

Automated Search for Periodic Functions in Quantum Attacks: Distinguishers and Key-Recovery Attacks

Qun Liu^{1,2,4}, Haoyang Wang³, Boyun Li^{1,2,4}, and Meiqin Wang(✉)^{1,2,4}

¹ School of Cyber Science and Technology, Shandong University, Qingdao, China
{qunliu,mqwang}@sdu.edu.cn

² State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao, China
boyunli@mail.sdu.edu.cn

³ Shanghai Jiao Tong University, Shanghai, China
haoyang.wang@sjtu.edu.cn

⁴ Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China

Abstract. Quantum attacks based on Simon’s algorithm rely on Boolean functions with hidden XOR periods. At ASIACRYPT 2025, Liu et al. introduced a symbolic automated search model for periodic functions used in distinguishers. However, for distinguishers on target ciphers, the search mainly works with truncated differences and does not jointly constrain exact differential propagation. A separate modeling gap arises in the key-recovery setting: the model does not cover the key-dependent periodic functions used in the polynomial-time key-recovery attacks presented by Canale et al. at CRYPTO 2022. In this work, we study automated search for two classes of periodic functions: periodic functions for distinguishers and key-dependent periodic functions for polynomial-time key-recovery attacks. For the former, we refine the symbolic search model of Liu et al. by adding fixed input differences, differential propagation, and probability constraints. Based on this model, we extend the periodic distinguishers of TWINE and LBlock from 10 rounds to 12 rounds, obtain a 12-round periodic distinguisher for LBlock-S, and find a full-round periodic distinguisher for ALLPC with verified probability $2^{-9.06}$. For key recovery, we adapt the symbolic-state tracking used for distinguishers to key-dependent values. The model tracks key-dependent shifts and checks that the periodic core is exposed through an output-computable expression. With this model, we extend the attack on MISTY L-FK from 5 to 6 rounds and obtain new polynomial-time key-recovery attacks on Type-1/2 GFS and Skipjack-B variants with FK or KF round functions.

Keywords: Quantum cryptanalysis · Simon’s algorithm · Periodic functions · Automated search · Distinguishers and key-recovery attacks

1 Introduction

Grover’s algorithm gives a quadratic speedup for unstructured key search [15], but some symmetric constructions additionally induce Boolean functions with hidden algebraic structure that quantum algorithms can exploit. In Simon-based cryptanalysis, the attacker constructs a vector-valued Boolean function from the cipher computation. If this function can be evaluated coherently under the attack model and satisfies Simon’s promise for a hidden nonzero XOR period, Simon’s algorithm [28] recovers the period with polynomially many oracle queries to this function. Depending on the construction, the recovered period can yield a quantum distinguisher or reveal key information. Kuwakado and Morii [22,23] gave a quantum distinguisher for 3-round Feistel and a key-recovery attack on Even–Mansour. Kaplan et al. [21] applied period-finding attacks to message authentication codes, authenticated-encryption modes, and slide attacks. Leander and May [24] combined Grover search with Simon’s algorithm to attack the FX construction. Bonnetain et al. [3] extended Simon-based cryptanalysis to hidden linear structures and developed quantum linearization attacks against parallelizable MACs. Further period-finding attacks were developed for Type-1 GFSSs and CAST-256 [26], and for reduced-round SMS4 [17]. These attacks depend both on the required hidden structure and on coherent evaluation of the corresponding function under the stated oracle-access model.

In the Q2 model, the adversary has coherent quantum-oracle access to the keyed construction and may query it in superposition. Under more restricted oracle access, Hosoyamada and Sasaki [19] developed quantum meet-in-the-middle attacks using online classical queries followed by offline quantum computation. Within period-finding cryptanalysis, Bonnetain et al. [2] introduced the offline Simon algorithm. Bonnetain, Schrottenloher, and Sibleyras [5] later used this approach to obtain a beyond-quadratic time speedup against the 2XOR-Cascade construction. Yu et al. [33] analyzed the data and quantum resources required by offline Simon. For suitable constructions, offline Simon exploits periodic structure without Q2 access to the keyed construction, with construction-dependent costs in classical data, offline quantum time, and memory.

One direction studies periodic functions used to distinguish symmetric constructions. Kuwakado and Morii [22] gave a 3-round Feistel distinguisher, while Ito et al. [20] gave a 4-round quantum chosen-ciphertext distinguisher for Feistel ciphers. Ni et al. [26] studied Type-1 GFSSs and CAST-256. Sun et al. [29] further analyzed Type-1 GFSSs. Dong et al. [13] studied Type-2 GFSSs, and Cui et al. [10] studied Skipjack-type ciphers. For Type-3 GFSSs, Hodzic et al. [18] identified separability conditions supporting quantum distinguishers. Xiang et al. [32] established links between Simon-based distinguishers and truncated differentials. Liu et al. [25] introduced a symbolic SMT model for automated search. They applied it to probabilistic periodic distinguishers for TWINE [30] and LBlock [31]. The model propagates symbolic states and uses truncated-difference collision constraints to bound a symbolic probability estimate. It first returns a symbolic candidate and instantiates exact differences and DDT transitions only afterward, so exact-difference feasibility and DDT-based probabilities do not constrain the

symbolic solve. Without exact differences, the search enforces period consistency by allowing only one period-generating transition.

A second direction studies quantum key-recovery attacks under different oracle-access assumptions and using different quantum subroutines. Existing results include Simon-based period-finding attacks [21], Grover-meets-Simon attacks [24], Q1 attacks based on Grover search [12], and offline-Simon attacks [1]. This paper focuses on polynomial-time Q2 attacks using vector-valued Boolean functions whose nonzero XOR periods depend on secret subkeys. When Simon’s promise holds, period finding recovers a subkey or a relation among subkeys with polynomially many coherent queries to the keyed construction. The remaining algebraic steps and, when needed, nested period-recovery steps [4,9] also have polynomial overhead in Q2. Canale et al. [6] automated the search for key-dependent periodic functions by representing candidates as circuits, enumerating circuits up to a gate bound, and testing their periodicity for small input sizes. Their procedure found polynomial-time key-recovery attacks on Feistel-FK and MISTY L/R-FK constructions, but circuit enumeration becomes difficult to scale as the target computation grows. Liu et al. [25] instead developed symbolic-state propagation for periodic distinguishers, but their states do not represent key variables or relations among subkeys. Our setting therefore requires symbolic propagation of key-dependent expressions through FK/KF computations and a check that the resulting periodic function can be computed from public inputs and output branches.

1.1 Our Contributions

We develop separate symbolic SMT search models for periodic distinguishers and key-dependent periodic functions used in polynomial-time Q2 key-recovery attacks. Both models propagate symbolic states, but use different state representations and constraints for the two search targets.

For small output branches, such as the 4-bit branches of TWINE, the additional Simon evaluations required within the Grover-meets-Simon procedure can offset the advantage of recovering an unknown period. We therefore use the candidate period returned by the SMT search as a prescribed period and test the corresponding quartet relation with classical oracle queries. To search for such relations, we extend the symbolic model of Liu et al. [25] with exact-difference variables, DDT constraints, and constraints that link every accepted period-generating transition to a common prescribed period. The solver jointly determines the symbolic propagation, an exact-differential path, the prescribed period, and the corresponding DDT path weight in one SMT instance. Each candidate is then evaluated by the prescribed-period quartet test. Our model finds 12-round periodic distinguishers for TWINE and LBlock, extending the previous 10-round results, and yields a 12-round periodic distinguisher for LBlock-S and a full-round periodic distinguisher for ALLPC. For full-round ALLPC, the quartet relation has probability $2^{-9.06}$, compared with 2^{-24} for the existing full-round differential distinguisher [11] and $2^{-16.07}$ for the existing full-round boomerang

distinguisher [35]. Table 1 compares the rounds and probabilities of periodic distinguishers, and Table 6 gives the corresponding time-complexity comparison.

For polynomial-time key recovery in Q2, we introduce a symbolic SMT search model for a class of key-dependent periodic functions that includes the Feistel and MISTY constructions of Canale et al. [6]. Given fixed round counts for the input, core, and output parts, the model tracks x -independent key-dependent values and returns a key-dependent period expression together with a function having that period. Its constraints check that the required shifted core is generated by the actual FK/KF computation and that the function can be computed from public inputs and the output branches of the queried oracle. We instantiate the returned expressions and check their claimed periods experimentally. The resulting procedures combine period recovery with round peeling to recover all round keys. The model extends the MISTY L-FK attack from 5 to 6 rounds and yields new 5-round attacks on MISTY L-KF and MISTY R-KF. It also gives 17-round FK and 13-round KF attacks on Type-1 GFS under qCCA access, together with 7-round FK and 5-round KF attacks on Type-2 GFS and 17-round FK and 14-round KF attacks on Skipjack-B under qCPA access.

Table 1. Rounds and probabilities of periodic distinguishers.

Cipher	#Rounds	Probability	Reference
TWINE	8	1	[7]
	9	2^{-2}	[25]
	10	2^{-8}	[25]
	11*	$2^{-13.98}$	This work
	12*	$2^{-23.12}$	This work
LBlock	8	1	[32]
	9	2^{-2}	[25]
	10	2^{-8}	[25]
	11*	$2^{-14.02}$	This work
	12*	$2^{-22.77}$	This work
LBlock-S	10*	$2^{-8.00}$	This work
	11*	$2^{-13.97}$	This work
	12*	$2^{-21.34}$	This work
ALLPC	25* (full)	$2^{-9.06}$	This work

* Prescribed-period distinguishers obtained in this work. Probabilities are experimental estimates for the public keyless structural round functions.

1.2 Organization

Section 2 introduces the notation, quantum preliminaries, and two classes of periodic functions. Sections 3 and 4 develop the respective search models for prescribed-period distinguishers and key-dependent periodic functions. Section 5 presents the applications and results, and Section 6 concludes the paper.

2 Preliminaries

2.1 Notation

The following notation is used for an r -round structure with d branches.

- For $0 \leq i \leq r$, let $x^i = (x_0^i, x_1^i, \dots, x_{d-1}^i)$ denote the state after i rounds, with x_j^i its j -th branch for $0 \leq j < d$. The input and output states are x^0 and x^r . For two-branch structures, write $x^i = (x_L^i, x_R^i)$.
- In the distinguisher search model and its round-state tables, X^i and Y^i denote the input and internal output states of round i for $1 \leq i \leq r$. Their j -th branches are $X^i[j]$ and $Y^i[j]$ for $0 \leq j < d$.
- F denotes a public function. Whenever F^{-1} appears, F is bijective and F^{-1} denotes its inverse. For $t \geq 1$, F^t and F^{-t} denote the t -fold iterations of F and F^{-1} , respectively.
- For $0 \leq i < r$, k_j^i denotes the j -th round key of round i . If round i uses one key, it is denoted by k^i .
- E^r denotes the r -round structure, and E_j^r denotes its j -th output branch for $0 \leq j < d$. For two-branch structures, the output branches are denoted by E_L^r and E_R^r .

2.2 Quantum Preliminaries

Quantum access. Let $h : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ be a classical function. Coherent access to h is represented by the unitary oracle

$$U_h(|x\rangle|z\rangle) = |x\rangle|z \oplus h(x)\rangle,$$

where $x \in \{0, 1\}^n$ and $z \in \{0, 1\}^\ell$. In the Q2 model, the adversary may query the keyed target construction in superposition, while Q1 permits only classical queries. We write qCPA for quantum access to the encryption oracle and qCCA for quantum access to both the encryption and decryption oracles. Any function used by Simon's algorithm must be implemented coherently using only the permitted target oracle and efficient public reversible computations.

