*Computer Systems Science & Engineering*
DOI: 10.32604/csse.2023.035576
***Article***

Tech Science Press

# An Anti-Physical Attack Scheme of ARX Lightweight Algorithms for IoT Applications

**Qiang Zhi[1], Xiang Jiang[1], Hangying Zhang[2], Zhengshu Zhou[3], Jianguo Ren[1] and Tong Huang[4],[*]**

[1]School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, 221116, China
[2]State Key Laboratory of Precision Measurement Technology and Instruments, Tsinghua University, Beijing, 100084, China
[3]Graduate School of Informatics, Nagoya University, Nagoya, 4648601, Japan
[4]College of Chemical Engineering, Nanjing Tech University, Nanjing, 211816, China
*Corresponding Author: Tong Huang. Email: jsnu_cs@163.com

Received: 26 August 2022; Accepted: 25 October 2022

**Abstract:** The lightweight encryption algorithm based on Add-Rotation-XOR (ARX) operation has attracted much attention due to its high software affinity and fast operation speed. However, lacking an effective defense scheme for physical attacks limits the applications of the ARX algorithm. The critical challenge is how to weaken the direct dependence between the physical information and the secret key of the algorithm at a low cost. This study attempts to explore how to improve its physical security in practical application scenarios by analyzing the masking countermeasures of ARX algorithms and the leakage causes. Firstly, we specify a hierarchical security framework by quantitatively evaluating the indicators based on side-channel attacks. Then, optimize the masking algorithm to achieve a trade-off balance by leveraging the software-based local masking strategies and non-full-round masking strategies. Finally, refactor the assembly instruction to improve the leaks by exploring the leakage cause at assembly instruction. To illustrate the feasibility of the proposed scheme, we further conducted a case study by designing a software-based masking method for Chaskey. The experimental results show that the proposed method can effectively weaken the impact of physical attacks.

**Keywords:** IoT security; lightweight encryption; anti-physical attack; ARX algorithms

## 1 Introduction

With the promotion and development of the Internet of Things (IoT), data security problem in the hardware devices are becoming more and more prominent. However, due to the strict limitations of economy and energy consumption for various reasons (economy, energy consumption, etc.), the performance of hardware devices in IoT, such as sensor networks and distributed systems, are usually orders of magnitude lower than server or client PCs. Since these performance-constrained devices may not be enough to support these mainstream security protection measures, many traditional data protection schemes have not satisfied the security necessary [1–3]. To this end, some professional organizations, such as the National Institute of Standards and Technology (NIST) information technology laboratory [4], have actively established the standard lightweight encryption algorithms for IoT devices. In the decade, a plenty of the candidate encryption algorithms have been widely applied in practical scenarios and making a great contribution for related IoT researches. However, these algorithms are facing the serious threat of side-channel attacks (also known as physical attacks) [5,6], and their strategies to deal with physical attacks need to be verified and further optimized [7].

ARX algorithm, composed of ADD-ROTATION-XOR operation, is one of the popular methods among the NIST candidate lightweight algorithms. Due to its high software affinity and fast operation speed, ARX algorithm has achieved excellent performance in software and special instruction architecture [8]. However, the drawback of ARX algorithm is that it is usually vulnerable while attacked by the power analysis (DPA), and how to defense the DPA has become the key issue in related area [9,10]. The major challenge in solving this problem lies in how to weaken or even eliminate the direct dependence between the physical information and the secret key of algorithm at a low cost. The core countermeasures adopted can be divided into masking countermeasures and hiding countermeasures [11]. Specifically, by modifying the algorithm, masking strategy decouples sensitive intermedia variables to protect operation results from intermedia variables affecting. Hiding countermeasures aim to reduce the distinguishability of the data by averaging the “0” and “1”.

Furthermore, the masking strategies of ARX algorithm can be divided into software-based masking strategies [12,13] and hardware-based (Threshold Implementation) masking strategies [14,15]. Software-based masking countermeasures are highly portable, but their security may be affected by the way they are compiled and optimized. Hardware-based masking countermeasures need to consider the impact of the difference in instruction architecture, and will be affected by computing power, circuit scale, energy consumption, etc. [16,17]. Regardless of the implementation, ARX masking strategies may be limited due to its cost several times the algorithm itself [10]. In addition, although the theoretical security of some mainstream encryption algorithms’ masking countermeasures has been verified, they still leak when subjected to DPA, and even threaten the security [18,19]. Therefore, it is necessary to explore more effective methods to reduce the cost of ARX algorithm masking countermeasures on the one hand, and improve its physical security on the other hand.

## 2 Related Research

- Status Quo of ARX lightweight algorithms in IoT applications

In recent years, many well-designed lightweight block cipher algorithms have emerged in the field of IoT applications [20–22]. Some lightweight block ciphers use the S-box as the nonlinear part, and some block ciphers such as ARX algorithms use the addition operation as the nonlinear part. The cryptographic algorithm based on ARX realizes the nonlinearity, ambiguity, and diffusion of the algorithm through the combination of Addition, Rotation, and XOR. According to the report of FelICS [23], the block cipher implementation of ARX is superior to other S-box-based ciphers, and some ARX algorithms specially designed for IoT nodes have emerged [24–27].

- Problems existing in ARX lightweight algorithms in the field of IoT security

ARX algorithm itself has a fast operation speed, a small amount of calculation, and the mathematical theory is safe, but the resistance to physical attacks is weak [28]. ARX algorithms are different from the block ciphers with S-box, and their masking strategy needs to be aimed at the whole algorithm rather than the S-box part, which will lead to the cost of masking strategy being too high [10]. Therefore, the application of ARX algorithms in highly sensitive scenarios will be limited.

- Research status of ARX lightweight algorithms in physical security

1) For the problem that the cost of the masking strategy of ARX algorithms is too expensive, there are few studies at present. In addition to improving the ability of the ARX algorithm itself to resist physical attacks [29], it is generally believed that the masking strategy of ARX algorithms needs to be implemented as a whole algorithm, which needs to involve arithmetic masking (AM), Boolean masking (BM) [30], and Conversion between arithmetic and Boolean masking [31]. Some studies have implemented local masking (initialization phase and finalization phase) countermeasures to special algorithms based on software to reduce the overhead of masking countermeasures while ensuring security [32]. The hardware-based masking scheme is mainly implemented based on TI (Threshold Implementation) [33,34], and there are studies using the second operand feature of the ARM instruction set to reduce the computational complexity of Boolean masking [35]. Inserting pseudo-instructions into an algorithm to hide side-channel information is a means of resisting high-order physical attacks, but this method often causes concerns in security evaluation and scalability [36]. It is also an effective protection method to find codes that may lead to leaks during the compilation process and then take specific measures [37,38].
2) Compared with other block ciphers with S-box, the leakage model of ARX algorithms has more leakage points [39,40], and the leakage trajectory is easy to track [41,42]. ARX algorithms with masking countermeasure are found to be leaking by some new leakage models [18,19,43]. Therefore, in view of the problems and shortcomings of the above research, it is necessary to explore a new idea to deal with the physical security of ARX algorithms.

