substitution boxes using the SPECK-R strategy [19] produce fewer failures in reduced-round versions of the ciphers.

Table 3. AIS.31 Statistical Test Suite—number of failed randomness tests for full round ciphers (lower is better).

SIMECK	SIMON	SPECK	TEA	XTEA	XXTEA	Rounds
11	9	9	9	9	9	1
9	9	9	9	9	9	2
8	10	6	8	6	7	3
9	10	3	9	0	1	4
9	7	5	0	0	0	5
8	7	1	0	0	0	6
9	5	0	0	0	0	7
9	5	0	0	0	0	8
8	1	0	0	0	0	9
7	1	0	0	0	0	10
7	0	0	0	0	0	11
6	0	0	0	0	0	12
2	0	0	0	0	0	13

Table 4. AIS.31 Statistical Test Suite—number of failed randomness tests for reduced round ciphers (lower is better).

XXTEA-R	TEA-R	SIMECK-R	SIMON-R	SPECK-R	XTEA-R	Rounds
9	9	0	6	7	8	1
8	8	0	6	5	3	2
2	2	0	3	3	0	3
0	0	0	2	0	0	4

The SPECK and SIMON ciphers are used with a 64-bit block size and a key length of 96 bits just as the original SPECK-R cipher [19]. SIMECK has a 64-bit block size and a key length of 128 bits, just like the TEA family of ciphers.

It can be observed that the best cipher in terms of producing random output using as few rounds as possible is SIMECK-R, according to the data in Tables 2 and 4. This cipher is going to be used in order to compare it with SIMECK-T.

4.1.3. Analysis with the Law of Iterated Logarithm

For resources-constrained devices, the SIMECK-R dynamic substitution boxes are an obvious bottleneck. Not only must the three tables be memorized and exchanged based on certain thresholds, but each table takes 1024 bits, which may not be affordable for medical implants and other healthcare devices, or even for mass production in smart agriculture systems.

The proposed SIMECK-T cipher has none of these problems when at least 18 rounds are used (3 rounds of SIMECK each containing 5 rounds of TEA). Although standardized randomness tests are a norm for validating the output of ciphers [88], it has been shown, with an emphasis on the NIST statistical tests [24], that weak ciphers can also successfully pass them [89]. In order to further investigate the randomness of the SIMECK-T output, the law of the iterated logarithm (LIL) suggested by Wang as a stronger validation was used [27,28].

Figure 2 shows that the density function for the output of SIMECK-T closely follows the expected curve. The results have been generated using the Java tool developed by Wang [89,90]. A minimum of 1000 sequences is recommended for accurate testing and

10000 have been used for the SIMECK-T cipher using a fixed number of SIMECK-3 and TEA-5 rounds. That is, the *pmul* and *psum* variables in Algorithm 1 have been forced to zero. The total variance (TV), the Hellinger distance (HD), and the root mean square deviation (RMSD) show values lower than the required limit for true random data, which means that SIMECK-T passes all tests. For random data obtained with the OneRNG true random number generator (TRNG) [91], TV is 0.05325 < 0.09161, HD is 0.04715 < 0.08054 and RMSD is 0.00306 < 0.00607 for 1000 sequences.

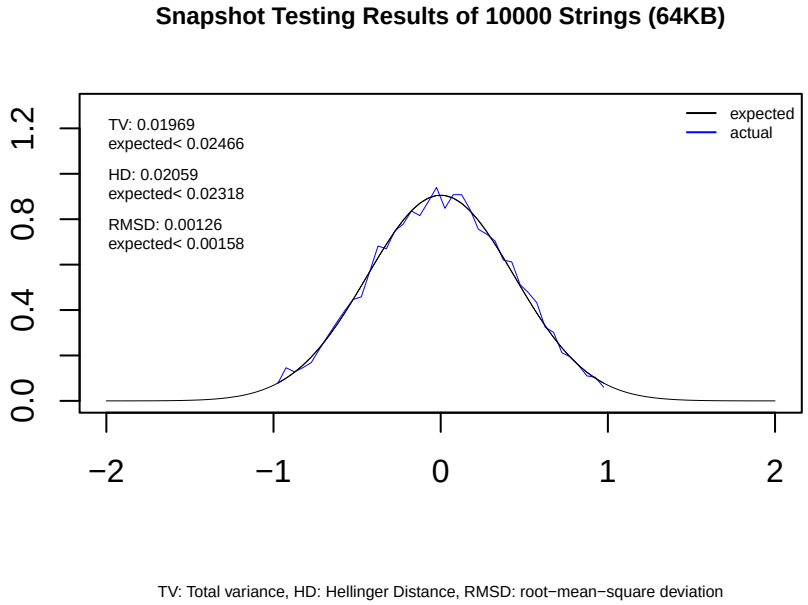


Figure 2. Density function from the output of SIMECK-T.

Table 5 presents the minimal number of rounds that are necessary for each cipher, augmented or not, to pass the LIL test. SIMECK-R is once again the best cipher, showing that SIMECK is the best candidate for augmentation into a reduced round lightweight cipher.

Table 5. Law of iterated logarithm—minimal number of rounds for passing.

SIMECK	SIMON	SPECK	TEA	XTEA	XXTEA
16	11	4	6	4	5
1	7	4	4	3	5
SIMECK-R	SIMON-R	SPECK-R	TEA-R	XTEA-R	XXTEA-R

4.2. Cipher Strength Analysis

4.2.1. Avalanche Effect

SIMECK-T fails 0 tests from the NIST Statistical Test Suite [71] and the AIS.31 test suite [72]. It also passes the randomness tests of other known test suites like Dieharder [92] and ENT [93].

Tables 6 and 7 show the avalanche effect and the time required to compute the round function using various numbers of rounds. It can be observed that, on average, half of the bits are flipped when only 1 input bit is modified. This shows that the two ciphers have good avalanche properties.

Table 6. Avalanche effect data for SIMECK-R cipher across rounds.

Rounds	64 bit Blocks	Avg Bit Flips	Min B. Fl.	Max B. Fl.	Time (s)
1	10^6	32.00	3	55	12.44
2	10^6	32.00	10	54	12.77
3	10^6	32.00	10	55	13.13
4	10^6	32.00	8	55	13.73
5	10^6	32.00	11	54	13.77
5	10^2	32.04	18	49	0.0014

Table 7. Avalanche effect data for SIMECK-T cipher.

