Balanced Dual-Mask Protection Scheme for GIFT Cipher Against Power Attacks

Aljuffri, A.A.M.; Reinbrecht, Cezar; Hamdioui, S.; Taouil, M.; Sepulveda, Johanna

DOI
10.1109/VTS52500.2021.9794230

Publication date
2022

Document Version
Final published version

Published in
2022 IEEE 40th VLSI Test Symposium (VTS)

Balanced Dual-Mask Protection Scheme for GIFT Cipher Against Power Attacks

Abdullah Aljuffri, Cezar Reinbrecht, Said Hamdioui, Mottaqiallah Taouil
Delft University of Technology – EEMCS Faculty
Delft, The Netherlands
{a.a.aljuffri,c.r.wedigreinbrecht,s.hamdioui,m.taouil}@tudelft.nl

Johanna Sepúlveda
Airbus Defense and Space
Munich, Germany
johanna.sepulveda@airbus.com

***Abstract*—Currently NIST is working towards the standardization of lightweight cryptography (LWC). Although the cryptanalytic strength of LWC is currently under deep scrutiny, the LWC implementation security has not been yet widely explored. GIFT block cipher is the main building block of many of the LWC NIST candidates and therefore has the potential to be part of the next lightweight crypto-standard. Hence it is important to understand its implementation vulnerabilities such as side-channel attacks (SCAs). Although SCAs have been evaluated for hardware implementations, no analysis or countermeasures have been proposed yet for software implementations. This work evaluates GIFT 128-bit software implementations (protected and unprotected) against power-based SCAs. Our protected implementation is based on a new lightweight countermeasure consisting of two balanced and masked SBoxes. Our results show that GIFT's SBox (or SubCell function) is vulnerable against profiled and non-profiled attacks when unprotected or protected implementations based on existing balancing or masking techniques are used. On the other hand, our proposed countermeasure that smartly combines balancing and masking offers full protection with negligible overhead.**

***Index Terms*—Lightweight cipher, Side channel analysis, GIFT, Deep Learning, Countermeasure**

I. Introduction

A recent study made by the Dutch software company Irdeto showed that cyberattacks targeting the Internet of Things (IoT) devices could cost the UK's economy alone 1 billion pounds annually [1]. When it comes to IoT devices' security, lightweight ciphers play an essential role. The U.S. National Institute of Standards and Technology (NIST) started a process for selecting the next lightweight cipher (LWC) standard [2]. GIFT, a lightweight block cipher devised by Banik et al. [3], is currently used by seven of LWC NIST candidates as the underlying cryptographic block of their security solution (e.g., SUNDAE-GIFT [4], HYENA [5], ESTATE [6], GIFT-COFB [7], etc.). Therefore, the probability that GIFT makes it into the LWC standard is high. Although GIFT developers studied the security details of the algorithm's mathematical part well, the resistance against implementation vulnerabilities such as side channel has not received enough attention. Side channels are unintended physical leakage sources such as power, heat, etc. that can be exploited to retrieve secret information. In the past two decades, many different side channel attacks (SCAs) have been performed. For example, power [8] and electromagnetic [9] based side channel attacks. SCAs have been proven to be a powerful tool. The various SCAs techniques were successful in attacking different cryptographic algorithms, including many protected implementations [10, 11]. In the lightweight domain, there is only limited research that focuses on side channel attacks, especially for GIFT cipher.

978-1-6654-1060-1/22/$31.00 ©2022 IEEE

To the best of our knowledge, there are only three articles published in the literature that address SCAs against the GIFT block cipher. The authors in [12] and in [13] presented a correlation power analysis on GIFT 64-bit hardware implementation. Both papers show that by targeting the register values of selected intermediate functions (e.g., SubCell and/or AddRoundKey) within a round of the algorithm process, the secret key can be recovered. However, their methodology is strongly dependent on the hardware implementation, which makes it for example not applicable to software-based implementations. With respect to software, the authors in [14] presented a cache memory attack to recover the key of a GIFT-128. Although successful, the authors showed that such cache attacks can be avoided when the implementation does not use tables or when bigger cache lines are used. Overall, we observe that implementation based security evaluations are very limited, and until now, no secure software implementations have been proposed.

