RESEARCH

Open Access

Improved conditional differential attacks on lightweight hash family QUARK

Xiaojuan Lu[1,2], Bohan Li[1,2], Meicheng Liu[1,2*] and Dongdai Lin[1,2*]

Abstract
Nonlinear feedback shift register (NFSR) is one of the most important cryptographic primitives in lightweight cryptography. At ASIACRYPT 2010, Knellwolf et al. proposed conditional differential attack to perform a cryptanalysis on NFSR-based cryptosystems. The main idea of conditional differential attack is to restrain the propagation of the difference and obtain a detectable bias of the difference of the output bit. QUARK is a lightweight hash function family which is designed by Aumasson et al. at CHES 2010. Then the extended version of QUARK was published in *Journal of Cryptology* 2013. In this paper, we propose an improved conditional differential attack on QUARK. One improvement is that we propose a method to select the input difference. We could obtain a set of good input differences by this method. Another improvement is that we propose an automatic condition imposing algorithm to deal with the complicated conditions efficiently and easily. It is shown that with the improved conditional differential attack on QUARK, we can detect the bias of output difference at a higher round of QUARK. Compared to the current literature, we find a distinguisher of U-QUARK/D-QUARK/S-QUARK/C-QUARK up to 157/171/292/460 rounds with increasing 2/5/33/8 rounds respectively. We have performed the attacks on each instance of QUARK on a 3.30 GHz Intel Core i5 CPU, and all these attacks take practical complexities which have been fully verified by our experiments. As far as we know, all of these results have been the best thus far.

Keywords: Conditional differential attack, Lightweight hash function, Automatic condition imposing algorithm, NFSR, QUARK

Introduction

Nonlinear feedback shift register (NFSR) is one of the most important cryptographic primitives in the modern cryptosystem. As the main building block of a cryptosystem, the NFSR contains an *internal state* and a non-linear feedback function. The internal state is updated at each clock cycle by left/right shifting one *stage* and computing a new one by the non-linear feedback function. The NFSR can provide high level of security and privacy while achieving a good performance in hardware implementation. Therefore, many cryptographic algorithms are designed based on NFSR, such as the stream ciphers Trivium (De Canniere and Preneel 2008) and Grain v1 (Hell et al. 2008) in the eSTREAM portfolio 2 (hardware oriented), and the lightweight block cipher KATAN/KTANTAN (De Canniere et al. 2009). Especially, each instance of the lightweight hash function family QUARK (Aumasson et al. 2010; Aumasson et al. 2013; Aumasson et al. 2012) contains two NFSRs. In light of this, the NFSR-based cryptosystems have been a hot research topic for many cryptanalysts.

In cryptanalysis, the NFSR-based cryptosystems are often analyzed as Boolean functions in which the input, output, and internal state bits are viewed as variables. In light of this, different types of attacks are proposed, such as cube attack (Dinur and Shamir 2009) and conditional differential attack (CDA) (Knellwolf et al. (2010)). The framework of CDA on NFSR-based cryptosystems was first proposed in Knellwolf et al. (2010) at ASIACRYPT

*Correspondence: liumeicheng@iie.ac.cn; ddlin@iie.ac.cn
[1] Institute of Information Engineering, Chinese Academy of Sciences, No. 65 Xingshikou Road, Haidian District, Beijing 100093, People's Republic of China
Full list of author information is available at the end of the article

2010. The CDA is based on the differential attack by introducing the technique of condition imposing at the first few rounds of the cryptosystem. When applying the CDA, the attacker first chooses a proper *input difference*. Then, the attacker traces the *difference characteristic* round by round and imposes some conditions at proper rounds so that he/she can control the difference propagation as many rounds as possible. Finally, a detectable *bias* of the *output difference* can be obtained. This may result in a distinguishing attack or a key recovery attack depending on the types of the conditions.

In the traditional CDA, the attacker usually chose the single-bit input difference because it is believed to propagate slower than the multi-bit input difference. Lately, in order to trace the difference characteristic of the cryptosystem as many rounds as possible, two approaches are applied: *forward* approach and *backward* approach (Knellwolf 2012; Li and Guan 2018). The forward approach aims to control the propagation of the single-bit input difference e_i as far as possible, and backward approach aims to find an input difference (with arbitrary weight) which can lead to such single-bit difference e_i after Q_{e_i} rounds. By concatenating the difference characteristic obtained from the forward and backward approach, the distinguisher of more rounds may be found. In Zhang et al. (2015), Yang et al. (2018), both their forward approaches are to find the maximum round of each single-bit input difference, where a *deterministic difference* still exists in the internal state. Then, the top k ($k \geq 1$) are chosen to be the candidates. In this paper, we apply a different forward approach to find the maximum round of of each single-bit input difference. Compared to the methods in Zhang et al. (2015), Yang et al. (2018), we consider a *probabilistic difference characteristic* rather than a deterministic one. In order to estimate the bias of output difference, some theoretical frameworks are proposed. For example, Banik (2014) proposed a theoretical framework to compute the biases of the output difference of Grain v1, and Liu et al. (2021) proposed a new theory of estimation of the differential-linear bias at CRYPTO 2021. Since a truncated differential attack can be considered as an extreme case of the differential-linear attack (Blondeau et al. 2017), the theory of estimation of the differential-linear bias can also be applied to the CDA (Liu et al. 2021). In this paper, we apply the technique proposed in Liu et al. (2021) to estimate the bias of output difference in the CDA.

Another difficulty in the attack is the condition imposing. On the one hand, both the number of the imposed conditions and the complexity of the difference expression increase rapidly with each increasing round which may exceed the attacker's computing capability. On the other hand, according to the security model, all the conditions should be replaced by the *initial variables* (input bits) which may lead to contradictions (e.g. an initial variable needs to be set as 0 and 1 at the same time). In Ma et al. (2017), the authors chose to nullify the common initial vector (IV) bits and directly nullify the internal variables. Their strategy is to make the number of Type 1 conditions as large as possible and limit the number of all conditions as small as possible at the same time. In Li and Guan (2018), the authors factored the target condition expression to the form $f = f_1 \cdot f_2 + f_3$ with a manually elaborative analysis. Their strategy is to make the number of Type 1 conditions as small as possible and limit the number of all conditions as small as possible at the same time, which is different from the one in Ma et al. (2017). However, compared to the previous manual analyses, we propose the first algorithm to automatically impose the conditions. By applying such automatic algorithm, we can handle the much more complex condition expressions and improve the efficiency of the condition imposing process in CDA.

The lightweight hash function family QUARK was first proposed by Aumasson et al. (2010) at CHES 2010. Then its extended version was published in *Journal of Cryptology* 2013 (Aumasson et al. 2013), in which the specification of QUARK was slightly modified, more accurately, the digest length has been increased. Another instance C-QUARK (Aumasson et al. 2012) is proposed in 2012, which is designed for the scenario of authenticated encryption scheme with associated data (AEAD). The CDA was first applied on the QUARK instances by the designers of QUARK (Aumasson et al. 2013; Aumasson et al. 2012) providing the primary security evaluation against CDA. In 2015, Zhang et al. (2015) improved the CDA on QUARK by a forward computation algorithm to obtain the distinguishers for the U-QUARK, D-QUARK, S-QUARK and C-QUARK up to 153, 159, 248 and 445 rounds respectively. Then, Yang et al. (2018) proposed a technique called symbolic-like computation to improve the corresponding distinguishers up to 155, 166, 259 and 452 respectively. In order to analyze the security of QUARK further, we apply our improved CDA on each instance of QUARK. Furthermore, we perform the corresponding experiments to verify the validity and efficiency of the attacks. From the results of the practical experiments, we make improvements on the cryptanalysis of all the instances of QUARK, which are compared with the cryptanalytic results in current literature.

Our contributions

All in all, our contributions can be described as follows.

- We propose a method to obtain a set of good input differences. The procedure of the method contains the forward search part and backward search part. In the forward search part, we aim to obtain a set of good single-bit differences denoted as S. Such a single-bit difference will be chosen as a potential condidate if it follows the rule that this single-bit difference will result in a large bias of output difference at a high enough round. Futhermore, the bias is estimated theoretically by the technique proposed by Liu et al. (2021) at CRYPTO 2021. In terms of the rule used in the forward search part, we consider a probabilistic difference characteristic (e.g. $|\varepsilon| > 2^{-12}$) rather than a deterministic one (i.e. $|\varepsilon| = \frac{1}{2}$). For a single-bit difference e_i in S, we take advantage of the backward approach to derive an input difference (with arbitrary weight) which will result in the single-bit difference e_i after Q_{e_i} round. In the backward search part, we aim to expand rounds as many as possible, that is we make Q_{e_i} as large as possble by imposing some proper conditions. By such method, a better difference characteristic can be obtained which is expected to lead to a detectable bias of the output difference at a higher round.
- We propose an automatic condition imposing algorithm to handle the complex conditions. In the previous research work, conditions imposing is a time consuming and complicated process. In order to explain the efficiency and advancement of the algorithm, we revisit the work of D-QUARK in Yang et al. (2018) with our proposed automatic condition imposing algorithm. Consequently, we find a distinguisher of D-QUARK at the same round by imposing different conditions with lower complexity (see Tables 1 and 2).
- We propose the improved CDA on QUARK and obtain the current best cryptanalytic results on all instances of QUARK. We can distinguish U-QUARK, D-QUARK, S-QUARK and C-QUARK up to 157, 171, 292 and 460 rounds with the complexity of 2^{28}, 2^{19}, 2^{22}, and 2^{23} respectively, which respectively increases 2/5/33/8 rounds. For example, we find a distinguisher of C-QUARK up to 460 rounds with complexity 2^{23}, compared to the 452 rounds with higher complexity 2^{28} in the current literature.