## 3 Three Aspects to Improve the Physical Security of ARX Algorithms

Based on the research status mentioned above, we summarize the key problems and scientific significance of ARX algorithms for IoT applications in three different aspects.

1) In the NIST lightweight encryption algorithm solicitation activity, the physical security performance of candidate ARX algorithms is yet to be verified. In the view of the characteristics of ARX algorithms, it appears necessary to refer to the actual security standards required by the industry, combine the algorithms' computing speed and required platform performance, explore the relationship between the cost of masking countermeasures and actual security level, and create a set of reasonable security classifications. The framework aims to provide a reference for the application of ARX algorithms and the cost of masking countermeasures for different IoT application scenarios. Therefore, trying to achieve a breakthrough in the physical security evaluation standard of ARX algorithms must be significant.
2) The cost of the masking strategy of ARX algorithms is too expensive. In view of the consideration of the performance of IoT devices, the practical application significance should be limited. Therefore, the masking strategy of ARX algorithms needs further study and optimization. Combined with the security classifications in the first aspect, try to explore new anti-side-channel attack methods such as software-based local masking strategy, aiming to reduce the cost of defending against physical attacks, making it more suitable for IoT platforms in different scenarios.
3) The problem of leakage of ARX algorithms with masking countermeasures indicates that the leakage may not be completely eliminated, and it reflects the necessity of the security classification framework. Therefore, it is indispensable to explore the causes of the leakage caused by ARX algorithms at the assembly instruction level and the corresponding protection countermeasures.

The relationship between the three aspects is shown in Fig. 1.

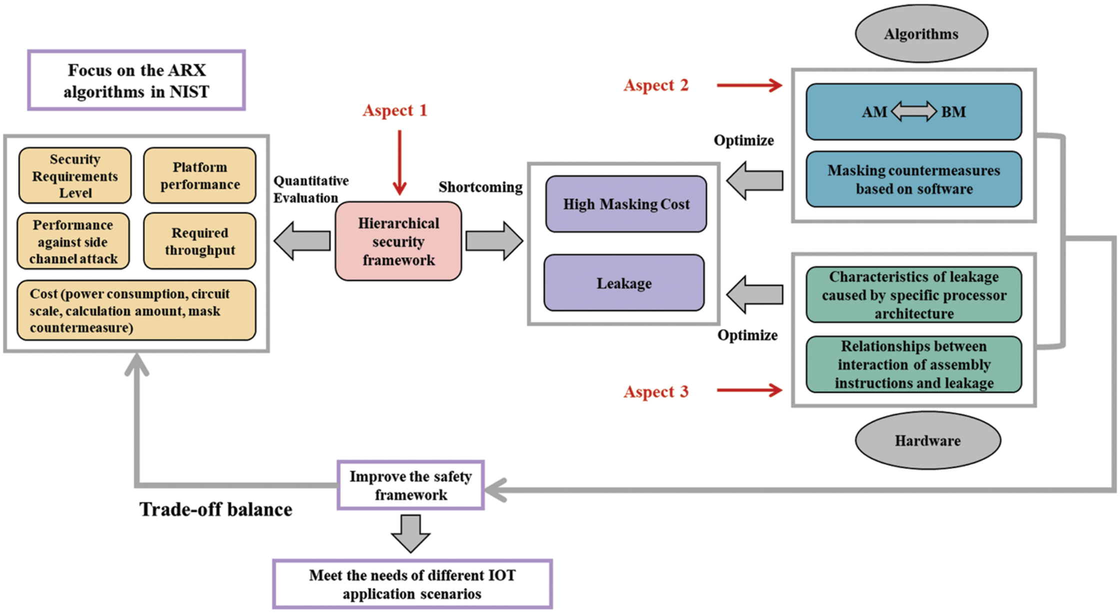

Algorithms
Focus on the ARX algorithms in NIST
Aspect 2
Aspect 1
AM ⟺ BM
Optimize
Security Requirements Level
Platform performance
Quantitative Evaluation
Shortcoming
Masking countermeasures based on software
High Masking Cost
Performance against side channel attack
Required throughput
Hierarchical security framework
Cost (power consumption, circuit scale, calculation amount, mask countermeasure)
Leakage
Characteristics of leakage caused by specific processor architecture
Optimize
Relationships between interaction of assembly instructions and leakage
Aspect 3
Hardware
Improve the safety framework
Trade-off balance
Meet the needs of different IOT application scenarios

**Figure 1:** Schematic diagram of improving the physical security of ARX algorithms

ARX algorithm security classification frameworks: It aims to not only consider various attribute indicators of ARX lightweight algorithms, but also combine the actual needs of life in the industry to propose a complete set of security classifications frameworks to reasonably apply different algorithms and anti-physical attack strategies in different scenarios to achieve a coordinated balance.

Optimizing the masking countermeasures of ARX algorithms: It aims to reduce the cost of ARX algorithms in terms of physical attack resistance. However, few studies have discussed the dedicated masking approach of the ARX algorithms. To solve that, the key issue is how to optimize the number of operations in the transformation between Boolean operations and arithmetic operations in masking countermeasures, and explore whether software-based local masking countermeasures can greatly improve the resistance to physical attacks while reducing the computational cost.

Exploring the cause of leakage: It aims to explore the possible causes of security problems caused by leakage even after the algorithm is masked. The key part is to explore whether the types of ARX algorithm assembly instructions and specific combinations will obviously cause potential leakage based on specific processor architecture, and what measures should be taken to optimize such assembly instructions with the least cost.