This paper analyzes vulnerabilities of GIFT software implementations. Our evaluations consist of both non-profiled (i.e., CPA) and profiled attacks (i.e., deep learning attacks). We analyze two of the most popular existing countermeasures techniques (i.e., balancing and masking) and propose a new variant by combining them, referred to as balanced dual-mask. The contributions of this paper can be summarized as follows:

- Proposal of balanced dual-mask countermeasure based on masking and balancing.
- Evaluation of naive GIFT software implementation against a non-profiled SCA (namely CPA) and a profiled SCA (namely deep learning power attack) for the unprotected and protected implementation.
- Comparison with existing countermeasure techniques.

The paper is organized as follows. Section II provides a background on GIFT cipher and power attacks. Section III describes the proposed countermeasure. Section IV explains the attack methodology. Section V describes the experiments and results. Finally, Section VI concludes this paper.

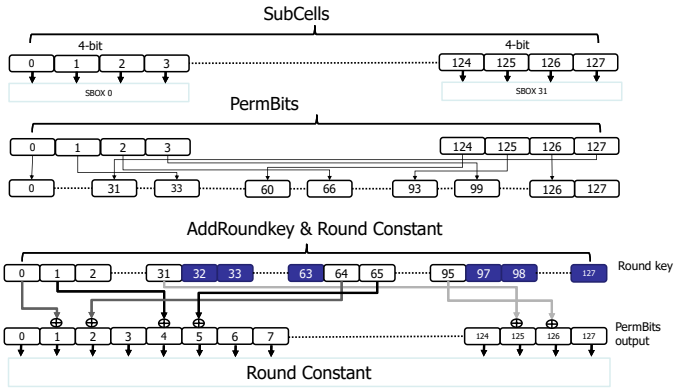


Fig. 1: Round Operations of GIFT-128 Cipher

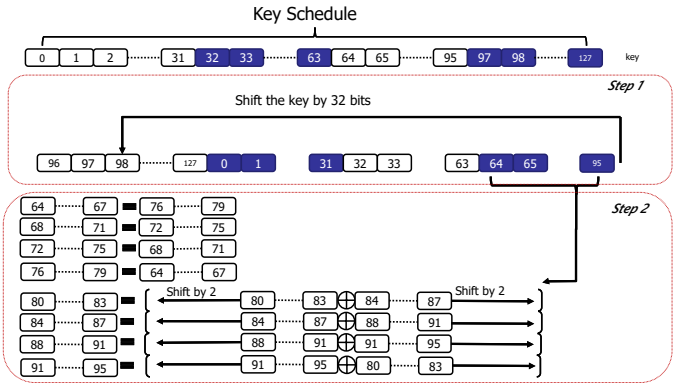


Fig. 2: Round Key Update Process

II. Background

This section provides the necessary background for this paper. Section II-A explains how the GIFT cipher works and Section II-B how side channel attacks can be performed.

A. GIFT Cipher

GIFT is a lightweight block cipher designed by Banik et al [3] as an improved to PRESENT. For performance reasons, GIFT substitution blocks are smaller than PRESENT and use less number of rounds. This makes the GIFT design very compact with a high throughput. There are two versions of GIFT, namely GIFT-64 and GIFT-128. The GIFT-64 uses 28 rounds with a 64-bit block size, while the GIFT-128 40 rounds with a 128-bit block size. The key size is the same in both versions (i.e., 128 bits). As shown in Figure 1, each round of the GIFT cipher consists of four functions [3]: *SubCells*, *PermuBits*, *AddRoundKey*, and Round Constant. Each function will be described briefly next.

SubCells: This function processes the 128-bit round input based on 4-bit data segments. Each 4-bit segment is replaced using a substitution box (SBox). The inputs and outputs of the SBox have a non-linear relation.

PermBits: This function processes its input state at bit-level. The SubCells' outputs are shuffled based on a fixed reordering scheme.

