Differential Power Analysis on ZUC Algorithm

2012

TANG Ming[1,2], CHENG PingPan[2] ,QIU ZhenLong[2]

[1]*State Key Lab. of AIS & TC, Ministry of Education, Wuhan University, Wuhan 430072, China;*

[2]*School of Computers, Wuhan University, Wuhan 430072, China;*

Abstract. Stream cipher ZUC plays a crucial role in the next generation of mobile communication as it has already been included by the 3GPP LTE-Advanced, which is a candidate standard for the 4G network. Through a long-time evaluation program, ZUC algorithm is thought to be robust enough to resist many existing cryptanalyses, but not for DPA, one of the most powerful threat of SCAs(Side Channel Analysis).Up to the present, almost all the work on DPA is for block ciphers, such as DES and AES, a very few work has been done on stream ciphers, such as ZUC algorithm, for particular reasons that would be illustrated in the later section. In this paper, we generally study the security of unprotected ZUC hardware implementation against DPA. Our theoretical analysis and experimental results show that ZUC algorithm is potentially vulnerable to this kind of attack. Furthermore, kinds of common countermeasures are discussed when we try to apply them to ZUC hardware implementations, both the security and tradeoffs are considered. The experiments are given in the last section to verify our conclusions, which would undoubtedly provide some guidance to the corresponding designers.

Key words: SCA, DPA, Stream Cipher, ZUC algorithm

1 Introduction

In the field of telecommunications, the world is stepping into 4th Generation (4G for short) standard. During the last few years, the 3rd Generation Partnership Project (3GPP) has submitted Long Term Evaluation Advanced(LTE-Advanced) [1], which is the enhancement of the LTEstandard, as a candidate for the 4G network. Particularly for the radio interface, theLTE-Advanced standard needs a standardized encryption

and integrity algorithmset to guarantee its security.The latest algorithm set isthe 128-EEA3 (EEA is short for the Encryption Algorithm)[2] and the 128-EIA3(EIA is short for the Integrity Algorithm) [2].Both of the two algorithmsare based on a proposedstream cipher algorithm ZUC [3] with an internal state of 496 bits initialized from a 128-bit secret key and a 128-bit initialization vector (*IV* for short).

To ensure the security of the proposed ZUC algorithm, it was agreed that a robust, three–phase evaluation program would be needed, including the evaluations by an ETSI SAGE task force, the evaluation by two funded teams of academic experts, and a public evaluation phase.Having experienced such a long-time evaluation program, up to the present, ZUC algorithm is thought to be robust enough to resist against many existing cryptanalyses such asWeak Key Attacks, Guess-and-Determine Attacks, Algebraic Attacks, Timing Attacks, etc.[4]. However, whether the algorithm is vulnerable to Power Analysis (PA) [5], one of the most effective SCAs（SCA is short for side channel analysis）[5,6,7] to recover secret keys in numerous cryptosystems, is not known yet.One point should be noted is that in terms of Power Analysis, we usually distinguish between Simple Power Analysis (SPA) [5] and the other more powerful analysis methods, such as Differential Power Analysis (DPA for short) [5], Correlation Power Analysis (CPA for short) [8], .etc. However, throughout this paper, we study more particularly about DPA, since it does not depend on the leakage function and has a better practicability than other PAs.

In [5] P. Kocher et al. firstproposed DPA, which performs a statistical analysisof the power consumptionfora smart card and finally retrieve the secret key. The attack exploits the fact that the power consumption of a cryptographic device is dependent on the intermediate values operated during the encryption, and it has been shown to be a practical threat to the security of cryptosystems.

In this paper, we generally studied the security of the unprotected ZUC algorithm againstDPA. Furthermore, kinds of effective countermeasuresweirediscussed to protect the hardware implementations of ZUCalgorithm from DPA. Experimental results are also made to verify our conclusions.

1.1 Related Work

1) DifferentialPower Analysis. During the last decade, sidechannel analysishas been shown to be a major threat towards the cryptographic implementations, regardless of software and hardware.Among SCAs, DPA is one of the most commontechniques for its high efficiency, good applicabilityand convenient performance compared with other methods. However, almost all the work on DPA is for block ciphers (e.g. DESand AES) and public key algorithms (e.g. RSA), there is a very few researches on stream ciphers. The difficulty for DPA against stream ciphers lies in the fact thatthe key steam generated by a stream cipher is independent from the plaintext and the ciphertext,so the prerequisites of DPA attacks are not present, because we cannot use the known values (i.e. the plaintext or the ciphertext) and the guessed values (i.e. a part of the secret key needed to be recovered) to determine the intermediate result during the encryptions.

2) DPA on Stream Ciphers. However,Joseph Lanoet al. [9] point out that, in real applications, for the purpose to keep synchronization between the sender and the receiver, stream cipher usually needs to be frequently resynchronized. In this situation, the initial state of the stream cipher is frequently changed with different *IV*s while the secret key(the secret key here is actually our attacking target in DPA, it was used to initial the internal state of ZUC algorithm combined with the *IV*) is identical. So, we can just treat the initial value as the plaintext in DPA while regard the generated key stream as the ciphertext, and use the *IV* and the guessed key together to determine the intermediate value of stream ciphers. So, DPA is still possible to attack stream ciphers.In this paper, actually, our DPA strategy is based on this thought.

From another perspective, Sanjay Burman et al. proposed a PA based SCA method to attack LFSR-based stream ciphers, which can precisely recover the state of an n-bit LFSR through collecting the power of the LFSR in each clock period over consecutive periods linear in n [10].And in [11], with a similar strategy, the authors use DPA successfully break down the Grain-80 stream cipher and recover the secret key.All the attacks are based on the fact that there is a high correlation between the

cipher's power and the switching activity of the state bits of FSR.

3) DPA countermeasures. As the serious threat brought by DPA, lots of efforts have already been dedicated towards the development of the corresponding countermeasures on block ciphers [12, 13, 14, 15, 16, 17], all these countermeasures are supposed to be able to be applied on stream ciphers as well [11].One subject of research has been on algorithmic countermeasures that try to randomize all the intermediate results during the processes. Thiskind of countermeasure isusually called masking [12,13, 14], which can be applied either at algorithm level or at gate level. However in practice, masking strategy is usually very specific and requires additional cost of reduced performance, so we often combine this kind of countermeasure and othercountermeasures for better performance. Hardware countermeasures try to make the power consumption of a device independent of the data being processed by the device. Typical examples are noise generatorsand the insertion of random delays [15, 16],logic styles with data-independent power consumption [17, 18], etc. Among which, in the real applications, the insertion of random delays is the most popular countermeasure for its simplicity while logic styles with data-independent power consumption are the most efficient ones for its best performance.

There are also some countermeasures specifically targeting at protecting stream ciphers. To our best knowledge, in order to counteract the potential threat of power analysis methods in [10], the author in this paper first proposed an architecture level countermeasure specifically targeting at LFSR-based stream ciphers. And in COSADE 2012,Shohreh Sharif Mansouri et al. [11] proposed another architectural countermeasure against power analysis for FSR-based stream ciphers, which can resist against DPA, MIA and several more complex power attacks.