## 4 Discussion of Research Methods and Technical Routes

### *4.1 Research Methods*

Under the current research background of IoT security issues, the existing ARX algorithms and NIST candidate ARX algorithms can be used as research and experimental targets to analyze the operation speed, calculation amount, number of instructions, energy consumption, physical attack resistance, and the masking cost. Then a hierarchical security framework based on performance and security (Cost-Security Balance) can be proposed by quantitative evaluation combined with actual industry standards. For the problem that the masking strategy of the ARX algorithm is too expensive, it is necessary to consider optimizing AM, BM, and the conversion between AM and BM to reduce the cost. Additionally, considering the problem of leakage after algorithm masking, we recommend analyzing and collecting assembly instructions (or some fixed instruction pairs) that may cause leakage, and verify their effectiveness through manual intervention and assembly instruction reorganization. The research methods and their relationships are shown in Fig. 2.

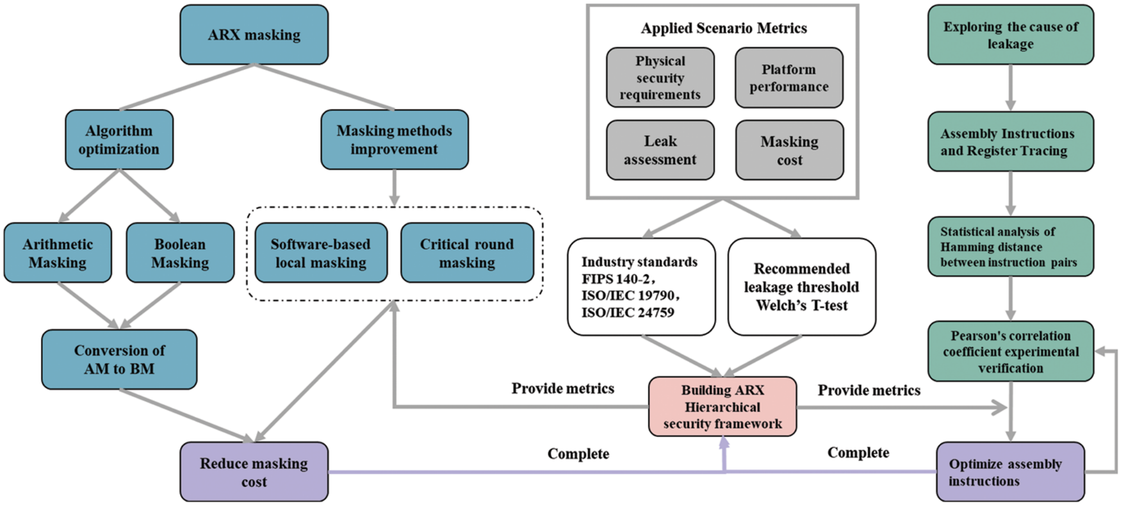

ARX masking
Algorithm optimization
Masking methods improvement
Arithmetic Masking
Boolean Masking
Software-based local masking
Critical round masking
Conversion of AM to BM
Reduce masking cost
Applied Scenario Metrics
Physical security requirements
Platform performance
Leak assessment
Masking cost
Industry standards FIPS 140-2, ISO/IEC 19790, ISO/IEC 24759
Recommended leakage threshold Welch's T-test
Building ARX Hierarchical security framework
Provide metrics
Provide metrics
Complete
Complete
Exploring the cause of leakage
Assembly Instructions and Register Tracing
Statistical analysis of Hamming distance between instruction pairs
Pearson's correlation coefficient experimental verification
Optimize assembly instructions

**Figure 2:** Schematic diagram of research methods and technical routes

## *4.2 Technical Routes*

### *4.2.1 Building ARX Algorithm Security Hierarchical Framework*

To construct the ARX algorithm security hierarchical framework, it is first necessary to evaluate the operation speed, calculation amount, instruction number, energy consumption, and physical attack resistance of existing ARX algorithms or NIST candidate ARX lightweight algorithms, and then combine the evaluation criteria such as the security requirement level, platform performance, anti-side channel attack requirements, and computational cost, to finally define the security hierarchical framework according to specific application scenarios by quantitative evaluation. (See Table 1).

**Table 1:** Core evaluation metrics of ARX hierarchical security framework

| | Physical security requirements | Platform performance | Masking cost | Leak assessment | Algorithm computational cost | Leak assessment after masking | Threshold |
|---|---|---|---|---|---|---|---|
| Scenario metrics | H/M/L/N | (P) | (CO) | (L) | (O) | (Lm) | 0.01–0.1 |

Welch's T-test could be utilized for leak assessment of ARX algorithms. In formula (1), A represents fixed input, B represents random input, $X_A$ represents the average of A, $X_B$ represents the average of B, and $S_A$ and $S_B$ represent the standard deviation of A and B, respectively. The T value could be flexibly set according to the security classification framework and application scenarios (The recommended value is 0.045).

$$T = \frac{X_A - X_B}{\sqrt{\frac{S_A^2}{N_A} + \frac{S_B^2}{N_B}}} \quad (1)$$

### *4.2.2 Optimizing Masking Strategies of ARX Algorithms*

Masking is a strategy against side-channel attacks. It consists of two or more (first order/higher order) shares to represent sensitive values in cryptographic primitives. The sum or XOR of these shares is usually equal to the variable before being split value. The intermediate variable values resulting from the splitting of the primitives in the encryption algorithm into shares will not correlate with the actual (no masking countermeasures) intermediate variable values. Since the ARX algorithm uses a combination of two different types of operations, Boolean and arithmetic, two masking strategies are required: arithmetic masking and Boolean masking. Furthermore, to guarantee that all intermediate variables are independent of the masked data, the conversion between arithmetic masking and Boolean masking should be considered.

AM includes addition and subtraction operations, and BM includes XOR, rotation, and shift. In order to guarantee that intermediate variables are resistant to power analysis attacks (DPA), the correlation between arithmetic and Boolean operations should be eliminated. Due to the particularity of ARX algorithms, high-order masking algorithms are usually not practical (in view of the cost-effectiveness of computation and security) Therefore, the optimization of the computation amount in the conversion process between AM and BM is a necessary method in the first-order masking countermeasure. Fig. 3 illustrates a schematic diagram of the masking strategy of ARX algorithms.