AddRoundKey: This function processes its input state in a 4-bit segment based fashion. Only the middle two bits of each segment are XORed with specific bits of the key as illustrated in Figure 1. Note that in each round only 64 bits of the key are used. For round 0 these are bits 0-31 and 64-95 as indicated in the figure. A key scheduler is used to update the round key.

Round Constant: After the XOR operation with the key, a selected number of bits (i.e., 3,7,11, 15,19, 23, and 127) are XORed with the round constant.

Key Schedule After each round, the key is updated based on two steps as illustrated in Figure 2. In the first step, the key is circularly rotated by 32 bits to the right. In the second step, the last eight segments (i.e., bits 64-95) are updated as follows (see also the bottom part of the figure): (1) the first four segments are reversed, i.e., bits 64-67 are moved to bits 76-79, bits 68-71 to bits 72-75, bits 72-75 to 68-71, and bits 76-79 to 64-67; (2) the last four segments (i.e. bits 80-95) are updated by circular shifting a segment to the left by two and XORing it with the next segment that is circularly shifted by two to the right (see Figure 5c). Note that the last segment (i.e., bits 91 to 95) uses the segment with bits 80-83 as its neighbour.

B. Side Channel Attacks (SCAs)

There are various ways of implementing side channel attacks, which can be classified into two groups, namely non-profiled and profiled attacks. Non-profiled attacks are attacks that do not require prior access to the device and/or similar alternatives. These attacks are carried directly to the target device by analyzing the side channel traces and identifying distinguishable patterns (e.g., power amplitude or execution time). There are many examples of such attacks. The most popular ones in the power domain are DPA [8] and CPA [15]. Since these attacks do not know the real power behavior of the target device, they rely on power models. The most common models use the hamming weight or hamming distance of the result of a target operation. This target operation must have a relation with the secret key. Consequently, the attacker creates a hypothetical power estimation for each possible key/sub-key and correlates each with the collected power traces.

In contrast, profiled attacks create a profiled template that can be used later to attack the target device. Profile side channel attacks consist of two phases namely **profiling** and **extraction** phase [16]. In the **profiling** phase, the attacker focuses on a similar or identical device to collect traces. Thereafter, the attacker chooses an intermediate point of attack (e.g., *SubCells* or *AddRoundKey* operations in GIFT cipher). Next, the attacker records power traces and controls all inputs, outputs, and used keys. With such information, a profile template can be built. The template can be a statistical model like multivariate distribution [15], or more recently, a trained neural network [16]. Neural networks, especially deep neural networks, have been gaining popularity due to their high efficiency even against complex countermeasures [11]. After creating the template, the attacker can start with the extraction phase. In the **extraction** phase, the attacker runs the target device, collects power traces, and identifies the target

operation. Thereafter, the attacker uses the template to guess the key.

III. Balanced Dual-Mask Countermeasure

This section discusses the proposed countermeasure approach. First, we motivate the rationale behind it and then discuss its design and implementation.

A. Motivation

When it comes to securing crypto algorithms from power attacks, most countermeasures revolve around one of two techniques: 1) randomizing the power behavior or 2) balancing the power behavior for every key and plaintext/ciphertext pair. In our study, we considered one popular approach from each technique. For the power randomization approach, we use masking. Masking [17] introduces multiple randomized shares called the mask. For the power-balance approach, we apply the method in [18] where the output of the sensitive function that leaks the most (i.e., SBox) always results in the same number of ones. Hence, the leakage model will always be the same. Unfortunately, neither method was successful in securing the SubCell function as we will see in the up coming sections. Therefore, a new more robust countermeasure is needed.