1.2 Our Contribution

This is the first paper trying to perform DPA on ZUC algorithm. There havealready been a few publications towards the security of the ZUC algorithm from traditional mathematic analyses. Especially, the official design and evaluation report [4]gives the

evaluation of the ZUC algorithm on the resistance against several cryptanalytic attacks, weak key attacks, guess-and-determine attacks, algebraic attacks, timing attacks, etc. ChunfangZhou et al. extend the differentialpropertiesof the initialization stage of ZUC algorithmfrom 20 roundsto four more rounds and shows that ZUC can still resist against chosen-*IV* attacks [19]. However, whether ZUC can resist against DPA is rather unknown.In this paper, this topic is generally studied.Ourfinal analysis results show that ZUC algorithm is to some extent vulnerable to DPA.

Since DPA is a potential serious threat of ZUC algorithm,it is necessaryfor designers to add theeffective countermeasures to the ZUC implementation to guarantee its security in real applications.However, all the countermeasures proposed in [10, 11] specifically targeting at stream ciphers cannot resist our DPA strategy proposed in this paper, because our attacking point is not the state of FSR or LFSR but the outputs of S-box in ZUC stream cipher. Thus, these kinds of countermeasures which decrease the correlation between the power of FSR and the total power of cipher cannot work on our attack. In this paper, we generally apply several of the countermeasures which are initially designed for block ciphers but still working on stream ciphers to the implementations of ZUC algorithm, and study the ability of protected ZUC algorithm against DPA.We generally take the tradeoffs and security into consideration, which wouldundoubtedly provide some guidance to the designers.

1.3 Organization of This Paper

The remainder of this paper is organized as follows. Section 2 presents the introduction ofthe hardware implementation of ZUC algorithm and the general steps of DPA [5]. Our detailed DPA strategy on ZUC algorithm is described in Section 3.In Section 4,we discussthree kinds of countermeasures trying to protect ZUC implementations from DPA. Our experiments resultsare in Section 5 and finally, we draw our conclusions in Section 6.

2 Preliminaries

2.1 The Definition of ZUC Algorithm

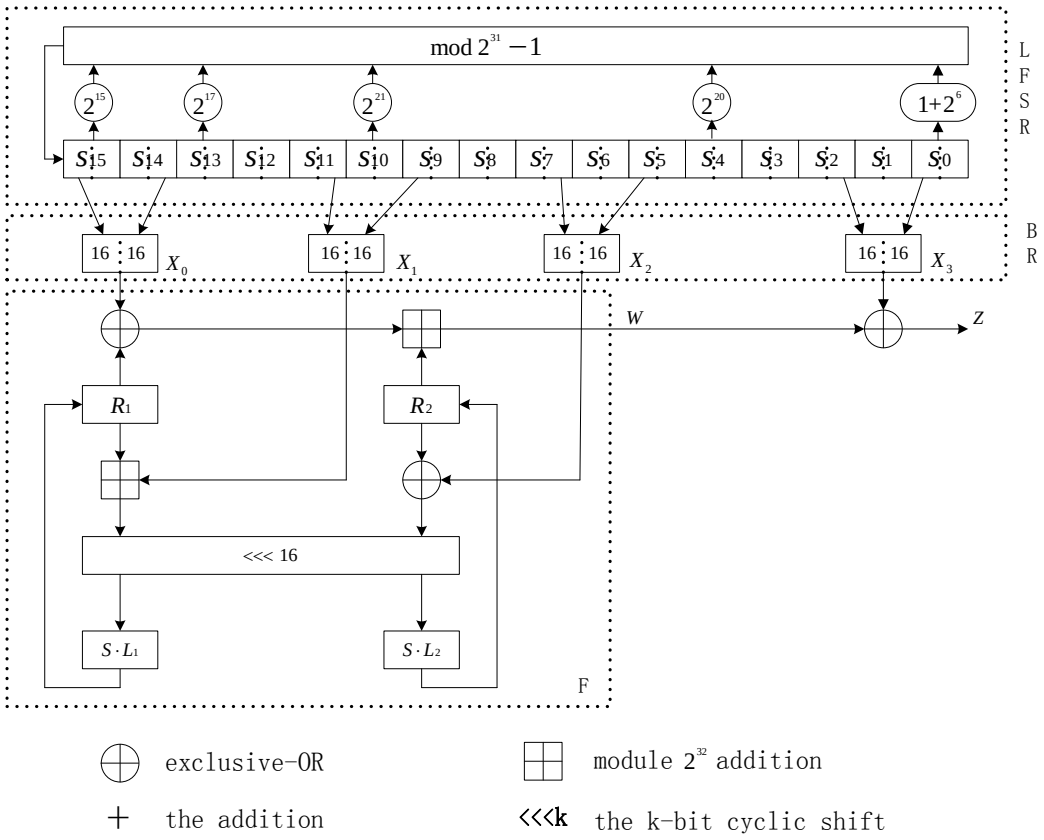


Fig 2-1the structure of ZUC Algorithm

In this section, we briefly introduce the latest ZUC algorithm, for detailed phase please refer to [2]. The new stream cipher ZUC is a world–orientedstream cipher [ ], taking a 128-bit secret key and a 128-bit *IV*as input, and outputs a keystream of 32-bit words, which is used to encrypt or decrypt the data.According to the official ZUC specification [3], ZUC is composed of three logical layers, among whichthe top layer is a linear feedback shift register (LFSR for short) of 16 cells, the middle layer is the bit-reorganization (BR for short) operation, and the bottom layer is a nonlinear function F. The structure of ZUC is illustrated in **Fig 2-1**.

2.1.1 The Linear Feedback Shift Register (LFSR)

The linear feedback shift register (LFSR) has 16 of 31-bit cells($S_0, S_1, \ldots, S_{15}$). Each

register cell $S_i (0 \le i \le 15)$ is restricted to take values from the following set: $\{1, 2, 3, \ldots, 2^{31}-1\}$. The LFSR has two modes of operations: the initialization mode and working mode. The initialization mode works as Algorithm 1 shown.

Algorithm 1. *LFSRWithInitialisationMode*(u){

1. $v = 2^{15} s_{15} + 2^{17} s_{13} + 2^{21} s_{10} + 2^{20} s_4 + (1+2^8) \bmod (2^{31}-1)$;
2. $s_{16} = (u+v) \bmod (2^{31}-1)$;
3. If $s_{16} = 0$, then set $s_{16} = 2^{31}-1$;
4. $(s_1, s_2, ..., s_{15}, s_{16}) \to (s_0, s_1, \ldots, s_{14}, s_{15})$.

}

In the working mode, the LFSR does not receive any input, and it works as Algorithm 2 shown.