Table 1 Results of QUARK

Cipher	Round	Complexity	References
U-QUARK	136	2^{27}	Aumasson et al. (2013)
	153	2^{18}	Zhang et al. (2015)
	155	2^{27}	Yang et al. (2018)
	157	2^{28}	This paper
D-QUARK	159	2^{27}	Aumasson et al. (2013)
	159	2^{22}	Zhang et al. (2015)
	166	2^{25}	Yang et al. (2018)
	166*	2^{22}	This paper
	171	2^{19}	This paper
S-QUARK	237	2^{27}	Aumasson et al. (2013)
	248	2^{24}	Zhang et al. (2015)
	259	2^{15}	Yang et al. (2018)
	273	2^{10}	This paper
	292	2^{22}	This paper
C-QUARK	396	2^{20}	Aumasson et al. (2012)
	445	2^{20}	Zhang et al. (2015)
	452	2^{28}	Yang et al. (2018)
	460	2^{23}	This paper

*We revisit the work in Yang et al. (2018) by imposing different conditions with lower complexity

Table 2 Cryptanalytic Results of QUARK

Cipher	Input difference bits	Ouput difference bit	Round	Bias	Pairs of input samples
U-QUARK	$s_{15}, s_{17}, s_{64}, s_{70}, s_{71}, s_{72},$		157	$2^{-14.01}$	2^{33}
	$s_{76}, s_{82}, s_{84}, s_{89}, s_{93}, s_{95},$	s_0			
	s_{97}, s_{99}, s_{101}				
D-QUARK	s_{41}, s_{89}	s_{88}	166*	$2^{-10.78}$	2^{30}
	s_{89}, s_{144}	s_{88}	171	$2^{-9.67}$	2^{28}
S-QUARK	$s_7, s_{16}, s_{19}, s_{107}, s_{129}, s_{130},$	s_0	273	$2^{-4.84}$	2^{28}
	$s_{130}, s_{140}, s_{143}, s_{144}, s_{145},$				
	$s_{146}, s_{147}, s_{150}, s_{156}, s_{162},$		292	$2^{-10.88}$	2^{30}
	$s_{165}, s_{168}, s_{171}, s_{174}, s_{177}$				
C-QUARK	$s_{24}, s_{32}, s_{34}, s_{38}, s_{40}, s_{91},$	s_0	460	$2^{-11.72}$	2^{30}
	$s_{125}, s_{194}, s_{195}, s_{202}, s_{204},$				
	$s_{212}, s_{220}, s_{225}, s_{227}, s_{229},$				
	$s_{243}, s_{244}, s_{249}, s_{250}, s_{255},$				
	$s_{256}, s_{273}, s_{275}, s_{277}, s_{281},$				
	$s_{283}, s_{285}, s_{287}, s_{289}$				

*We revisit the work in Yang et al. (2018) by imposing different conditions with lower complexity

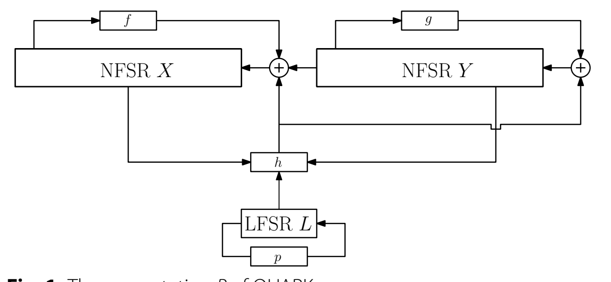

Fig. 1 The permutation *P* of QUARK

- In order to verify the validity and the efficiency of the above attacks, we perform the corresponding experiments on a 3.30 GHz Intel Core i5 CPU. Moreover, these experiments also confirm that our attacks take practical complexities. So as to show the advancement of our improved CDA on QUARK, our cryptanalytic results and the previous results are listed in Table 1.

Preliminaries

In this section, we first make a description of lightweight hash family QUARK. Then, we introduce the basic framework of CDA.

Description of QUARK

The lightweight hash function family QUARK was first proposed by Aumasson et al. (2010) at CHES 2010. Then its extended version was published in *Journal of Cryptology* 2013 (Aumasson et al. 2013), in which the specification of QUARK was slightly modified, more accurately, the digest length has been increased. The authors finally confirm three instances: U-QUARK, D-QUARK, and S-QUARK with the digest lengths defined respectively as 136, 176 and 256 bits. Another instance C-QUARK (Aumasson et al. 2012) is proposed in 2012, which is designed for the scenario of authenticated encryption scheme with associated data (AEAD).

Sponge construction

QUARK makes use of the sponge construction with a b-bit permutation P. The notations as follows were introduced in Aumasson et al. (2013). The structure of each instance of QUARK depends on the *capacity* c, *rate* (or block length) r, and *digest length* n. In Aumasson et al. (2012, 2013), the designers gave the b-bit predefined initial state of each instance of QUARK. In order to deal with a message m, three phases will be applied i.e. *initialization*, *absorbing phase* and *squeezing phase*. During the initialization, the message m is supposed to be padded according to the rate r. The padding rule is to append an '1' bit and the least '0' bits to make the total length be divided by the rate r. After the initialization, the r-bit message block is absorbed in the construction by XOR with the last r bits of the internal state. Then, the internal state is updated by the permutation P in the absorbing phase. During the squeezing phase, the last r bits of the internal state are squeezed. Then the internal state continues to be updated by the permutation P until the total n bits are squeezed as the digest.

Permutation P

The permutation P used in the QUARK is inspired by the stream cipher Grain v1 and the block cipher KATAN, as depicted in Fig. 1. The permutation P is designed based on three feedback shift registers (FSRs). Two are non-linear denoted as X and Y, and one is linear denoted as L. Both the length of the two non-linear feedback shift registers (NFSRs) are $b/2$, and the length of the linear feedback shift register (LFSR) is defined as $l = \lceil log4b \rceil$ bits. Thus, the internal state of permutation P at round $t \geq 0$ can be denoted as (X^t, Y^t, L^t) whose components are listed as follows. The components in X at round t can be denoted as $X^t = (X_0^t, \ldots, X_{b/2-1}^t)$. The components in Y at round t can be denoted as $Y^t = (Y_0^t, \ldots, Y_{b/2-1}^t)$ and the components in L at round t can be denoted as $L^t = (L_0^t, \ldots, L_{l-1}^t)$.

Considering the permutation P as an independent cryptographic primitive, its input can be viewed as a b-bit initial vector (IV). Then the process of permutation P is divided into three phase: *initialization*, *state update* and *computation of the output*. During the initialization, P initializes the internal state (X^0, Y^0, L^0) with the b-bit input denoted as $IV = (s_0, \ldots, s_{b-1})$. In more detail, X is initialized as $(X_0^0, \ldots, X_{b/2-1}^0) := (s_0, \ldots, s_{b/2-1})$. Y is initialized as $(Y_0^0, \ldots, Y_{b/2-1}^0) := (s_{b/2}, \ldots, s_{b-1})$ and L is initialized as $(L_0^0, \ldots, L_{l-1}^0) := (1, \ldots, 1)$.

During the state update phase, the internal state is updated according to the corresponding Boolean functions and the previous state. First, the update value of h^t is computed as $h^t := h(X^t, Y^t, L^t)$. Then, X is clocked with the newly updated bit computed by using Y_0^t, the function f, and the value of h^t. Y is clocked with the newly updated bit computed by using the function g, and the value of h^t. Finally, L is clocked with the newly updated bit computed by using the function p. To show the update process more clearly, the update of the state is listed in (1).

$$\begin{aligned}(X_0^{t+1}, \ldots, X_{b/2-1}^{t+1}) &:= (X_1^t, \ldots, X_{b/2-1}^t, Y_0^t + f(X^t) + h^t),\\ (Y_0^{t+1}, \ldots, Y_{b/2-1}^{t+1}) &:= (Y_1^t, \ldots, Y_{b/2-1}^t, g(Y^t) + h^t),\\ (L_0^{t+1}, \ldots, L_{l-1}^{t+1}) &:= (L_1^t, \ldots, L_{l-1}^t, p(L^t)).\end{aligned} \tag{1}$$

For QUARK, the state of permutation P is updated $4b$ times (especially $2b$ times for C-QUARK). In the state update phase, the final state of the NFSRs X and Y is defined as the output of the permutation P. Then, during the computation of the output phase, such output is used to set to the new internal state of the sponge construction.

Proposed instances

Until now, the hash function family QUARK contains four instances of different flavors: U-QUARK, D-QUARK, S-QUARK, and C-QUARK. The first three instances use the same data-independent LFSR L with the length defined as $l = \lceil log4b \rceil = 10$. The function p of L is defined as $p(L^t) = L_0^t + L_3^t$. For C-QUARK, the length of L is defined as $l = \lceil log4b \rceil = 16$, and the corresponding feedback polynomial $p(L^t)$ returns $L_0^t + L_2^t + L_3^t + L_5^t$.

The functions *f*, *g*, and *h* for each instance of QUARK are nonlinear functions of internal state variables, which are presented in Appendix. For more details of QUARK family, we can refer to the C source code (Aumasson https://github.com/veorq/Quark/) and the original papers (Aumasson et al. 2012, 2013) which are given by the designers.

Backward update functions

As mentioned above, *forward* approach and *backward* approach are used to trace the difference characteristic of the cryptosystem as many rounds as possible. For example, given an single-bit difference at round t, the *forward* approach obtains difference characteristic from round t to round $t+1$. In order to expand the difference characteristic, *backward* approach is proposed to trace the difference characteristic from round t to round $t-1$.