Simon's algorithm. Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ satisfy Simon's promise [28] for a nonzero $s \in \{0, 1\}^n$. For all $x, x' \in \{0, 1\}^n$, $f(x) = f(x')$ if and only if $x' = x$ or $x' = x \oplus s$. One Simon sample uses one query to U_f and returns a uniformly random vector $y \in \{0, 1\}^n$ satisfying $y \cdot s = 0$, where $\cdot$ denotes the inner product over $\mathbb{F}_2$. $O(n)$ independent samples determine s with high probability by solving the resulting linear system.

Grover-meets-Simon. Let $\kappa \in \{0, 1\}^\mu$ index the candidate parameters, and let f_κ be the corresponding function. The Grover-meets-Simon technique of Leander and May [24] applies amplitude amplification over κ using a coherent Simon-type periodicity test on f_κ as the predicate. The test uses parallel Simon circuits with measurements deferred until the end. The outer search requires $O(2^{\mu/2})$ predicate evaluations, each incurring the polynomial cost of the periodicity test.

2.3 Periodic Functions for Distinguishers

In the distinguisher setting, a periodic function computed from the target construction is used to distinguish it from a random permutation. We recall the 3-round Feistel distinguisher of Kuwakado and Morii [22].

Let $f^0, f^1, f^2 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be independent permutations defining a 3-round Feistel structure E^3 , with round update

$$\begin{cases} x_L^{i+1} = x_R^i, \\ x_R^{i+1} = x_L^i \oplus f^i(x_R^i), \end{cases} \quad \text{for } i = 0, 1, 2.$$

Let $\mathcal{O}$ be either E^3 or a uniformly random permutation on $\{0, 1\}^{2n}$, and let $\mathcal{O}_L$ denote its left output branch. Fix distinct $\alpha_0, \alpha_1 \in \{0, 1\}^n$ and define $h_{\mathcal{O}} : \{0, 1\}^{n+1} \rightarrow \{0, 1\}^n$ by

$$h_{\mathcal{O}}(b, x) = \mathcal{O}_L(x, \alpha_b) \oplus \alpha_b, \quad b \in \{0, 1\}, \quad x \in \{0, 1\}^n.$$

For $\mathcal{O} = E^3$, expanding the rounds gives

$$h_{E^3}(b, x) = f^1(x \oplus f^0(\alpha_b)).$$

Set $\delta = f^0(\alpha_0) \oplus f^0(\alpha_1)$. Then

$$h_{E^3}(b, x) = h_{E^3}(b \oplus 1, x \oplus \delta).$$

Since f^1 is a permutation, these are the only collisions. Hence h_{E^3} satisfies Simon's promise with the nonzero period $s_{\text{sel}} = 1 \parallel \delta$, where $\parallel$ denotes concatenation. The resulting distinguisher uses $O(n)$ quantum queries and has error probability approximately 2^{-n} .

For an r -round structure E^r , parameterize its input through a public input mapping by

$$(x, \alpha_b, c), \quad b \in \{0, 1\},$$

where x varies, $\alpha_b \in \{\alpha_0, \alpha_1\}$ selects one of two distinct fixed values, and c is fixed. The x - and α_b -components may each contain multiple input branches.

Definition 1 (Periodic function). Let $g : \{0, 1\}^m \rightarrow \{0, 1\}^\ell$. A nonzero vector $s \in \{0, 1\}^m$ is an XOR period of g if $g(x) = g(x \oplus s)$ for every $x \in \{0, 1\}^m$. A function with such a period is periodic.

Liu et al. [25] used the following criterion to construct a periodic function from a truncated output branch.

Theorem 1 (Periodic-function criterion). Fix c and two distinct constants α_0 and α_1 . Suppose that, for some ℓ -bit output branch j , there exist functions G, H, P, Q such that, for every $x \in \{0, 1\}^m$ and $b \in \{0, 1\}$,

$$E_j^r(x, \alpha_b, c) = H(x \oplus G(\alpha_b, c), c) \oplus P(x, c) \oplus Q(\alpha_b, c), \quad (1)$$

where $G(\alpha_b, c) \in \{0, 1\}^m$, and H, P, Q have ℓ -bit outputs. Define $g : \{0, 1\}^m \rightarrow \{0, 1\}^\ell$ by

$$g(x) = E_j^r(x, \alpha_0, c) \oplus E_j^r(x, \alpha_1, c).$$

Let $s = G(\alpha_0, c) \oplus G(\alpha_1, c)$. If $s \neq 0$, then s is an XOR period of g .

Proof. Write $G_b = G(\alpha_b, c)$ and $Q_b = Q(\alpha_b, c)$. After the two $P(x, c)$ terms cancel,

$$g(x) = H(x \oplus G_0, c) \oplus H(x \oplus G_1, c) \oplus Q_0 \oplus Q_1.$$

Replacing x by $x \oplus s$, where $s = G_0 \oplus G_1$, exchanges the two inputs to H . Therefore,

$$g(x \oplus s) = H(x \oplus G_1, c) \oplus H(x \oplus G_0, c) \oplus Q_0 \oplus Q_1 = g(x).$$

□

For the 3-round Feistel structure, the theorem applies with c empty, $j = L$, $H = f^1$, $G(\alpha_b) = f^0(\alpha_b)$, $P = 0$, and $Q(\alpha_b) = \alpha_b$. The corresponding XOR function is

$$g(x) = E_L^3(x, \alpha_0) \oplus E_L^3(x, \alpha_1).$$

It has the nonzero period $s = f^0(\alpha_0) \oplus f^0(\alpha_1)$.

2.4 Key-Dependent Periodic Functions for Polynomial-Time Key-Recovery Attacks

We next consider key-dependent periodic functions, whose hidden periods contain secret-subkey information. Under Simon's promise, each period is recovered with $O(n)$ queries. The 5-round MISTY L-FK attack of Canale et al. [6] illustrates how successive period recovery yields all round keys in polynomial time.

Let $x^i = (x_L^i, x_R^i) \in \{0, 1\}^{2n}$, let $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a public permutation, and let $k^0, \dots, k^4 \in \{0, 1\}^n$ be the round keys. For $i = 0, \dots, 4$, the state is updated as

$$\begin{cases} x_L^{i+1} = x_R^i, \\ x_R^{i+1} = F(x_L^i) \oplus k^i \oplus x_R^i. \end{cases}$$

Canale et al. constructed the function

$$\begin{aligned} g(x) &= E_L^5(F^{-1}(F^{-1}(x) \oplus x), F^{-1}(x)) \oplus F(x) \\ &= F(x \oplus k^0) \oplus F(x) \oplus \eta_K, \end{aligned}$$

where $K = (k^0, \dots, k^4)$ and

$$\eta_K = F(k^0 \oplus k^1) \oplus k^0 \oplus k^1 \oplus k^2 \oplus k^3$$

is independent of x . Hence $g(x \oplus k^0) = g(x)$, so k^0 is an XOR period of g and is recovered by Simon's algorithm.

After recovering k^0 , the first round is efficiently inverted using the public permutation F and the known round key. A public zero-key round is then appended to the output, giving an effective 5-round oracle $\hat{E}^5$ with round keys $k^1, \dots, k^4, 0$. Applying the same construction recovers k^1 , and repeating this procedure recovers $k^2, \dots, k^4$. Thus the complete attack uses $O(n)$ oracle queries and polynomial time.

3 Differential-Based Periodic-Function Search for Distinguishers

This section develops an automated search for prescribed-period distinguishers. After defining the quartet test, we review the symbolic model of Liu et al. [25], which instantiates exact differences after finding a candidate and allows only one period-generating transition. Our SMT model jointly encodes symbolic states, exact differences, DDT-based probability constraints, and prescribed-period matching, extending the periodic distinguishers for TWINE and LBlock from 10 to 12 rounds.

3.1 Prescribed-Period Distinguishers

We express a probabilistic periodic distinguisher as a quartet relation on a selected output branch. Throughout this subsection, $x \in \{0, 1\}^n$, and the selected output branch is also n bits. This quartet relation is the target of the search models below.

Definition 2 (Probabilistic periodic distinguisher). *Fix a nonzero input difference δ_i . Choose α_0 and c uniformly, set $\alpha_1 = \alpha_0 \oplus \delta_i$, and let $g_{\alpha_0, c} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be the function constructed from a truncated output branch of E^r for the resulting auxiliary tuple (α_0, α_1, c) . For a nonzero $s \in \{0, 1\}^n$, define $\rho_s = \Pr_{\alpha_0, c}[g_{\alpha_0, c}(x) = g_{\alpha_0, c}(x \oplus s)]$ for every $x \in \{0, 1\}^n$. We call this family a probabilistic periodic distinguisher with period s and probability ρ_s .*

Consider an output branch j . For a fixed tuple (α_0, α_1, c) , define

$$g(x) = E_j^r(x, \alpha_0, c) \oplus E_j^r(x, \alpha_1, c).$$

If $s \neq 0$ is a period of g , then $g(x) = g(x \oplus s)$ for every x , or equivalently,

$$E_j^r(x, \alpha_0, c) \oplus E_j^r(x, \alpha_1, c) \oplus E_j^r(x \oplus s, \alpha_0, c) \oplus E_j^r(x \oplus s, \alpha_1, c) = 0.$$

Let $\delta_i = \alpha_0 \oplus \alpha_1$. Since x , α_b , and c vary by s , δ_i , and 0, respectively, we write the corresponding relation as

$$(s, \delta_i, 0) \xrightarrow{2^{-p}} (0, ?, \dots, ?), \quad (2)$$

where the output 0 refers to branch j , while ? leaves the other output branches unrestricted. The annotation 2^{-p} denotes the probability, over the choices of α_0 and c with $\alpha_1 = \alpha_0 \oplus \delta_i$, that the relation holds for every x . The quartet test below checks this relation for a chosen tuple.

When the period is not fixed in advance, such a distinguisher can be used within the Grover-meets-Simon algorithm [24]. Each Grover predicate evaluation then requires several parallel Simon circuits to recover and test the hidden period. For a small branch, such as a 4-bit branch of TWINE, this coherent period-recovery cost can be comparable to direct evaluation over the small domain and may offset the concrete advantage of Simon's algorithm.

Prescribed-period distinguisher. We therefore prescribe the period s and the input difference δ_i , and test the corresponding quartet relation directly on output branch j . This test uses only classical oracle queries. Let N be the number of candidate tuples tested and $1 \leq q \leq 2^{n-1}$ the number of test points for each tuple. Let $\mathcal{O}$ denote either E^r or a random permutation, and let $\mathcal{O}_j$ denote output branch j . For uniformly chosen α_0 and c , set $\alpha_1 = \alpha_0 \oplus \delta_i$ and define

$$Q_j^{\mathcal{O}}(x) = \mathcal{O}_j(x, \alpha_0, c) \oplus \mathcal{O}_j(x, \alpha_1, c) \oplus \mathcal{O}_j(x \oplus s, \alpha_0, c) \oplus \mathcal{O}_j(x \oplus s, \alpha_1, c).$$

Because $Q_j^{\mathcal{O}}(x) = Q_j^{\mathcal{O}}(x \oplus s)$, the domain splits into 2^{n-1} cosets of $\{0, s\}$, and one representative from each coset suffices. We choose $q \leq 2^{n-1}$ distinct coset representatives and test whether $Q_j^{\mathcal{O}}(x) = 0$ for all of them. The distinguisher returns **real** if at least one of the N candidate tuples passes every test, and returns **random** otherwise.