Algorithm 2. *LFSRWithWorkMode*(){

1. $v = 2^{15} s_{15} + 2^{17} s_{13} + 2^{21} s_{10} + 2^{20} s_4 + (1+2^8) \bmod (2^{31}-1)$;
2. If $s_{16} = 0$, then set $s_{16} = 2^{31}-1$;
3. $(s_1, s_2, ..., s_{15}, s_{16}) \to (s_0, s_1, \ldots, s_{14}, s_{15})$.

}

2.1.2 The Bit-Reorganization

The middle layer of ZUC is the bit-reorganization (BR) procedure. It extracts 128 bits from the cells of the LFSR and form four32-bit words, where the first three words will be passed to the next layer, nonlinear function F, and the last word will be involved

in producing the key stream. Assuming that s_2, s_5, s_7, s_9, s_{11}, s_{14}, and s_{15} are eight cells of LFSR as in section 2.1.1. Then the bit-reorganizationforms 4 of 32-bit words X_0, X_1, X_2 and X_3 from the above cells as Algorithm 3.

Algorithm 3. *Bitreorganization*(){

1. $X_0 = s_{15H} \| s_{14L}$;
2. $X_1 = s_{11L} \| s_{9H}$;
3. $X_2 = s_{7L} \| s_{5H}$;
4. $X_3 = s_{2L} \| s_{0H}$.

}

2.1.3 The Nonlinear Function *F*

There are two 32-bit memory cells, R_1 and R_2, in the nonlinear function F. The input of F is X_0, X_1, X_2, which are the first three words ofoutput of the BR procedure, and it outputs a 32-bit word W. The detailed processof the nonlinear function F is described in Algorithm 4, where S is a 32×32 S-box.

Algorithm 4. $F(X_0, X_1, X_2)$**{**

1. $W = (X_0 \oplus X_1 \boxplus) R_{21}$;
2. $W_1 = R_1 \boxplus X_1$;
3. $W_2 = R_2 \oplus X_2$;
4. $R_1 = S(L_1(W_{1L} \| W_{2H}))$;
5. $R_2 = S(L_2(W_{2L} \| W_{1H}))$.

}

Both L_1 and L_2 are linear transforms from 32-bit words, and are defined as follows:

$$L_1(X) = X \oplus (X <<<_{32} 2) \oplus (X <<<_{32} 10) \oplus (X <<<_{32} 18) \oplus (X <<<_{32} 24)$$
$$L_2(X) = X \oplus (X <<<_{32} 8) \oplus (X <<<_{32} 14) \oplus (X <<<_{32} 22) \oplus (X <<<_{32} 30).$$

2.1.5 The Key Loading

The key loading procedure will expand the initial key and the initial vector into 16 31-bit integers as the initial state of the LFSR. Let the 128-bit initial key k and the 128-bit initial vector iv be $k = k_0 \| k_1 \| k_2 \| ... \| k_{15}$ and $iv = iv_0 \| iv_1 \| iv_2 \| ... \| iv_{15}$ separately, where k_i and iv_i, $0 \le i \le 15$, are all bytes. Then k_i and iv_i are loaded into the cells S_i as $s_i = k_i \| d_i \| iv_i$, where d_i is a known constant.

2.1.6 The Execution of ZUC

The execution of ZUC is composed of two stages: the initialization stage and working stage. During the initialization stage, the cipher algorithm runs the following operations 32 times to finish the initialization:

1. $Bitreorganization()$;

2. $W = F(X_0, X_1, X_2)$;

3. $LFSRWithInitialisationMode(w >> 1)$.

After the initialization stage, the algorithm moves into the working stage. At the beginning of this stage, the algorithm executes the following operations once, and discards the output W of nonlinear function F:

1. $Bitreorganization()$;

2. $F(X_0, X_1, X_2)$;

3. $LFSRWithInitialisationMode()$.

Then the algorithm goes into the stage of producing keystream, i.e., for eachiteration, the following operations are executed once, and a 32-bit word Z is produced as an output:

1. *Bitreorganization*();

2. $Z = F(X_0, X_1, X_2) \oplus X_3$;

3. *LFSRWithInitialisationMode*().

2.2 Implementation in Hardware

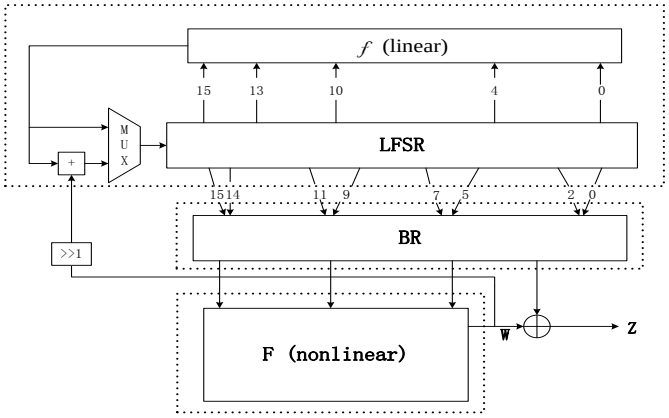


Fig 2-2implementation of ZUC Algorithm in Hardware

Thestructure of the hardware implementation of ZUC algorithm is illustrated in Fig 2-2. It is mainly made up of the following parts:

- a LFSR of 16 cells $s_0, s_1, \ldots, s_{15}$;
- acombinatorial logic block *BR*, implementing the bit-reorganizationoperation;
- a combinatorial logic block *F*, implementing the nonlinear function *F*;
- a combinatorial logic block *f*, implementing $\mathrm{mod}(2^{31}-1)$ addition operation;
- some extra XORs and other logic operations.

2.3 The Differential Power Analysis Attacks

Over the past ten years, there have been some new DPA attacks, but the main idea and principles of most DPA attacks [20] are originated from Kocher [5]. Like

other SCAs, DPAs are composed of two phases: the first is data collection, and the other is statistical analyzing. The following steps provide an example of a DPA processes proposed by Kocher:

1) Attackers choose a key-dependent selection function D. In this case, the selection function would have the form $D(C_i, b, k_s)$;
2) Attackers could observe m encryption operations and capture two kinds of information as following:
   - C_i represents cipher text which is corresponding to one power trace;
   - k samplings are collected and each sampling is related to a certain time point.
3) Statistically analyzing to get k_s:
   - Firstly, to get the value of $T_i[j]$ for a certain power sampling, i is the i th power sampling and j represents the j thsampling point;
   - Secondly, to compute the value of differential power based on function (1), and only C_i and $T_i[j]$ are variable;

$$\Delta D[j] = \frac{\sum_{i=1}^{m} D(C_i, b, k_s) T_i[j]}{\sum_{i=1}^{m} D(C_i, b, k_s)} - \frac{\sum_{i=1}^{m} (1 - D(C_i, b, k_s)) T_i[j]}{\sum_{i=1}^{m} (1 - D(C_i, b, k_s))} \quad (1)$$

   - Eventually, if k_s is incorrect, $\lim_{m \to \infty} \Delta D[j] \approx 0$; adversely if k_s is correct, the computed value of $\Delta D[j]$ will not be zero and show spikes in regions where D is correlated to the values being processed. This conclusion has been proofed by Kocher[5].