The backward update functions of permutation P used in the backward approach can be derived from the feedback functions by the simple computation. Suppose the state of X, Y and L at round t is $(X_0^t, \ldots, X_{b/2-1}^t)$ and $(Y_0^t, \ldots, Y_{b/2-1}^t)$, and $(L_0^t, \ldots, L_{l-1}^t)$ respectively. Then, the backward update functions can be derived as follows.

$$(L_0^{t-1}, \ldots, L_{l-1}^{t-1}) := (L_{l-1}^t + p^*(L^{t-1}), L_0^t, \ldots, L_{l-2}^t),$$
$$(Y_0^{t-1}, \ldots, Y_{b/2-1}^{t-1}) := (Y_{b/2-1}^t + g^*(Y^{t-1}) + h^{t-1}, Y_0^t, \ldots, Y_{b/2-2}^t),$$
$$(X_0^{t-1}, \ldots, X_{b/2-1}^{t-1}) := (Y_0^{t-1} + X_{b/2-1}^t + f^*(X^{t-1}) + h^{t-1}, X_0^t, \ldots, X_{b/2-2}^t).$$

where $p^*(L^{t-1}) = L_0^{t-1} + p(L^{t-1}), g^*(Y^{t-1}) = Y_0^{t-1} + g(Y^{t-1})$ and $f^*(X^{t-1}) = X_0^{t-1} + f(X^{t-1})$. Notably, according to the definition of p, f, g and h in the previous part, h^{t-1} is independent on X_0^{t-1} and Y_0^{t-1}, $p^*(L^{t-1})$ is independent on L_0^{t-1}, $g^*(Y^{t-1})$ is independent on Y_0^{t-1}, and $f^*(X^{t-1})$ is independent on X_0^{t-1}.

Framework of CDA on NFSR-based cryptosystems

Consider that $\mathcal{C}$ is an NFSR-based cryptosystems and without loss of generality, let the size of internal state be l. Suppose that the cipher $\mathcal{C}$ takes as input a public initial vector $v = (v_1, \ldots, v_a)$ and secret key $k = (k_1, \ldots, k_b)$. Note that the secret key may not be necessary for some hash function cryptosystems. Assume that the internal state of $\mathcal{C}$ at round r is denoted as $(s_r, s_{r+1}, \ldots, s_{r+l-1})$, the newly generated state bit s_{r+l} at round r is updated according to $s_{r+l} = g(s_r, s_{r+1}, \ldots, s_{r+l-1})$, and the output bit z at round r is defined as: $z_r = h(s_r, s_{r+1}, \ldots, s_{r+l-1})$.

Then, we introduce some useful notations with respect to CDA. Let Δ_{in} be the input difference whose '1' differences are only inserted in v, let Δs_{r+l} be the difference of newly generated state bit s_{r+l} at round r, and let Δz_r represent the difference of the output bit of the cipher at round r. Note that the output bit z_r can also be viewed as a Boolean function of k and v denoted as $f(k, v)$. Therefore, the output difference Δz_r can be denoted as: $\Delta z_r = \Delta f(k, v) = f(k, v) + f(k, v + \Delta_{in})$. For $r \geq 0$, the bias ε of Δz_r is defined as: $\varepsilon = Pr(\Delta z_r = 0) - \frac{1}{2}$.

The critical technique of CDA is to control the difference propagation by imposing some conditions (restrictions) on the initial IV/key variables. In order to achieve such goal, we should trace the difference characteristic round by round and make the newly updated uncertain difference expressions Δs_{i+l}, $(0 \leq i \leq r_1 - 1)$ to be 0 or 1 in the first r_1 rounds. Although these equations might contain internal state variables, only the initial public variables can be controlled in a chosen IV scenario. Thus, these equations are supposed to be analyzed carefully to make sure that they are satisfied by imposing conditions only on the initial IV/key variables. In terms of the involved initial variables, the conditions can be divided into three separate types. Type 0 is defined as the condition only including public variables (e.g. IV). Type 1 is defined as the condition including both of public variables (e.g. IV) and secret variables (e.g. key). Type 2 is defined as the condition including only secret variables (e.g. key).

Now we describe the framework of CDA on the NFSR-based cryptosystem. In general, the main idea of CDA (Knellwolf et al. 2010) is to distinguish the output difference Δz_r from a random Boolean function by imposing some conditions to avoid the propagation of the input difference Δ_{in}. The detailed steps of CDA are described as follows.

1 The attacker chooses an initial input difference to trace the difference characteristic of the cipher;
2 Impose conditions on the newly generated state bits to control the early difference propagation;
3 Verify this difference characteristic by experiments to show the rounds where the bias of the difference of output bit can still be observed.
4 Repeat step 1–step 3 until finding a difference characteristic good enough;
5 The attacker performs a distinguishing attack or a key recovery attack, according to the imposed conditions only involve IV bits or also key bits.

Owing to the advancement of the CDA, this cryptanalytic method was also applied to QUARK. In Zhang et al. (2015), proposed an improved CDA and applied it to QUARK. They proposed an algorithm to search for the maximum round based on a rule that there is still a deterministic difference in the internal state for a given difference as input. They applied this algorithm to filter the single-bit difference and select some potential ones which may attack more rounds of the cipher. Then for each single-bit difference candidate, they derived the corresponding input difference by backward computation and imposing conditions from round q to round 0. Consequently, they gave a distinguisher of U-QUARK/D-QUARK/S-QUARK/ C-QUARK up to 153/159/248/445 rounds respectively. In Yang et al. (2018), proposed a technique called symbolic-like computation to simulate the difference propagation. Based on the same rule in Zhang et al. (2015), the technique was used to choose the potential single-bit differences as candidates. After obtaining the single-bit difference, then they made a tradeoff between the number of the backward round and that of the forward round according to the conditions needed to be imposed. Finally they distinguished U-QUARK/D-QUARK/S-QUARK/C-QUARK up to 155/166/259/452 rounds respectively.

Improved CDA on QUARK

In this section, we introduce the the improved CDA on QUARK, which is divided into four phases: *input difference choosing phase* (consists of *forward search part* and *backward search part*), *difference retracing phase*, *condition imposing phase* and *verifying phase*, and its schematic diagram is depicted in Fig. 2.

Compared with the work in Zhang et al. (2015), Yang et al. (2018) mentioned above, two improvements are applied in our CDA on QUARK. One improvement is that we choose the single-bit differences as the candidates following the rule that this single-bit difference will result in a large bias of output difference at a high enough round. Furthermore, the bias is estimated by the technique proposed by Liu et al. (2021) at CRYPTO 2021. In other words, we consider a probabilistic difference characteristic (e.g. $|\varepsilon| > 2^{-12}$) rather than a deterministic one (i.e. $|\varepsilon| = \frac{1}{2}$) in the input difference choosing phase. The other improvement is that we proposed an algorithm to automatically impose conditions to deal with the complex condition expressions. In the following, we introduce the four phases of improved CDA on QUARK in detail and the two improvements will be further illustrated in the input difference choosing phase and condition imposing phase respectively.

Input difference choosing phase

In this phase, we target at choosing a good input difference which owns a good difference characteristic leading to a detectable bias of output difference at rounds as high as possible. The improved procedure is divided into the following two parts: *forward search part* and *backward search part*.

Before introducing the forward search part, some notations should be defined first. The single-bit difference inserted in the i-th bit of IV is denoted by e_i, where $0 \le i \le b-1$. Let R_{e_i} denote the maximum round at which the bias ε_{e_i} of output difference can be detected with difference e_i. We call e_i is a *good* single-bit difference if R_{e_i} satisfies $R_{e_i} > TH$ and ε_{e_i} satisfies $|\varepsilon_{e_i}| > \varepsilon_0$, where TH is a predefined constant threshold and the ε_0 is usually a relatively large bias (e.g. 2^{-10}).

Forward search part

At CRYPTO 2021, Liu et al. (2021) studied the differential-linear attack from an algebraic perspective. They

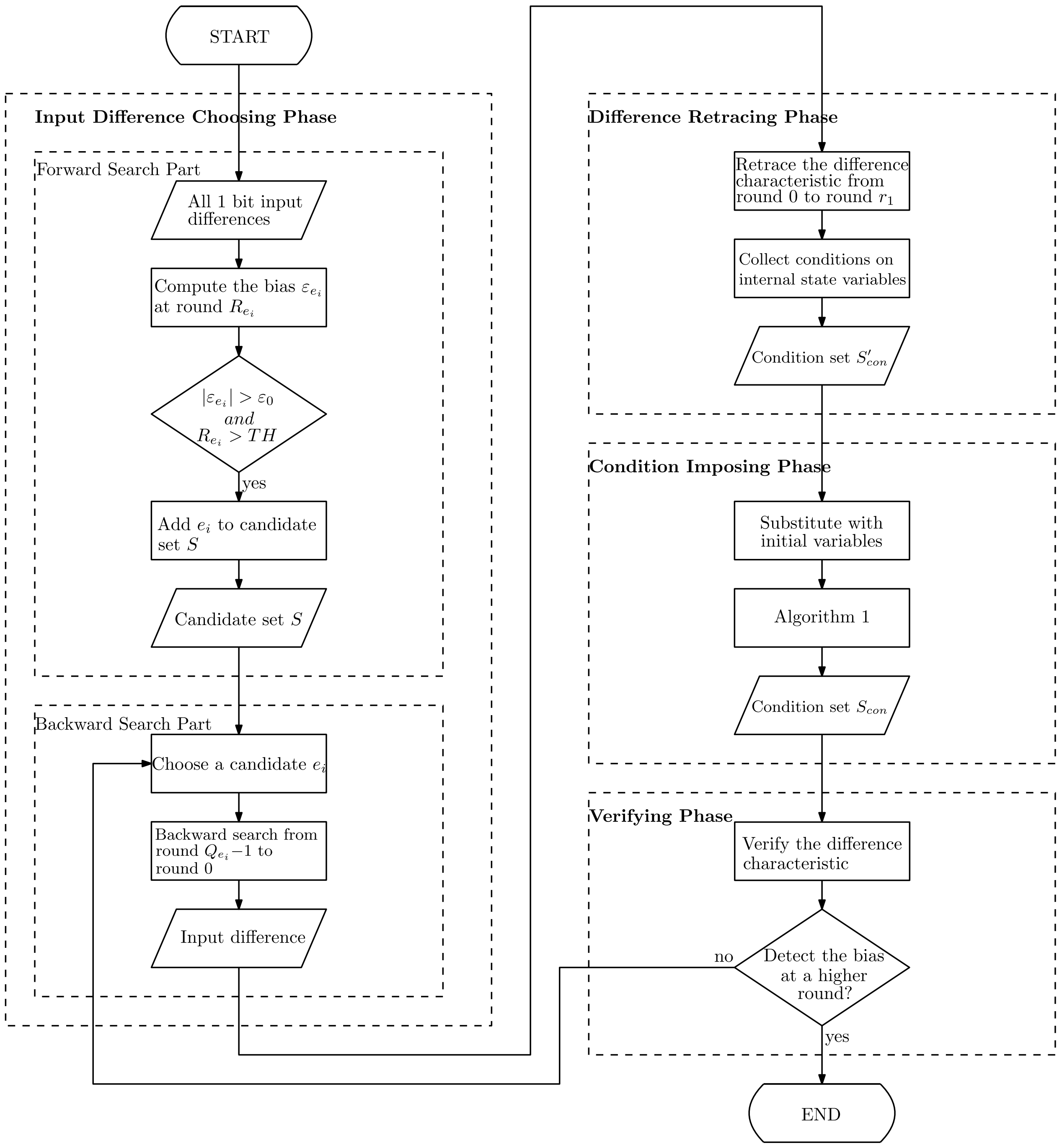
START
Input Difference Choosing Phase
Forward Search Part
All 1 bit input differences
Compute the bias ε_{e_i} at round R_{e_i}
$|\varepsilon_{e_i}| > \varepsilon_0$ *and* $R_{e_i} > TH$
yes
Add e_i to candidate set S
Candidate set S
Backward Search Part
Choose a candidate e_i
Backward search from round $Q_{e_i}-1$ to round 0
Input difference
Difference Retracing Phase
Retrace the difference characteristic from round 0 to round r_1
Collect conditions on internal state variables
Condition set S'_{con}
Condition Imposing Phase
Substitute with initial variables
Algorithm 1
Condition set S_{con}
Verifying Phase
Verify the difference characteristic
Detect the bias at a higher round?
no
yes
END