64 bit Blocks	Avg Bit Flips	Min Bit Flips	Max Bit Flips	Time (s)
10^6	32.00	10	54	13.88
10^6	32.00	10	52	13.84
10^6	32.00	11	54	13.90
10^2	32.06	18	48	0.0038
10^2	32.00	18	47	0.0038
10^2	32.02	18	48	0.0038

4.2.2. Linear Cryptanalysis

Given a sufficiently large number of random samples, a secure cryptosystem will produce a bias closer to 0%.

Table 8 shows the results obtained for large numbers of pairs (plaintext, ciphertext), with a percentage bias that converges to 0%. The largest pool of pairs randomly sampled is 2^{32} using memory-mapped files for hash tables. The goal is to identify suspect bit-level linear relationships between the input and the output of the cipher.

Table 8. Best key guesses and biases.

Key Guess	Number of Pairs	Bias
EE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	10^8	0.011891%
1A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	10^9	0.004373%
89 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	10^9	0.004798%
AC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	2^{32}	0.001758%

An interesting pattern in Table 8 would require finding a bias that is not close to zero—-no interesting pattern has been found, suggesting that the encryption scheme is resistant to linear trails.

A randomly generated key is kept constant during statistical searches, so that any pattern has a sufficiently large collection of pairs to surface.

4.2.3. Differential Cryptanalysis

The results in Table 9 show the frequency of appearance and the respective percentage for differential trails using different attack strategies.

For an ARX cipher, rotational, additive, and XOR-based differences are expected to trigger a pattern. However, our results show that OR- and AND-based operations are more susceptible to producing trails. This may be due to the external rounds of SIMECK, but we expect that most trails are canceled by performing the XOR operation with a constant inside the TEA inner rounds.

Table 9. Most frequent output differences for different attacks.

Attack Type	Most Freq. Output Diff. (Hex)	Frequency	Percentage
XOR-based	(79357347, 018D019B)	1	1.0×10^{-5}%
Rotational	(3D3E7347, 3479019B)	1	1.0×10^{-5}%
Additive	(86B57347, CC39019B)	1	1.0×10^{-5}%
OR-based	(B27D7347, B079019B)	2	2.0×10^{-5}%
AND-based	(00208007, 4D39019B)	2	2.0×10^{-5}%

Table 9 shows no bias far from zero, which means that no interesting pattern is found. The goal is to identify suspect relationships (differential trails) when the difference between two plaintext blocks encrypted with the same password is known.

4.3. Statistical Evaluation Metrics

4.3.1. Chi-Squared

For 1 GB of data, we obtain a value of 251.546, which is greater than 0.05, meaning that the histogram is uniform. The full-round SIMECK has a value of 244.252 and full round TEA is at 268.299.

4.3.2. Encryption Quality

Our cipher yielded an encryption quality of 8.853×10^6. The higher this value, the better the encryption quality of the cryptosystem. For full-round SIMECK, the encryption quality is 7.273×10^6, while full-round TEA has an encryption quality of 7.467×10^6. The proposed SIMECK-T has a better encryption quality.

4.3.3. Mean Square Error

The mean square error obtained from our experiments is 1.09221×10^4. The full-round SIMECK has a mean square error of 1.09225×10^4, while the full-round TEA has a mean square error of 1.09222×10^4. These values are comparable with those of SIMECK-T.

4.3.4. Peak Signal-to-Noise Ratio

Low values of the PSNR indicate a secure cryptosystem. The value resulting from our experiments is 25.7373. The full-round SIMECK has a PSNR value of 25.7368, while the full-round TEA has a PSNR of 25.7372. These values are similar to SIMECK-T.

4.3.5. Structural Similarity Index Measure

The value obtained in our experiments for the measure of the structural similarity index is 1.362×10^{-2}. Full round SIMECK has 1.362×10^{-2} and full round TEA has 1.362×10^{-2}. The values are comparable with those of SIMECK-T.

4.3.6. Normalized Cross-Correlation

Our cryptosystem yielded a normalized cross-correlation value of 2.630×10^{-5}. The normalized cross-correlation obtained by full-round SIMECK is -7.306×10^{-6}, while full round TEA obtains a value of 5.086×10^{-6}. In all cases, the normalized cross-correlation is close to 0.

4.3.7. Mean Absolute Error

The mean absolute error for our cryptosystem is 85.330. The full-round SIMECK has a mean absolute error of 85.332, and the full-round TEA has a mean absolute error of 85.330. The values are identical/very close to each other.

4.3.8. Normalized Absolute Error

The value of the normalized absolute error obtained through our experiments is 6.692×10^{-1}. The same value is obtained for the full-round SIMECK and full-round TEA.

4.3.9. Maximum Difference

Our experiments resulted in a maximum difference value of 2.55×10^{2}. The value for SIMECK for all round and TEA for all round is the same.

4.3.10. Average Difference

Our result for the average difference is -7.548×10^{-3}. The full-round SIMECK has an average difference of 3.169×10^{-4}, and the full-round TEA has an average difference of -3.505×10^{-3}. All values are very close to 0.

4.3.11. Structural Content

During our experiments, we obtained a value of 9.9990×10^{-1}. Full round SIMECK has a structural content value of 9.9999×10^{-1}, while full round TEA has a structural content value of 9.9995×10^{-1}.

4.3.12. Two-Dimensional Correlation Coefficient

Through our experiments, we have obtained a value of 2.630×10^{-5}. The value for the full-round SIMECK is -7.306×10^{-6} and for full round TEA is 5.086×10^{-6}. These values are close to 0.

4.3.13. Next Pixel Change Rate

We obtained a value of 9.9609×10^{-1} for NPCR and 3.3463×10^{-1} for UACI. The same value was obtained for the full-round SIMECK and full-round TEA.

4.3.14. Correlation Coefficient

The value we obtained through our experiments is 2.630×10^{-5}. The SIMECK correlation coefficient is -7.306×10^{-6}, while the correlation coefficient for full-round TEA is 5.086×10^{-6}. All values are close to 0.

4.3.15. Entropy

Our value is 7.99999983 (N = 8). The entropy for full round SIMECK is 7.99999983, and for full round TEA, it is 7.99999981. All values are very close to the ideal value of 8.

4.4. TETRA Cryptographic Primitives

TETRA ciphers are stream ciphers. Both TETRA TEA2 and TEA3 pass the NIST statistical tests [71] but TEA3 behaves better when using a reduced number of rounds. Since both ciphers are considered unbroken, a reduced-round version of TEA3 has been tested as a candidate for lightweight cryptography. The measurements have been made on an Intel i7 running GNU/Linux using the reverse-engineered source code published by [32]. The results for the reduced-round TETRA TEA3 cipher are shown in Table 10.