While the effects of a single transistor switching would be normally be impossible to identify from direct observations of a device's power consumption, the statistical operations used in DPA are able to reliably identify extraordinarily slight differences in power consumption.

3 DPA attacks on the ZUC Algorithm

As mentioned above, since stream ciphers requirefrequent resynchronizations in many

applications, it is possible for attackers to obtain a huge number of power traces of a stream cipher using different initial values but the same secret key. In this case, we can choose a selection function D combined with the *IV* and the guessed key to determine the intermediate value of the algorithm.

This section presentsour DPA strategyonZUC algorithm.

3.1 Overview

The main stages of the attack are as follows, we generally recover the secret key bytes by bytes:

1) Data collection stage: collect power consumptions;
2) Use DPA against the first round of the initialization stage to recover k_9 and k_5.

   The value of R_1 and R_2 registers and the state of the 16 LFSR cells derived in the first round are used in the second round of the initialization stage, to recover k_{10} and k_6;
3) Use the same strategy in the next few rounds, and determine the correct key information k_{11}, k_7, k_{12}, k_8, and k_{13} separately;
4) Performthe DPA in the sixth round of the initialization stage to make an exhaustive search of k_{15}, k_{14}, k_4, k_0 and recover them;
5) Use the same strategy as step 4) to get the values of k_{11} and k_1 in the next round of the initialization stage;
6) Finally, determine the only left byte of the secret key k_2 in the eighth round.

The attack details are given in the next section and the experiment results are presented in **Section 5**.

3.2 Attack Details

Step 1: Power measurements are performed for 10000 samplings, each encrypted with the same secret key and different*IVs*.

Collecting the power traces covering the first eight rounds of ZUC algorithm, the initialization and key-loading operations of EEA3-128 algorithm should be also covered.

Step 2: Use DPA to recover k_9 and k_5. In the first round of the initialization stage of ZUC algorithm, the inputs of eachS-box are solely related to the correct key information k_9 or k_5. Therefore, for each byte of the secret key,the DPA can perform an 8-bit exhaustive search over the bits, and use one bit of the outputs of the corresponding S-box as the partition bit to recover the correct information.The data flow related with the secret key bitsof k_9 and k_5 in each operationis as follows:

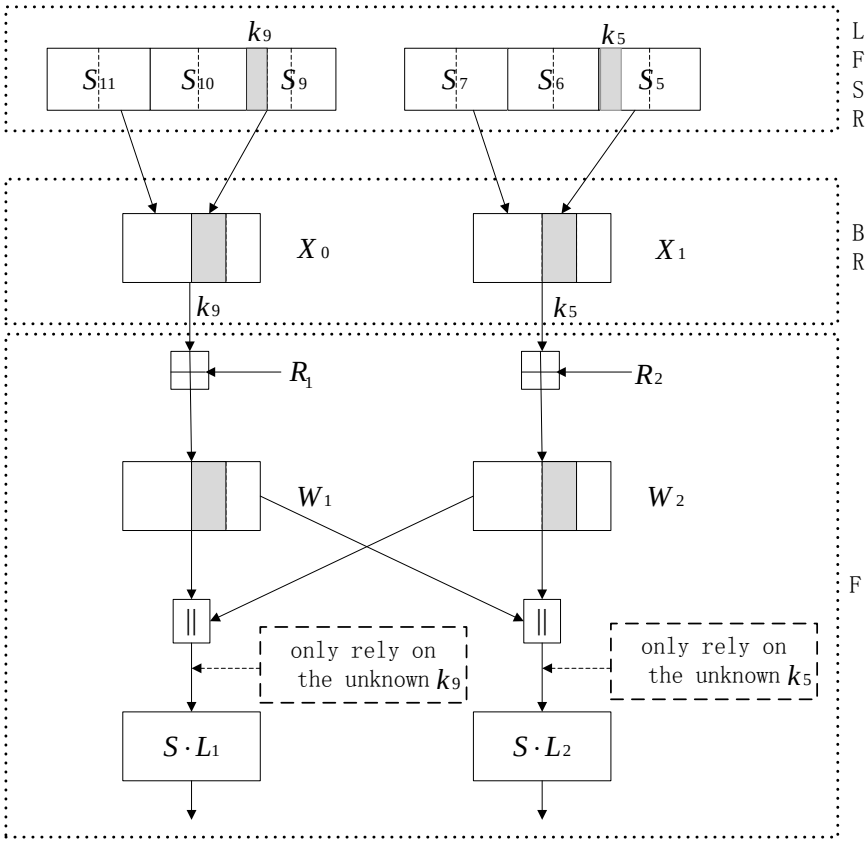


Fig3-1the data flow related to the secret k_9 and k_5 in DPA

- $KeyLoading(IV, d_i, KEY)$.This operation makes each cell s_0, …, s_{14}, s_{15} of the LFSR solely related with the correct information k_0, …, k_{14}, k_{15} respectively, and the highesteight bits of each cell are thecorrect key information;

- *Bitreorganization*() . Afterthe combinations in this step:

$$\begin{cases} X_0[30:23]=k_{15} \\ X_1[15:8]=k_9 \\ X_2[15:8]=k_5 \end{cases},$$

where $X[A:B]$ represents the highest[A:B] bits of X ($A>B$). We do not consider X_3 in our attacks for it does not involve in the nonlinear function *F*(our attacking operation) operation of initialization stage.

- $W=F(X_0,X_1,X_2)$ .The analysis of the nonlinear function *F*displays the most crucial part of the total attacks. X_0 , X_1 , and X_2 derived in the above processes are needed in this step:

  1) **In the first three functionsofthe *F*function**（see**Section 2.1.3**）, since the initial state of R_1 and R_2 are both zero, it is easy to find that the topeight-bit of W is identical with the secretkeyinformation in X_0, while W_1 and W_2 cover the same correctkey information with that in X_1 and X_2, respectively.This relationship can be expressed as:

$$\begin{cases} W_1[15:8]=k_9 \\ W_2[15:8]=k_5 \end{cases};$$

  2) Thus, the inputs of **the fourth function of the *F*function** $W_{1L}\|W_{2H}$ are solely related to k_9 , and the inputs of the fifth function $W_{2L}\|W_{1H}$ are only related to k_5 . The two selection functions in these twice DPAs to recover k_9 and k_5 can be represented in the following forms:

$$\begin{cases} b_1=D_1(IV_i,k_9) \\ b_2=D_2(IV_i,k_5) \end{cases},$$

where b_1 and b_2 are one-bit of the outputs of the corresponding S-boxes; IV_i represents the different and known initial values; k_9 and k_5 arethe

parts of the only unknown secret key;

3) It is easy to acquire the corresponding values of R_1 and R_2, which will be used in the following attacks.