Algorithm 1 Prescribed-Period Distinguisher

Require: An oracle $\mathcal{O}$, a prescribed-period relation $(s, \delta_i, 0) \xrightarrow{2^{-p}} (0, ?, \dots, ?)$ with $s \neq 0$ and $\delta_i \neq 0$, a target branch j , a number N of candidate tuples, and an integer $1 \leq q \leq 2^{n-1}$

Ensure: **real** or **random**

- 1: **for** $t = 1$ to N **do**
 - 2: Choose α_0 uniformly at random from the domain of α_b
 - 3: Set $\alpha_1 \leftarrow \alpha_0 \oplus \delta_i$ and choose c uniformly at random from the domain of c
 - 4: Define $Q_j^{\mathcal{O}}(x) \leftarrow \mathcal{O}_j(x, \alpha_0, c) \oplus \mathcal{O}_j(x, \alpha_1, c) \oplus \mathcal{O}_j(x \oplus s, \alpha_0, c) \oplus \mathcal{O}_j(x \oplus s, \alpha_1, c)$
 - 5: Choose q representatives from distinct cosets of $\{0, s\}$
 - 6: **if** $Q_j^{\mathcal{O}}(x) = 0$ for every chosen representative x **then**
 - 7: **return real**
 - 8: **end if**
 - 9: **end for**
 - 10: **return random**
-

Each candidate tuple requires at most $4q$ oracle calls, so testing N tuples requires at most $4Nq$ calls. Let ρ be the probability that a sampled tuple satisfies the quartet relation. Choosing $N \approx \rho^{-1}$ is sufficient for constant success probability on E^r . Under the approximation that the selected n -bit outputs of a random permutation are independent, the union bound gives a false-positive probability at most $N2^{-nq}$. Hence, for $\rho = 2^{-p}$, the query complexity is approximately $4q2^p$. For TWINE, $n = 4$, and full-domain verification uses $q = 8$, giving approximately 2^{p+5} oracle calls. All periodic distinguishers considered in the following applications are represented by relations of the form (2) and evaluated by the prescribed-period quartet test.

3.2 Symbolic Model and Its Limits

We recall the symbolic search model of Liu et al. [25] and its limitations for the prescribed-period relations defined above. The model propagates symbolic XOR expressions using truncated rather than exact difference information. Each branch is represented by a 6-bit state, as listed in Table 2. An encoding whose leftmost bit is 1 denotes the untracked state ?. When this bit is 0, the remaining five bits record the XOR combination of the basic symbolic terms.

Table 2. Basic symbols and their 6-bit encodings

Symbol	Encoding	Meaning
0	000000	zero symbolic difference
δ	000001	marker for the fixed input difference $\alpha_0 \oplus \alpha_1$
$\mathbf{x}$	000100	marker for dependence on the variable input x
$\mathbf{R}(\mathbf{x})$	001000	symbolic term depending on the variable input, e.g., $P(x, c)$
$\mathbf{R}(\delta)$	000010	symbolic term depending only on the selected constants, e.g., $Q(\alpha_b, c)$
0_s	010000	separable shifted term of the form $H(x \oplus G(\alpha_b, c), c)$
?	1*****	untracked state in which the remaining five bits are ignored

Composite states are obtained by XORing the encodings of their basic symbols. For example,

$$\begin{aligned} \mathbf{x} \oplus \delta &= 000101, & \mathbf{x} \oplus \mathbf{R}(\delta) &= 000110, \\ \mathbf{x} \oplus \mathbf{R}(\delta) \oplus \delta &= 000111, & \mathbf{R}(\mathbf{x}) \oplus \mathbf{x} &= 001100. \end{aligned}$$

Under the symbolic propagation rules, applying $\mathbf{R}$ to $\mathbf{x} \oplus \delta$ may yield 0_s , while the $\mathbf{R}(\delta)$ component is propagated separately from the x -dependent components.

Symbolic propagation rules. To encode the conditions of Theorem 1, the symbolic model uses four constraint groups $\mathcal{C}_{\text{input}}$, $\mathcal{C}_{\text{prop}}$, $\mathcal{C}_{\text{output}}$, and $\mathcal{C}_{\text{coll}}$. The constraint $\mathcal{C}_{\text{input}}$ fixes the round-0 symbolic pattern, while $\mathcal{C}_{\text{output}}$ imposes the symbolic output condition. The constraint $\mathcal{C}_{\text{prop}}$ propagates symbolic states through the round wiring using three operations. Here **COPY** duplicates a state, **R** applies the round function, and **XOR** combines two states bitwise. Finally, $\mathcal{C}_{\text{coll}}$ accounts for truncated-difference collisions and bounds the resulting probability estimate. The symbolic model is therefore

$$\mathcal{M}_{\text{sym}} := \mathcal{C}_{\text{input}} \wedge \mathcal{C}_{\text{prop}} \wedge \mathcal{C}_{\text{output}} \wedge \mathcal{C}_{\text{coll}}.$$

The use of truncated differences leads to two limitations when the model is applied to ciphers such as TWINE and LBlock.

Limitation 1. The first limitation concerns the creation of 0_s . Several branches may contain terms of the form $H(x \oplus G(\alpha_b, c), c)$, but the symbolic model records only truncated differences and cannot determine whether the exact differences at these transitions match the same period.

Limitation 2. The second limitation concerns ciphers with S-boxes. Since the symbolic search keeps only truncated differences, it does not record the exact input and output differences at each S-box. The symbolic model may therefore overestimate the differential weight of a candidate. To address this issue, Liu et al. [25] use a two-step procedure.

- Solve $\mathcal{M}_{\text{sym}}$ and extract a symbolic truncated-difference candidate.
- Instantiate exact differences for this candidate and check the S-box transitions with the DDT.

The second step may find an exact instantiation with a lower differential weight than the symbolic estimate. However, exact differences are introduced only after the symbolic candidate has been fixed and therefore cannot guide the search for symbolic candidates.

3.3 Differential-Based Periodic Distinguisher Search

Both limitations arise because exact differences are introduced only after the symbolic search. We retain the symbolic propagation rules and introduce exact-difference variables in the same SMT instance. The linking constraints require every selected period-generating transition to match the prescribed period, while the DDT constraints exclude infeasible S-box transitions and accumulate their differential weights during the solve.

Model overview. Let $\mathbf{s}$ denote the symbolic states, and let Δ denote exact differences attached to branches and internal nodes. Let Δ_{per} denote the common period difference determined by the solver and subsequently used in the prescribed-period quartet test. Every accepted period-generating transition must match this value. The variable p denotes the accumulated DDT probability exponent, and τ is its upper bound. The model is

$$\mathcal{M}_{\text{diff}} := \mathcal{C}_{\text{input}} \wedge \mathcal{C}_{\text{prop}} \wedge \mathcal{C}_{\text{output}} \wedge \mathcal{C}_{\text{diff}}(\mathbf{s}, \Delta) \wedge \mathcal{C}_{\text{DDT}}(\mathbf{s}, \Delta, p, \tau) \wedge \mathcal{C}_{\text{link}}(\mathbf{s}, \Delta, \Delta_{\text{per}}).$$

The differential-based model retains the three symbolic constraint groups $\mathcal{C}_{\text{input}}$, $\mathcal{C}_{\text{prop}}$, and $\mathcal{C}_{\text{output}}$. It omits the symbolic term $\mathbf{R}(\delta)$, since the corresponding fixed-difference behavior is represented by exact-difference variables together with the linking constraints. It also removes $\mathcal{C}_{\text{coll}}$, which estimates probability from truncated-difference collisions. The three added constraint groups have the following roles.

- $\mathcal{C}_{\text{diff}}(\mathbf{s}, \Delta)$ associates an exact-difference variable with each branch and internal node. It propagates exact differences along wires and through COPY, and

through XOR whenever its symbolic output is tracked. Values at untracked or difference-free states are set to zero, and the exact input-output pair at each call to $\mathbf{R}$ is supplied to $\mathcal{C}_{\text{DDT}}$.

- $\mathcal{C}_{\text{DDT}}(\mathbf{s}, \Delta, p, \tau)$ replaces the truncated-collision estimate of $\mathcal{C}_{\text{coll}}$ with exact DDT checks for calls to $\mathbf{R}$ whose input state is δ . For each such call, it selects the S-box DDT entry using the exact input-output differences supplied by $\mathcal{C}_{\text{diff}}$. A zero entry makes the candidate infeasible, whereas a nonzero entry contributes to p . Only candidates satisfying $p \leq \tau$ are retained. Other symbolic inputs make no DDT contribution.
- $\mathcal{C}_{\text{link}}(\mathbf{s}, \Delta, \Delta_{\text{per}})$ handles the two input states $\mathbf{x} \oplus \delta$ and $\mathbf{0}_s \oplus \mathbf{x} \oplus \delta$ at calls to $\mathbf{R}$. These states describe shifted inputs but do not record the exact difference between the two shifts. The constraint outputs $\mathbf{0}_s$ only when the exact input difference recorded in Δ equals Δ_{per} ; otherwise, the output is set to ?. Thus, every newly created $\mathbf{0}_s$ is linked to the same Δ_{per} , which is determined by the solver.

Algorithm 2 summarizes the joint search. The solver determines the symbolic states, exact differences, the period difference Δ_{per} , the DDT probability exponent p , and the periodic output branch in one SMT constraint system. Each solution therefore specifies a periodic distinguisher without a separate exact-difference instantiation step. The 12-round TWINE distinguisher below illustrates the three added constraint groups. Supporting Material B gives its full SMT encoding.

Algorithm 2 Differential-Based Joint Search

Require: A target structure, a number of rounds, and a DDT probability-exponent bound τ

Ensure: A periodic distinguisher or **NoFeasibleCandidate**

- 1: Declare the symbolic states $\mathbf{s}$, exact differences Δ , prescribed period Δ_{per} , and DDT probability exponent p
 - 2: Add the inherited symbolic constraints $\mathcal{C}_{\text{input}}$, $\mathcal{C}_{\text{prop}}$, and $\mathcal{C}_{\text{output}}$
 - 3: Add $\mathcal{C}_{\text{diff}}(\mathbf{s}, \Delta)$ for exact-difference propagation
 - 4: Add $\mathcal{C}_{\text{DDT}}(\mathbf{s}, \Delta, p, \tau)$ for probability constraints and the condition $p \leq \tau$
 - 5: Add $\mathcal{C}_{\text{link}}(\mathbf{s}, \Delta, \Delta_{\text{per}})$ for prescribed-period matching inside $\mathbf{R}$
 - 6: Solve $\mathcal{M}_{\text{diff}}$
 - 7: **if** $\mathcal{M}_{\text{diff}}$ is satisfiable **then**
 - 8: **return** $\mathbf{s}$, Δ , Δ_{per} , p , and the periodic output branch
 - 9: **end if**
 - 10: **return** **NoFeasibleCandidate**
-

Example 1. Figure 1 shows the propagation path for the 12-round TWINE example, and Table 3 lists the corresponding round states. TWINE has 16 branches of 4 bits each. In the initial state X^1 , branches 3, 5, 7, 8, 12, 13, and 15 carry δ , while branch 9 carries $\mathbf{x}$. Let α_b consist of the inputs on these seven δ branches