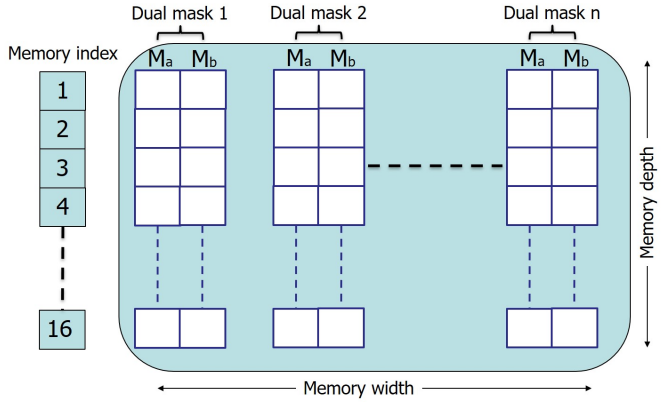


Fig. 3: Balanced Dual Masks Scheme

B. Design and Implementation

Both power-balancing and masking countermeasures failed to protect the SubCells function (see Section V-B). To overcome their limitations, we propose a balanced dual-masks scheme. In this technique, as illustrated in Figure 3, we apply two masks to each SubCells index; both masks together contain 4-bits of actual data and 4-bits of dummy data. This dummy data can reside (partly) in either of the masks. On top of that, instead of using a single set of dual masks we can integrate n different sets. Therefore, in the figure, M_a and M_b of dual mask 1 are not equal to M_a and M_b of dual masks 2. During run-time, only one of the outputs related to the n sets will be used and the dummy bits will be filtered out.

IV. Attack Methodology

This section describes the performed attack methodology. First, it explains the threat model and thereafter the attacks.

A. Threat Model

Our threat model consists of the following assumptions:

- The attacker has the ability to measure the side channel source (i.e., power consumption) from the target device during the execution of GIFT.
- The attacker can observe the plaintext of the encryptions and their associated power traces.
- The attacker has the ability to procure a similar or an identical device to the target to perform profiled attacks.

B. Non-profiled Attack

The concept behind GIFT cipher is similar to most block ciphers such as DES [19], AES [19], PRESENT [20], etc. where the plaintext is encrypted using several confusion and diffusion blocks such as substitution box (which is called *SubCells* in GIFT), add round key and permutation operation. Similar to DES cipher, the add round key operation occurs only at the end of the round which makes it unpractical to target any first round operation as a target intermediate function (with the exception of the add round key itself). Therefore, the first part of the sub-key used by GIFT in the first round (key-bits from 31 to 0 for GIFT-64 and key-bits from 63 to 0 for GIFT-128), can only be attacked through the following intermediate functions:

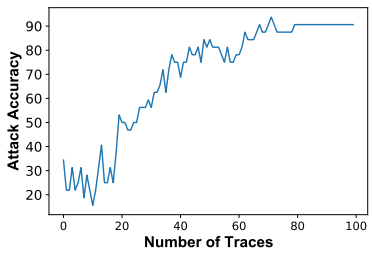
- 1st Round *AddRoundKey*: In this case, the attacker calculates the inputs of the *AddRoundKey* operation based on the used plaintext. Hence, the hypothetical hamming weight (HW) of the XOR result can be estimated for all possible key-bits.
- 2nd Round *SubCells*: In this case, the attacker attacks the output of the SBox operations of the second round. The attacker first needs to calculate the possible inputs at the *SubCells* operation of the second round. As a result, each guessed key-bit (considered in the *AddRoundKey* of the first round) will result in an input to the SBox. Thereafter, the HW or HD of the output of the SBox can be used to estimate the power consumption.
- 2nd Round *PermBits*: We use the outputs of the estimated *SubCells* SBox operation to calculate the outputs of the permutation operation. Thereafter, the HW or HD of this result can be used as target operation.

Next, the attacker can apply a correlation process of the targeted operation with the collected power traces. Pearson correlation is commonly used in CPA for this task [21]. Finally, after retrieving the sub-keys used in the first round, the same methodology can be repeated to target the sub-keys of the following rounds.

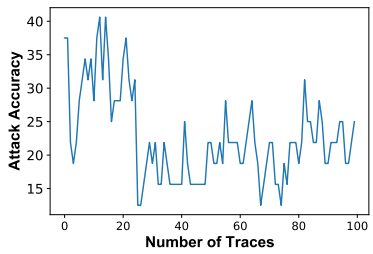
C. Profiled Attack

The intermediate functions discussed for non-profiled attacks can be also used for profiled attacks. Next, we describe the profiling and extraction phase in more detail.