Fig. 2 Schematic diagram of improved CDA on QUARK

proposed a new theory of estimation of the differential-linear bias. Since a truncated differential attack can be considered as an extreme case of the differential-linear attack (Blondeau et al. 2017), the theory of estimation of the differential-linear bias can also be applied to the CDA (Liu et al. 2021). During the forward search part, we applied the technique proposed in Liu et al. (2021) to estimate the bias of output difference in the CDA. The detailed procedure is presented as follows.

We apply the Algorithm 1 and Algorithm 2 in Liu et al. (2021) to all single-bit differences to add the good ones to the candidate set S. For each $e_i \in S$, R_{e_i} of the single-bit difference e_i should satisfy $R_{e_i} > TH$, and the bias ε_{e_i} of output difference at round R_{e_i} should satisfy $|\varepsilon_{e_i}| > \varepsilon_0$. In addition, the number of '0' differences in its difference characteristic should be as large as possible while the number of '1' differences should be as small as possible. Note that in the forward search part, the previous research work (Zhang et al. 2015; Yang et al. 2018) aimed to find the maximum round of each single-bit difference, where a deterministic difference still exists in the internal state (i.e. $|\varepsilon_{e_i}| = \frac{1}{2}$). Then the top k ($k \geq 1$) are chosen to be the candidates. Compared with their methods, we aim to search for the candidates which the biases of output differences are still large enough at the rounds as high as possible. From this perspective, we consider a probabilistic difference characteristic (e.g. $|\varepsilon_{e_i}| > 2^{-12}$) rather than a deterministic one (i.e. $|\varepsilon_{e_i}| = \frac{1}{2}$) in the improved CDA on QUARK. Therefore, the candidates we choose are expected to lead to a detectable bias of output difference at a higher round.

Backward search part

The aim of the backward search part is to expand rounds as many as possible. However, with the increasing round of the cipher, the symbolic expressions of the output bit get more and more complex. Furthermore, the symbolic expressions of the output bit are too complex to compute after some rounds letting alone the derivation of the required conditions. In light of this, in the backward search part, we should choose the rounds at which the symbolic expressions of the output bit and the required conditions can both be derived. During this part, we choose a candidate $e_i \in S$, we take advantage of the backward approach to trace back the difference characteristic for Q_{e_i} rounds obtaining the input difference (with arbitrary weight), that is such input difference will lead to the single-bit difference e_i after Q_{e_i} rounds. Now, we derive an input difference.

Up to now, we can obtain the single-bit difference candidate set S in the forward search part and could obtain the corresponding input difference in the backward search part. Such obtained input difference is expected to result in a detectable bias of output difference at a higher round of QUARK.

Difference retracing phase

In order to prevent the difference propagation as many rounds as possible, the difference retracing phase should be performed. During this phase, for an input difference obtained in the input difference choosing phase, we retrace the difference characteristic from the round 0 to round r_1 and finally collect a set $S_{con}^{'}$ of conditions on the internal state variables. In other words, we prevent difference propagation and collect conditions from round 0 to round r_1. When we retrace the difference characteristic for an input difference, we should impose the conditions as simple as possible and restrain the number of '1' difference in the newly updated bits as small as possible. This is because of the following two findings. First, the difference propagates more rapidly with the increasing number of difference '1', which can also lead to the increasing number of conditions. Second, the more complex conditions we want to impose, the more likely the initial bits will be conflicts. Then we can obtain the rounds of an input difference at which a detectable bias of the output difference can be detected.

Condition imposing phase

After collecting the set $S_{con}^{'}$ which are conditions on the internal state variables in the difference retracing phase, the next task is to analyze carefully to ensure that the conditions only contain IV variables. Since condition imposing is a very difficult and time-consuming process, we first introduce the method to automatically impose conditions. we propose an algorithm to automatically obtain a set of initial variables satisfying these conditions while keeping total number of conditions as small as possible. In order to describe the algorithm clearly, we first give a definition as follows. Given the algebraic normal form f of a Boolean function, the definition of *maximum term* of f is defined in Definition 1.

Definition 1 (*Maximum Term*) Given the algebraic normal form f of a Boolean function, in all the decomposition forms of $f = g_1 \cdot g_2 + h$ with g_1 being a linear function containing at most n_1 variables, $(g_1 \cdot g_2)_{max}$ is said to be *maximum term* of f if the number of terms of $g_1 \cdot g_2$ is the maximum.

Note that the maximum term can be nullified in f by imposing the conditions on g_1, that is let $g_1 = 0$. Since g_1 is a linear Boolean polynomial, conditions imposed on it can be easily derived. Then, the complexity of imposing conditions on f can be transformed to that of imposing conditions on the remaining part h which can also be viewed as a new Boolean function. Therefore, by iteratively nullifying the maximum terms in the Boolean functions, the relative fewer and simpler conditions can be derived. The whole procedure is depicted as Algorithm 1.

Algorithm 1: Automatic Condition Imposing Algorithm

```
Input: A Boolean polynomial $f$, a constant $C$, a set $G$.
Output: Imposed conditions set $I$.
$C$ can be specifically selected according to $f$;
$G$ enumerates the linear expressions containing at most $n_1$ variables of $f$;
$I = \emptyset$;
while $len(f) > C$ do /* $len()$ returns the number of terms */
    $max = 0$;
    for $g$ in $G$ do
        $h \leftarrow f$ mod $g$;
        if $len(f - h) > max$ then
            $max = len(f - h)$;
            $h_{min} \leftarrow h$;
            $cond \leftarrow g$;
        end
    end
    $f \leftarrow h_{min}$;
    Add $cond$ to $I$;
end
Add $f$ to $I$;
Return $I$.
```

In condition imposing phase, when applying the Algorithm 1 to QUARK, $|G|$ becomes too large if $n_1 > 4$ in the case of condition expressions containing many variables. Therefore, n_1 is usually set as $n_1 = 4$ for the sake of the efficiency. Furthermore, according to some tests and experiments, the parameter $C < 25$ can usually lead to satisfying results. Thus, from an empirical perspective, C is usually set as $C < 25$. By applying these settings, we can derive a set S_{con} in which conditions are on IV variables.

Verifying phase

When we obtain the input difference and a set S_{con} of conditions on IV variables, we should verify this difference characteristic by experiments to show the rounds at which the bias of output difference can still be observed with enough input samples. For each pair of input samples, the initial state bits are set as follows: a bit included in the initial input difference is set to be the difference bits, bits given in the condition imposing phase are set to satisfy conditions and the other IV bits are set to random values. Note that the experiments also provide a demonstration of the practical complexity of our improved CDA. If we can not observe a bias at a higher round from this input difference, we should go back to backward search part to select another single-bit difference in candidate set S and continue the subsequent phases until the bias at a higher round can be detected.

Analysis of QUARK

In this section, we give the cryptanalysis of the improved CDA on all instances of QUARK. In Aumasson et al. (2012); Aumasson et al. (2013), the authors proposed a standard attack model for QUARK, in which the initial state of the permutation P is assumed chosen uniformly and independently at random. The previous cryptanalytic researches of QUARK all followed this standard attack model. By following the standard attack model, our cryptanalysis also aims at the permutation P. We first take S-QUARK as an example to show the whole analysis progress.

Analysis of S-QUARK

The procedure of the our improved CDA on QUARK can be divided into four phases: input difference choosing phase (consists of forward search part and backward search part), difference retracing phase, condition imposing phase and verifying phase. We give the analysis of S-QUARK by illustrating these four phases of S-QUARK in detail as follows.