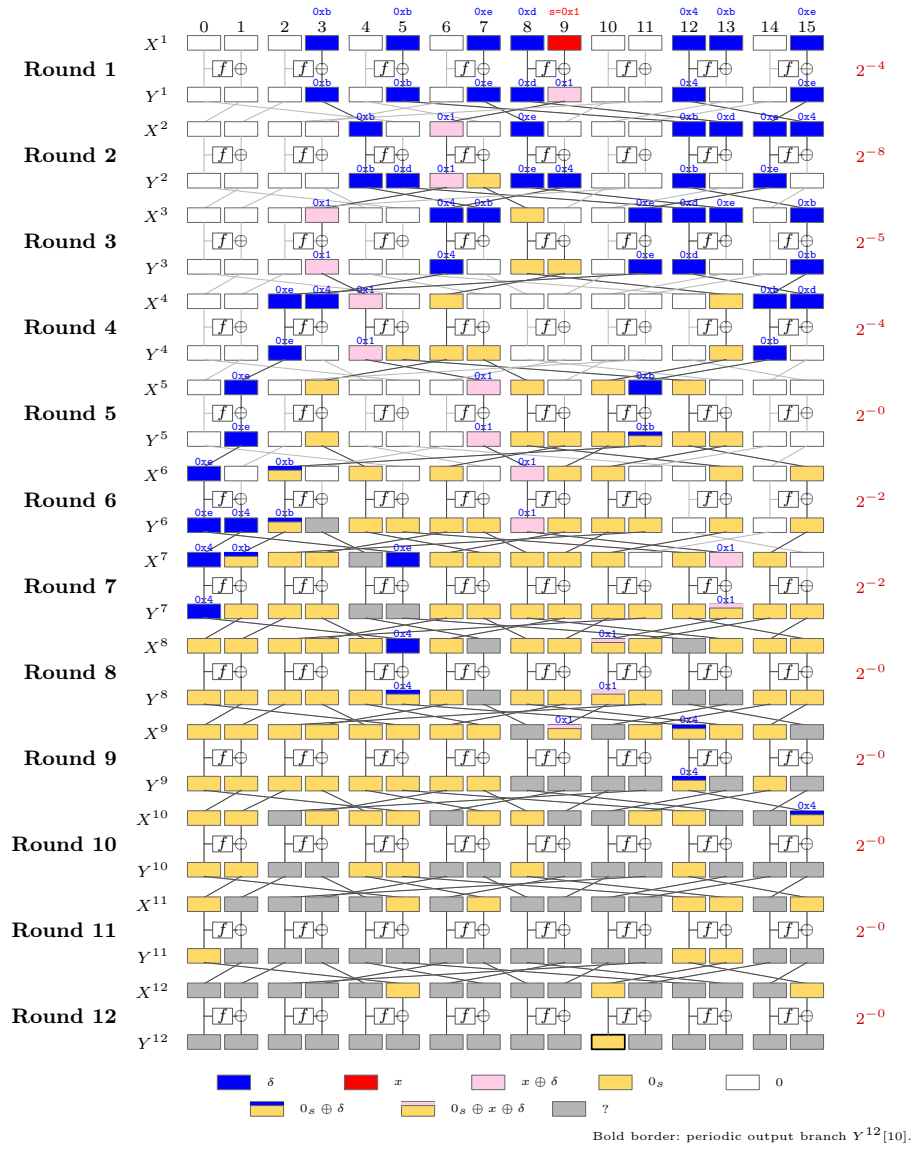


Fig. 1. 12-round periodic distinguisher for TWINE. Each box f denotes the 4-bit TWINE S-box.

in the order listed above. The corresponding input difference is

$$\alpha_0 \oplus \alpha_1 = (0xb, 0xb, 0xe, 0xd, 0x4, 0xb, 0xe).$$

The solver returns $\Delta_{\text{per}} = 0x1$, $p = 25$, and the output condition $Y^{12}[10] = 0_s$. We omit the final branch permutation and regard Y^{12} as the output of the reduced-round structure. Thus, E_{10}^{12} denotes branch $Y^{12}[10]$, and the corresponding function is

$$g(x) = E_{10}^{12}(x, \alpha_0, c) \oplus E_{10}^{12}(x, \alpha_1, c),$$

and the prescribed-period relation is

$$g(x) = g(x \oplus 0x1)$$

for every x . The differential-based model thus reaches 12 rounds for TWINE, compared with 10 rounds for the symbolic model of Liu et al. [25].

Table 3. Round states of the 12-round TWINE distinguisher.

Round i	Nonzero entries of X^i
1	3 : $\delta(0xb)$, 5 : $\delta(0xb)$, 7 : $\delta(0xe)$, 8 : $\delta(0xd)$, 9 : $x(0x0)$, 12 : $\delta(0x4)$, 13 : $\delta(0xb)$, 15 : $\delta(0xe)$
2	4 : $\delta(0xb)$, 6 : $x \oplus \delta(0x1)$, 8 : $\delta(0xe)$, 12 : $\delta(0xb)$, 13 : $\delta(0xd)$, 14 : $\delta(0xe)$, 15 : $\delta(0x4)$
3	3 : $x \oplus \delta(0x1)$, 6 : $\delta(0x4)$, 7 : $\delta(0xb)$, 8 : $0_s(0x0)$, 11 : $\delta(0xe)$, 12 : $\delta(0xd)$, 13 : $\delta(0xe)$, 15 : $\delta(0xb)$
4	2 : $\delta(0xe)$, 3 : $\delta(0x4)$, 4 : $x \oplus \delta(0x1)$, 6 : $0_s(0x0)$, 13 : $0_s(0x0)$, 14 : $\delta(0xb)$, 15 : $\delta(0xd)$
5	1 : $\delta(0xe)$, 3 : $0_s(0x0)$, 7 : $x \oplus \delta(0x1)$, 8 : $0_s(0x0)$, 10 : $0_s(0x0)$, 11 : $\delta(0xb)$, 12 : $0_s(0x0)$
6	0 : $\delta(0xe)$, 2 : $0_s \oplus \delta(0xb)$, 4 : $0_s(0x0)$, 6 : $0_s(0x0)$, 8 : $x \oplus \delta(0x1)$, 9 : $0_s(0x0)$, 10 : $0_s(0x0)$, 13 : $0_s(0x0)$, 15 : $0_s(0x0)$
7	0 : $\delta(0x4)$, 1 : $0_s \oplus \delta(0xb)$, 2 : $0_s(0x0)$, 3 : $0_s(0x0)$, 4 : $?(0x0)$, 5 : $\delta(0xe)$, 6 : $0_s(0x0)$, 7 : $0_s(0x0)$, 8 : $0_s(0x0)$, 9 : $0_s(0x0)$, 10 : $0_s(0x0)$, 12 : $0_s(0x0)$, 13 : $x \oplus \delta(0x1)$, 14 : $0_s(0x0)$
8	0 : $0_s(0x0)$, 1 : $0_s(0x0)$, 2 : $0_s(0x0)$, 3 : $0_s(0x0)$, 4 : $0_s(0x0)$, 5 : $\delta(0x4)$, 6 : $0_s(0x0)$, 7 : $?(0x0)$, 8 : $0_s(0x0)$, 9 : $0_s(0x0)$, 10 : $0_s \oplus x \oplus \delta(0x1)$, 11 : $0_s(0x0)$, 12 : $?(0x0)$, 13 : $0_s(0x0)$, 14 : $0_s(0x0)$, 15 : $0_s(0x0)$
9	0 : $0_s(0x0)$, 1 : $0_s(0x0)$, 2 : $0_s(0x0)$, 3 : $0_s(0x0)$, 4 : $0_s(0x0)$, 5 : $0_s(0x0)$, 6 : $0_s(0x0)$, 7 : $0_s(0x0)$, 8 : $?(0x0)$, 9 : $0_s \oplus x \oplus \delta(0x1)$, 10 : $?(0x0)$, 11 : $0_s(0x0)$, 12 : $0_s \oplus \delta(0x4)$, 13 : $0_s(0x0)$, 14 : $0_s(0x0)$, 15 : $?(0x0)$
10	0 : $0_s(0x0)$, 1 : $0_s(0x0)$, 2 : $?(0x0)$, 3 : $0_s(0x0)$, 4 : $0_s(0x0)$, 5 : $0_s(0x0)$, 6 : $?(0x0)$, 7 : $0_s(0x0)$, 8 : $0_s(0x0)$, 9 : $?(0x0)$, 10 : $?(0x0)$, 11 : $0_s(0x0)$, 12 : $0_s(0x0)$, 13 : $?(0x0)$, 14 : $?(0x0)$, 15 : $0_s \oplus \delta(0x4)$
11	0 : $0_s(0x0)$, 1 : $?(0x0)$, 2 : $?(0x0)$, 3 : $?(0x0)$, 4 : $?(0x0)$, 5 : $0_s(0x0)$, 6 : $?(0x0)$, 7 : $0_s(0x0)$, 8 : $?(0x0)$, 9 : $?(0x0)$, 10 : $?(0x0)$, 11 : $?(0x0)$, 12 : $0_s(0x0)$, 13 : $0_s(0x0)$, 14 : $?(0x0)$, 15 : $0_s(0x0)$
12	0 : $?(0x0)$, 1 : $?(0x0)$, 2 : $?(0x0)$, 3 : $?(0x0)$, 4 : $?(0x0)$, 5 : $0_s(0x0)$, 6 : $?(0x0)$, 7 : $?(0x0)$, 8 : $?(0x0)$, 9 : $?(0x0)$, 10 : $0_s(0x0)$, 11 : $?(0x0)$, 12 : $?(0x0)$, 13 : $?(0x0)$, 14 : $?(0x0)$, 15 : $0_s(0x0)$

Exact-difference constraints $\mathcal{C}_{\text{diff}}(s, \Delta)$. The constraint associates an exact-difference variable with each branch and each internal node of COPY, R, and XOR. Round connections and COPY operations preserve these values. At an XOR with

a tracked symbolic output, the model enforces $\Delta_{\text{out}} = \Delta_{\text{in},1} \oplus \Delta_{\text{in},2}$. When the symbolic output is $?$, its exact-difference variable is assigned the dummy value zero and is ignored by the subsequent DDT and linking constraints. At each call to R , the constraint records an exact input-output difference pair for the DDT constraints. In the first round of the TWINE example, the R call fed by $X^1[12]$ has input difference $0x4$ and output difference $0xb$. Its output and $X^1[13]$, whose exact difference is also $0xb$, enter the same XOR. Hence, $Y^1[13]$ has exact difference $0xb \oplus 0xb = 0x0$, although both XOR inputs have nonzero differences.

DDT constraints $\mathcal{C}_{\text{DDT}}(s, \Delta, p, \tau)$. For TWINE, this constraint group uses the DDT of the 4-bit S-box specified by Suzuki et al. [30]. The S-box has maximum nontrivial differential probability 2^{-2} . For each round-function call whose input symbolic state is δ , the exact input-output difference pair selects the corresponding DDT entry, which must be nonzero. Calls with other symbolic input states contribute zero. For a nonzero input difference, every nonzero TWINE DDT entry has count 4 or 2, corresponding to probabilities 2^{-2} and 2^{-3} . These entries are encoded by the values 2 and 3, respectively. The model accumulates the encoded values over all round-function calls and enforces $p \leq \tau$. In the running example, the accumulated value is $p = 25$.

Prescribed-period linking $\mathcal{C}_{\text{link}}(s, \Delta, \Delta_{\text{per}})$. The linking constraints apply only at calls to R . Outside R , $\mathcal{C}_{\text{prop}}$ and $\mathcal{C}_{\text{diff}}$ propagate the symbolic states and exact differences, respectively, without comparison to Δ_{per} .

For ordinary input states to R , the inherited symbolic transitions remain unchanged. The two shifted states $x \oplus \delta$ and $0_s \oplus x \oplus \delta$ are instead resolved by the following prescribed-period rule.

Property 1 (Prescribed-period propagation through R). Let u and v denote the input and output symbolic states of a call to R , and let Δ_u denote the exact difference carried by the δ component of u . Assume that $\Delta_{\text{per}} \neq 0$. For $u \in \{x \oplus \delta, 0_s \oplus x \oplus \delta\}$, the linking constraint sets

$$v = \begin{cases} 0_s, & \Delta_u = \Delta_{\text{per}}, \\ ?, & \Delta_u \neq \Delta_{\text{per}} \end{cases}.$$

Proof. Let D_b denote the fixed value represented by the δ component of u . By the exact-difference invariant, $\Delta_u = D_0 \oplus D_1$.