– $LFSRWithInitializationMode(W >> 1)$. The value of W derived in the above step is the input of this operation. After the first two functions of this step, we could get the results as follows:

$$\begin{cases} v = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1+2^8) \bmod (2^{31}-1) \\ s_{16} = (u+v) \bmod (2^{31}-1) \end{cases},$$

where the generated s_{16} covers the correct key information of k_{15}, k_{13}, k_{10}, k_4, and k_0; Then, the next two functions are used to update the 16 cells of LFSR, and the leftmost cell of LFSR is updated with the value of s_{16}.

Note:It should be noted that the S-box Scontains 4 8×8 S-boxes, namely S_0, S_1, S_2, S_3, where $S_1 = S_3$, $S_0 = S_2$. Actually, we can either choose the outputs of the S-box S or the outputs of one amongthe 4 S-boxes as the attacking point for there always exists a S-box whose one-bit output covers the whole correct key information of k_9 or k_5; Moreover, the specialtyof the analysis on the first round lies in the fact that the initial values of R_1 and R_2 are both zeros, thus they would not affect the inputs of the S-box(that is the values of W_1 and W_2). So typically, we further analyze the attacking details on the second round of the initialization stage.

Step 3:Use DPA to recover k_{10} and k_6. Similar with the analysis of the first round, the DPA on the second round can also choose one bit in the outputs of the S-boxes as the partition bit to recover the correct key. But there exists a slight difference for some special operations and the values of R_1 and R_2 are not zero any longer. The data flow related to k_{10} and k_6 in the DPA is just like the **Fig 3-1**.

Like the analysis in **Step 2**, after the bit-reorganization operation, we can get the

results:

$$\begin{cases} X_1[15:8] = k_{10} \\ X_2[15:8] = k_6 \end{cases},$$

In fact, the differences lie in the analysis of nonlinear function *F*:

- After **the middle twofunctions** of $W_1 = R_1 \boxplus X_1$ and $W_2 = R_2 \oplus X_2$, W_1 covers the correct key information of k_{10}. More precisely, if the outgoing carry bit of $R_1 + X_1$ is not zero,then $W_1[16:8]$ covers the information of k_{10}. On the other hand, $W_1[15:8]$ will cover this information; and it is obvious that $W_2[15:8]$ will cover the correct information k_6.
- In**the last two functions**, the result of $W_{1L} \| W_{2H}$ is solely related with the secret k_{10}, we can use DPA to recover it; with the same way, k_6 can be retrieved in the next step. However, it should be noted that in the process to recover k_6, the result of $W_{2L} \| W_{1H}$ maybe rely on both k_6 and k_{10}, for the existence of the carry bitof $R_1 + X_1$. Since k_{10} is retrieved in the former step, this does not have influence on our attacks.

The final operation will update the LFSR, and we use S_{16}' to represent the new S_{16}.

Step 4: Usethe same strategies as in **Step 3**, we are capable to recover k_{11} and k_7 in the third round, k_{12} and k_8 in the fourth round, k_{13} and k_9 (in fact, k_9 has already been retrieved in the first round) in the fifth round. So far, 72 bits out of the 108 bits secret information have already been recovered in the previous processes. (128?)

Step 5: Use DPA to recover k_0, k_4, and k_{15}. In the sixth round of the initialization stage of ZUC algorithm, the attacks become more difficult, for S_{16} generated in the first round of the current stage involves in the operations.

More precisely, S_{16} covers more correctkey informationthan that in the other cells, whichinvolves 32-bit correct key information of k_{15}, k_{13}, k_{10}, k_4, and k_0. Fortunately, k_{13} and k_{10} have already been recovered in the previous analyses, moreover,only k_0, k_4, k_{15} are still unknown. S_{16} makes X_1 cover much more correct key information than that in the bit-reorganization operationof the previous rounds, which is 32 bits (besides the correct information in S_{16}, it also covers the secret k_{14}). So the adversary has to perform the32-bit exhaustive search over the bits, which is still reasonable. ???

Step 6:Use DPA to recover k_{11} and k_1 in the seventh round. The operations in the seventh round of the current stage involves S_{16}' (only covers the correctkey information of k_{11} and k_1) generated in the second round.Like the analysis towards the sixth round, DPA is capable to retrieve the values with the16-bit exhaustive searches.

Step 7: After the DPA on the first seven rounds, there only leaves k_2 uncovered. It is easy to determine the value in the next round, and we do not give the details.

The overall attacks need 10 times of 8-bit exhaustive search, a 32-bit exhaustive search, and a 16-bit exhaustive search. So the time complexity is $(2^8 \times 10 + 2^{32} + 2^{16})$.

It should be noted that, since our attacking point is the outputs of the S-box in the nonlinear function *F*, which can only be implemented by looking up tables and cannot be implemented by logics, our DPA strategy actually do not rely on the specific ZUC hardware implementations, such as the three optimized implementations proposed in [21].

4 The Probably Secure Hardware Implementations of ZUC Algorithm

The DPA strategydescribed in Section 3 is towards on unprotected ZUC

implementations. However, in real applications, countermeasures are usually added to the hardware implementations of the algorithm to guarantee its security. So, studying the effectiveness of different common countermeasures is of great value for it can provide some imperative guidance for the hardware implementations of ZUC algorithm in applications.

In this section, three kinds of popular countermeasures, the insertion of random delays, WDDL, masking respectively, are discussed as different countermeasures in hardware implementations of ZUC algorithm.DPAs results on protected circuits are given in Section 5.

4.1The Insertion of Random Delays

One of the most classic countermeasures against DPA in the real world applications is the insertion of random delays. With the insertion of random delays, instead of executing all the operations sequentially, the CPU interleaves the code's execution with that of dummy instructions so that the corresponding operation cycles do not match because of time shifts[15].

Such countermeasures are effective because all the intermediate results are no longer computed at a fixed instance. It rather occurs at a set of different time instants with probability distribution, thus we cannot use DPA to attack a fixed point. This kind of countermeasure is simple, and very convenient to implement with limited hardware resource increase.

So far, several strategies for this kind of countermeasure have already been proposed [22, 23, 24]. From [23] and [24], it is clear that the complexity of a DPA attack grows quadratically or linearly with the standard deviation of the trace displacement in the attacking point. Among theexisting methods based on this idea, the method proposed in [24]gets the best performance for achieving a relatively higher standard deviation of the trace displacement with reasonable resource. In our corresponding experiment, we implement the strategy for random delaygeneration proposed in [22] for its best performance in hardware implementation.And we

generally study the ability of the protected circuits to resist DPA.

4.2Wave Dynamic Differential Logic

Using logic styles with data-independent power consumption in hardware implementations is a rather common hardware countermeasure. Such technologies have been proposed trying to keep the power of the whole circuits at a constant value to prevent DPA attacks, andwave dynamic differential logic (WDDL for short)first proposed in [25]is a typical logic of this kind of countermeasures.The circuits using WDDL [25, 26, 27] have the following characteristics: whenever an operation is performed in hardware, a complementary operation should be performed on a dummyelement to assure that the total power consumption of the unit remains balanced. The WDDL logics have shown almost the constant power value, thus it canresist DPA fundamentally. While at the same time, this kind of countermeasure will definitely increase hardware resources at least three times[17].