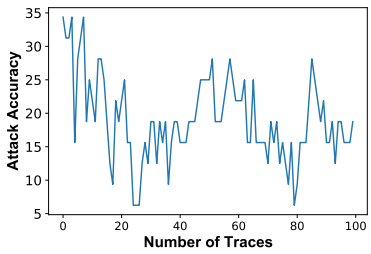
Profiling Phase: In this phase we prepare the neural network model to attack the GIFT cipher algorithm. It consists of the following steps (S):



(a) CPA on SubCells



(b) CPA on PermBits



(c) CPA on AddRoundKey

Fig. 4: Accuracy Analysis of Non-profiled Attacks

S1: Select a target intermediate function. In our case we tried all the three intermediate functions: the *AddRoundKey* of the first round, and *PermBits* and *SubCells* of the second round, respectively. For brevity, the next steps are described only for the *SubCells* intermediate function. However, the same steps can be applied to the *PermBits* and *AddRoundKey*.

S2: Use the outputs of the SubCells operation in the second and third rounds as labels during the training of the profile model. Since the key is known in this phase, it is easy to compute the labels. In this step we need to create labels for all the different *SubCells* functions (i.e., every two bits of the key). Since the key is 128-bit we need 64 sets of labels. Note that during this profile phase both the plaintext and key values are randomly selected. As a leakage model we considered the HW and HD.

S3: Record power traces during the execution of the targeted intermediate function. The traces are divided into two datasets, one for each round.

S4: Construct a neural network and determine its structure, i.e., the number of input and output neurons, the number of layers, activation function, loss function, etc. The number of input neurons is determined by the number of samples in the recorded power trace. The number of output neurons is determined by the total number of labels. The structural parameters of our neural network can be seen in Table I.

S5: Train the neural network. In this step we train the neural network based on the labels generated from *Step 2* and use the corresponding recorded traces from *Step 3* as input. As the labels generated from *Step 2* are different for each sub-key, the neural network is trained 64 times each time targeting a different sub-key.

Extraction Phase: In this phase, we use the trained neural network to extract the key from the target device. This phase consists of two parts. First, we recover the key bits used in the first round by attacking the *SubCells* function of the second round. Next, we use the recovered bits to attack the key bits used in the second round by attacking the *SubCells* function of the third round. To predict the key value, we use Algorithm 1. First, the output of the permutation of the first round is calculated. Then, for each sub-key, we loop through the traces of the targeted round and predict the output of the SBox using the trained neural network. The probability results of all traces are accumulated for all key possibilities. The actual key is most likely the one with the highest occurrence probability. For the *SubCells* intermediate function, we have tried different leakage functions denoted by leakage function lkf() on line 10. Note that the algorithm is used to retrieve the keys of both rounds.

TABLE I: Deep Learning Specifications

Layer (type)	**Output Shape**	**Param #**
input_25 (InputLayer)	(None, 3000, 1)	0
conv1d_73 (Conv1D)	(None, 1000, 64)	256
max_pooling1d_73 (MaxPooling)	(None, 333, 64)	0
activation_73 (Activation)	(None, 333, 64)	0
conv1d_74 (Conv1D)	(None, 111, 128)	24704
max_pooling1d_74 (MaxPooling)	(None, 37, 128)	0
activation_74 (Activation)	(None, 37, 128)	0
conv1d_75 (Conv1D)	(None, 13, 256)	98560
max_pooling1d_75 (MaxPooling)	(None, 4, 256)	0
activation_75 (Activation)	(None, 4, 256)	0
flatten_25 (Flatten)	(None, 1024)	0
dense_25 (Dense)	(None,16)	16400
Total params: 139,920	Trainable params: 139,920	

V. Experiment Results

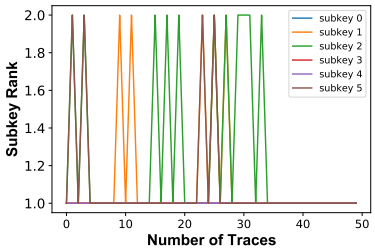
This section first describes the experimental setup. Subsequently, it presents the security and performance analysis.