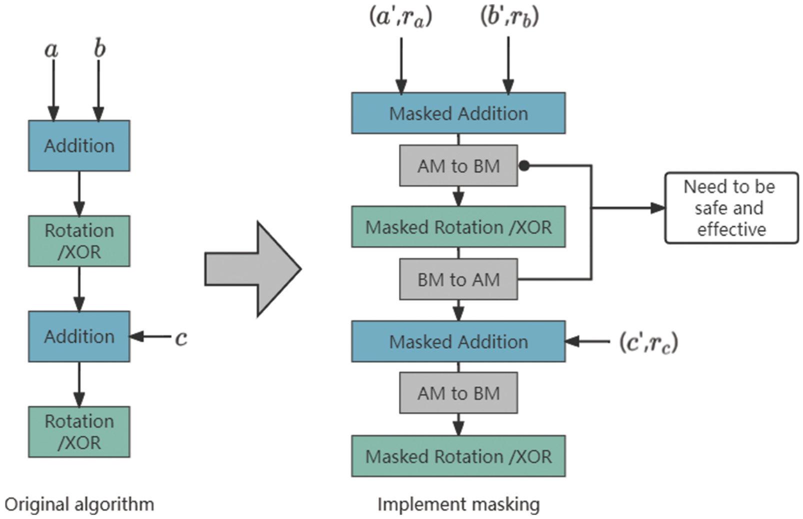


**Figure 3:** ARX algorithms masking strategy

Arithmetic operations in the ARX algorithm refer to addition and subtraction operations. Let the original intermediate variable be $x$, the intermediate variable after implementing masking is $x'$, and the random number is $r$. Then the arithmetic masking method can be expressed as formula (2). Boolean operations refer to rotation and shift, which can be expressed as formula (3).

$$x = x' + r_x mod 2^n \quad (2)$$

$$x = x' \oplus r_x \tag{3}$$

Fig. 4 presents the masking process of ARX algorithms and the location and relationship of the object of this study (conversion between arithmetic operations and Boolean operations).

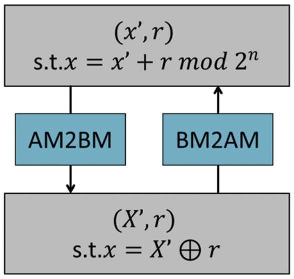


**Figure 4:** The position and relationship between AM and BM

Table 2 reveals the conditions and guaranteed objects of arithmetic operations and Boolean operations for the masking countermeasure.

**Table 2:** Conversion between AM and BM

***BM to AM***

**Require:** *(x′,r)* such that $x = x' \oplus r_x$
**Ensure:** *(A, r)* such that $x = A + r_x mod2^n$
**Algorithm of BM to AM /* need to be optimized */**

***AM to BM***

**Require:** *(A, r)* such that $x = A + r_x mod2^n$
**Ensure:** *(x′,r)* such that $x = x' \oplus r_x$
**Algorithm of AM to BM /* need to be optimized */**

### *4.2.3 Optimization of ARX Masking Algorithm Based on Software*

The ARX algorithm is designed to be very efficient in software, but it will loss advantage once it is subject to DPA. However, the ARX algorithm is different from AES or other algorithms with S-boxes, and cannot perform masking countermeasures for S-boxes. It is generally considered that the entire algorithm process needs to be masked, which will lead to high costs. In order to improve the DPA resistance and maintain the advantages based on software, we study the feasibility of the local masking countermeasure of the ARX algorithm in combination with the security hierarchical framework.

The following discussion takes the ARX algorithm Chaskey as an example. Chaskey is a permutation-based ARX algorithm for MAC authentication. It is designed to run on 32-bit processors. The Chaskey algorithm is shown in Table 3, which identifies the part of the algorithm that attempts the masking strategy. The verification of security after masking can be achieved through DPA attacks and Test Vector Leakage Assessment (TVLA). Its effectiveness can be evaluated under the ARX algorithm security hierarchical framework.

**Table 3:** Chaskey's permutation process and the local masking strategy

***Chaskey Algorithm***

```
TimesTwo(a) /* Initialization phase */
  **if** a [127] = 0 **then return** (a << 1) $\oplus$ $0^{128}$
  **else return** (a << 1) $\oplus$ $0^{120}10000111$
SubKeys(K) /* Key generation phase */
  $K_1 \leftarrow$ TimesTwo(K)
  $K_2 \leftarrow$ TimesTwo(K1)
  **return** $(K_1, K_2)$
$\text{Chaskey}^{\pi}$(K, m) /* Permutation phase, π represents permutation round, m represents plaintext */
  $(K_1, K_2) \leftarrow$ SubKeys(K)
  $m_1 \ldots km_\ell \leftarrow m$
  $h_1 \leftarrow K$
  **for** $i = 1, \ldots, \ell - 1$ **do** $h_{i+1} \leftarrow \pi(h_i \oplus m_i)$ **/* Masking Target */**
  **if** $|m_\ell| = n$ **then** $L \leftarrow K_1$ /* Final phase */
  **else**
    $m_\ell \leftarrow m_\ell k10^{n-|m\ell|-1}$
    $L \leftarrow K_2$
    $h_{\ell+1} \leftarrow \pi(h_\ell \oplus m_\ell \oplus L) \oplus L$
    **return** $\tau \leftarrow \text{right}_t(h_{\ell+1})$
```

### *4.2.4 The Cause of Leakage*

In recent years, with the gradual deepening of Internet of Things security research, some new leakage models have been proposed and it has been found that the masked algorithm will still leak and even threaten security. This is often due to design problems in the hardware and unexpected interactions in the hardware and is unavoidable. We suggest that possible leaks can be explored at the assembly instruction level of the ARX algorithm, analyze and summarize the possible causes of leaks, and try to optimize assembly instructions to reduce such leaks. A feasible technical route is as follows:

Since the interaction between instruction pairs may cause leakage, the ARX algorithm assembly instructions can be analyzed by tracing instructions and registers, the Hamming distance between related instruction pairs can be counted, and then the effect of interactions between instruction pairs on leakage can be examined by the Pearson correlation coefficient Experiments (Eq. (4)).