Table 10. TErrestrial Trunked RAdio (TETRA) TEA3 cipher performance across rounds with a keystream size of 1 KB.

Rounds	64 bit Blocks	Time (s)
1	1,048,576	8.97
2	1,048,576	17.84
3	1,048,576	28.34
4	1,048,576	35.00

The TETRA TEA3 cipher has 80-bit encryption keys and shows promising protection levels when four rounds are used, which makes it a bit slower than the SIMECK-R and SIMECK-T ciphers. The table used for the substitution box must be stored in memory, and has been criticized in [32], and also requires access to memory and storage space. This level of protection is not achievable on the default 51-byte keystream size; a minimum of 1 KB is recommended according to the NIST statistical tests [24].

4.5. Performance Evaluation on Raspberry PICO

The PBKDF2 function with 65.000 iterations takes 7.572 milliseconds to run. For 30.000 iterations, it takes 3.494 milliseconds. For 10.000 iterations, it takes 1.165 milliseconds, and for 1.000 iterations, it takes 0.116 milliseconds.

A single encryption in counter mode takes 0.00873 milliseconds and in electronic codebook mode 0.00327 milliseconds.

By modifying the number of SIMECK rounds to five and the number of internal TEA rounds to seven, the following results are obtained:

A single encryption in counter mode takes 1.038×10^{-2} milliseconds and in electronic codebook mode 3.31×10^{-3} milliseconds.

The PBKDF2 function with 65.000 iterations takes 10.422 milliseconds to run. For 30.000 iterations, it takes 4.810 milliseconds, and for 10.000 iterations, it takes 1.603 milliseconds.

5. Discussion

Lenstra [94] estimates the strength of a key size based on the cost of breaking a DES [95] cipher key. The original name of this cipher is LUCIFER [96], having a key size of 128 bits. This key was reduced to 56 bits. Sufficient protection is defined as the cost involved in breaking the DES cipher in 1982, where y is the current year and λ is the key length.

$$y(\lambda) = 1982 + \frac{3(\lambda - 56)}{2} \tag{24}$$

$$\lambda(y) = 56 + \frac{2(y - 1982)}{3} \tag{25}$$

Although outdated, Moore's law [97] is taken into account. According to Equation (24), $y(80) = 2018$, which means that 80-bit keys should no longer be used, starting with 2019.

Equation (25) predicts that, for 2025, a minimum key size of $\lambda(2025) = 85$ bits is required.

However, an 80-bit key can be accepted as a decent security level for IoT devices, as shown in the following analysis.

The power of TOP500 [98] contains exascale-capable supercomputers. For a cryptographic cipher designer, this means, in the worst-case scenario, that such a machine can be used to brute force more than 1 quintillion passwords per second. For an 80-bit key size, this means that it can be recovered theoretically in $2^{80}/2^{18} = 2^{62}$ seconds. The measured performance in TOP500 [98] is for floating point operations in double precision. Simple operations like ARX ciphers can reasonably be expected to happen orders of magnitude

faster, and specialized hardware can be designed for even faster brute-forcing capacity. But the costs for such a computation simply outweighs any hopes for finding a needle in a 2^{80} space.

However, for such a key length to offer protection, the key values need to be random. The ASCII table has 95 printable characters, which means that, if the key is 10 bytes, each byte will be covered by 95 symbols instead of the full 256.

A 95 symbol byte for a key size of 80 bits is 95^8 which lies between 2^{52} and 2^{53} practically reducing the key size to something easy to brute force. This is true when the symbols are placed with equal probability. If not, the actual search space for the key is smaller.

A 96-bit key with 95 symbols means 95^{12} possibilities which lie between 2^{78} and 2^{79} when the symbols have the same probability of appearing in the password.

As a consequence, a tunable password derivation function is required to mitigate brute-force attacks. The number of rounds in the password derivation function could increase depending on the quality of the password and the current year [34].

A 64 bit block cipher in counter mode requires a new key every $2^{n/2-16}$ blocks [99], which means that a new key should be deployed/generated for every 512 KB of encrypted data. A strategy like this cannot be envisioned on ciphers with a 32-bit block size because given $n = 32$, a new key would be required for every 4 bytes of plaintext.

An improved SIMECK round function [100] could be important for the proposed SIMECK-T cipher, where a block size of 64 bits is used instead of 32 as in [100], and the encryption key is 128 bits. An increase in the number of rounds to 5 SIMECK rounds, each containing 7 TEA rounds (a total of 40 rounds), should be applied whenever this is not a drawback for the limited resources.

The fact that SIMECK-T successfully produces random output is very important. The cipher passes the χ^2 test introduced by Vuadenauy [73], and the law of the iterated logarithm tests in all three forms, weak, medium, and strong, introduced by Wang [28,89]. The metrics used by Oladipupo et al. [31] are also passed.

The net advantage of SIMECK-T over SPECK-R [19], or over the newly discovered SIMECK-R cipher, which has also been made available as free software [101], is the lack of substitution boxes. No lookup memory is required, there are no side channels, and no communication bottleneck is perceived since all operations can be performed in local 32-bit registers. This also means that any available memory on a resource-constrained device can be used for other purposes. For IoT devices, variables that are stored in memory—and more importantly the tables—constantly draw power from scavenged resources and battery supplies. For this reason, even though lightweight cryptography approaches like that demonstrated by SPECK-R [19] can offer decent protection levels, an ARX solution like SIMECK-T is lighter in resources and easier to implement.

A 128-bit block size for lightweight cryptography is not always applicable. A 32-bit block size is sometimes used, but it is too little by modern standards. A 64-bit block size like the one in SIMECK-T seems to be the right choice, and a 128-bit key that changes every 512 KB offers sufficient protection [99]. The SPECK-R algorithm only uses a 96-bit key [19].

The role of the *psum* and *pmul* variables that depend on the derived key and the password in Algorithm 1 is to defeat neural classifiers which have been shown to successfully attack reduced rounds of ciphers with small block sizes [102].

6. Conclusions

Lightweight cryptography is an important challenge for resource-constrained devices and the Internet of Things.

A selection of Addition–Rotation–XOR (ARX) block ciphers, designed for software implementations, were improved using the dynamic substitution layer of SPECK-R [19]. Reduced-round versions of these augmented ciphers were compared to produce pseudo-random output, and SIMECK was found to be the best performing candidate in the form of SIMECK-R.