In the input difference choosing phase, we first choose some single-bit differences as candidates in the forward search part. First, we choose $TH = 100$ and $\varepsilon_0 = 2^{-12}$. Then we exhaustively search all single-bit differences to obtain the candidates by applying the technique proposed by Liu et al. at CRYPTO 2021. The set of candidates is

$$S = \{e_{57}, e_{89}, e_{196}, e_{70}\}.$$

In the backward search part, for the obtained candidate set S, we choose candidate e_{57} using the backward approach to expand more rounds. For S-QUARK, we choose $Q_{e_{57}} = 51$ and obtain the input difference bits in the following:

$$s_7, s_{16}, s_{19}, s_{107}, s_{129}, s_{130}, s_{140}, s_{143}, s_{144}, s_{145}, \\ s_{146}, s_{147}, s_{150}, s_{156}, s_{162}, s_{165}, s_{168}, s_{171}, s_{174}, s_{177}. \quad (2)$$

When the input difference is obtained, we enter into difference retracing phase. We collect the conditions from round 0 to round 68, which are complex Boolean polynomials of internal state variables. We first replace the internal state variables by initial IV variables.

Then we enter into condition imposing phase. We apply the automatic condition imposing Algorithm 1 to these complex conditions and the detailed conditions are listed as follows.

At round 0 (i.e. after 1 round), in order to control the difference ($\Delta X^1_{127} = 0, \Delta Y^1_{127} = 0$), we impose the conditions as follows:

$$s_{28} = s_{52} = s_{141} = s_{184} = s_{229} = 0.$$

At round 2, in order to control the difference ($\Delta Y^3_{127} = 0$) , we impose the conditions as follows:

$$s_{195} = s_{209} = 0, s_{158} = 1.$$

At round 3, in order to control the difference ($\Delta X^4_{127} = 0, \Delta Y^4_{127} = 0$) , we impose the conditions as follows:

$$s_{31} = s_{55} = s_{159} = s_{210} = 0, s_{187} = s_{196} = 1.$$

At round 4, in order to control the difference ($\Delta X^5_{127} = 0, \Delta Y^5_{127} = 0$), we impose the conditions as follows:

$$s_{73} = s_{76} = s_{88} = s_{104} = s_{115} = s_{188} = 0, s_{160} = 1.$$

At round 5, in order to control the difference ($\Delta Y^6_{127} = 0$), we impose the conditions as follows:

$$s_{198} = s_{212} = 0, s_{161} = 1.$$

At round 6, in order to control the difference ($\Delta Y^7_{127} = 0$) , we impose the conditions as follows:

$$s_{162} = s_{190} = s_{243} = 0, s_{147} = 1.$$

At round 7, in order to control the difference ($\Delta X^8_{127} = 0$) , we impose the conditions as follows:

$$s_{79} = s_{107} = s_{143} = s_{246} = 0.$$

At round 8, in order to control the difference ($\Delta X^9_{127} = 0$) , we impose the conditions as follows:

$$s_{108} = s_{247} = 0.$$

At round 9, in order to control the difference ($\Delta X^{10}_{127} = 0, \Delta Y^{10}_{127} = 0$), we impose the conditions as follows:

$$s_{81} = s_{150} = s_{202} = 0, s_{109} = s_{165} = 1.$$

At round 10, in order to control the difference ($\Delta X^{11}_{127} = 0$) , we impose the conditions as follows:

$$s_{62} = s_{71} = s_{82} = s_{110} = s_{113} = 0.$$

At round 11, in order to control the difference ($\Delta X^{12}_{127} = 0$) , we impose the conditions as follows:

$$s_{83} = s_{111} = 0.$$

At round 12, in order to control the difference ($\Delta X^{13}_{127} = 0$) , we impose the conditions as follows:

$$s_{153} = s_{205} = s_{241} = 0.$$

At round 14, in order to control the difference ($\Delta X^{15}_{127} = 0$) , we impose the conditions as follows:

$$s_{86} = 0, s_{242} = 1.$$

At round 15, in order to control the difference ($\Delta Y^{16}_{127} = 0$) , we impose the conditions as follows:

$$s_{222} = 0, s_{156} = s_{171} = s_{180} = s_{199} = s_{208} = s_{244} = s_{252} = 1.$$

At round 20, in order to control the difference ($\Delta X^{21}_{127} = 0$), besides imposing the conditions $s_{67} = 0, s_{92} = s_{120} = 1$, we impose the conditions on X^0_4 as follows:

$$s_4 = s_{10} + s_{50}s_{139} + s_{50} + s_{61} + s_{93} + s_{131} + s_{134} + s_{139} + s_{144} \\ + s_{149} + s_{200}s_{227}s_{232} + s_{223} + s_{227} + s_{232}s_{240} + s_{232}.$$

At round 21, in order to control the difference ($\Delta Y^{22}_{127} = 0$) , we impose the conditions as follows:

$$s_{177} = s_{250} = 0.$$

At round 23, in order to control the difference ($\Delta X^{24}_{127} = 0$), besides imposing the conditions $s_{126} = 0, s_{39} = s_{75} = 1$, we impose the conditions on X^0_6 as follows:

$$\begin{aligned} s_6 = {} & s_{13} + s_{22}s_{34} + s_{22}s_{58}s_{90}s_{117} + s_{22} \\ & + s_{32} + s_{34}s_{45}s_{117} + s_{45} \\ & + s_{53}s_{78}s_{142} + s_{53}s_{78}s_{245} + s_{58} \\ & + s_{64} + s_{78}s_{106}s_{142} + s_{78}s_{106} \\ & + s_{78}s_{142}s_{245} + s_{90}s_{103}s_{117} \\ & + s_{90}s_{103} + s_{90} + s_{100} + s_{103} \\ & + s_{106}s_{142} + s_{106}s_{245} + s_{117} \\ & + s_{134} + s_{137} + s_{152} + s_{168} + s_{205} \\ & + s_{225} + s_{245}. \end{aligned}$$

At round 24, in order to control the difference ($\Delta Y^{25}_{127} = 0$) , we impose the conditions as follows:

$$s_{217} = s_{231} = 0.$$

At round 26, in order to control the difference ($\Delta X^{27}_{127} = 0$), we impose the conditions on Y^0_{39} as follows:

$$\begin{aligned} s_{167} = s_{10} &+ s_{16} + s_{114} + s_{137} + s_{140} \\ &+ s_{145} + s_{155} + s_{174} + s_{193} \\ &+ s_{206} + s_{216}s_{233}s_{238} + s_{216} \\ &+ s_{228} + s_{233} + s_{238}. \end{aligned}$$

At round 27, in order to control the difference ($\Delta Y^{28}_{127} = 0$) , we impose the conditions as follows:

$$s_{211} = 0, s_{183} = 1.$$

At round 29, in order to control the difference ($\Delta X^{30}_{127} = 0$), besides imposing the conditions $s_{101} = 0$, we impose the conditions on X^0_1, X^0_{19} as follows:

$$\begin{aligned} s_1 = s_2 &+ s_8 + s_{17}s_{29} + s_{17}s_{53}s_{112} \\ &+ s_{17} + s_{27} + s_{40} + s_{53} + s_{59} \\ &+ s_{70} + s_{91} + s_{95} + s_{101}s_{137} \\ &+ s_{101}s_{240} + s_{106} + s_{129} + s_{132} \\ &+ s_{163} + s_{200} + s_{220} + s_{240} + 1, \\ s_{19} = s_{13} &+ s_{59}s_{84}s_{148} + s_{59} + s_{70} \\ &+ s_{84}s_{112}s_{148} + s_{84}s_{112} \\ &+ s_{84}s_{148}s_{251} + s_{102} + s_{112}s_{148} \\ &+ s_{112}s_{251} + s_{117} + s_{140} \\ &+ s_{153}s_{168} + s_{153}s_{219}s_{249} + s_{153} \\ &+ s_{170} + s_{174} + s_{205} \\ &+ s_{219}s_{236}s_{241} + s_{219} + s_{220} \\ &+ s_{232} + s_{236} + s_{241}s_{249} + s_{241} \\ &+ s_{251} + 1. \end{aligned}$$

At round 30, in order to control the difference ($\Delta Y^{31}_{127} = 0$) , we impose the conditions as follows:

$$s_{214} = 0, s_{186} = 1.$$

At round 32, in order tocontrol the difference ($\Delta X^{33}_{127} = 0$), we impose the conditions on X^0_5 as follows:

$$\begin{aligned} s_5 = s_4 &+ s_{11} + s_{20}s_{32}s_{43}s_{56}s_{65} \\ &+ s_{20}s_{32} + s_{20} + s_{30} + s_{43}s_{56}s_{65} \\ &+ s_{43} + s_{56} + s_{65} + s_{101} + s_{132} \\ &+ s_{135} + s_{166} + s_{203} + s_{223}. \end{aligned}$$

At round 33, in order to control the difference ($\Delta Y^{34}_{127} = 0$) , we impose the conditions as follows:

$$s_{189} = 1.$$

At round 36, in order to control the difference ($\Delta Y^{37}_{127} = 0$), we impose the conditions as follows:

$$s_{192} = 1.$$

At round 38, in order to control the difference ($\Delta X^{39}_{127} = 0$), besides imposing the conditions $s_{99} = 1$, we impose the conditions on X^0_{14}, X^0_{36} as follows:

$$\begin{aligned} s_{14} = s_8 &+ s_{23}s_{35}s_{46}s_{59}s_{68} \\ &+ s_{23}s_{35} + s_{23}s_{59}s_{91}s_{118} + s_{23} + s_{33} \\ &+ s_{46}s_{59}s_{68} + s_{46} + s_{59} + s_{65} \\ &+ s_{68} + s_{91} + s_{97} + s_{101} + s_{112} \\ &+ s_{135} + s_{138} + s_{153} + s_{169} \\ &+ s_{206} + s_{215} + s_{226} + 1 \\ s_{36} = s_{10} &+ s_{11} + s_{17} + s_{26}s_{38} + s_{26} \\ &+ s_{49} + s_{57} + s_{68} + s_{100} + s_{138} \\ &+ s_{172} + s_{218} + s_{249} \end{aligned}$$