In the experiments, based on WDDL principles, the original ZUC circuits are disintegrated. More precisely, the design uses input signals to replace NOT gates and logic inverters, and replaces the complicated logics (i.e. NOR gates and NAND gates) with simple logics (i.e. AND gates and OR gates). And then, built in complementary principle, the AND gates are added with OR gates while the OR gates are added with AND gates. After these operations, the overall circuits would be capable to resist DPA for the power of each operation keeps balanced. Experiment results are shown in Section 5 to confirm the analysis.

4.3Masking

The masking technique [12, 13, 14] is the most widely used countermeasure against power analysis. In a masking design, for every execution of the algorithm a new mask is randomly generated and applied to the input data and to the secret key. All internal computations are masked from then on and the final resultsare unmasked after the last round. In this case, the attacker cannot extract any correlation between the secret key

in running and the actual power curves. If the masking scheme is implemented properly, it has shown that this kind of countermeasure can render DPA rather complicated or even impossible [13, 14].

However, the existing masking strategies almost allconcentrate on block ciphers and each cipher usually needs a specific masking scheme.We find it hard to propose a proper masking strategy for the steam cipher ZUC.There are several distinct reasons: Firstly,in a masking scheme, the nonlinear operations are generally the most resource-consuming parts. Particularly for the implementation of ZUC algorithm, thenonlinear operations - thetwo 8×8 LUTs (LUT is short for look up table) are just too large to implement in a rather limited hardware resource;More importantly,the worst cases for a specific masking scheme are the algorithms that repeatedly require switching between different masking types, such as additive masking to multiplicative masking or arithmetic masking. Taking the AES masking schemes for example, Mehdi-Laurent Akkar et al.proposed a masking implementation in [12] and Johannes Blomeret al. introduced a probably secure masking strategy in [28], both of which require switching between different kinds of masking types. They were believed to be secure for a period. However, it was shown in later research that both of these two strategies are not secure enough to resist DPA for the existence of zero-value attacks caused by the switching [13, 29].Unfortunately, ZUC algorithm is this kind of algorithm for it involves several different kinds of operations which needs frequent switching.

So, we do not apply this kind of countermeasure to the implementations of ZUC algorithm.

5 Experiments

In this section, two kinds of experiments are conducted to verify our conclusions:the attacks on the original ZUC hardware implementation, and the attacks on the protected hardware implementations of ZUC algorithm, respectively. The test environments as follows:

- CPU: Pentium(R)Dual-Core 2.60GHz
- Memory: 2.0GB RAM
- OS: Windows 7
- Simulation database: TSMC 0.18 μm database
- EDA compiler: Synopsys Design Compiler 2008.9
- Wave simulator: Modelsim SE 6.5b
- Power simulator: Synopsys PrimePower 2004.12
- Program Language: Visual C#

The flow of the whole experimentis illustrated as**Fig 5-1**:

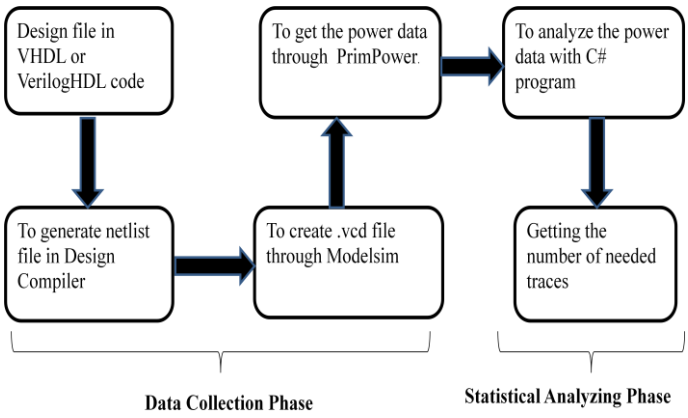


Fig 5-1the flow of our experiment

Asimplified descriptionof the process of our experiment is as follows:

Step1: In data collection phase, power measurements are performed for *N*(typically 10000)samplings,each encryptedwith a randomly selected initial value*IV* and the same secret key.Then, use EDA simulation tools (i.e. Modersim, PrimePower) to collect the power data;

Step2: Our attacking point is set at the first bit of outputs of the leftmost S-boxin the first round of ZUC algorithm in the initialization stage.

Note: Since the DPAfor each round are virtually the same and just differs at the target bits of the secret key, it makes sensefor ustoonly attack the first round of the initialization stage for simplicity. Here, in all the experiments, we only try to recover

k_9 in the first roundas we described in **Section 3.**

Step3: Use power analyzerto collect certain information such as initial values IVsand power $T_i[j]$ on line;

Step4: To put theguess k_s into D function and compute value of b .

Step5: To compute the differential power traces based on the power model proposed by Kocher [5],and judge whether the guessed k_s is correct by comparing all the spikes for each hypothesis k_s .

The details for each experiment are presented in the following parts.

5.1 Attacks on Original ZUC Hardware Implementation

In the experiment,the correctkey is set “C15CB7421B980FD5438D2972F86BE0E4”, and the guess key is randomly selected.In our hardware implementation, the clock cycle is 20ns, and we collect the power each 1 ns.We use about 10 clock periods to implement the first two rounds of the initialization stage of ZUC algorithm, so we collect about 240 power points in each sampling. The power tracescollected inPrimePowerare illustrated as**Fig 5-2**, and the analysis results in thepower analyzer are shown in**Fig 5-3**.

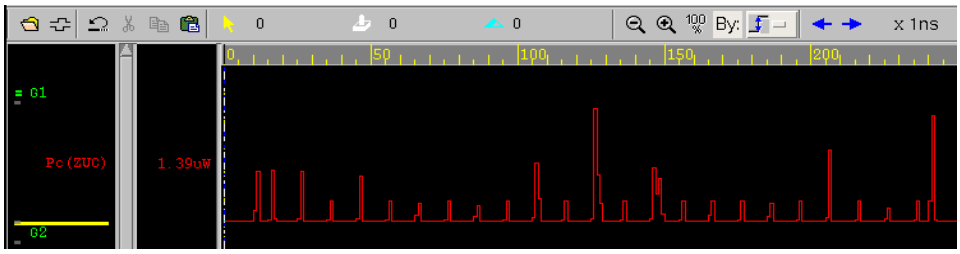
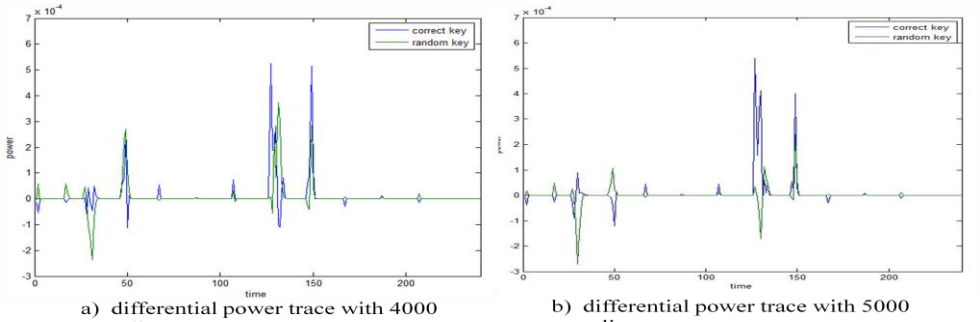