Based on the operators within the round function of SIMECK and the one used by the TEA cipher, also performing very well in our tests, a lightweight encryption scheme was obtained using only ARX instructions based on existing SIMECK [29] and TEA [30] algorithms. The encryption scheme was based on a 64-bit block size and 128-bit keys, suitable for resource-constrained devices.

The performance of the new algorithm in electronic code book (ECB) mode and in counter mode (CTR), was studied on a Raspberry PICO 2040 device. Implementation of a password-based key derivation function (PBKDF) was also evaluated, using SIMECK-T as the back-end and MDC-2 to create a hash function. The results show that the encryption scheme is very fast.

The proposed encryption scheme combines 3 rounds of SIMECK with 5 internal rounds of TEA for a total of 18 rounds. Where possible, we recommend that 5 rounds of SIMECK be combined with 7 rounds of TEA for a total of 40 rounds. Our tests on Raspberry PICO show that the performance impact is negligible.

The NIST statistical tests [24] and the AIS.31 [25,26] standards have been used to validate the randomness of the output produced by SIMECK-T. Most importantly, SIMECK-T has been shown not only to be very fast but also to successfully pass the law of the iterated logarithm (LIL) proposed by Wang as a stronger requirement for pseudorandom generators [28,89].

Several evaluation metrics have been used to show that the encryption scheme is secure compared to other solutions. The χ^2 test proposed by Vaudenay [73] shows that SIMECK-T produces random output.

Linear and differential cryptanalysis have been attempted in the proposed encryption scheme using custom tools made available as free software [34]. A number of 2^{32} pairs of random plaintext and corresponding ciphertext blocks were sampled without finding any linear patterns. For differential cryptanalysis, a pool of 10^7 differential pairs was sampled without detecting any relationship.

Characteristic Automated Search of Cryptographic Algorithms for Distinguishing Attacks (CASCADA) is a Python-based software library that can be used for the automatic analysis of ARX ciphers [103] or encryption schemes such as SIMECK-T. It is based on modulo theories of satisfiability (SMT) and SMT solvers, and it requires that encryption primitives are ported within the framework [104].

The proposed encryption scheme was implemented in block counter mode (CTR), which is parallelizable but does not provide authentication. Galois/Counter Mode (GCM) can be implemented in order to achieve both parallelization and authentication, although devices with very limited resources may be better suited for the actual mode of operation.

White-box cryptography aims to implement algorithms in software so that secret keys remain secure, even when faced with white-box attempts to extract them from the software itself [105]. A white-box implementation of SIMECK-T should add an additional layer of security [106], most importantly one-wayness [107], preventing decryption when the algorithm runs in hostile environments.

Author Contributions: Conceptualization, A.-A.A.; methodology, A.-A.A.; software, A.-A.A., P.C. and R.-D.C.; validation, A.-A.A., P.C. and R.-D.C.; investigation, A.-A.A., P.C. and E.-A.C.; resources, A.-A.A., P.C. and E.-A.C.; data curation, P.C. and E.-A.C.; writing—original draft preparation, A.-A.A. and E.-A.C.; writing—review and editing, P.C. and E.-A.C.; visualization, A.-A.A.; supervision,

A.-A.A.; project administration, A.-A.A.; funding acquisition, A.-A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This paper benefited from financial support through the Program on "Supporting the research activity by funding an internal grant competition—SACER 2023", Politehnica University Timișoara, Competition 2022.

Data Availability Statement: The data associated with this article are fully reproducible. The true hardware number generator data and the analysis provided by statistical tests for encrypted files are available at https://staff.cs.upt.ro/~alin.anton/BANPUMP/datasets/SIMECK-T accessed on (11 November 2024). Data are available under the terms of the Creative Commons Zero "No rights reserved" data waiver (CC0 1.0 Public domain dedication).

Conflicts of Interest: The authors declare no conflicts of interest. Sponsors had no role in the design of the study; in the collection, analysis, or interpretation of the data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

2DCC	Two-Dimensional Correlation Coefficient
AD	Average Difference
AES	Advanced Encryption Standard
ARX	Addition–Rotation–XOR
BSI	Federal Office for Information Security
CASCADA	Characteristic Automated Search of Cryptographic Algorithms for Distinguishing Attacks
CTR	Counter Mode
DES	Data Encryption Standard
ECT	Electronic Code Book
EQ	Encryption Quality
ETSI	European Telecommunications Standards Institute
FELICS	Fair Evaluation of Lightweight Cryptographic Systems
FNV	Fowler-No-Vo
GCM	Galois Counter Mode
GPUs	Graphic Processing Units
HD	Hellinger Distance
HXP	Hash XOR Permutation
IIoT	Industrial Internet of Things
IoT	Internet of Things
IRC6	Improved-RC6
IV	Initialization Vector
KSA	Key Scheduling Algorithm
LIL	Law of the Iterated Logarithm
MAE	Mean Absolute Error
MD	Maximum Difference
MDC	Meyer–Schilling Detection Code
MHF	Memory Hard Function
MSE	Mean Square Error
NAE	Normalized Absolute Error
NCC	Normalized Cross-Correlation
NDA	Non-Disclosure Agreement
NIST	National Institute of Standards and Technology
NPCR	Next Pixel Change Rate
NSA	National Security Agency

PBKDF	Password-Based Key Derivation Function
PSNR	Peak Signal-to-Noise Ratio
PXB	Permuted XORed Bytes
RAM	Random Access Memory
RFID	Radio-Frequency IDentification
RMSD	Root-Mean-Square Deviation
ROM	Read-Only Memory
SC	Structural Content
SHA	Secure Hash Algorithm
SIMD	Single Instruction Multiple Data
SMT	Satisfiability Modulo Theory
SSIM	Structural Similarity Index Measure
STS	Statistical Test Suite
TETRA	TErrestrial Trunked RAdio
TRNG	True Random Number Generator
TV	Total Variance
UACI	Unified Average Change Intensity