At round 41, in order to control the difference ($\Delta X^{42}_{127} = 0$), we impose the conditions on Y^0_{47} as follows:

$$\begin{aligned} s_{175} = s_{13} &+ s_{14} + s_{20} + s_{29}s_{41} + s_{29} \\ &+ s_{60}s_{149} + s_{60} + s_{65} + s_{74} \\ &+ s_{97} + s_{103} + s_{118} + s_{144} \\ &+ s_{149} + s_{221} + s_{232} + 1. \end{aligned}$$

At round 46, in order to control the difference ($\Delta X^{47}_{127} = 0$) , we impose the conditions as follows:

$$s_{85} = s_{98} = 1.$$

At round 55, in order to control the difference ($\Delta X^{56}_{127} = 0$) , we impose the conditions as follows:

$$s_{124} = 0, s_{94} = s_{116} = 1.$$

At round 68, in order to control the difference ($\Delta X^{69}_{127} = 0$), besides imposing the conditions $s_{96} = 0$, we impose the conditions on X^0_9 as follows:

$$\begin{aligned} s_9 = s_{10} &+ s_{16} + s_{25}s_{37}s_{48}s_{61}s_{70} \\ &+ s_{25}s_{37} + s_{25}s_{61}s_{93} + s_{25} + s_{35} \\ &+ s_{37}s_{48}s_{112} + s_{48}s_{61}s_{70}s_{78}s_{93}s_{106} \\ &+ s_{48}s_{61}s_{70} + s_{48} + s_{61} \\ &+ s_{70}s_{78}s_{106}s_{112} + s_{70}s_{78} + s_{70} \\ &+ s_{78}s_{93}s_{106}s_{112} + s_{78} \\ &+ s_{93}s_{106}s_{112} + s_{93} + s_{103} \\ &+ s_{106} + s_{114} + s_{137} + s_{140} \\ &+ s_{145} + s_{155} + s_{228}. \end{aligned}$$

So far, the input difference and conditions on IV bits of S-QUARK are derived. Now, we start to the last verifying phase. We perform experiments 10 times each with enough pairs of input samples. For each pair of input samples, the IV bits are set as follows: a bit described in (2) are set to be the difference bit, bits given in the condition imposing phase are set to satisfy conditions and the other IV bits are set to random values. We randomly generate 2^{28} pairs of input samples and observe the bias of the difference of output bit round by round. After 273 rounds, the average bias of difference in state bit $s_0(\Delta X_0^{273})$ is $2^{-4.84}$ with standard deviation of $2^{-15.47}$. Note that the number of pairs of input samples are large enough to ensure the validity of our results.

Further, we randomly generate 2^{30} pairs of input samples to perform the above experiments and after 292 rounds, the average bias of difference in state bit $s_0(\Delta X_0^{292})$ is $2^{-10.88}$ with standard deviation of $2^{-15.34}$. Also, we have randomly generate 2^{35} pairs of input samples to perform the above experiments and do not detect any bias at higher round.

Cryptanalytic results of QUARK

In the above subsection, we have presented the detailed analysis of S-QUARK. We also perform the cryptanalysis of our improved CDA on the other instances of QUARK. The cryptanalytic results for all instances of QUARK are showed in Table 2. In addition, we list the imposed conditions for U-QUARK, D-QUARK and C-QUARK in Appendix. All the attacks are performed on a 3.30 GHz Intel Core i5 CPU, and all these attacks take practical complexities which have been fully verified by our experiments. Compared to the existing cryptanalytic results of QUARK, we can distinguish U-QUARK/D-QUARK/S-QUARK/C-QUARK up to 157/171/292/460 rounds with increasing 2/5/33/8 rounds respectively. The complexities of the attacks of U-QUARK/D-QUARK/S-QUARK/C-QUARK on the above rounds are $2^{28}/2^{19}/2^{22}/2^{23}$ respectively. As far as we know, all of these results have been the best thus far.

Conclusion

In this paper, we propose an improved CDA on QUARK. One improvement is that we could obtain a set of good input differences by the proposed method to select the input difference. Another improvement is that we proposed an algorithm to automatically impose conditions to deal with the complex condition expressions. Then, we apply the improved CDA on QUARK and obtain the corresponding improved analytic results. With the two improvements, the bias of the difference of the output bit at a higher round can be detected. Furthermore, all the cryptanalytic results are verified by practical experiments and to the best of our knowledge, these results have been the best so far. In the future, the conditions of other cryptosystems can hopefully be dealt with by our automatic condition imposing algorithm in the CDA.

Appendix

Feedback functions of QUARK

U-QUARK

U-QUARK is the lightest flavor of QUARK. It was designed to provide 64-bit security, and to admit a parallelization degree of 8. It has sponge numbers $r = 8, c = 128, b = 136, n = 136$. The feedback functions for U-QUARK is defined as follows.

$$\begin{aligned}
f(X) = {} & X_0 + X_9 + X_{14} + X_{21} + X_{28} + X_{33} + X_{37} + X_{45} + X_{50} + X_{52} \\
& + X_{55} + X_{55}X_{59} + X_{33}X_{37} + X_9X_{15} + X_{45}X_{52}X_{55} + X_{21}X_{28}X_{33} \\
& + X_9X_{28}X_{45}X_{59} + X_{33}X_{37}X_{52}X_{55} + X_{15}X_{21}X_{55}X_{59} \\
& + X_{37}X_{45}X_{52}X_{55}X_{59} + X_9X_{15}X_{21}X_{28}X_{33} \\
& + X_{21}X_{28}X_{33}X_{37}X_{45}X_{52}, \\
g(Y) = {} & Y_0 + Y_7 + Y_{16} + Y_{20} + Y_{30} + Y_{35} + Y_{37} + Y_{42} + Y_{49} + Y_{51} + Y_{54} \\
& + Y_{54}Y_{58} + Y_{35}Y_{37} + Y_7Y_{15} + Y_{42}Y_{51}Y_{54} + Y_{20}Y_{30}Y_{35} \\
& + Y_7Y_{30}Y_{42}Y_{58} + Y_{35}Y_{37}Y_{51}Y_{54} + Y_{15}Y_{20}Y_{54}Y_{58} \\
& + Y_{37}Y_{42}Y_{51}Y_{54}Y_{58} + Y_7Y_{15}Y_{20}Y_{30}Y_{35} + Y_{20}Y_{30}Y_{35}Y_{37}Y_{42}Y_{51}, \\
h(X,Y,L) = {} & L_0 + X_1 + Y_2 + X_4 + Y_{10} + X_{25} + X_{31} + Y_{43} + X_{56} + Y_{59} \\
& + Y_3X_{55} + X_{46}X_{55} + X_{55}Y_{59} + Y_3X_{25}X_{46} + Y_3X_{46}X_{55} \\
& + Y_3X_{46}Y_{59} + L_0X_{25}X_{46}Y_{59} + L_0X_{25}.
\end{aligned}$$

D-QUARK

D-QUARK is the second-lightest flavor of QUARK. It was designed to provide 80-bit security, and to admit a parallelization degree of 8. It has parameters $r = 16, c = 160, b = 176, n = 176$. D-QUARK uses the same function *f* as U-QUARK, but with taps 0, 11, 18, 19, 27, 36, 42, 47, 58, 64, 67, 71, 79 instead of 0, 9, 14, 15, 21, 28, 33, 37, 45, 50, 52, 55, 59, respectively. The function *g* is also the same as U-QUARK, but with taps 0, 9, 19, 20, 25, 38, 44, 47, 54, 63, 67, 69, 78 instead of 0, 7, 15, 16, 20, 30, 35, 37, 42, 49, 51, 54, 58, respectively. The function *h* is defined as:

$$\begin{aligned} h(X, Y, L) = {} & L_0 + X_1 + Y_2 + X_5 + Y_{12} + Y_{24} \\ & + X_{35} + X_{40} + X_{48} + Y_{55} \\ & + Y_{61} + X_{72} + Y_{79} + Y_4X_{68} \\ & + X_{57}X_{68} + X_{68}Y_{79} + Y_4X_{35}X_{57} \\ & + Y_4X_{57}X_{68} + Y_4X_{57}Y_{79} \\ & + L_0X_{35}X_{57}Y_{79} + L_0X_{35}. \end{aligned}$$

S-QUARK

S-QUARK is the second-heaviest flavor of QUARK. It was designed to provide 112-bit security, and to admit a parallelization degree of 16. It has parameters $r = 32, c = 224, b = 256, n = 256$. S-QUARK uses the same function *f* as U-QUARK, but with taps 0, 16, 26, 28, 39, 52, 61, 69, 84, 94, 97, 103, 111 instead of 0, 9, 14, 15, 21, 28, 33, 37, 45, 50, 52, 55, 59, respectively. The function *g* is also the same as U-QUARK, but with taps 0, 13, 28, 30, 37, 56, 65, 69, 79, 92, 96, 101, 109 instead of 0, 7, 15, 16, 20, 30, 35, 37, 42, 49, 51, 54, 58, respectively. The function *h* is defined as:

$$\begin{aligned} h(X, Y, L) = {} & L_0 + X_1 + Y_3 + X_7 + Y_{18} + Y_{34} \\ & + X_{47} + X_{58} + Y_{71} + Y_{80} + X_{90} \\ & + Y_{91} + X_{105} + Y_{111} + Y_8X_{100} \\ & + X_{72}X_{100} + X_{100}Y_{111} + Y_8X_{47}X_{72} \\ & + Y_8X_{72}X_{100} + Y_8X_{72}Y_{111} \\ & + L_0X_{47}X_{72}Y_{111} + L_0X_{47}. \end{aligned}$$

C-QUARK

C-QUARK is the heaviest flavor of QUARK. It was designed to admit a parallelization degree of 32. It has parameters $r = 64, c = 320, b = 384, n = 384$. The feedback functions for C-QUARK is defined as follows.