For $u = x \oplus \delta$, we have $u_b = x \oplus D_b$ and take $F(z, c) = z$. For $u = 0_s \oplus x \oplus \delta$, write the existing 0_s term as $S(x \oplus A_b, c)$. By induction over the preceding propagation, every such term is linked to the same prescribed period, so $A_0 \oplus A_1 = \Delta_{\text{per}}$. If $\Delta_u = \Delta_{\text{per}}$, then

$$A_0 \oplus D_0 = A_1 \oplus D_1 =: t.$$

Hence

$$u_b = S(x \oplus D_b \oplus t, c) \oplus x \oplus D_b = F(x \oplus D_b, c),$$

where $F(z, c) = S(z \oplus t, c) \oplus z$. Thus, in either case,

$$v_b = R(u_b) = H(x \oplus D_b, c),$$

where $H(z, c) = R(F(z, c))$.

Let $g_v(x) = v_0(x) \oplus v_1(x)$. Since $D_0 \oplus D_1 = \Delta_{\text{per}}$, we have

$$\begin{aligned} g_v(x \oplus \Delta_{\text{per}}) &= H(x \oplus D_1, c) \oplus H(x \oplus D_0, c) \\ &= g_v(x). \end{aligned}$$

Therefore, Δ_{per} is an XOR period of g_v , and the output can be represented by 0_{s} . If $\Delta_u \neq \Delta_{\text{per}}$, the linking constraint does not establish this period and therefore sets the output to ?.

This is the only rule in $\mathcal{C}_{\text{link}}(\mathbf{s}, \Delta, \Delta_{\text{per}})$ that determines an output symbolic state using both the input symbolic state and its attached exact difference. Together with the inherited symbolic transitions, the complete transition rule for a call to R is

$$v = \begin{cases} 0, & u = 0, \\ 0_{\text{s}}, & u = 0_{\text{s}}, \\ R(\mathbf{x}), & u \in \{\mathbf{x}, R(\mathbf{x}), R(\mathbf{x}) \oplus \mathbf{x}\}, \\ \delta, & u = \delta, \\ 0_{\text{s}}, & u \in \{\mathbf{x} \oplus \delta, 0_{\text{s}} \oplus \mathbf{x} \oplus \delta\} \text{ and } \Delta_u = \Delta_{\text{per}}, \\ ?, & u \in \{\mathbf{x} \oplus \delta, 0_{\text{s}} \oplus \mathbf{x} \oplus \delta\} \text{ and } \Delta_u \neq \Delta_{\text{per}}, \\ ?, & \text{otherwise.} \end{cases}$$

The truncated-difference model of Section 3.2 does not perform this comparison and therefore restricts the creation of 0_{s} to a single occurrence. In the differential-based model, every occurrence of $\mathbf{x} \oplus \delta$ or $0_{\text{s}} \oplus \mathbf{x} \oplus \delta$ is checked when it enters R . If $\Delta_u \neq \Delta_{\text{per}}$, the output is set to ? and no new 0_{s} is created. For the 12-round TWINE solution in Figure 1, the equality $\Delta_u = \Delta_{\text{per}} = 0\mathbf{x}1$ holds at four such calls to R . Embedding these exact-difference checks in the SMT search permits multiple creations of 0_{s} while requiring each of them to match the same prescribed period.

4 Key-Dependent Periodic-Function Search

We now turn from prescribed-period distinguishers to key-dependent periodic functions used in polynomial-time Q2 key-recovery attacks. In Section 3, a key-independent period difference is represented explicitly and matched against propagated exact differences by the linking constraints. For each fixed key, the key-dependent period is unknown to the adversary, and recovering it reveals a subkey or a relation among subkeys. The search requires a sufficient algebraic criterion that plays the same role as Theorem 1 does for distinguishers. It must also track

key-dependent shift values whose XOR forms the period and check that the periodic core occurs in the actual keyed computation and is exposed through a function computable from public inputs and ciphertext branches. We encode these conditions and the propagation rules in a single SMT instance for the FK/KF structures defined below.

4.1 FK/KF Structures

Let $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a public function and $k \in \{0, 1\}^n$ a round subkey. We call $x \mapsto F(x) \oplus k$ an FK call and $x \mapsto F(x \oplus k)$ a KF call. Since the subkey is added after F in an FK call and before F in a KF call, the two forms require different symbolic propagation rules for key-dependent values.

Section 2.4 shows how a key-dependent period can support polynomial-time Q2 key recovery under Simon's promise. The search must identify more than an internal expression satisfying the periodicity criterion. It must verify that the actual FK/KF computation produces the two shifted inputs and that a function with the resulting period can be computed from public inputs and ciphertext branches.

4.2 Key-Dependent Periodicity Criterion

The following theorem gives a sufficient algebraic condition for a function to have a nonzero period that may depend on the secret key. The condition can be encoded by the symbolic SMT search developed later in this section.

Theorem 2 (Key-Dependent Periodicity Criterion). *Fix a secret key K . Let $F_0 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be publicly computable, and let $F_1, F_2 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be functions that may depend on K but are fixed for this key. Let $\psi_1, \psi_2 \in \{0, 1\}^n$ be constants with respect to x , possibly depending on K . Set $s_K = \psi_1 \oplus \psi_2$ and assume $s_K \neq 0$. Suppose that $f_K : \{0, 1\}^n \rightarrow \{0, 1\}^n$ satisfies, for every $x \in \{0, 1\}^n$,*

$$f_K(x) = F_2(F_1(x \oplus \psi_1) \oplus F_1(x \oplus \psi_2)) \oplus F_0(x). \quad (3)$$

Define $g_K : \{0, 1\}^n \rightarrow \{0, 1\}^n$ by $g_K(x) = f_K(x) \oplus F_0(x)$. Then s_K is a nonzero XOR period of g_K .

Proof. Let $s_K = \psi_1 \oplus \psi_2$. For every $x \in \{0, 1\}^n$, Equation (3) gives

$$g_K(x) = F_2(F_1(x \oplus \psi_1) \oplus F_1(x \oplus \psi_2)).$$

Since $\psi_1 \oplus s_K = \psi_2$ and $\psi_2 \oplus s_K = \psi_1$,

$$g_K(x \oplus s_K) = F_2(F_1(x \oplus \psi_2) \oplus F_1(x \oplus \psi_1)) = g_K(x).$$

Since $s_K \neq 0$, it is a nonzero XOR period of g_K . $\square$

Algebraic condition. The search uses the sufficient condition in Theorem 2. The Feistel and MISTY expressions of Canale et al. [6] satisfy this condition. Cancelling the public term $F_0(x)$ from $f_K(x)$ leaves $g_K(x) = F_2(T_K(x))$, where the *paired-shift core* is

$$T_K(x) = F_1(x \oplus \psi_1) \oplus F_1(x \oplus \psi_2). \quad (4)$$

The shift $x \mapsto x \oplus s_K$, where $s_K = \psi_1 \oplus \psi_2$, exchanges the two inputs of F_1 . Hence both $T_K(x)$ and $F_2(T_K(x))$ have period s_K . The search therefore identifies $F_0(x)$, the two shifted F_1 calls forming $T_K(x)$, and the resulting expression $F_2(T_K(x))$.

After identifying the paired-shift core in Equation (4), the search tracks subsequent operations on $F_2(T_K(x))$ that preserve s_K . The following property gives the closure rules applied after the algebraic cancellation of $F_0(x)$.

Property 2 (Period-Preserving Closure Rules). Fix a secret key K . Let $T_K : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be the paired-shift core in Equation (4) with nonzero period s_K . Let $F_2, F'_2, H : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be functions that may depend on K but are fixed for this key, and let $c \in \{0, 1\}^n$ be constant with respect to x . Define

$$f_K(x) = F_2(T_K(x)), \quad f'_K(x) = F'_2(T_K(x)).$$

Then the following functions have period s_K .

- The function

$$x \mapsto H(f_K(x) \oplus c).$$

- The function

$$x \mapsto f_K(x) \oplus f'_K(x).$$

Proof. Since T_K has period s_K , $T_K(x \oplus s_K) = T_K(x)$ for every $x \in \{0, 1\}^n$. Hence $f_K(x \oplus s_K) = f_K(x)$ and $f'_K(x \oplus s_K) = f'_K(x)$. Since c is constant with respect to x ,

$$H(f_K(x \oplus s_K) \oplus c) = H(f_K(x) \oplus c).$$

Moreover,

$$f_K(x \oplus s_K) \oplus f'_K(x \oplus s_K) = f_K(x) \oplus f'_K(x).$$

Thus both functions have period s_K . $\square$

Remark 1 (Earlier Key-Recovery Attacks). For a fixed secret key K , the Feistel and MISTY attacks of Canale et al. [6] construct periodic functions from expressions $f_K : \{0, 1\}^n \rightarrow \{0, 1\}^n$ of the form

$$f_K(x) = F'_0(x \oplus \psi) \oplus F_0(x) \oplus \psi_0,$$

where $F_0, F'_0 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ are public functions and $\psi, \psi_0 \in \{0, 1\}^n$ may depend on K but are fixed for this key, with $\psi \neq 0$. Using these public functions, define

$$g_K(x) = f_K(x) \oplus F_0(x) \oplus F'_0(x) = F'_0(x \oplus \psi) \oplus F'_0(x) \oplus \psi_0.$$

Taking $\psi_1 = \psi$, $\psi_2 = 0$, $F_1 = F'_0$, $F_2(z) = z \oplus \psi_0$, and $x \mapsto F_0(x) \oplus F'_0(x)$ as the removable public term, Theorem 2 shows that g_K has nonzero period ψ .

4.3 Constructing Periodic Functions in FK/KF Computations

The criterion in Section 4.2 gives a sufficient condition for an algebraic expression to have a key-dependent period. This subsection gives conditions for constructing an output-computable function with the corresponding period in an FK/KF computation.

Fix a secret key K , and let E_K^r denote the r -round target primitive. Suppose an internal branch function, after being XORed with a publicly computable function, satisfies the algebraic relation in the criterion. If an expression computable from the public input and ciphertext branches differs from this internal branch by a value independent of x , then XORing the same publicly computable function yields an output-computable function with the period given by the criterion. The following definition formalizes this construction.

Definition 3 (Construction). *For a fixed key K , a criterion-based periodic-function construction consists of an initial-state map x^0 , functions h_K, h_K^{out} and F_0, F_1, F_2 , and constants ψ_1, ψ_2 satisfying the following three conditions. The map x^0 sends an n -bit input to d branches of n bits each. Each of the five functions maps n -bit strings to n -bit strings, and $\psi_1, \psi_2 \in \{0, 1\}^n$.*

- Input part. *The initial state $x^0(x)$ can be computed from x using public operations.*
- Core part. *The function h_K is an internal branch function from the evaluation of E_K^r on $x^0(x)$. The function F_0 is publicly computable, while F_1, F_2 are fixed once K is fixed. The values ψ_1, ψ_2 are constant with respect to x , may depend on K , and satisfy $\psi_1 \oplus \psi_2 \neq 0$. For every x , define*

$$T_K(x) := F_1(x \oplus \psi_1) \oplus F_1(x \oplus \psi_2).$$

The core condition requires

$$h_K(x) \oplus F_0(x) = F_2(T_K(x))$$

for every x .

- Output part. *The expression $h_K^{\text{out}}(x)$ can be computed from the public input and the ciphertext branches of $E_K^r(x^0(x))$ using public operations. There exists $\eta_K \in \{0, 1\}^n$, independent of x , such that, for every x ,*

$$h_K^{\text{out}}(x) = h_K(x) \oplus \eta_K.$$

The construction yields the output-computable function

$$G_K(x) := h_K^{\text{out}}(x) \oplus F_0(x).$$

By Theorem 2, G_K has the nonzero XOR period $s_K = \psi_1 \oplus \psi_2$.

The adversary need not know η_K , since it is independent of x and does not affect the period.

The following example shows why these conditions are needed in a 5-round Feistel-FK computation.