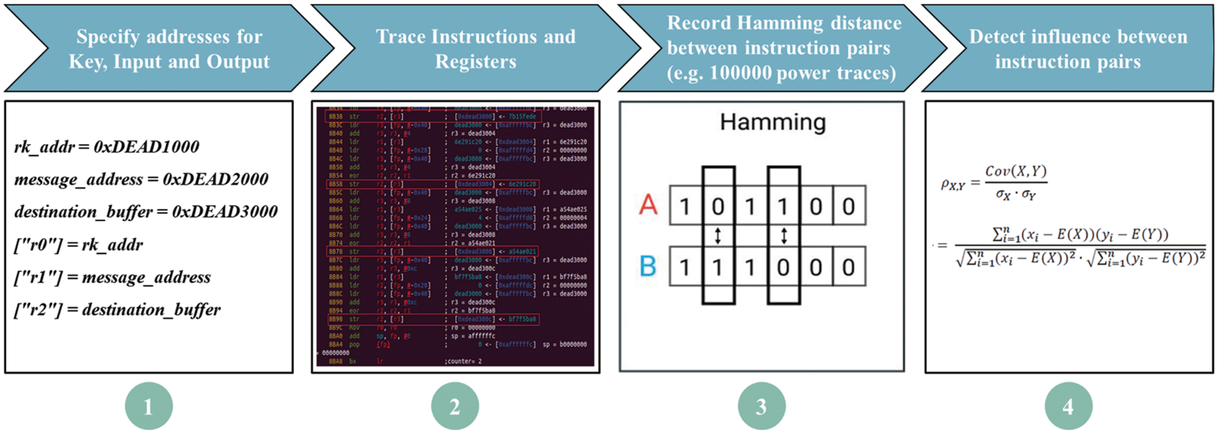
$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \cdot \sigma_Y} = \frac{|X-\bar{X}|\cdot|Y-\bar{Y}|}{||X-\bar{X}||_2||Y-\bar{Y}||_2} \tag{4}$$

The assembly instructions could be interfered with by screening, refactoring, optimization, etc., and the ability of anti-physical attack of the algorithm could be improved at a reasonable cost by combining the aforementioned security hierarchical framework indicators. Fig. 5 reveals the detailed steps of this technical route.

## 5 Experiments

Following the discussion of masking strategies in above section, we conduct an empirical study on the Chaskey algorithm. Fig. 6 shows the power consumption trajectory (with random input) of Chaskey's original algorithm and the masked algorithm (Table 3) under ARM-CortexM0 processor architecture.

Industrial IoT hardware devices need to consider device performance and energy consumption requirements, compilation and optimization could be performed when processing algorithms. Therefore, we also consider power consumption trajectories under different compilation optimization levels and the influence of noise on the trajectory as well. The noise set follows the security standards of ISO/IEC 17825 (SNR = 0.1~1).


Specify addresses for Key, Input and Output

Trace Instructions and Registers

Record Hamming distance between instruction pairs (e.g. 100000 power traces)

Detect influence between instruction pairs

*rk_addr = 0xDEAD1000*

*message_address = 0xDEAD2000*

*destination_buffer = 0xDEAD3000*

*["r0"] = rk_addr*

*["r1"] = message_address*

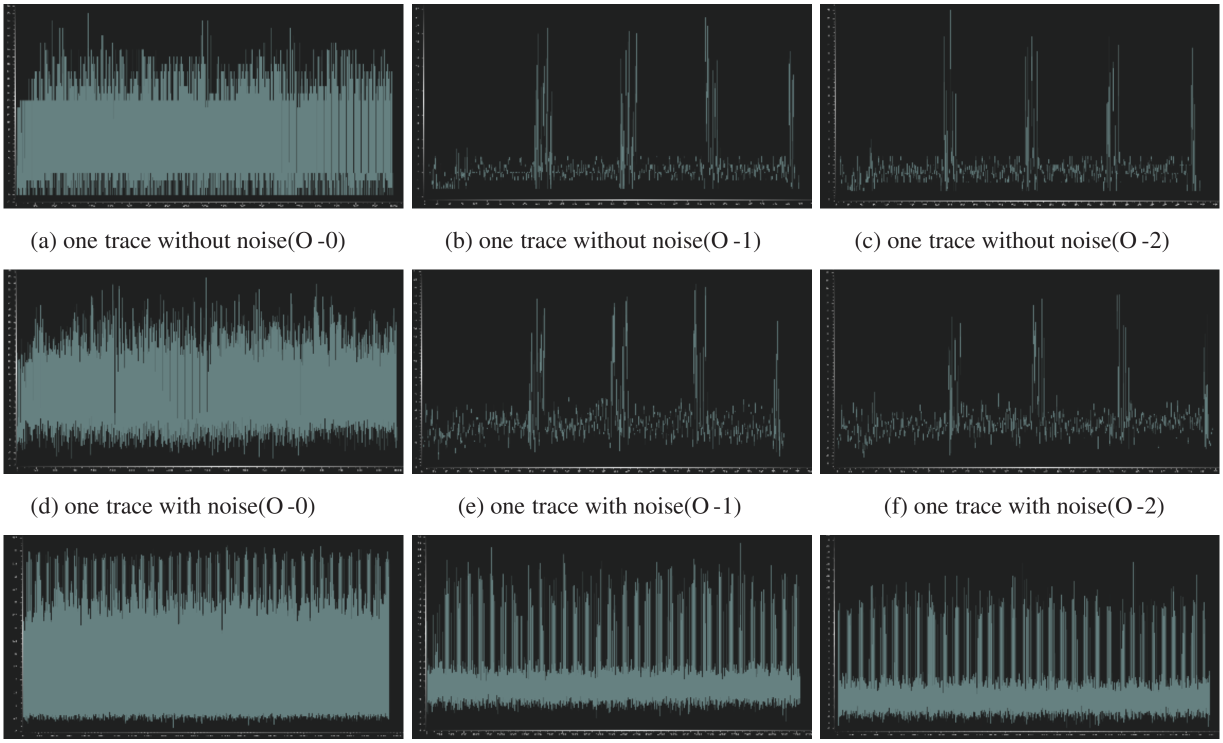
*["r2"] = destination_buffer*

```
8B34 ldr    r3, [fp, #-0x40]   ; dead3000 <- [0xafffffbc]  r3 = dead3000
8B38 str    r2, [r3]           ; [0xdead3000] <- 7b15fede
8B3C ldr    r3, [fp, #-0x40]   ; dead3000 <- [0xafffffbc]  r3 = dead3000
8B40 add    r3, r3, #4         ; r3 = dead3004
8B44 ldr    r1, [r3]           ; 6e291c20 <- [0xdead3004]  r1 = 6e291c20
8B48 ldr    r2, [fp, #-0x28]   ;        0 <- [0xafffffd4]  r2 = 00000000
8B4C ldr    r3, [fp, #-0x40]   ; dead3000 <- [0xafffffbc]  r3 = dead3000
8B50 add    r3, r3, #4         ; r3 = dead3004
8B54 eor    r2, r2, r1         ; r2 = 6e291c20
8B58 str    r2, [r3]           ; [0xdead3004] <- 6e291c20
8B5C ldr    r3, [fp, #-0x40]   ; dead3000 <- [0xafffffbc]  r3 = dead3000
8B60 add    r3, r3, #8         ; r3 = dead3008
8B64 ldr    r1, [r3]           ; a54ae025 <- [0xdead3008]  r1 = a54ae025
8B68 ldr    r2, [fp, #-0x24]   ;        4 <- [0xafffffd8]  r2 = 00000004
8B6C ldr    r3, [fp, #-0x40]   ; dead3000 <- [0xafffffbc]  r3 = dead3000
8B70 add    r3, r3, #8         ; r3 = dead3008
8B74 eor    r2, r2, r1         ; r2 = a54ae021
8B78 str    r2, [r3]           ; [0xdead3008] <- a54ae021
8B7C ldr    r3, [fp, #-0x40]   ; dead3000 <- [0xafffffbc]  r3 = dead3000
8B80 add    r3, r3, #0xc       ; r3 = dead300c
8B84 ldr    r1, [r3]           ; bf7f5ba8 <- [0xdead300c]  r1 = bf7f5ba8
8B88 ldr    r2, [fp, #-0x20]   ;        0 <- [0xafffffdc]  r2 = 00000000
8B8C ldr    r3, [fp, #-0x40]   ; dead3000 <- [0xafffffbc]  r3 = dead3000
8B90 add    r3, r3, #0xc       ; r3 = dead300c
8B94 eor    r2, r2, r1         ; r2 = bf7f5ba8
8B98 str    r2, [r3]           ; [0xdead300c] <- bf7f5ba8
8B9C mov    r0, r0             ; r0 = 00000000
8BA0 add    sp, fp, #0         ; sp = affffffc
8BA4 pop    {fp}               ;        0 <- [0xaffffffc]  sp = b0000000
= 00000000
8BA8 bx     lr                 ;counter= 2
```