A. Experiment Setup

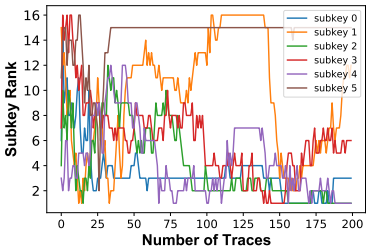
To validate the proposed attack scheme, the publicly available open-source software implementation of the GIFT 128-bit cipher [22] was used. The GIFT 128-bit program is written in C code. The power traces have been collected by running the implementation on the Chipwhisperer board from NewAE Technology Inc [23]. Chipwhisperer is a development board that comes with an Atmel XMEGA micro-controller that is used as a target device. It has been used in many attacks such as ECC [24]. The power traces were captured by an Analogue-to-Digital Converter with a sample rate of 105 MS/s. Both the target chip and the measurements setup are connected to the computer using a USB interface, to execute the program and transfer the recorded traces. The proposed attack was implemented in Python using the Keras library. This is an open-source software library that can be used to create, train, and run artificial neural networks.

B. Security Analysis of Naive Implementation

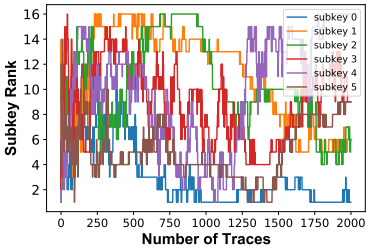
To validate the GIFT cipher against power attacks, three functions were selected as targets: 1) the *SubCells* function



(a) Subcells of Second Round

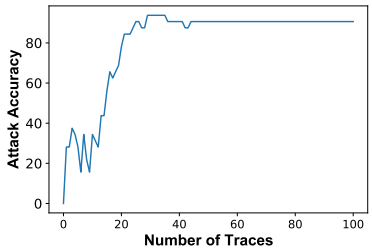


(b) PermBits

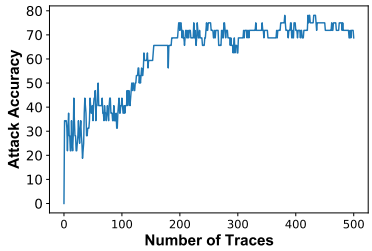


(c) AddRoundKey

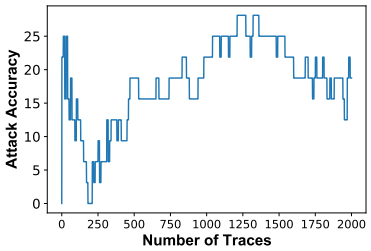
Fig. 5: Rank Analysis of Profiled Attacks



(a) Power balancing



(b) Power Randomizing



(c) Balanced Dual Masks

Fig. 6: Countermeasures Analysis using Non-profiled Attacks

Algorithm 1 Extract Key bits

procedure KET_EXTRACT($Traces_{set}, pt_{array}$)
  pt = output of Permutation XORed with constant
  $P_k[0,15]$ = key probability
  $predict$ = is the trained model the sub-key
  for each sub-key **do**
    $P_k[0,15] = 0$
    for each trace in trace-set **do**
      $X_{0,15} = predict(trace)$
      for k=0 to 15 **do**
        $y = lkf(SBOX[pt[sub-key] \bigoplus k])$
        $P_k[k] = P_k[k] + X[y]$
      end for
    end for
    $guess_{sub-key} = max(P_k)$
  end for
end procedure

during second and third rounds; 2) the *PermBits* of the second round; and 3) the *AddRoundKey* in first round. They have been used in both non-profiled (i.e., CPA) and profiled attacks(i.e., deep learning power attack). Their results are discussed next.