Fig 5-2 power trace collected in PrimePower in about 12 clock cycles



a) differential power trace with 4000 samplings

b) differential power trace with 5000 samplings

Fig 5-3thedifferential power traces to recover k_9

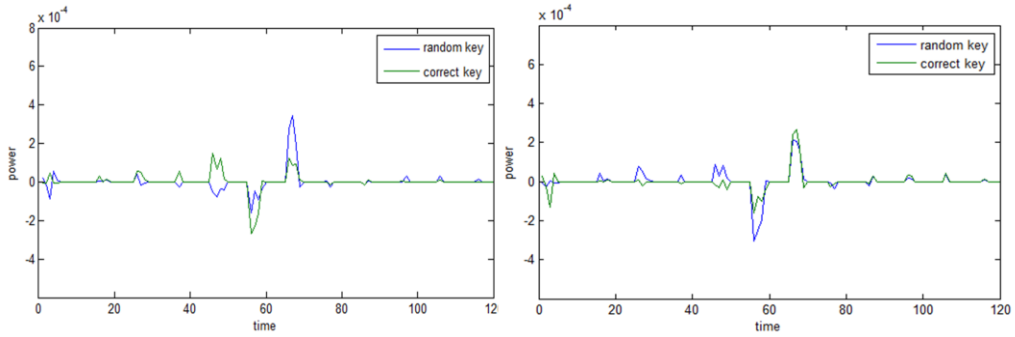
As shown in**Fig 5-3 b)**, the spike generated by the correct key is clearly visible when the sampling number is 5000, while in **Fig 5-3a),** the spike is not so obvious for the sampling number is not enough.Furthermore, we can see that the spike shows at about time point of 130ns, which is exactly the point when the targeted S-box is operated. Thus, we can successfully recover the secret k_9 . Just at the same way, we are capable to recover the other bits of the secret key.

5.2 Attacks on Protected ZUC Hardware Implementations

Experiment 1: In the ZUC hardware Implementation of the first experiment, we implement the strategy for random delaygeneration proposed in [ ].

The core idea of this strategy is to use a configurable switch matrix to control the position of registers in between functional blocks of an algorithm. Since each register causes a delay of one clock cycle, these randomly poisoning registers shift the code's executions and desynchronize attackers' observations. And the number of possible configurations depends on the number m of registers and n of functional blocks. In our implementation, there are five functional blocks, and we use two registers: one is between the bit-reorganization operation and the F function, and the other register is ahead of the nonlinear operation S-box. The clock cycle is set 15ns, and we collect power point each 1ns. As there are at most two registers in each round, it would take 120ns (i.e. 8 clock cycles, 6 among which implements the original ZUC circuits, and 2 among which implements the delays at most) to implement the first round of the initialization stage at most. In the statistical analyzing phase,the partition bit is set at

the first bit of the outputs of S_1. The analysis results in thepower analyzer are shown in**Fig 5-4.**



a) differential power trace with 5000 samplings

b) differential power trace with 10000 samplings

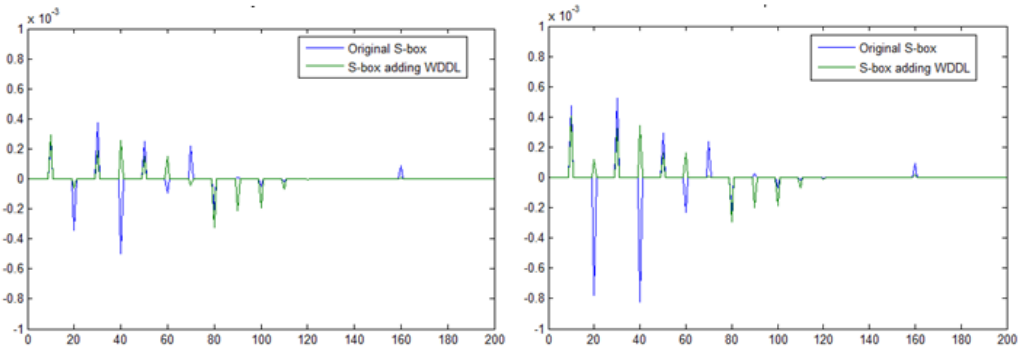
Fig 5-4differential power traces to recover k_9

As we can see in both **Fig 5-4** a) and b), there is no obvious spikes in all the differential traces, and the spikes generated by the random key traces even cover the spikes generated by the correct key traces. When the sampling number increases from 5000 (which is actually the number of samplings needed to break the original ZUC hardware implementation in our design) to 10000, the trend is still not clear. The DPA attacks failed when we use this kind of countermeasure.

Experiment 2: In this experiment, WDDL is used to protect ZUC hardware implementation from DPA.

WDDL is a rather resource-consuming countermeasure.It has been proved that the nonlinear operations in an algorithm are the most crucial parts to resist DPA [], for simplicity, it makes sense for us to only apply this kind of countermeasure to one of the S-boxes of ZUC algorithm, and we choose S_1. As mentioned above, in our implementation of WDDL, the original netlist of S_1 generated in Design Compiler is disintegrated into simple circuit units,and each circuit unit is added with a complementary unit to keep the total power balanced. The clock period is set 20ns, and we collect the power each 0.1ms in the data collection phase.In the statistical analyzing phase, we set the first bit of the outputs of S_1 as the partition bit and use

our power analyzer to attack the design. The analysis results in thepower analyzer are shown in**Fig 5-4.**



a) differential power traces using 5000 samplings

a) differential power traces using 10000 samplings

Fig 5-4 differential power traces to attack the S-boxes

As is shown in **Fig 5-4 a)**, when the sampling number is 5000, the original S-box generates an obvious spike while the S-box added WDDL does not. It means DPA is successfully implemented on the original S-box, but the attack on the S-box added WDDL failed. When we increase the sampling number to 10000, as is shown in **Fig 5-4 b)**,thespike generated by the original S-box is more clearly visible, while DPA on the S-box added WDDL is still unsuccessful for the corresponding trace does not produce a visible spike yet. The results confirmed our analysis.

6 Conclusions

This is the first paper trying to perform DPA on the new proposed standardized algorithm ZUC.Almost all the present DPA research is towards on block ciphers, however, based on the frequentresynchronization of stream ciphers in real applications, we showed that the stream cipher ZUC is still potentially vulnerable to DPA. Kinds of effective countermeasures are also discussed trying to render the attack more complicated,and we mounted stimulated DPA attacks to prove the effectiveness of these countermeasures in our experiments. Our work would undoubtedly provide some guidance to the hardware implementation designers of ZUC algorithm in real applications.