References

1. Volosciuc, C.; Bogdan, R.; Blajovan, B.; Stângaciu, C.; Marcu, M. GreenLab, an IoT-Based Small-Scale Smart Greenhouse. *Future Internet* **2024**, *16*, 195. [CrossRef]
2. Stefan, I.; Aldea, C.L.; Nechifor, C.S. Web platform architecture for ambient assisted living. *J. Ambient Intell. Smart Environ.* **2018**, *10*, 35–47. [CrossRef]
3. Wiedemann, T.; Vlaicu, C.; Josifovski, J.; Viseras, A. Robotic Information Gathering with Reinforcement Learning Assisted by Domain Knowledge: An Application to Gas Source Localization. *IEEE Access* **2021**, *9*, 13159–13172. [CrossRef]
4. Levina, A.; Mukhamedjanov, D.; Bogaevskiy, D.; Lyakhov, P.; Valueva, M.; Kaplun, D. High-Performance Parallel Pseudorandom Number Generator on Cellular Automata. *Symmetry* **2022**, *14*, 1869. [CrossRef]
5. Lee, H.; Moon, S. Parallel stream cipher for secure high-speed communications. *Signal Process.* **2002**, *82*, 259–265. [CrossRef]
6. Velea, R.; Gurzău, F.; Mărgărit, L.; Bica, I.; Patriciu, V.V. Performance of parallel ChaCha20 stream cipher. In Proceedings of the 2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 12–14 May 2016; pp. 391–396. [CrossRef]
7. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK lightweight block ciphers. In Proceedings of the 52nd Annual Design Automation Conference, New York, NY, USA, 7–11 June 2015; DAC '15. [CrossRef]
8. Dinu, D.; Corre, Y.L.; Khovratovich, D.; Perrin, L.; Großschädl, J.; Biryukov, A. Triathlon of Lightweight Block Ciphers for the Internet of Things. *J. Cryptogr. Eng.* **2015**, *9*, 283–302. [CrossRef]
9. Rivest, R.L. The RC5 Encryption Algorithm. In Proceedings of the Fast Software Encryption: Second International Workshop, Leuven, Belgium, 14–16 December 1994; Proceedings; Springer: Berlin/Heidelberg, Germany, 1994; Lecture Notes in Computer Science, Volume 1008, pp. 86–96. [CrossRef]
10. Ronald L.; Rivest, M.J.B.; Robshaw, R.S.Y.Y. *The RC6 Block Cipher*; Technical Report; MIT Laboratory for Computer Science: Cambridge, MA, USA, 1998. Available online: https://people.csail.mit.edu/rivest/pubs/RRSY98.pdf (accessed on 11 November 2024)
11. Sklavos, N.; Koufopavlou, O. Data dependent rotations, a trustworthy approach for future encryption systems/ciphers: Low cost and high performance. *Comput. Secur.* **2003**, *22*, 585–588. [CrossRef]
12. Faragallah, O.S.; Elashry, I.F.; AlGhamdi, A.; El-Shafai, W.; El-Rabaie, S.; Abd El-Samie, F.E.; El-Sayed, H.S.; Elaskily, M.A. Improved RC6 Block Cipher Based on Data Dependent Rotations. *Comput. Mater. Contin.* **2022**, *70*, 1921–1934. [CrossRef]
13. Baysal, A.; Şahin, S. RoadRunneR: A Small and Fast Bitslice Block Cipher for Low Cost 8-Bit Processors. In *Lightweight Cryptography for Security and Privacy*; Güneysu, T., Leander, G., Moradi, A., Eds.; Springer: Cham, Switzerland, 2016; pp. 58–76.
14. Guo, J.; Peyrin, T.; Poschmann, A.; Robshaw, M. The LED Block Cipher. In *Cryptographic Hardware and Embedded Systems—CHES 2011*; Preneel, B., Takagi, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 326–341.
15. Shibutani, K.; Isobe, T.; Hiwatari, H.; Mitsuda, A.; Akishita, T.; Shirai, T. Piccolo: An Ultra-Lightweight Blockcipher. In *Cryptographic Hardware and Embedded Systems—CHES 2011*; Preneel, B., Takagi, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 342–357.

16. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems—CHES 2007*; Paillier, P., Verbauwhede, I., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 450–466.
17. Mouha, N.; Mennink, B.; Van Herrewege, A.; Watanabe, D.; Preneel, B.; Verbauwhede, I. Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In *Selected Areas in Cryptography—SAC 2014*; Joux, A., Youssef, A., Eds.; Springer: Cham, Switzerland, 2014; pp. 306–323.
18. Dinu, D.D.; Biryukov, A.; Groszschädl, J.; Khovratovich, D.; Le Corre, Y.; Perrin, L.P. FELICS—Fair Evaluation of Lightweight Cryptographic Systems. In *NIST Workshop on Lightweight Cryptography*; NIST: Gaithersburg, MD, USA, 2015.
19. Sleem, L.; Couturier, R. Speck-R: An ultra light-weight cryptographic scheme for Internet of Things. *Multimed. Tools Appl.* **2021**, *80*, 17067–17102. [CrossRef]
20. Jindal, P.; Singh, B. RC4 Encryption-A Literature Survey. *Procedia Comput. Sci.* **2015**, *46*, 697–705. [CrossRef]
21. Anton, A.; Csereoka, P. Enhanced Speck-R Implementation. 2024. Available online: https://github.com/BANPUMP-team/speckr (accessed on 11 November 2024).
22. Alsharida, R.; Hammood, M.; Ahmed, M.A.; Thamer, B.; Shakir, M. RC4D: A New Development of RC4 Encryption Algorithm. In *Selected Papers from the 12th International Networking Conference*; Ghita, B., Shiaeles, S., Eds.; Springer: Cham, Switzerland, 2021; pp. 19–30.
23. Anton, A.A.; Csereoka, P.; Capota, E.A.; Cioargă, R.D. Enhancing Syslog Message Security and Reliability over Unidirectional Fiber Optics. *Sensors* **2024**, *24*, 6537. [CrossRef] [PubMed]
24. Bassham, L.E.; Rukhin, A.L.; Soto, J.; Nechvatal, J.R.; Smid, M.E.; Barker, E.B.; Leigh, S.D.; Levenson, M.; Vangel, M.; Banks, D.L.; et al. *SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Technical Report; National Institute of Standards & Technology: Gaithersburg, MD, USA, 2010.
25. Matthias, P.; Werner, S. A Proposal for: Functionality Classes for Random Number Generators—Version 2.36 Current Intermediate Document for the AIS 20/31 Workshop. *Ser. BDI Bonn*. 2023. Available online: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Certification/Interpretations/AIS_31_Functionality_classes_for_random_number_generators_e_2023.pdf?__blob=publicationFile&v=2 (accessed on 11 November 2024)
26. Park, H.; Kang, J.S.; Yeom, Y. Probabilistic Analysis of AIS.31 Statistical Tests for TRNGs and Their Applications to Security Evaluations. *J. Korea Inst. Inf. Secur. Cryptology* **2016**, *26*, 49–67. [CrossRef]
27. Matos-Rodríguez, Y.; Madarro-Capó, E.J.; Legón-Pérez, C.M.; Rojas, O.; Sosa-Gómez, G. Improvements in the computing efficiency of the probabilities of the LIL test for the PRNG evaluation. *Theor. Comput. Sci.* **2022**, *902*, 29–40. [CrossRef]
28. Wang, Y.; Nicol, T. On statistical distance based testing of pseudo random sequences and experiments with PHP and Debian OpenSSL. *Comput. Secur.* **2015**, *53*, 44–64. [CrossRef]
29. Yang, G.; Zhu, B.; Suder, V.; Aagaard, M.D.; Gong, G. The Simeck Family of Lightweight Block Ciphers. In *Cryptographic Hardware and Embedded Systems—CHES 2015*; Güneysu, T., Handschuh, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 307–329.
30. Wheeler, D.J.; Needham, R.M. TEA, a Tiny Encryption Algorithm. In Proceedings of the Fast Software Encryption: Second International Workshop, Leuven, Belgium, 14–16 December 1994; Proceedings; Springer: Berlin/Heidelberg, Germany, 1994; Volume 1008, pp. 363–366. [CrossRef]
31. Oladipupo, E.T.; Abikoye, O.C.; Awotunde, J.B. A Lightweight Image Cryptosystem for Cloud-Assisted Internet of Things. *Appl. Sci.* **2024**, *14*, 2808. [CrossRef]
32. Meijer, C.; Bokslag, W.; Wetzels, J. All cops are broadcasting: TETRA under scrutiny. In Proceedings of the 32nd USENIX Conference on Security Symposium, Anaheim, CA, USA, 9–11 August 2023; SEC '23.
33. Leander, G.; Paar, C.; Poschmann, A.; Schramm, K. New lightweight DES variants. In Proceedings of the Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, 26–28 March 2007; Revised Selected Papers 14; Springer: Berlin/Heidelberg, Germany, 2007; pp. 196–210.
34. Anton, A.A. SIMECK-T Cipher. 2024. Available online: https://github.com/BANPUMP-team/simeck-tea (accessed on 11 November 2024).
35. Mohd, B.J.; Hayajneh, T.; Vasilakos, A.V. A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues. *J. Netw. Comput. Appl.* **2015**, *58*, 73–93. [CrossRef]
36. Eisenbarth, T.; Gong, Z.; Güneysu, T.; Heyse, S.; Indesteege, S.; Kerckhof, S.; Koeune, F.; Nad, T.; Plos, T.; Regazzoni, F.; et al. Compact Implementation and Performance Evaluation of Block Ciphers in ATtiny Devices. In *Progress in Cryptology—AFRICACRYPT 2012*; Mitrokotsa, A., Vaudenay, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 172–187.
37. Cazorla, M.; Marquet, K.; Minier, M. Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks. In *International Conference on Security and Cryptography (SECRYPT) 2013*; Cazorla, M., Marquet, K., Minier, M., Eds.; IEEE: Reykjavik, Iceland, 2013; pp. 1–6.
38. Eisenbarth, T.; Kumar, S.; Paar, C.; Poschmann, A.; Uhsadel, L. A survey of lightweight-cryptography implementations. *IEEE Des. Test Comput.* **2007**, *24*, 522–533. [CrossRef]