*1) **Non-profiled Attacks**:* The attacks were evaluated initially with a single trace and iteratively reevaluated by adding each time a single trace until 100 traces have been used. Figure 4 show the accuracy analysis of CPA attacks for the three attacked functions, respectively. A higher accuracy value means that more sub-keys were correctly guessed; e.g., a 100% accuracy means that all sub-keys were correctly guessed. We observe that CPA attack was successful in recovering the majority of sub-keys values for SubCells while targeting other functions was unsuccessful. The more traces are used, the closer the guessed sub-key is from the correct sub-key. The results indicate that the software implementation of GIFT is attackable using non-profiled techniques. Although a few sub-keys have not been attacked successfully, it could be possible to attack the entire key when more traces are added.

*2) **Profiled Attacks**:* The performed profile attacks are described in Section II. The results for *SubCells* in terms of rank analysis is shown in Figure 5a; the lower the rank, the better the guessed sub-key.. All sub-keys reached a rank of 1 (i.e., they were fully recovered) using only 50 traces. Similarly, the results for *PermuBits* and *AddRoundKey* are shown in Figures 5b and 5c. The key ranking results show random behaviour which means that it is difficult to retrieve the correct sub-key value.

C. Security Analysis of Proposed Implementation

First, we separately evaluated the balancing and masking countermeasures presented in [18] and [17], respectively. Using CPA, we were able to achieve a high accuracy with only a few traces for both countermeasures as shown in Figures 6a and 6b. Next, we evaluated the security analysis using one single dual mask only (i.e., only Dual mask 1) to validate the minimum security level of the proposed approach. Note that only the SubCells function was targeted as both AddRoundKey and PermBits functions were unattackable in the naive implementation. In the non-profiled attack (i.e., CPA) the approach was secure as the maximum accuracy the attack could reach was 25% as shown in Figure 6c. However, using the profiled attack (i.e., deep learning) the attack was not fully secure as some of the sub-keys were recovered as can be seen

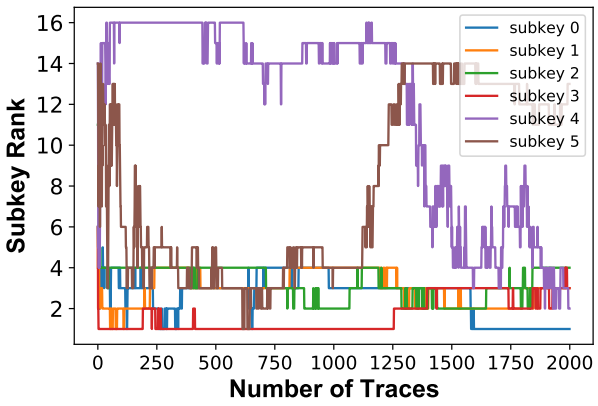


Fig. 7: Single Dual Masks Analysis

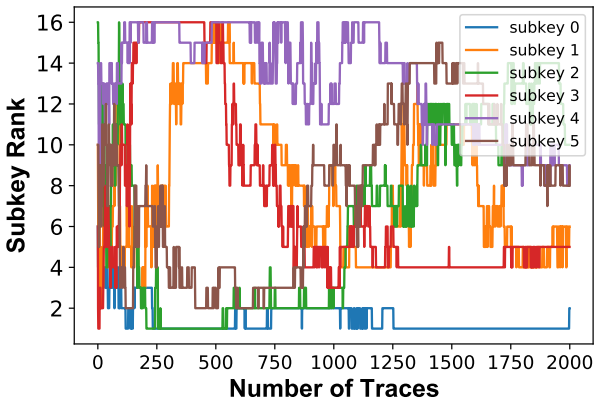


Fig. 8: Double Dual Masks Analysis

in Figure 7. To solve this issue, we increased the number of dual masks to two, i.e., we used different masks for profiling and attack phases where n=2 in both cases. The results shows that the sub-keys are secure as shown in Figure 8. In order for an attacker to create a successful template, he needs to consider all 16^n combinations for the different masks. This becomes quickly infeasible for n=8.

D. Area overhead and Performance Analysis

Our proposed technique only increases the width of the SBox table. Therefore, there is no area overhead unless the number of dual masks exceeds the word size (i.e., 4 dual masks in 32-bit wide memories and 8 dual masks in 64-bit wide memories). The performance overhead is measured by the additional number of instructions added to the baseline execution. Since these extra instructions are only required to multiplex the dual masks and select one of them. Hence, the increase in execution time is negligible.