Our DPA strategy is built in the standard DPA proposed by Kocher, and the attack

complexity is in a reasonable level. However, we believe that some other techniques might be used to reduce the DPA complexity to an even lower level. Additionally, as hardware design techniques develop, more work could be done on the countermeasures of ZUC algorithm to against DPA, especially masking. If properly implemented, masking is undoubtedly one of the most efficient countermeasures against DPA.

References

[1] 3rd Generation Partnership Project. Long Term Evaluation Release 10 and beyond (LTE-Advanced). Proposed to ITU at 3GPP TSG RAN Meeting, Spain (2009)

[2] Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 1: 128-EEA3 and 128-EIA3 Specification. ETSI/SAGE Specification, Version: 1.5 (January 4, 2011)

[3] Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification. ETSI/SAGE Specification, Version: 1.5 (January 4, 2011)

[4] Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 4: Design and Evaluation Report. ETSI/SAGE Specification, Version: 2.0 (September 9, 2011)

[5] P.C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In Advances in Cryptology – CRYPTO 1999, volume 1666 of Lecture Notes in Computer Science (LNCS), pages 388–397. Springer, 1999

[6] D. Agrawal, B. Archambeault, J.R. Rao, and P.Rohatgi. The EM Side-channel(s). In Cryptographic Hardware and Embedded Systems – CHES 2002, Lecture Notes in Computer Science (LNCS). Springer, 2002

[7] P.C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Related Attacks. In Advances in Cryptology – CRYPTO 1996, volume 1109 of Lecture Notes in Computer Science(LNCS), pages 104–113. Springer, 1996

[8] E. Brier, C. Clavier, F. Olivier. Correlation Power Analysis with a Leakage Model, In Cryptographic Hardware and Embedded Systems – CHES 2004, volume 3156 of

Lecture Notes in Computer Science (LNCS), pages 16–29. Springer, 2004

[9] J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede. Power Analysis of Synchronous StreamCiphers with Resynchronization Mechanism. In The State of the Art of StreamCiphers–SASC 2004, Workshop Record, pages 327–333

[10] S. Burman, D. Mukhopadhyay, and K. Veezhinathan. LFSR Based Stream Ciphers Are Vulnerable to Power Attacks. In INDOCRYPT 2007, volume 4859 of Lecture Notes in Computer Science(LNCS), pages 384–392. Springer, 2007

[11] S. S. Mansouri and E. Dubrova.An Architectural Countermeasure against Power Analysis Attacks for FSR-Based Stream Ciphers. In COSADE 2012, volume 7275 of Lecture Notes in Computer Science (LNCS), pages 54–68. Springer, 2012

[12] M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Cryptographic Hardware and Embedded Systems – CHES 2001, volume 2162 of Lecture Notes in Computer Science (LNCS), pages 309–318. Springer, 2001

[13] J. D. Goli ́c and C. Tymen. Multiplicative Masking and Power Analysis of AES. In Cryptographic Hardware and Embedded Systems – CHES 2002, volume 2535 of Lecture Notes in Computer Science (LNCS), pages 198–212. Springer, 2003

[14] L. Genelle, E. Prouff and M. Quisquater.Thwarting Higher-Order Side Channel Analysis with Additive and Multiplicative Maskings. In Cryptographic Hardware and Embedded Systems – CHES 2011, volume 6917 of Lecture Notes in Computer Science (LNCS), pages 240–255. Springer, 2011

[15] C. Clavier, J.-S.Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Cryptographic Hardware and Embedded Systems – CHES2000, volume 1965 of Lecture Notes in Computer Science (LNCS), pages 252–263. Springer, 2000

[16] S. Mangard. Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In Cryptology – CTRSA2004, The Cryptographers' Track at the RSA Conference 2004, volume 2964 of Lecture Notes inComputer Science (LNCS), pages 222–235. Springer, 2004

[17] K. Tiri and I. Verbauwhede.Securing Encryption Algorithms against DPA at the

Logic Level: Next Generation Smart Card Technology. In Cryptographic Hardware and EmbeddedSystems – CHES 2003, volume 2779 of Lecture Notes in Computer Science (LNCS), pages 137–151. Springer, 2003

[18] K. Tiri and I. Verbauwhede.A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In 2004 Design, Automation andTest in Europe Conference and Exposition (DATE 2004), 16-20 February 2004,Paris, France, pages 246–251. IEEE Computer Society, 2004

[19] Chunfang Zhou, XiutaoFeng, and Dongdai Lin.The Initialization Stage Analysis of ZUC v1.5*'**. In CANS 2011, volume 7092 of Lecture Notes in Computer Science (LNCS), pages 40–53. Springer, 2011

[20] T. S. Messerges, E. A. Dabbish and R. H. Sloan.Investigations of Power Analysis Attacks on Smart Cards.Usenix Workshop on Smartcard Technology, USA, pages151-162,1999

[21] Lei Wang, Jiwu Jing, Zongbin Liu, Lingchen Zhang, and Wuqiong Pan. Evaluating Optimized Implementations of Stream Cipher ZUC Algorithm on FPGA*. In ICICS 2011, volume 7043 of Lecture Notes in Computer Science (LNCS), pages 202–215. Springer, 2011

[22] N. Mentens, B. Gierlichs, and I. Verbauwhede.Power and Fault Analysis Resistance in Hardware through Dynamic Reconfiguration.In Cryptographic Hardware and EmbeddedSystems – CHES 2008, volume 5154 ofLecture Notes in Computer Science (LNCS), pages 346–362. Springer, 2008

[23]M.Tunstall, O.Benoit.Efficient Use of Random Delays in Embedded Software. In WISTP 2007, volume 4462of Lecture Notes in Computer Science (LNCS), pages 27–38. Springer, 2007

[24] J.-S. Coron and I. Kizhvatov.An Efficient Method for Random Delay Generation in Embedded Software. In Cryptographic Hardware and EmbeddedSystems – CHES 2009, volume 5747 ofLecture Notes in Computer Science (LNCS), pages 156–170. Springer, 2009

[25] K. Tiri, and I. Verbauwhede.A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: Design, Automation and Test in

Europe, DATE’04. (2004) pages 246–251

[26] K. Tiri, and I. Verbauwhede.A VLSI Design Flow for Secure Side-Channel Attack Resistant ICs. In: Design, Automation and Test in Europe, DATE’05. Vol. 3. (2005) pages 58–63

[27] K. Tiri, and I. Verbauwhede.Place and Route for Secure Standard Cell Design. In: 6th International Conference on Smart Card Research and Advanced Applications,CARDIS’04. (2004) pages 143–158

[28] J.Bl ̈omer, J. G.Merchan, and V.Krummel. Provably SecureMasking of AES. Cryptology ePrint Archive (http://eprint.iacr.org/), Report2004/101, 2004

[29] E. Oswald, S.Mangard, N.Pramstaller,and V.Rijmen. A Side-Channel Analysis Resistant Descriptionof the AES S-Box*.In FSE 2005, volume 3557 ofLecture Notes in Computer Science (LNCS), pages 413–423. Springer, 2005.