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \cdot \sigma_Y}$$

$$= \frac{\sum_{i=1}^{n}(x_i - E(X))(y_i - E(Y))}{\sqrt{\sum_{i=1}^{n}(x_i - E(X))^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - E(Y))^2}}$$

**Figure 5:** ARX algorithms leakage countermeasure process at assembly instruction level


(a) one trace without noise(O -0)

(b) one trace without noise(O -1)

(c) one trace without noise(O -2)

(d) one trace with noise(O -0)

(e) one trace with noise(O -1)

(f) one trace with noise(O -2)

(g) one masking trace with noise(O -0)

(h) one masking trace with noise(O -1)

(i) one masking trace with noise(O -2)

**Figure 6:** Power consumption traces of Chaskey and the masked algorithm under ARM-CortexM processor architecture

All parameters of this experiment are shown in Table 4. The simulation environment of the experiment is based on the microprocessor ARM-CortexM0, the input is random 16*32 bit, and the key is fixed 4*32 bit.

**Table 4:** Experimental parameters

| | a | b | c | d | e | f | g | h | i |
|---|---|---|---|---|---|---|---|---|---|
| Instructions | ≈10000 | ≈750 | ≈720 | ≈10000 | ≈750 | ≈720 | ≈570000 | ≈75500 | ≈71500 |
| Optimization level | Lv 0 | Lv 1 | Lv 2 | Lv 0 | Lv 1 | Lv 2 | Lv 0 | Lv 1 | Lv 2 |
| Noise (SNR) | 0 | 0 | 0 | 0.1–1 | 0.1–1 | 0.1–1 | 0.1–1 | 0.1–1 | 0.1–1 |

Fig. 7 shows the power consumption trace of Chaskey and the part where the masking strategy is to be implemented (where module 1 and module 2 represent the masking countermeasure parts).

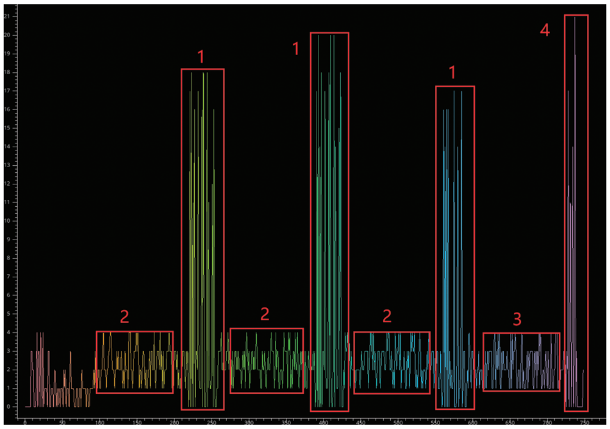


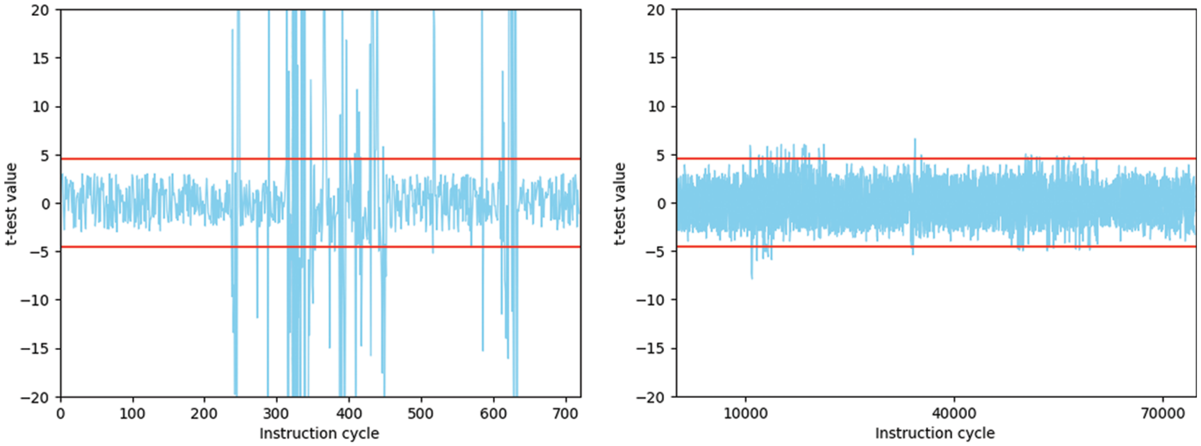
**Figure 7:** Chaskey masking strategy in the power consumption traces under ARM-CortexM processor architecture

We then performed Welch's t-test on the experimental subjects. The experimental method, source code, and datasets generated in this study are accessible (refer to the Data Availability Statement). We collected 100,000 traces for each algorithm. To reduce memory usage, we calculate the t-test values incrementally by using Welford's algorithm. The experimental results are shown in Fig. 8.