VI. Conclusion

In this paper, we analyzed the security of a software implementation of GIFT 128. Our proposed countermeasure, consisting of a masking and balancing technique, is secure against power SCA while having a low overhead. This increases the chances of having GIFT integrated in coming lightweight security standards.

Acknowledgments. This work was labelled by the EUREKA cluster PENTA and funded by Dutch authorities under grant agreement PENTA-2018e-17004-SunRISE.

References

[1] Infosecurity Magazine, "IoT Attacks Cost UK Firms Over £1bn." Available at: https://www.infosecurity-magazine.com/news/iot-attacks-cost-uk-firms-over-1bn-1/. Accessed: 2020-02-24.
[2] National Institute of Standards and Technology, "Lightweight cryptography." Available at:https://csrc.nist.gov/projects/lightweight-cryptography, 2020.
[3] S. Banik *et al.*, "Gift: A small present - towards reaching the limit of lightweight encryption," in *CHES*, 2017.
[4] S. Banik *et al.*, "SUNDAE-GIFT." Available at: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/SUNDAE-GIFT-spec-round2.pdf. Accessed: 2020-02-24.
[5] A. Chakraborti *et al.*, "HyENA." Available at: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/hyena-spec-round2.pdf. Accessed: 2020-02-24.
[6] A. Chakraborti *et al.*, "ESTATE." Available at: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/estate-spec-round2.pdf. Accessed: 2020-02-24.
[7] S. Banik *et al.*, "GIFT-COFB ." Available at: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/gift-cofb-spec-round2.pdf. Accessed: 2020-02-24.
[8] P. C. Kocher *et al.*, "Differential Power Analysis," in *CRYPTO*, 1999.
[9] J.-J. Quisquater and S. David, *Electromagnetic Attack*. 2005.
[10] A. Bogdanov, "Improved side-channel collision attacks on aes," in *Selected Areas in Cryptography*, 2007.
[11] R. Gilmore *et al.*, "Neural network based attack on a masked implementation of aes," in *HOST*, 2015.
[12] V. Satheesh and D. Shanmugam, "Secure realization of lightweight block cipher: A case study using gift," in *Security, Privacy, and Applied Cryptography Engineering*, 2018.
[13] J. Zhang *et al.*, "Power analysis attack on a lightweight block cipher gift," in *Proceedings of the 9th International Conference on Computer Engineering and Networks*, 2021.
[14] C. R. W. Reinbrecht *et al.*, "Grinch: A cache attack against gift lightweight cipher," in *DATE*, 2021.
[15] S. Chari *et al.*, "Template attacks," in *Cryptographic Hardware and Embedded Systems*, 2002.
[16] H. Maghrebi *et al.*, "Breaking cryptographic implementations using deep learning techniques," in *IACR Cryptology ePrint Archive*, 2016.
[17] H. S. Kim and S. Hong, "New type of collision attack on first-order masked aess," in *ETRI Journal*, 2016.
[18] Y.-S. Won *et al.*, "On the security of balanced encoding countermeasures," in *Revised Selected Papers of the 14th International Conference on Smart Card Research and Advanced Applications - Volume 9514*, 2015.
[19] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. 2014.
[20] A. Bogdanov *et al.*, "Present: An ultra-lightweight block cipher," in *CHES*, 2007.
[21] E. Brier *et al.*, "Correlation power analysis with a leakage model," in *CHES*, 2004.
[22] T. Peyrin, "Gift block cipher." https://github.com/giftcipher/gift, 2019.
[23] NewAE Technology Inc, "Chipwhisperer-Lite two part board." Available at: http://store.newae.com/chipwhisperer-lite-cw1173-two-part-version/. Accessed: 2020-01-31.
[24] S. Paramasivam Karthikeyan and H. El-Razouk, "Horizontal correlation analysis of elliptic curve diffie hellman," in *ICICT*, 2020.