Example 2 (5-round Feistel-FK). For a fixed key K , consider the 5-round Feistel-FK construction analyzed by Canale et al. [6], with round keys $k^0, \dots, k^4$. The round update is

$$x_L^{i+1} = x_R^i, \quad x_R^{i+1} = x_L^i \oplus F(x_R^i) \oplus k^i, \quad i = 0, \dots, 4.$$

Input part. For input x , use the public initial state

$$x^0(x) = (x_L^0, x_R^0) = (F(x), x).$$

After the first round,

$$x_L^1 = x_R^0 = x, \quad x_R^1 = x_L^0 \oplus F(x_R^0) \oplus k^0 = F(x) \oplus F(x) \oplus k^0 = k^0.$$

Thus the first round produces $(x_L^1, x_R^1) = (x, k^0)$, with k^0 as the key-dependent branch used in the core.

Core part. The update with $i = 1$ gives

$$x_R^2 = x_L^1 \oplus F(x_R^1) \oplus k^1 = x \oplus F(k^0) \oplus k^1.$$

The updates with $i = 2$ and $i = 3$ then give the internal branch

$$h_K(x) = x_L^4 = x_R^3 = F(x \oplus F(k^0) \oplus k^1) \oplus k^0 \oplus k^2.$$

Set

$$\begin{aligned} F_0(x) &= F(x), & F_1(z) &= F(z), & F_2(z) &= z \oplus k^0 \oplus k^2, \\ \psi_1 &= F(k^0) \oplus k^1, & \psi_2 &= 0. \end{aligned}$$

Let $s_K = \psi_1 \oplus \psi_2 = F(k^0) \oplus k^1$ and assume $s_K \neq 0$. Then $T_K(x) = F(x \oplus \psi_1) \oplus F(x)$ and $h_K(x) \oplus F_0(x) = F_2(T_K(x))$. Since $F_0(x) = F(x)$ is publicly computable, this verifies the core condition of Definition 3. It remains to express $h_K(x)$, up to a value independent of x , from the ciphertext branches.

Output part. The internal expression $h_K(x) = x_L^4$ is not itself an output branch. After the fifth round, the ciphertext branches satisfy $x_R^5 = x_L^4 \oplus F(x_L^4) \oplus k^4$. Hence the output expression $h_K^{\text{out}}(x) = F(x_L^5) \oplus x_R^5$ is computable from the ciphertext branches and satisfies $h_K^{\text{out}}(x) = h_K(x) \oplus k^4$, so $\eta_K = k^4$. Thus the function computed from the public input and ciphertext branches

$$G_K(x) = h_K^{\text{out}}(x) \oplus F(x) = F(x \oplus F(k^0) \oplus k^1) \oplus F(x) \oplus k^0 \oplus k^2 \oplus k^4$$

has the nonzero XOR period $s_K = F(k^0) \oplus k^1$. This completes the construction required by Definition 3.

4.4 SMT Encoding for Key-Dependent Periodic-Function Search

This subsection encodes the three-part construction in Definition 3 as a single SMT instance. Each branch is represented by a symbolic state and an assigned value for its x -independent key-dependent part. The propagation and consistency

constraints encode the input, core, and output conditions. Supporting Material C gives the full SMT encoding.

Symbolic states. Table 4 lists the 7-bit symbolic states used by the SMT encoding. In this encoding, the function F_1 in the criterion is instantiated as the public round function F of the target primitive. The states $\mathbf{F}(\mathbf{x} \oplus \psi_1)$ and $\mathbf{F}(\mathbf{x} \oplus \psi_2)$ record the two calls to F forming the paired-shift core $T_K(x)$. The states $\mathbf{F}_0(\mathbf{x})$ and $\mathbf{F}_2(\mathbf{T}_K(\mathbf{x}))$ record the removable public term and the periodic expression tracked after forming $T_K(x)$, respectively. The states $\mathbf{x}$, $\mathbf{k}$, and $\mathbf{0}$ record the public input, an x -independent key-dependent component, and either zero or an x -independent value with no key tag, respectively. A code whose leftmost bit is one denotes $?$, and the remaining bits are ignored. When the leftmost bit is zero, the lower six bits indicate the symbolic terms present in a branch. Their combination is governed by the propagation rules below.

Table 4. Symbolic states for key-dependent periodic-function search.

State	Meaning	Encoding
$\mathbf{0}$	zero or an x -independent value with no key tag	0000000
$\mathbf{k}$	key-dependent component independent of x	0000001
$\mathbf{x}$	public input variable x	0000010
$\mathbf{F}(\mathbf{x} \oplus \psi_2)$	second term of $T_K(x)$, equal to $F(x)$ when $\psi_2 = 0$	0000100
$\mathbf{F}_0(\mathbf{x})$	removable public term	0001000
$\mathbf{F}(\mathbf{x} \oplus \psi_1)$	first term of $T_K(x)$	0010000
$\mathbf{F}_2(\mathbf{T}_K(\mathbf{x}))$	periodic expression tracked after forming $T_K(x)$	0100000
$?$	untracked symbolic expression	1 * * * * *

Assigned-value rules for key-dependent terms. Unlike the exact differences in Section 3.3, the values tracked here depend on the round keys but not on x . Since their actual values are unknown, we associate an n -bit tag with each round subkey and propagate these tags together with the symbolic states. Supporting Material C gives the concrete tag width used in the implementation.

- Before solving, one run chooses an auxiliary permutation $\hat{F} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ uniformly at random. It is used only to update a tag when an x -independent key-dependent expression passes through the public function F .
- Assign a tag $e_i \in \{0, 1\}^n$ to each round subkey k^i by setting $\text{val}(k^i) = e_i$. For multiple subkeys per round, set $\text{val}(k_j^i) = e_j^i$ with $e_j^i \in \{0, 1\}^n$. All tags and the permutation $\hat{F}$ are constants in the SMT instance.
- For x -independent key-dependent expressions u, w , propagate the tags by

$$\text{val}(u \oplus w) = \text{val}(u) \oplus \text{val}(w), \quad \text{val}(F(u)) = \hat{F}(\text{val}(u)).$$

A branch without an x -independent key-dependent component starts with tag 0. XORing the subkey k^i into a branch XORs e_i into its tag.

The tags serve as fingerprints during the SMT search. The expressions returned by the search are checked separately against the corresponding cipher computations.

Propagation rules for the three parts. Each SMT instance uses a prescribed round split $(r_{\text{in}}, r_{\text{core}}, r_{\text{out}})$. We first set $r_{\text{in}} = r_{\text{out}} = 0$ and search for the largest feasible r_{core} . We then vary r_{in} and r_{out} to extend the core toward the input and output. The input part checks whether the core can be reached from a publicly computable initial state. The output part checks whether it can be expressed from the public input and ciphertext branches up to an x -independent key-dependent value. At each boundary, the final state of the preceding part is equated with the initial state of the following part. We next give the propagation rules for the core, input, and output parts.

Core-search rules. The core part extends the symbolic propagation rules in Section 3.3. Each branch carries a pair (σ, v) , where σ records its symbolic state and $v \in \{0, 1\}^n$ is the tag of its x -independent key-dependent component. A round subkey k^i is represented by $(\mathbf{k}, e_i)$, where $\mathbf{k}$ marks a key-dependent term and $e_i = \text{val}(k^i)$ is its assigned tag.

The assigned tags Ψ_1, Ψ_2 represent the two x -independent shifts in the paired-shift core. We impose $\Psi_1 \neq 0$ and $\Psi_1 \neq \Psi_2$, reserving the zero tag, when used, for Ψ_2 . These constraints apply to the fingerprints used in the SMT search. For a concrete key instantiation, the corresponding candidate period is $s_K = \psi_1 \oplus \psi_2$, and the periodic-function construction requires $s_K \neq 0$.

The COPY operation copies both components of (σ, v) . For an XOR with inputs (σ, v) and (ρ, w) , let (ω, z) denote the output. The bits representing $\mathbf{x}$, $\mathbf{F}(\mathbf{x} \oplus \psi_1)$, and $\mathbf{F}(\mathbf{x} \oplus \psi_2)$ are combined modulo two. The states $?$, $\mathbf{F}_0(\mathbf{x})$, and $\mathbf{F}_2(\mathbf{T}_K(\mathbf{x}))$ are retained whenever they occur. The $\mathbf{k}$ bit and the output tag satisfy

$$v = w \Rightarrow \omega_0 = \sigma_0 \oplus \rho_0, \quad v \neq w \Rightarrow \omega_0 = \sigma_0 \vee \rho_0, \quad z = v \oplus w.$$

When a subkey with tag e_i is XORed into a branch with tag v , the $\mathbf{k}$ component cancels and the output tag becomes 0 if $v = e_i$. Otherwise, the $\mathbf{k}$ component remains and the output tag becomes $v \oplus e_i$.

The F rule recognizes the two shifted calls and the paired-shift core. For x -independent inputs,

$$(0, 0) \mapsto (0, 0), \quad (\mathbf{k}, a) \mapsto (\mathbf{k}, \widehat{F}(a)).$$

Since $\Psi_1 \neq 0$, the public input follows $(\mathbf{x}, 0) \mapsto (\mathbf{F}(\mathbf{x} \oplus \psi_2), 0)$ when $\Psi_2 = 0$, and $(\mathbf{x}, 0) \mapsto (\mathbf{F}_0(\mathbf{x}), 0)$ otherwise. For an input $(\mathbf{x} \oplus \mathbf{k}, a)$,

$$(\mathbf{x} \oplus \mathbf{k}, a) \mapsto \begin{cases} (\mathbf{F}(\mathbf{x} \oplus \psi_1), 0), & a = \Psi_1, \\ (\mathbf{F}(\mathbf{x} \oplus \psi_2), 0), & a = \Psi_2, \\ (?, 0), & a \notin \{\Psi_1, \Psi_2\}. \end{cases}$$

The combined shifted-call state is mapped by F as $(F(\mathbf{x} \oplus \psi_1) \oplus F(\mathbf{x} \oplus \psi_2), 0) \mapsto (F_2(T_K(\mathbf{x})), 0)$. For inputs labelled $F_0(\mathbf{x})$ or $F_2(T_K(\mathbf{x}))$, F preserves the symbolic label. Supporting Material C gives the remaining state combinations.

Input-realization rules. The input part checks whether the symbolic state required at the core boundary can be obtained through the r_{in} FK/KF rounds preceding the core. It uses the core-search rules for COPY, XOR, and ADK. Its F rule is more restrictive. An input containing an untracked component, an x -independent key-dependent component, the paired-shift core, or an incompatible shifted-call component is rejected, while 0 remains unchanged and every other admissible nonzero input is mapped to $F_0(\mathbf{x})$. The final state/tag pairs of the input part are equated with the initial state/tag pairs of the core part. Target-specific constraints restrict the admissible states at the primitive-input boundary. Each extracted initial-state map is then checked for public computability as required by Definition 3.

Output-realization rules. When $r_{\text{out}} > 0$, the output part checks whether a branch representing $h_K(x)$ at the core boundary can be expressed from the ciphertext branches up to an x -independent key-dependent value. Each branch carries a label $\lambda = (\lambda_1, \lambda_0) \in \{00, 01, 11\}$. The labels 11, 01, and 00 denote exact computability from ciphertext branches by public operations, computability up to an x -independent key-dependent value, and a branch not tracked as usable, respectively. Each observable ciphertext branch provides label 11, while a lower label discards information that is not needed by the selected expression. The target-specific constraints propagate these labels from the ciphertext boundary back to the core boundary.

The model selects exactly one branch whose symbolic state satisfies the core-output conditions in Supporting Material C and requires its λ_0 bit to be one. The propagation rules are

$$\text{COPY :} \quad \lambda' = \lambda'' = \lambda,$$

$$\text{ADK :} \quad 00 \mapsto 00, \quad 01 \mapsto 01, \quad 11 \mapsto 01,$$

$$F : \quad 00 \mapsto 00, \quad 01 \mapsto 00, \quad 11 \mapsto 11,$$

and, for XOR inputs $\lambda^{(1)}$ and $\lambda^{(2)}$,

$$\lambda_{\text{out},i} = \lambda_i^{(1)} \wedge \lambda_i^{(2)}, \quad i \in \{0, 1\}.$$

Consequently, $\lambda_0 = 1$ at the selected core branch gives an expression $h_K^{\text{out}}(x) = h_K(x) \oplus \eta_K$, where η_K is independent of x .