39. Malina, L.; Clupek, V.; Martinasek, Z.; Hajny, J.; Oguchi, K.; Zeman, V. Evaluation of Software-Oriented Block Ciphers on Smartphones. In *Foundations and Practice of Security*; Danger, J.L., Debbabi, M., Marion, J.Y., Garcia-Alfaro, J., Zincir Heywood, N., Eds.; Springer: Cham, Switzerland, 2014; pp. 353–368.
40. Zhong, Y.; Gu, J. Lightweight block ciphers for resource-constrained environments: A comprehensive survey. *Future Gener. Comput. Syst.* **2024**, *157*, 288–302. [CrossRef]
41. Hong, D.; Sung, J.; Hong, S.; Lim, J.; Lee, S.; Koo, B.S.; Lee, C.; Chang, D.; Lee, J.; Jeong, K.; et al. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Cryptographic Hardware and Embedded Systems—CHES 2006*; Goubin, L., Matsui, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 46–59.
42. Hong, D.; Lee, J.K.; Kim, D.C.; Kwon, D.; Ryu, K.H.; Lee, D.G. LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors. In *Information Security Applications*; Kim, Y., Lee, H., Perrig, A., Eds.; Springer: Cham, Switzerland, 2014; pp. 3–27.
43. Dinu, D.; Perrin, L.; Udovenko, A.; Velichkov, V.; Großschädl, J.; Biryukov, A. Design Strategies for ARX with Provable Bounds: Sparx and LAX. In *Advances in Cryptology—ASIACRYPT 2016*; Cheon, J.H., Takagi, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 484–513.
44. Koo, B.; Roh, D.; Kim, H.; Jung, Y.; Lee, D.G.; Kwon, D. CHAM: A Family of Lightweight Block Ciphers for Resource-Constrained Devices. In *Information Security and Cryptology—ICISC 2017*; Kim, H., Kim, D.C., Eds.; Springer: Cham, Switzerland, 2018; pp. 3–25.
45. Prathiba, A.; Bhaaskaran, V.S.K. Lightweight S-Box Architecture for Secure Internet of Things. *Information* **2018**, *9*, 13. [CrossRef]
46. ETSI. TETRA Air Interface Security, Algorithms Specifications; Part 1: TETRA Encryption Algorithms Set A. 2024. Available online: https://www.etsi.org/deliver/etsi_ts/104000_104099/10405301/01.01.01_60/ts_10405301v010101p.pdf (accessed on 11 November 2024).
47. Hong, D.; Koo, B.; Kwon, D. Biclique attack on the full HIGHT. In Proceedings of the 14th International Conference on Information Security and Cryptology, Xi'an, China, 26–29 October 2011; ICISC'11, pp. 365–374. [CrossRef]
48. Koo, B.; Hong, D.; Kwon, D. Related-key attack on the full HIGHT. In Proceedings of the 13th International Conference on Information Security and Cryptology, Seoul, Republic of Korea, 1–3 December 2010; ICISC'10, pp. 49–67.
49. Roh, D.; Koo, B.; Jung, Y.; Jeong, I.; Lee, D.; Kwon, D.; Kim, W. Revised Version of Block Cipher CHAM. In *Information Security and Cryptology—ICISC 2019—22nd International Conference, Revised Selected Papers*; Seo, J., Ed.; Springer: Cham, Switzerland, 2020; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 1–19. [CrossRef]
50. Raja, K.P.; Mishra, Z.; Singh, P.; Acharya, B. Efficient hardware implementations of lightweight Simeck Cipher for resource-constrained applications. *Integration* **2023**, *88*, 343–352. [CrossRef]
51. Mishra, Z.; Acharya, B. High throughput novel architectures of TEA family for high speed IoT and RFID applications. *J. Inf. Secur. Appl.* **2021**, *61*, 102906. [CrossRef]
52. Zafar, F.; Olano, M.; Curtis, A. GPU random numbers via the tiny encryption algorithm. In Proceedings of the Conference on High Performance Graphics, Saarbrücken, Germany, 25–27 June 2010; Eurographics Association: Goslar, Germany, 2010; HPG '10; pp. 133–141.
53. Needham, R.M.; Wheeler, D.J. *Tea Extensions*; Report; Cambridge University: Cambridge, UK, 1997.
54. Wheeler, D.; Needham, R.M. *Correction to XTEA*; Report; Cambridge University: Cambridge, UK, 1997.
55. Bernstein, D.J. ChaCha, a Variant of Salsa20. In *Workshop Record of SASC*; Special Workshop Hosted by the ECRYPT Network of Excellence: Lausanne, Switzerland, 2008; Volume 8, pp. 3–5.
56. Ferguson, N.; Lucks, S.; Schneier, B.; Whiting, D.; Bellare, M.; Kohno, T.; Callas, J.; Walker, J. The skein hash function family. *Submiss. NIST (Round 3)* **2010**, *7*, 3.
57. Lara, E.; Aguilar, L.; García, J.A.; Sanchez, M.A. A Lightweight Cipher Based on Salsa20 for Resource-Constrained IoT Devices. *Sensors* **2018**, *18*, 3326. [CrossRef]
58. National Institute of Standards and Technology (NIST); Dang, Q. Secure Hash Standard. 2015. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf (accessed on 11 November 2024).
59. Diffie, W.; Hellman, M. Privacy and authentication: An introduction to cryptography. *Proc. IEEE* **1979**, *67*, 397–427. [CrossRef]
60. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed.; Cambridge University Press: New York, NY, USA, 2007.
61. Fowler, G.; Noll, L.C.; Vo, K.P.; Eastlake, D.E., 3rd; Hansen, T. The FNV Non-Cryptographic Hash Algorithm. Internet-Draft draft-eastlake-fnv-24. *Internet Eng. Task Force* (IETF) 2024, *in press*. Available online: https://ftp.fi.muni.cz/pub/internet-drafts/draft-eastlake-fnv-20.html (accessed on 11 November 2024).
62. Zhu, B. The Reference Implementations (in C and Python) of the Simeck Family of Block Ciphers. 2024. Available online: https://github.com/bozhu/Simeck (accessed on 11 November 2024).
63. Roos, A. *A Class of Weak Keys in the RC4 Stream Cipher*; Technical report; Vironix Software Laboratories, Greyville: Durban, South Africa, 1995.

64. Fluhrer, S.R.; Mantin, I.; Shamir, A. Weaknesses in the Key Scheduling Algorithm of RC4. In Proceedings of the Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography, Toronto, ON, Canada, 16–17 August 2001; SAC '01, pp. 1–24.
65. Mironov, I. (Not So) Random Shuffles of RC4. In Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, CA, USA, 18–22 August 2002; Springer: Berlin/Heidelberg, Germany, 2002; CRYPTO '02, pp. 304–319.
66. Maximov, A. Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of RC4 Family of Stream Ciphers. In *Fast Software Encryption*; Gilbert, H., Handschuh, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 342–358.
67. Wheeler, D.; Needham, R. *XXTEA: Correction to XTEA*; Computer Laboratory, University of Cambridge: Cambridge, UK, 1998.
68. Boneh, D.; Corrigan-Gibbs, H.; Schechter, S. Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks. In Proceedings of the 22nd Annual International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT), Hanoi, Vietnam, 4–8 December 2016; [CrossRef]
69. Biryukov, A.; Dinu, D.; Khovratovich, D. Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy, Saarbruecken, Germany, 21–24 March 2016; pp. 292–302. [CrossRef]
70. Turan, M.S.; Barker, E.B.; Burr, W.E.; Chen, L. *SP 800-132. Recommendation for Password-Based Key Derivation: Part 1: Storage Applications*; Technical Report; NIST: Gaithersburg, MD, USA, 2010.
71. Arcetri, T. NIST Statistical Test Suite. 2024. Available online: https://github.com/arcetri/sts (accessed on 11 November 2024).
72. Colombier, B. AIS-31 Statistical Tests. 2024. Available online: https://gitlab.com/BColombier/ais-31-statistical-tests (accessed on 11 November 2024).
73. Vaudenay, S. An experiment on DES statistical cryptanalysis. In Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, 14–16 March 1996; CCS '96, pp. 139–147. [CrossRef]
74. Matsui, M. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology—EUROCRYPT '93*; Helleseth, T., Ed.; Springer: Berlin/Heidelberg, Germany, 1994; pp. 386–397.
75. Junod, P. Statistical Cryptanalysis of Block Ciphers. Ph.D. Thesis, EPFL, Lausanne, Switzerland, 2005. [CrossRef]
76. Almukhlifi, R.; Vora, P.L. Linear Cryptanalysis of Reduced-Round Simeck Using Super Rounds. *Cryptography* **2023**, *7*, 8. [CrossRef]
77. Bagheri, N. Linear Cryptanalysis of Reduced-Round SIMECK Variants. In *Progress in Cryptology—INDOCRYPT 2015*; Biryukov, A., Goyal, V., Eds.; Springer: Cham, Switzerland, 2015; pp. 140–152.
78. Chen, J.; Wang, M.; Preneel, B. Impossible differential cryptanalysis of the lightweight block ciphers TEA, XTEA, and HIGHT. In Proceedings of the 5th International Conference on Cryptology in Africa, Ifrance, Morocco, 10–12 July 2012; AFRICACRYPT'12, pp. 117–137. [CrossRef]
79. Bogdanov, A.; Wang, M. Zero correlation linear cryptanalysis with reduced data complexity. In Proceedings of the 19th International Conference on Fast Software Encryption, Washington, DC, USA, 19–21 March 2012; FSE'12, pp. 29–48. [CrossRef]
80. Khovratovich, D.; Nikolić, I. Rotational Cryptanalysis of ARX. In *Fast Software Encryption*; Hong, S., Iwata, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 333–346.
81. Kaur, J.; Jindal, N. A secure image encryption algorithm based on fractional transforms and scrambling in combination with multimodal biometric keys. *Multimed. Tools Appl.* **2019**, *78*, 11585–11606. [CrossRef]
82. Xian, Y.; Wang, X. Fractal sorting matrix and its application on chaotic image encryption. *Inf. Sci.* **2021**, *547*, 1154–1169. [CrossRef]
83. Chen, G.; Mao, Y.; Chui, C.K. A symmetric image encryption scheme based on 3D chaotic cat maps. *Chaos Solitons Fractals* **2004**, *21*, 749–761. [CrossRef]
84. Rhouma, R.; Meherzi, S.; Belghith, S. OCML-based colour image encryption. *Chaos Solitons Fractals* **2009**, *40*, 309–318. [CrossRef]
85. Liu, H.; Zhao, B.; Huang, L. Quantum Image Encryption Scheme Using Arnold Transform and S-box Scrambling. *Entropy* **2019**, *21*, 343. [CrossRef] [PubMed]
86. Chai, X.; Bi, J.; Gan, Z.; Liu, X.; Zhang, Y.; Chen, Y. Color image compression and encryption scheme based on compressive sensing and double random encryption strategy. *Signal Process.* **2020**, *176*, 107684. [CrossRef]
87. Stoyanov, B.; Kordov, K. Image Encryption Using Chebyshev Map and Rotation Equation. *Entropy* **2015**, *17*, 2117–2139. [CrossRef]
88. Lustro, R.A.F.; Sison, A.M.; Medina, R.P. Performance Analysis of Enhanced SPECK Algorithm. In Proceedings of the 4th International Conference on Industrial and Business Engineering, Macao, China, 24–26 October 2018; ICIBE '18, pp. 256–264. [CrossRef]
89. Wang, Y. On the Design of LIL Tests for (Pseudo) Random Generators and Some Experimental Results. *arXiv* **2014**, arXiv:1401.3307.
90. Wang, Y. Law of the Iterated Logarithm Based Testing Tools. 2024. Available online: https://github.com/yonggewang/lil (accessed on 11 November 2024).
91. Cheetham, J. Open Hardware Random Number Generator. 2024. Available online: https://onerng.info/ (accessed on 11 November 2024).