## 6 Discussion

The experimental results show that the masking strategy for Chaskey can effectively reduce the leakage, but it leads to a new problem, i.e., the masking cost is too expensive. Specifically, the number of instructions for the mask is about 100 times that of the original algorithm and this will result in limited application on devices with strict resource constraints. We perform the same masking approach on SPECK and Simon, and the experimental results are shown in Table 5.

It is worth mentioning that the random numbers introduced in our experiments are not fixed, which is one of the reasons for the huge overhead. There have been related studies on the reuse of random numbers in masking strategies, but the research in this area is mainly focused on stream cipher and threshold implementation [44,45], and software-based optimization strategies for ARX block cipher deserve further research. Moreover, the leakage assessment model may have a large impact on the results, which also deserves further study.



(a) TVLA of original Chaskey algorithm (O-1) (b)TVLA of software-based masked Chaskey algorithm (O-1)

**Figure 8:** Leakage Assessment of Chaskey and masked Chaskey

**Table 5:** Cost comparison of ARX algorithms in experiments (O–1)

| Algorithm | Instructions (original) | Instructions (masked) | Key (bit) | Input (bit) | Cost |
|---|---|---|---|---|---|
| Chaskey32 | ≈750 | ≈75500 | 4 * 32 | 16 * 32 | ≈x100 |
| SPECK32 | ≈790 | ≈84200 | 4 * 32 | 16 * 32 | ≈x106 |
| Simon32 | ≈1050 | ≈114550 | 4 * 32 | 16 * 32 | ≈x109 |

## 7 Conclusion

In this study, we mainly discuss the anti-physical attack scheme design of ARX lightweight algorithm for IoT applications and focus on how to systematically optimize the application of ARX lightweight encryption algorithms in the context of IoT security. In addition, we give some different but valuable suggestions:

- Reasonably apply different algorithms and anti-physical attack strategies in different scenarios. It is necessary to comprehensively consider the computing speed, computing platform, masking cost and other indicators of the ARX algorithm, and combine industry standards and actual needs to define a hierarchical security framework.
- A strategy based on a combination of software and hardware should be adopted in optimizing the masking and reducing leakage of ARX algorithms.

While we have discussed various aspects of how to improve the physical security of the ARX algorithm across the board, we have only conducted a case study of software implementation-based masking countermeasures. The experimental results show that the software implementation-based masking countermeasure is an effective countermeasure against physical attacks, and the rest of this research requires further in-depth analysis and discussion, which will be future work.

**Funding Statement:** This work was partially supported by the Natural Science Foundation of Jiangsu Province under Grant No. BK20201462, and partially supported by the Scientific Research Support Project of Jiangsu Normal University under Grant No. 21XSRX001.

**Availability of Data and Materials:** The source code and datasets generated and analyzed during the current study are available in the [Zhi-JSNU/ARX-Masking] repository, [https://github.com/Zhi-JSNU/ARX-Masking].

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] S. Li and L. D. Xu, *Securing the Internet of Things*, 1st ed., Amsterdam, Netherlands: Syngress, Elsevier Science, 2017.

[2] M. N. Bhuiyan, M. M. Rahman, M. M. Billah and D. Saha, “Internet of Things (IoT): A review of its enabling technologies in healthcare applications, standards protocols, security, and market opportunities,” *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10474–10498, 2021.

[3] C. P. Sandhya and B. C. Manjith, “Challenging aspects of data preserving algorithms in IoT enabled smart societies,” in *Society 5.0: Smart Future Towards Enhancing the Quality of Society.* Singapore: Springer, pp. 87–111, 2022.

[4] M. S. Turan, K. McKay, D. Chang, Ç. Çalık, L. Bassham *et al.,* “Status report on the second round of the NIST lightweight cryptography standardization process,” NISTIR 8369, 2021.

[5] D. Pokorný, P. Socha and M. Novotný, “Side-channel attack on rainbow post-quantum signature,” in *Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, Grenoble, France, pp. 565–568, 2021.

[6] A. B. Hansen, M. Eskilden and E. H. Nielsen, “Toolchain for timing leakage analysis of NIST lightweight crypto candidates,” in *NIST Lightweight Cryptography Workshop*, Virtual Conference, 2020.

[7] W. Wu, Z. Liu, H. Yang and J. Zhang, “Survey of side-channel attacks and countermeasures on post-quantum cryptography (in Chinese),” *Journal of Software*, vol. 32, no. 4, pp. 1165–1185, 2021.

[8] C. Liu, Y. Wu, J. Wu and C. Zhao, “Survey on RISC-V system architecture research (in Chinese),” *Journal of Software*, vol. 32, no. 12, pp. 3992–4024, 2021.

[9] Y. Yan and O. Elisabeth, “Examining the practical side channel resilience of ARX-boxes,” in *ACM Int. Conf. on Computing Frontiers*, Alghero, Italy, pp. 373–379, 2019.

[10] F. Coleman, B. Rezvani, S. Sachin and W. Diehl, “Side channel resistance at a cost: A comparison of ARX-based authenticated encryption,” in *Int. Conf. on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, pp. 193–199, 2020.

[11] A. Singh, M. Kar, S. Mathew, A. Rajan, V. De *et al.,* “Exploiting on-chip power management for side-channel security,” in *Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, Dresden, Germany, pp. 401–406, 2018.

[12] P. Gao, J. Zhang, F. Song and C. Wang, “Verifying and quantifying side-channel resistance of masked software implementations,” *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 3, pp. 1–32, 2019.

[13] S. Gao, J. Großschädl, B. Marshall, D. Page, T. H. Pham *et al.,* “An instruction set extension to support software-based masking,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 4, pp. 283–325, 2021.

[14] B. Jungk, R. Petri and M. Stöttinger, “Efficient side-channel protections of ARX ciphers,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, pp. 627–653, 2018.

[15] B. Seok and C. Lee, “Fast implementations of ARX-based lightweight block ciphers (SPARX, CHAM) on 32-bit processor,” *International Journal of Distributed Sensor Networks*, vol. 15, no. 9, pp. no pagination, 2019.

[16] A. Heuser, S. Picek, S. Guilley and N. Mentens, “Lightweight ciphers and their side-channel resilience,” *IEEE Transactions on Computers*, vol. 69, no. 10, pp. 1434–1448, 2020.

[17] H. Chen, W. Xi, L. Fan, Z. Jiao and J. Feng, “Side channel analysis and evaluation on cryptographic products (in Chinese),” *Journal of Electronics & Information Technology*, vol. 42, no. 8, pp. 1836–1845, 2020.