$$\begin{aligned} f(X) = {} & X_0 + X_{13} + X_{34} + X_{65} + X_{77} \\ & + X_{94} + X_{109} + X_{127} + X_{145} + X_{157} \\ & + X_{140} + X_{159}X_{157} + X_{109}X_{94} \\ & + X_{47}X_{13} + X_{157}X_{145}X_{127} \\ & + X_{94}X_{77}X_{65} + X_{159}X_{127}X_{77}X_{13} \\ & + X_{157}X_{145}X_{109}X_{94} \\ & + X_{159}X_{157}X_{65}X_{47} \\ & + X_{159}X_{157}X_{145}X_{127}X_{109} \\ & + X_{94}X_{77}X_{65}X_{47}X_{13} \\ & + X_{145}X_{127}X_{109}X_{94}X_{77}X_{65}, \\ g(Y) = {} & Y_0 + Y_{21} + Y_{57} + Y_{60} + Y_{94} \\ & + Y_{112} + Y_{125} + Y_{133} + Y_{152} \\ & + Y_{157} + Y_{146} + Y_{159}Y_{157} \\ & + Y_{125}Y_{112} + Y_{36}Y_{21} + Y_{157}Y_{152}Y_{133} \\ & + Y_{112}Y_{94}Y_{60} + Y_{159}Y_{133}Y_{94}Y_{21} \\ & + Y_{157}Y_{152}Y_{125}Y_{112} \\ & + Y_{159}Y_{157}Y_{60}Y_{36} \\ & + Y_{159}Y_{157}Y_{152}Y_{133}Y_{125} \\ & + Y_{112}Y_{94}Y_{60}Y_{36}Y_{21} \\ & + Y_{152}Y_{133}Y_{125}Y_{112}Y_{94}Y_{60}, \\ h(X, Y, L) = {} & X_{25} + Y_{59} + Y_3X_{55} + X_{46}X_{55} \\ & + X_{55}Y_{59} + Y_3X_{25}X_{46} + \\ & Y_3X_{46}X_{55} + Y_3X_{46}Y_{59} \\ & + X_{25}X_{46}Y_{59}L_0 + X_{25}L_0 + L_0 \\ & + X_4 + X_{28} + X_{40} + X_{85} \\ & + X_{112} + X_{141} + X_{146} + X_{152} \\ & + Y_2 + Y_{33} + Y_{60} + Y_{62} \\ & + Y_{87} + Y_{99} + Y_{138} + Y_{148}. \end{aligned}$$

Conditions for U-QUARK, D-QUARK and C-QUARK

U-QUARK

The IV bits needed to set as '0' are listed as follows:

$$\begin{aligned} & s_{11}, s_{26}, s_{27}, s_{28}, s_{33}, s_{36}, s_{43}, s_{46}, s_{50}, s_{51}, \\ & s_{55}, s_{58}, s_{61}, s_{66}, s_{67}, s_{76}, s_{78}, s_{81}, s_{84}, s_{85}, \\ & s_{88}, s_{89}, s_{90}, s_{91}, s_{92}, s_{93}, s_{94}, s_{96}, s_{101}, s_{106}, \\ & s_{111}, s_{112}, s_{114}, s_{115}, s_{118}, s_{120}, s_{122}, s_{128}, s_{135}. \end{aligned}$$

The IV bits needed to set as '1' are listed as follows:

$$\begin{aligned} & s_9, s_{21}, s_{23}, s_{38}, s_{40}, s_{47}, s_{52}, s_{56}, s_{57}, s_{59}, s_{60}, s_{64}, s_{80}, s_{87}, s_{97}, s_{103}, \\ & s_{105}, s_{107}, s_{108}, s_{109}, s_{110}, s_{117}, s_{123}, s_{127}, s_{133}. \end{aligned}$$

The other condition expressions are listed as follows:

$$\begin{aligned}
s_0 &= s_1 + s_4 + s_{14} + s_{31} + s_{37} + s_{68} + s_{70},\\
s_1 &= s_2 + s_5 + s_{10}s_{16}s_{22}s_{29}s_{34} + s_{10}s_{16} + s_{10} + s_{16}s_{22} + s_{22}s_{29}s_{34}\\
&\quad + s_{22} + s_{29} + s_{32} + s_{34}s_{53} + s_{53} + s_{69} + s_{71} + s_{79} + 1,\\
s_3 &= s_2 + s_6 + s_{16} + s_{35} + s_{70} + s_{72} + s_{73} + s_{113},\\
s_4 &= s_7 + s_{34} + s_{71} + s_{73} + s_{113} + s_{125}s_{129} + s_{125} + s_{130},\\
s_8 &= s_{32}s_{53}s_{134} + s_{53}s_{62} + s_{62}s_{134} + s_{77} + s_{124} + s_{126}s_{129} + s_{126}\\
&\quad + s_{134} + 1,\\
s_{12} &= s_8 + s_{22} + s_{29} + s_{53} + s_{86} + s_{119} + 1,\\
s_{14} &= s_{13} + s_{17} + s_{22} + s_{34} + s_{44} + s_{65} + s_{83} + s_{124},\\
s_{17} &= s_3 + s_4 + s_7 + s_{12}s_{18} + s_{12} + s_{24} + s_{31} + s_{34} + s_{53} + s_{71}\\
&\quad + s_{73} + s_{130},\\
s_{19} &= s_{16} + s_{24}s_{30} + s_{24} + s_{29} + s_{65} + s_{83} + s_{126},\\
s_{20} &= s_6 + s_7 + s_{10} + s_{31}s_{77} + s_{31} + s_{34} + s_{37} + s_{62} + s_{77},\\
s_{42} &= s_{10} + s_{13} + s_{18}s_{24} + s_{18} + s_{37} + s_{54} + s_{65} + s_{77} + s_{79},\\
s_{44} &= s_{12} + s_{20} + s_{25} + s_{32} + s_{42} + s_{79} + 1,\\
s_{49} &= s_{12} + s_{13} + s_{16} + s_{37} + s_{62} + s_{82} + 1,\\
s_{82} &= s_4 + s_5 + s_8 + s_{13}s_{19}s_{25}s_{32}s_{37} + s_{13}s_{19} + s_{13} + s_{18}\\
&\quad + s_{25}s_{32}s_{37} + s_{25} + s_{32} + s_{35} + s_{37} + s_{54} + s_{72},\\
s_{83} &= s_{12} + s_{39} + s_{86} + s_{113} + s_{119} + s_{125} + s_{130}s_{134} + s_{130},\\
s_{95} &= s_{17} + s_{18} + s_{31} + s_{48} + s_{54} + s_{62},\\
s_{99} &= s_{22} + s_{25} + s_{119}s_{124} + s_{119} + s_{124}s_{126} + s_{124} + s_{126} + s_{131}\\
&\quad + s_{132} + 1,\\
s_{100} &= s_3 + s_6 + s_{70} + s_{72} + s_{73} + s_{77} + s_{86} + s_{113} + s_{119} + s_{121}s_{124}\\
&\quad + s_{121} + s_{124} + 1,\\
s_{102} &= s_5 + s_8 + s_{35} + s_{72} + s_{74} + s_{75} + s_{82} + s_{121} + s_{126}s_{130},\\
s_{104} &= s_7 + s_{10} + s_{31}s_{77} + s_{31} + s_{37} + s_{62} + s_{74} + s_{77} + s_{116} + s_{125} + 1,\\
s_{116} &= s_5 + s_6 + s_{14}s_{20} + s_{14} + s_{19} + s_{42} + s_{73} + s_{75} + s_{83},\\
s_{119} &= s_1 + s_4 + s_{31} + s_{68} + s_{70} + s_{98}.
\end{aligned}$$

D-QUARK

The IV bits needed to set as '0' are listed as follows:

$$s_{32}, s_{52}, s_{76}, s_{78}, s_{113}, s_{133}, s_{134}, s_{138}, s_{150}, s_{164}, s_{166}, s_{167}, s_{168}, s_{169}.$$

The IV bits needed to set as '1' are listed as follows:

$$s_{60}, s_{68}, s_{115}, s_{141}, s_{147}, s_{149}, s_{155}, s_{157}, s_{159}, s_{160}, s_{163}.$$

The other condition expressions are listed as follows:

$$
\begin{aligned}
s_0 = {} & s_1 + s_5 + s_{11}s_{19}s_{27}s_{36}s_{42} + s_{11}s_{19} + s_{11}s_{36}s_{58}s_{79} + s_{11} + s_{18} \\
& + s_{19}s_{27}s_{71}s_{79} + s_{27}s_{36}s_{42}s_{47}s_{58}s_{67} + s_{27}s_{36}s_{42} + s_{27} + s_{35}s_{57}s_{92} \\
& + s_{36} + s_{40} + s_{42}s_{47}s_{67}s_{71} + s_{42}s_{47} + s_{42} + s_{47}s_{58}s_{67}s_{71}s_{79} + s_{47} \\
& + s_{48} + s_{57}s_{92} + s_{57} + s_{58}s_{67}s_{71} + s_{58} + s_{64} + s_{67} + s_{71}s_{79} + s_{71} \\
& + s_{72} + s_{88} + s_{90} + s_{92} + s_{100} + s_{112} + s_{143}, \\
s_2 = {} & s_6 + s_{36}s_{58}s_{93} + s_{41} + s_{49} + s_{58}s_{69}s_{93} + s_{58}s_{69} + s_{69}s_{93} + s_{73} \\
& + s_{91} + s_{98}s_{108} + s_{98} + s_{101} + s_{109} + s_{114} + s_{127} + s_{136} \\
& + s_{143}s_{156}s_{158} + s_{143} + s_{144} + s_{152} + s_{156} + s_{158} + 1, \\
s_8 = {} & s_4 + s_{38}s_{95} + s_{38}s_{170} + s_{43} + s_{51} + s_{71}s_{170} + s_{71} + s_{75} + s_{91} \\
& + s_{93} + s_{95}s_{170} + s_{100}s_{110}s_{116}s_{129}s_{135} + s_{100}s_{110} + s_{100} + s_{103} \\
& + s_{111} + s_{116}s_{129}s_{135} + s_{116} + s_{129} + s_{135} + s_{145}s_{158} + s_{145} + s_{146} \\
& + s_{152} + s_{154} + s_{158} + s_{170} + 1, \\
s_9 = {} & s_{13} + s_{43}s_{65}s_{100} + s_{43}s_{65}s_{175} + s_{48} + s_{56} + s_{65}s_{100}s_{175} + s_{80} \\
& + s_{96} + s_{98} + s_{108} + s_{116} + s_{120} + s_{121}s_{165}s_{174} + s_{121} \\
& + s_{140}s_{143}s_{165} + s_{140}s_{143} + s_{140} + s_{143} + s_{151} + s_{165}s_{174} + s_{165} \\
& + s_{175} + 1, \\
s_{10} = {} & s_5 + s_6 + s_{16}s_{24} + s_{16}s_{41}s_{63}s_{84} + s_{16} + s_{23} + s_{40}s_{62}s_{97} \\
& + s_{40}s_{62}s_{172} + s_{41} + s_{45} + s_{47} + s_{53} + s_{62}s_{73}s_{97} + s_{62}s_{73} \\
& + s_{62}s_{97}s_{172} + s_{63} + s_{69} + s_{72} + s_{73}s_{97} + s_{73}s_{172} + s_{77} + s_{93} \\
& + s_{95} + s_{105} + s_{117} + s_{148} + s_{154} + s_{172} + 1, \\
s_{17} = {} & s_{13} + s_{47}s_{69}s_{104} + s_{47} + s_{69}s_{80}s_{104} + s_{69}s_{80} + s_{80}s_{104} + s_{84} + s_{100} \\
& + s_{102} + s_{109}s_{119} + s_{109} + s_{112} + s_{120} + s_{124} + s_{125} + s_{154} + s_{161}, \\
s_{21} = {} & s_{22} + s_{26} + s_{39} + s_{48}s_{57}s_{63} + s_{48} + s_{56} + s_{57} + s_{61} + s_{69} + s_{79} \\
& + s_{85} + s_{109} + s_{111} + s_{121} + s_{170}, \\
s_{44} = {} & s_4 + s_5 + s_9 + s_{15}s_{23}s_{31}s_{40}s_{46} + s_{15}s_{23} + s_{15}s_{40}s_{62}s_{83} + s_{15} \\
& + s_{22} + s_{23}s_{31}s_{75}s_{83} + s_{31}s_{40}s_{46}s_{51}s_{62}s_{71} + s_{31}s_{40}s_{46} + s_{31} \\
& + s_{39}s_{61}s_{96} + s_{39}s_{61}s_{171} + s_{40} + s_{46}s_{51}s_{71}s_{75} + s_{46}s_{51} + s_{46} \\
& + s_{51}s_{62}s_{71}s_{75}s_{83} + s_{51} + s_{61}s_{72}s_{96} + s_{61}s_{72} + s_{61}s_{96}s_{171} \\
& + s_{62}s_{71}s_{75} + s_{62} + s_{71} + s_{72}s_{96} + s_{72}s_{171} + s_{75}s_{83} + s_{75} + s_{92} \\
& + s_{94} + s_{104} + s_{116} + s_{153} + s_{171} + 1.
\end{aligned}
$$

C-QUARK

The IV bits needed to set as ’0’ are listed as follows:

$$s_{27}, s_{42}, s_{46}, s_{51}, s_{53}, s_{55}, s_{57}, s_{61}, s_{62}, s_{64}, s_{70}, s_{71}, s_{72}, s_{74}, s_{78}, s_{84}, s_{85},$$
$$s_{88}, s_{91}, s_{93}, s_{94}, s_{95}, s_{96}, s_{98}, s_{100}, s_{101}, s_{102}, s_{107}, s_{109}, s_{115}, s_{119}, s_{120},$$
$$s_{124}, s_{125}, s_{126}, s_{128}, s_{132}, s_{133}, s_{135}, s_{137}, s_{138}, s_{141}, s_{143}, s_{145}, s_{146}, s_{147},$$
$$s_{154}, s_{155}, s_{157}, s_{161}, s_{176}, s_{183}, s_{189}, s_{199}, s_{214}, s_{216}, s_{217}, s_{219}, s_{225}, s_{228},$$
$$s_{229}, s_{231}, s_{235}, s_{240}, s_{244}, s_{253}, s_{257}, s_{258}, s_{259}, s_{260}, s_{261}, s_{262}, s_{263}, s_{264},$$
$$s_{265}, s_{266}, s_{267}, s_{268}, s_{272}, s_{274}, s_{280}, s_{281}, s_{285}, s_{287}, s_{288}, s_{293}, s_{297}, s_{299},$$
$$s_{301}, s_{305}, s_{307}, s_{308}, s_{309}, s_{311}, s_{313}, s_{317}, s_{322}, s_{328}, s_{329}, s_{341}, s_{345}, s_{346},$$
$$s_{348}, s_{350}, s_{353}, s_{354}, s_{356}, s_{358}, s_{360}, s_{362}, s_{365}, s_{370}.$$

The IV bits needed to set as ’1’ are listed as follows:

$$s_{29}, s_{50}, s_{58}, s_{59}, s_{60}, s_{66}, s_{68}, s_{76}, s_{77},$$
$$s_{79}, s_{80}, s_{81}, s_{82}, s_{89}, s_{103}, s_{104},$$
$$s_{108}, s_{110}, s_{113}, s_{116}, s_{134}, s_{140},$$
$$s_{142}, s_{149}, s_{159}, s_{208}, s_{234}, s_{241}, s_{242},$$
$$s_{251}, s_{270}, s_{271}, s_{279}, s_{290}, s_{292}, s_{296},$$
$$s_{298}, s_{300}, s_{302}, s_{304}, s_{363}, s_{315}, s_{316},$$
$$s_{319}, s_{321}, s_{333}, s_{337}, s_{339}, s_{355},$$
$$s_{372}, s_{374}, s_{376}, s_{378}, s_{381}.$$

Acknowledgements
Not applicable.

Author contributions
Xiaojuan Lu proposed the improved CDA method and implemented the attacks in this paper. Xiaojuan Lu and Bohan Li drafted the manuscript. Dongdai Lin and Meicheng Liu inspired the work. Meicheng Liu offered some meaningful guidance and modifications. All of the authors took part in the discussion of the research and participated in the improvements of manuscript. All authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China (Grant No. 61872359, 62122085 and 61936008), the National Key R&D Program of China (Grant No. 2020YFB1805402), and the Youth Innovation Promotion Association of Chinese Academy of Sciences.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Institute of Information Engineering, Chinese Academy of Sciences, No. 65 Xingshikou Road, Haidian District, Beijing 100093, People’s Republic of China. [2]School of Cyber Security, University of Chinese Academy of Sciences, No. 65 Xingshikou Road, Haidian District, Beijing 100093, People’s Republic of China.

Received: 2 November 2021 Accepted: 27 December 2021
Published online: 02 June 2022

References

Aumasson J-P, Henzen L, Meier W, Naya-Plasencia M (2013) Quark: a lightweight hash. J Cryptol 26(2):313–339

Aumasson J-P. Github - veorq/quark: lightweight cryptographic hash functions (reference code). https://github.com/veorq/Quark/

Aumasson J-P, Henzen L, Meier W, Naya-Plasencia M (2010) Quark: a lightweight hash. In: International workshop on cryptographic hardware and embedded systems. Springer, pp 1–15

Aumasson J-P, Knellwolf S, Meier W (2012) Heavy Quark for secure AEAD. DIAC-Directions in Authenticated Ciphers

Banik S (2014) Some insights into differential cryptanalysis of Grain v1. In: Australasian conference on information security and privacy. Springer, pp 34–49

Blondeau C, Leander G, Nyberg K (2017) Differential-linear cryptanalysis revisited. J Cryptol 30(3):859–888

De Canniere C, Preneel B (2008) Trivium. In: New stream cipher designs. Springer, Heidelberg, pp 244–266

De Canniere C, Dunkelman O, Knežević M (2009) KATAN and KTANTAN-a family of small and efficient hardware-oriented block ciphers. In: International workshop on cryptographic hardware and embedded systems. Springer, pp 272–288

Dinur I, Shamir A (2009) Cube attacks on tweakable black box polynomials. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 278–299

Hell M, Johansso, T, Maximov A, Meier W (2008) The Grain family of stream ciphers. In: New stream cipher designs. Springer, Heidelberg, pp 179–190

Knellwolf S, Meier W, Naya-Plasencia M (2010) Conditional differential cryptanalysis of NLFSR-based cryptosystems. In: International conference on the theory and application of cryptology and information security. Springer, pp 130–145

Knellwolf S (2012) Cryptanalysis of hardware-oriented ciphers the knapsack generator, and sha-1. PhD thesis, ETH Zurich

Li J-Z, Guan J (2018) Advanced conditional differential attack on grain-like stream cipher and application on grain v1. IET Inf Secur 13(2):141–148

Li J-Z, Guan J (2018) Improved conditional differential attacks on round-reduced Grain v1. KSII Trans Internet Inf Syst (TIIS) 12(9):4548–4559

Liu M, Lu X, Lin D (2021) Differential-linear cryptanalysis from an algebraic perspective. In: Annual international cryptology conference. Springer, pp 247–277

Ma Z, Tian T, Qi W-F (2017) Improved conditional differential attacks on Grain v1. IET Inf Secur 11(1):46–53

Yang J, Liu M, Lin D, Wang W (2018) Symbolic-like computation and conditional differential cryptanalysis of quark. In: International workshop on security. Springer, pp 244–261

Zhang K, Guan J, Fei X (2015) Improved conditional differential cryptanalysis. Secur Commun Netw 8(9):1801–1811

Publisher’s Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.