92. Brown, R.G.; Eddelbuettel, D.; Bauer, D. Dieharder: A Random Number Test Suite. 2024. Available online: https://webhome.phy.duke.edu/~rgb/General/dieharder.php (accessed on 7 September 2024).
93. Walker, J. ENT—A Pseudorandom Number Sequence Test Program. 2024. Available online: https://www.fourmilab.ch/random/ (accessed on 7 September 2024).
94. Lenstra, A. Key lengths. In *Handbook of Information Security, Volume 1: Key Concepts, Infrastructure, Standards and Protocols*; Bidgoli, H., Ed.; Wiley: Hoboken, NJ, USA, 2006.
95. Biryukov, A.; De Cannière, C. Data encryption standard (DES). In *Encyclopedia of Cryptography and Security*; van Tilborg, H.C.A., Ed.; Springer: Boston, MA, USA, 2005; pp. 129–135. [CrossRef]
96. Smith, J. The Design of Lucifer, a Cryptographic Device for Data Communications. *Technical Report*. 1971. Available online: https://dominoweb.draco.res.ibm.com/reports/RC3326.pdf (accessed on 13 January 2025).
97. Schaller, R.R. Moore's law: Past, present, and future. *IEEE Spectr.* **1997**, *34*, 52–59. [CrossRef]
98. June 2024 | TOP500—top500.org. 2024. Available online: https://top500.org/lists/top500/2024/06/ (accessed on 7 September 2024).
99. Bhargavan, K.; Leurent, G. On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; CCS '16, pp. 456–467. [CrossRef]
100. Encarnacion, P.C.; Gerardo, B.D.; Hernandez, A.A. Performance Analysis on Enhanced Round Function of SIMECK Block Cipher. In Proceedings of the 2020 12th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 12–15 June 2020; pp. 270–275. [CrossRef]
101. Anton, A.A.; Csereoka, P. SIMECK-R Cipher. 2024. Available online: https://github.com/BANPUMP-team/simeckr (accessed on 11 November 2024).
102. Huang, Y.; Li, L.; Li, D.; Li, Y. IABC: A neural integral distinguisher for AND-RX Ciphers. *J. Intell. Fuzzy Syst.* **2024**, *47*, 451–465. [CrossRef]
103. Ranea, A.; Rijmen, V. Characteristic Automated Search of Cryptographic Algorithms for Distinguishing Attacks (CASCADA). Cryptology ePrint Archive, Paper 2022/513. 2022. Available online: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/ise2.12077 (accessed on 11 November 2024).
104. Ranea, A.; Liu, Y.; Ashur, T. An Easy-to-Use Tool for Rotational-XOR Cryptanalysis of ARX Block Ciphers. *Proc. Rom. Acad. Ser. A* **2017**, *18*, 307–316. Available online: https://acad.ro/sectii2002/proceedings/doc2017-4s/02artSupl.pdf (accessed on 11 November 2024)
105. Brecht, W. White-Box Cryptography: Hiding Keys in Software. *NAGRA Kudelski Group*. 2012. Available online: https://www.whiteboxcrypto.com/files/2012_misc.pdf (accessed on 11 November 2024).
106. Ranea, A.; Vandersmissen, J.; Preneel, B. Implicit White-Box Implementations: White-Boxing ARX Ciphers. In Proceedings of the Advances in Cryptology—CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, 15–18 August 2022; Proceedings, Part I; Springer: Berlin/Heidelberg, Germany, 2022; pp. 33–63. [CrossRef]
107. Delerablée, C.; Lepoint, T.; Paillier, P.; Rivain, M. White-Box Security Notions for Symmetric Encryption Schemes. In Proceedings of the Selected Areas in Cryptography Conference, Burnaby, BC, Canada, 14–16 August 2013; SAC '13.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.