[18] D. McCann, E. Oswald and C. Whitnall, “Towards practical tools for side channel aware software engineering: ‘Grey Box’ modelling for instruction leakages,” in *26th USENIX Security Symp.*, Vancouver, BC, pp. 199–216, 2017.

[19] M. A. Shelton, N. Samwel and L. Batina, “ROSITA: Towards automatic elimination of power-analysis leakage in ciphers,” in *Network and Distributed Systems Security (NDSS) Symp.*, Virtual Conference, pp. 1–17, 2021.

[20] B. Liu, L. Li, R. Wu, M. Xie and Q. Li, “Loong: A family of involutional lightweight block cipher based on SPN structure,” *IEEE Access*, vol. 7, pp. 136023–136035, 2019.

[21] L. Li, B. Liu, Y. Zhou and Y. Zou, “SFN: A new lightweight block cipher,” *Microprocessors and Microsystems*, vol. 60, no. 7, pp. 138–150, 2018.

[22] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim *et al.,* “GIFT: A small present,” in *Int. Conf. on Cryptographic Hardware and Embedded Systems*, Taipei, Taiwan, pp. 321–345, 2017.

[23] D. D. Dinu, A. Biryukov, J. Groszsch, D. Khovratovich and Y. L. Corre, “FELICS-fair evaluation of lightweight cryptographic,” in *NIST Workshop Lightweight Cryptogr*, Gaithersburg, MD, USA, 2015.

[24] B. Koo, D. Roh, H. Kim, Y. Jung, D. -G. Lee *et al.,* “CHAM: A family of lightweight block ciphers for resource-constrained devices,” in *Int. Conf. on Information Security and Cryptology*, Seoul, South Korea, pp. 3–25, 2017.

[25] Y. Guo, L. Li and B. Liu, “Shadow: A lightweight block cipher for IoT,” *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 13014–13023, 2021.

[26] N. Mouha, B. Mennink, A. V. Herrewege, D. Watanabe, B. Preneel *et al.,* “Chaskey: An efficient MAC algorithm for 32-bit microcontrollers,” in *Selected Areas in Cryptography*, Montreal, QC, Canada, pp. 306–323, 2014.

[27] D. Roh, B. Koo, Y. Jung, I. W. Jeong, D. -G. Lee *et al.,* “Revised version of block cipher CHAM,” in *Information Security and Cryptology*, Seoul, South Korea, pp. 1–19, 2019.

[28] M. Rodinko and R. Oliynykov, “Open problems of proving security of ARX-based ciphers to differential cryptanalysis,” in *Int. Scientific-Practical Conf. Problems of Infocommunications, Science and Technology*, Kharkov, Ukraine, pp. 228–231, 2017.

[29] M. Coutinho and T. C. S. Neto, “Improved linear approximations to ARX ciphers and attacks against chacha,” in *Annual Int. Conf. on the Theory and Applications of Cryptographic Techniques*, Cham, pp. 711–740, 2021.

[30] A. Biryukov, D. Dinu, Y. L. Corre and A. Udovenko, “Optimal first-order boolean masking for embedded IoT devices,” in *Int. Conf. on Smart Card Research and Advanced Applications*, Cham, Switzerland, pp. 22–41, 2017.

[31] M. V. Beirendonck, J. -P. D’Anvers and I. Verbauwhede, “Analysis and comparison of table-based arithmetic to boolean masking,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, pp. 275–297, 2021.

[32] A. Adomnicai, J. Fournier and L. Masson, “Masking the lightweight authenticated ciphers ACORN and Ascon in software,” *Cryptology ePrint Archive*, vol. 2018, pp. 708–723, 2018.

[33] I. Buhan, L. Batina, Y. Yarom and P. Schaumont, “SoK: Design tools for side-channel-aware implementations,” in *ACM on Asia Conf. on Computer and Communications Security*, NY, USA, pp. 756–770, 2022.

[34] S. Gao, B. Marshall, D. Page and T. Pham, “FENL: An ISE to mitigate analogue micro-architectural leakage,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 2, pp. 73–98, 2020.

[35] D. Dinu, J. Großschädl and Y. L. Corre, “Efficient masking of ARX-based block ciphers using carry-save addition on boolean shares,” in *Int. Conf. on Information Security*, Ho Chi Minh City, Vietnam, pp. 39–57, 2017.

[36] J. Lee and D. -G. Han, “Security analysis on dummy based side-channel countermeasures—Case study: AES with dummy and shuffling,” *Applied Soft Computing*, vol. 93, pp. 106352, 2020.

[37] A. Abromeit, F. Bache, L. A. Becker, M. Gourjon, T. Güneysu *et al.,* “Automated masking of software implementations on industrial microcontrollers,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, pp. 1006–1011, 2021.

[38] G. Barthe, B. Grégoire and V. Laporte, "Secure compilation of side-channel countermeasures: The case of cryptographic "constant-time"," in *IEEE 31st Computer Security Foundations Symp.*, Oxford, UK, pp. 328–343, 2018.

[39] Y. Yan, E. Oswald and S. Vivek, "An analytic attack against ARX addition exploiting standard side-channel leakage," in *Int. Conf. on Information Systems Security and Privacy*, Vienna, Austria, pp. 89–97, 2021.

[40] F. Bache, T. Schneider, A. Moradi and T. Giineysu, "SPARX—A side-channel protected processor for ARX-based cryptography," in *Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, Lausanne, Switzerland, pp. 990–995, 2017.

[41] L. Song, Z. Huang and Q. Yang, "Automatic differential analysis of ARX block ciphers with application to SPECK and LEA," in *Australasian Conf. on Information Security and Privacy*, Melbourne, Australia, pp. 379–394, 2016.

[42] Z. Liu, Y. Li, L. Jiao and M. Wang, "A new method for searching optimal differential and linear trails in ARX ciphers," *IEEE Transactions on Information Theory*, vol. 67, no. 2, pp. 1054–1068, 2021.

[43] G. Barthe, M. Gourjon, B. Grégoire, M. Orlt, C. Paglialonga *et al.,* "Masking in fine-grained leakage models: Construction, implementation and verification," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 2, pp. 189–228, 2021.

[44] B. Jungk, R. Petri and M. Stöttinger, "Efficient side-channel protections of ARX ciphers," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, pp. 627–657, 2018.

[45] S. Sallam and B. D. Beheshti, "A survey on lightweight cryptographic algorithms," in *TENCON, 2018-2018 IEEE Region 10 Conf.*, Jeju, Korea (South), pp. 1784–1789, 2018.