SMT search procedure. Algorithm 3 summarizes the joint SMT search for a prescribed round split. A satisfying assignment yields a candidate specification in the encoded construction class. The corresponding initial-state map, output expression, and period relation are then checked against the target computation. The return value `NoCandidateFound` indicates that no candidate satisfies

Algorithm 3 SMT Search for Key-Dependent Periodic-Function Candidates

Require: A target primitive, its wiring and boundary constraints, and a prescribed round split $(r_{\text{in}}, r_{\text{core}}, r_{\text{out}})$

Ensure: A candidate specification for Definition 3, or NoCandidateFound for the current SMT instance

- 1: Assign the subkey tags $(e_i)_i$ and choose the auxiliary permutation $\hat{F}$ uniformly at random.
- 2: Encode the r_{core} -round core-search constraints and require a core branch matching the symbolic pattern $h_K(x) = F_0(x) \oplus F_2(T_K(x))$, with $\Psi_1 \neq 0$ and $\Psi_1 \neq \Psi_2$.
- 3: Encode the r_{in} -round input-realization constraints and equate their final state/tag pairs with the initial state/tag pairs of the core part.
- 4: **if** $r_{\text{out}} > 0$ **then**
- 5: Initialize each observable ciphertext branch with label 11, propagate the output-realization constraints toward the core boundary, and select exactly one eligible core-output branch with $\lambda_0 = 1$.
- 6: **else**
- 7: Select one branch satisfying the accepted core-output condition and set $h_K^{\text{out}}(x) = h_K(x)$ for this branch.
- 8: **end if**
- 9: Solve the joint SMT constraint system.
- 10: **if** the constraint system is unsatisfiable **then**
- 11: **return** NoCandidateFound.
- 12: **end if**
- 13: Reconstruct the initial-state map, the selected core $h_K(x)$, $F_0(x)$, $F_2(T_K(x))$, $h_K^{\text{out}}(x)$, the output offset η_K , and the candidate shifts ψ_1, ψ_2 .
- 14: **return** the candidate specification, including $G_K(x) = h_K^{\text{out}}(x) \oplus F_0(x)$ and $s_K = \psi_1 \oplus \psi_2$.

the current SMT instance. This conclusion is restricted to the encoded construction class for the given target wiring, round split, boundary constraints, tag assignment, and auxiliary permutation.

A returned candidate is used only after direct substitution into the target round equations verifies the three conditions of Definition 3 and the condition $s_K \neq 0$. The assigned tags and $\hat{F}$ are used only during candidate generation.

5 Applications

This section applies the two SMT searches developed in Sections 3 and 4. For TWINE, LBlock, LBlock-S, and ALLPC, the differential-based search produces prescribed periods and exact-differential paths with their DDT weights. We evaluate the resulting candidates using Algorithm 1. For Feistel-FK and FK/KF variants of MISTY, Type-1 and Type-2 GFS, and Skipjack-B, we use the key-dependent search to construct output-computable functions and check their period expressions through period-recovery experiments. We solve the SMT formulas with STP [14] and provide the corresponding search and verification programs in the supplementary attachment.

5.1 Periodic Distinguishers and Verification

Table 5 lists the tested periodic relations for TWINE, LBlock, LBlock-S, and ALLPC. Each row gives the DDT probability exponent p , the prescribed period, the output condition, and an experimental estimate of the relation probability. The exponent p is accumulated from DDT entries along one exact-differential path returned by the SMT search. The value 2^{-p} gives the path-probability estimate from the DDT entries. The experimental relation probability may be higher because several compatible exact-difference paths can contribute to the same quartet test.

Table 5. Periodic distinguishers and experimental probabilities.

Target	Rounds	Periodic relation			Experiment	Verified
		p	Δ_{per}	Output condition	Experimental estimate	
TWINE	10	10	0xE	$Y^{10}[12] = 0_s$	$2^{-8.00}$	Yes
TWINE	11	14	0x5	$Y^{11}[10] = 0_s$	$2^{-13.98}$	Yes
TWINE	12	25	0x1	$Y^{12}[10] = 0_s$	$2^{-23.12}$	Yes
LBlock	10	10	0x1	$Y^{10}[6] = 0_s$	$2^{-8.00}$	Yes
LBlock	11	14	0x1	$Y^{11}[6] = 0_s$	$2^{-14.02}$	Yes
LBlock	12	24	0x2	$Y^{12}[6] = 0_s$	$2^{-22.77}$	Yes
LBlock-S	10	10	0x4	$Y^{10}[6] = 0_s$	$2^{-8.00}$	Yes
LBlock-S	11	14	0x9	$Y^{11}[5] = 0_s$	$2^{-13.97}$	Yes
LBlock-S	12	24	0x6	$Y^{12}[3] = 0_s$	$2^{-21.34}$	Yes
ALLPC	25	12	0x1	$Y^{25}[13] = 0_s \oplus x \oplus \delta$	$2^{-9.06}$	Yes

TWINE. TWINE, proposed by Suzaki et al. [30], is a 64-bit generalized-Feistel cipher with 16 four-bit branches and an 80- or 128-bit key. We obtain 11- and 12-round prescribed-period distinguishers, compared with the 10-round periodic distinguisher of Liu et al. [25]. In Algorithm 1, we take $q = 3, 5, 7$ for 10, 11, and 12 rounds, respectively, the smallest values satisfying $N2^{-nq} \leq 1/8$ under the estimate in Section 3.1. Table 6 includes the cost of testing the relation of Liu et al. [25] over the full domain. For the differential-linear distinguishers of Hadipour et al. [16], we convert the absolute correlations $|\text{Cor}| = 2^{-\gamma}$ to time complexities $2|\text{Cor}|^{-2} = 2^{2\gamma+1}$. The 11-round distinguisher appears in Table 10 and Fig. 2, and the 12-round distinguisher in Table 3 and Fig. 1.

LBlock and LBlock-S. LBlock, proposed by Wu and Zhang [31], is a 64-bit Feistel cipher with an 80-bit key and eight 4-bit S-boxes. LBlock-S is its simplified variant used in LAC [34]. We obtain 10-, 11-, and 12-round prescribed-period distinguishers for both ciphers. For LBlock, the 11- and 12-round results extend those of Xiang et al. [32] and Liu et al. [25]. We use the same values $q = 3, 5, 7$ as for TWINE at 10, 11, and 12 rounds, respectively. Table 6 also includes the costs

of full-domain tests for the 10-round LBlock relation of Liu et al. [25] and our 10-round LBlock-S relation. Supporting Material D.1 gives the branch convention, with round states and paths in Tables 11–14 and Figs. 3–6.

Table 6. Time-complexity analysis for TWINE, LBlock, LBlock-S, and ALLPC.

Cipher	Rounds	Distinguisher	Complexity
TWINE	10	Differential-linear distinguisher	$2^{16.70}$ [16]
		Periodic distinguisher	2^{13} [25]
		Prescribed-period distinguisher	$2^{11.59}$
	11	Differential-linear distinguisher	$2^{21.98}$ [16]
		Prescribed-period distinguisher	$2^{18.31}$
	12	Differential-linear distinguisher Prescribed-period distinguisher	$2^{23.98}$ [16] $2^{27.92}$
LBlock	10	Differential-linear distinguisher	$2^{13.06}$ [16]
		Periodic distinguisher	2^{13} [25]
		Prescribed-period distinguisher	$2^{11.59}$
	11	Differential-linear distinguisher	$2^{17.08}$ [16]
		Prescribed-period distinguisher	$2^{18.35}$
	12	Differential-linear distinguisher Prescribed-period distinguisher	$2^{20.96}$ [16] $2^{27.58}$
LBlock-S	10	Differential-linear distinguisher	$2^{11.86}$ [16]
		Periodic distinguisher	2^{13}
		Prescribed-period distinguisher	$2^{11.58}$
	11	Differential-linear distinguisher	$2^{15.86}$ [16]
		Prescribed-period distinguisher	$2^{18.29}$
	12	Differential-linear distinguisher Prescribed-period distinguisher	$2^{19.84}$ [16] $2^{26.15}$
ALLPC	25	Differential distinguisher	2^{25} [11]
		Boomerang distinguisher	$2^{18.07}$ [35]
		Prescribed-period distinguisher	$2^{13.06}$

ALLPC. ALLPC, proposed by Cheng et al. [8], is a lightweight generalized-Feistel cipher for IoT. We obtain a prescribed-period distinguisher for all 25 rounds. The final round ends at Y^{25} before the branch permutation. With the 4-bit branch numbering in Supporting Material D.2, its output condition is $Y^{25}[13] = 0_s \oplus x \oplus \delta$, with prescribed period $\Delta_{\text{per}} = 0x1$. The experimental probability is $2^{-9.06}$, and Algorithm 1 gives an estimated time complexity of $2^{13.06}$ for $q = 4$. The previous full-round differential and boomerang distinguishers have probabilities 2^{-24} [11] and $2^{-16.07}$ [35], respectively. For their costs in Table 6, we count two encryptions per differential pair and two encryptions plus two decryptions per boomerang quartet. Tables 15–16 and Fig. 7 give the round states and path of our distinguisher.

5.2 Search for Key-Dependent Periodic Functions

Table 7 compares the key-recovery attacks obtained from our SMT search with previous polynomial-time attacks and gives the round split $r_{\text{in}} + r_{\text{core}} + r_{\text{out}}$. We checked the stated key-dependent periods experimentally for randomly chosen keys. Below, we present three periodic functions and give the full key-recovery procedure for 6-round MISTY L-FK. The remaining functions and key-recovery procedures are given in Supporting Material F.

Table 7. Round comparison of polynomial-time key-recovery attacks.

Target	Variant	Previous rounds	Our rounds	$r_{\text{in}} + r_{\text{core}} + r_{\text{out}}$	Period tested
MISTY	L-FK	5 [6]	6	$2 + 4 + 0$	Yes
MISTY	L-KF	—	5	$0 + 4 + 1$	Yes
MISTY	R-FK	4 [6]	4	$0 + 3 + 1$	Yes
MISTY	R-KF	—	5	$0 + 2 + 3$	Yes
Feistel	FK	5 [6]	5	$1 + 3 + 1$	Yes
Type-1 GFS	FK	—	17	$3 + 13 + 1$	Yes
Type-1 GFS	KF	—	13	$0 + 13 + 0$	Yes
Type-2 GFS	FK	—	7	$1 + 5 + 1$	Yes
Type-2 GFS	KF	—	5	$0 + 5 + 0$	Yes
Skipjack-B	FK	—	17	$4 + 13 + 0$	Yes
Skipjack-B	KF	—	14	$0 + 13 + 1$	Yes

Type-1 and Type-2 GFS have four branches. Type-1 GFS uses the dec direction.

First 6-round periodic function for MISTY L-FK. For 6-round MISTY L-FK with input

$$(x_L^0, x_R^0) = (F^{-1}(F^{-1}(x)), F^{-1}(x)),$$

the left output after six rounds is

$$\begin{aligned} f(x) &= E_L^6(F^{-1}(F^{-1}(x)), F^{-1}(x)) \\ &= F(x \oplus F(k^0) \oplus k^0 \oplus k^1 \oplus k^2) \oplus F(x \oplus k^0 \oplus k^1) \\ &\quad \oplus x \oplus F(k^0) \oplus k^0 \oplus k^1 \oplus k^2 \oplus k^3 \oplus k^4. \end{aligned}$$

Let $s = F(k^0) \oplus k^2$. The shift $x \mapsto x \oplus s$ exchanges the two inputs to F , so $g(x) = f(x) \oplus x$ satisfies $g(x \oplus s) = g(x)$.

Full key recovery for 6-round MISTY L-FK. To recover k^0 , we use the nested Simon procedure of Bonnetain et al. [4] and Cid et al. [9]. For each y , define

$$G_y(x) = E_L^6(F^{-1}(F^{-1}(x) \oplus y), F^{-1}(x)) \oplus x.$$

Expanding the rounds gives the key-dependent period

$$g(y) = F(y \oplus k^0) \oplus k^2.$$

For the unique value of y satisfying $g(y) = 0$, G_y is constant and the inner procedure returns zero. We run the inner procedure without intermediate measurements and uncompute its workspace to evaluate $g(y)$ coherently. The outer function

$$H(y) = g(y) \oplus F(y) = F(y \oplus k^0) \oplus F(y) \oplus k^2$$

has period k^0 , which is recovered by the outer Simon procedure. After recovering k^0 , we remove the first round. A query to the remaining five rounds on (z_L, z_R) is implemented by querying the original primitive on

$$(F^{-1}(z_L \oplus z_R \oplus k^0), z_L).$$

Applying the 5-round recovery procedure of Canale et al. [6] then recovers $k^1, \dots, k^5$, completing the polynomial-time key-recovery attack on all six rounds.

First 5-round periodic function for MISTY R-KF. For 5-round MISTY R-KF, choose input $(x, F(x))$ and let

$$y_L = E_L^5(x, F(x)), \quad y_R = E_R^5(x, F(x)).$$

From these outputs, define

$$g(x) = F^{-1}(y_L \oplus y_R) \oplus y_L \oplus y_R \oplus F^{-1}(y_R).$$

Writing $A(x) = F(x \oplus k^0) \oplus F(x)$, we expand the five rounds to obtain

$$g(x) = F(A(x) \oplus k^1) \oplus k^3 \oplus k^4.$$

Since $A(x \oplus k^0) = A(x)$, we have $g(x \oplus k^0) = g(x)$.

First 17-round periodic function for Type-1 GFS-FK. Following previous attacks [26,29], we apply our search to the decryption structure of four-branch Type-1 GFS-FK. Choose the input

$$u(x) = (F^4(x) \oplus x, F(x), F^2(x), F^3(x)).$$

From the output branches, define

$$g_{\text{FK}}(x) = F(E_0^{17}(u(x))) \oplus E_1^{17}(u(x)) \oplus x.$$

Writing $A(x) = F(x \oplus k^0) \oplus F(x) \oplus k^3$, we expand the 17 rounds to obtain

$$\begin{aligned} g_{\text{FK}}(x) = & F(F(F(A(x)) \oplus k^2 \oplus k^6) \oplus F(k^2) \oplus k^1 \oplus k^5 \oplus k^9) \\ & \oplus F(F(k^2) \oplus k^1 \oplus k^5) \oplus F(k^1) \oplus k^0 \oplus k^4 \oplus k^8 \oplus k^{12} \oplus k^{16}. \end{aligned}$$

Since $A(x \oplus k^0) = A(x)$, we have $g_{\text{FK}}(x \oplus k^0) = g_{\text{FK}}(x)$.

6 Conclusion

We developed two SMT models for periodic distinguishers and key-dependent periodic functions. The distinguisher model combines symbolic propagation, exact differences, DDT-based probability constraints, and prescribed-period linking. It gives 12-round prescribed-period distinguishers for TWINE, LBlock, and LBlock-S and a full-round distinguisher for ALLPC, all of which use classical oracle queries. The key-recovery model tracks key-dependent values and checks that the corresponding periodic functions can be constructed from public inputs and oracle outputs. The resulting polynomial-time Q2 attacks extend key recovery for MISTY L-FK from five to six rounds and give new attacks on Type-1 GFS, Type-2 GFS, and Skipjack-B variants.

Future work could extend the search from periodic functions to complete quantum attack strategies, including the selection and combination of quantum algorithms. This requires jointly considering the exploitable properties of the target construction, the available oracle access, and the cost of each attack step. The resulting models could compare different strategies under the same access assumptions and identify combinations of quantum algorithms that reduce the overall attack complexity.

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Key R&D Program of China (Grant No. 2024YFA1013000), the National Natural Science Foundation of China (Grant No. 623B2067), the China Postdoctoral Science Foundation (Grant No. 2026T190826, 2025M783119), the Shandong Postdoctoral Science Foundation (Grant No. SDZZ-ZR-202501347), and the Qingdao Postdoctoral Science Foundation (Grant No. QDBSH20260101026). Haoyang Wang was supported by the National Natural Science Foundation of China (No. 62302293). We used generative AI tools for grammar checking. All content was written and verified by the authors, who take full responsibility for the final text.

References

1. Anand, R., Ghosh, S., Isobe, T., Shiba, R.: Quantum key recovery attacks on 4-round iterated Even-Mansour with two keys. In: Mouha, N., Nikiforakis, N. (eds.) ISC 2024, Part I. LNCS, vol. 15257, pp. 87–103. Springer, Cham (Oct 2024). https://doi.org/10.1007/978-3-031-75757-0_5
2. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline Simon’s algorithm. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 552–583. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_20
3. Bonnetain, X., Leurent, G., Naya-Plasencia, M., Schrottenloher, A.: Quantum linearization attacks. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 422–452. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_15

4. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: On quantum slide attacks. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019. LNCS, vol. 11959, pp. 492–519. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-38471-5_20
5. Bonnetain, X., Schrottenloher, A., Sibleyras, F.: Beyond quadratic speedups in quantum attacks on symmetric schemes. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 315–344. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_12
6. Canale, F., Leander, G., Stennes, L.: Simon’s algorithm and symmetric crypto: Generalizations and automatized applications. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 779–808. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_26
7. Chen, J., Liu, Q., Li, B., Qiao, J., Wang, J.: Simplified periodic distinguishers searching: Application to gfs-4f/2f and twine. In: Lin, D., Wang, M., Yung, M. (eds.) Information Security and Cryptology. pp. 425–443. Springer Nature Singapore, Singapore (2025)
8. Cheng, J., Guo, S., He, J.: ALLPC: a lightweight block cipher based on generalized feistel networks for iot. In: 40th IEEE International Performance Computing and Communications Conference. pp. 1–8. IEEE (2021)
9. Cid, C., Hosoyamada, A., Liu, Y., Sim, S.M.: Quantum cryptanalysis on contracting Feistel structures and observation on related-key settings. In: Bhargavan, K., Oswald, E., Prabhakaran, M. (eds.) INDOCRYPT 2020. LNCS, vol. 12578, pp. 373–394. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-65277-7_17
10. Cui, J., Guo, J., Ding, S.: Applications of Simon’s algorithm in quantum attacks on feistel variants. Quantum Inf. Process. **20**(3), 117 (2021). <https://doi.org/10.1007/S11128-021-03027-X>
11. Cui, T., Zhang, J., Jin, C., Chen, S., Yang, Y.: Practical distinguishing attack against the iot-friendly block cipher ALLPC. Electron. Lett. **58**(16), 612–613 (2022). <https://doi.org/10.1049/el12.12544>
12. Daiza, T., Yoneyama, K.: Quantum key recovery attacks on 3-round Feistel-2 structure without quantum encryption oracles. In: Cheng, C.M., Akiyama, M. (eds.) IWSEC 22. LNCS, vol. 13504, pp. 128–144. Springer, Cham (Aug / Sep 2022). https://doi.org/10.1007/978-3-031-15255-9_7
13. Dong, X., Li, Z., Wang, X.: Quantum cryptanalysis on some generalized feistel schemes. Sci. China Inf. Sci. **62**(2), 22501:1–22501:12 (2019). <https://doi.org/10.1007/S11432-017-9436-7>
14. Ganesh, V., Dill, D.L.: A decision procedure for bit-vectors and arrays. In: Damm, W., Hermanns, H. (eds.) Computer Aided Verification. pp. 519–531. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
15. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: 28th ACM STOC. pp. 212–219. ACM Press (May 1996). <https://doi.org/10.1145/237814.237866>
16. Hadipour, H., Derbez, P., Eichlseder, M.: Revisiting differential-linear attacks via a boomerang perspective with application to AES, Ascon, CLEFIA, SKINNY, PRESENT, KNOT, TWINE, WARP, LBlock, Simeck, and SERPENT. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14923, pp. 38–72. Springer (2024). https://doi.org/10.1007/978-3-031-68385-5_2

17. Hodzic, S., Knudsen, L.R.: A quantum distinguisher for 7/8-round SMS4 block cipher. *Quantum Inf. Process.* **19**(11), 411 (2020). <https://doi.org/10.1007/S11128-020-02929-6>
18. Hodzic, S., Knudsen, L.R., Kidmose, A.B.: On quantum distinguishers for type-3 generalized feistel network based on separability. In: Ding, J., Tillich, J. (eds.) *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, Paris, France, April 15-17, 2020, Proceedings. *Lecture Notes in Computer Science*, vol. 12100, pp. 461–480. Springer (2020). https://doi.org/10.1007/978-3-030-44223-1_25
19. Hosoyamada, A., Sasaki, Y.: Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In: Smart, N.P. (ed.) *CT-RSA 2018*. LNCS, vol. 10808, pp. 198–218. Springer, Cham (Apr 2018). https://doi.org/10.1007/978-3-319-76953-0_11
20. Ito, G., Hosoyamada, A., Matsumoto, R., Sasaki, Y., Iwata, T.: Quantum chosen-ciphertext attacks against Feistel ciphers. In: Matsui, M. (ed.) *CT-RSA 2019*. LNCS, vol. 11405, pp. 391–411. Springer, Cham (Mar 2019). https://doi.org/10.1007/978-3-030-12612-4_20
21. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: Robshaw, M., Katz, J. (eds.) *CRYPTO 2016, Part II*. LNCS, vol. 9815, pp. 207–237. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_8
22. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: *IEEE International Symposium on Information Theory, ISIT 2010*, June 13-18, 2010, Austin, Texas, USA, Proceedings. pp. 2682–2685. IEEE (2010). <https://doi.org/10.1109/ISIT.2010.5513654>
23. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: *Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012*, Honolulu, HI, USA, October 28-31, 2012. pp. 312–316. IEEE (2012), <https://ieeexplore.ieee.org/document/6400943/>
24. Leander, G., May, A.: Grover meets Simon - quantumly attacking the FX-construction. In: Takagi, T., Peyrin, T. (eds.) *ASIACRYPT 2017, Part II*. LNCS, vol. 10625, pp. 161–178. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70697-9_6
25. Liu, Q., Wang, H., Wang, J., Li, B., Wang, M.: Quantum periodic distinguisher construction: Symbolization method and automated tool. In: Hanaoka, G., Yang, B.Y. (eds.) *ASIACRYPT 2025, Part VIII*. LNCS, vol. 16252, pp. 224–255. Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5125-5_8
26. Ni, B., Ito, G., Dong, X., Iwata, T.: Quantum attacks against type-1 generalized Feistel ciphers and applications to CAST-256. In: Hao, F., Ruj, S., Sen Gupta, S. (eds.) *INDOCRYPT 2019*. LNCS, vol. 11898, pp. 433–455. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-35423-7_22
27. NIST: Skipjack and kea algorithm specifications (1998), <https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/skipjack/skipjack.pdf>
28. Simon, D.R.: On the power of quantum computation. In: *35th FOCS*. pp. 116–123. IEEE Computer Society Press (Nov 1994). <https://doi.org/10.1109/SFCS.1994.365701>
29. Sun, H.W., Cai, B.B., Qin, S.J., Wen, Q.Y., Gao, F.: Quantum attacks on type-1 generalized feistel schemes. *Advanced Quantum Technologies* **6**(10), 2300155 (2023)

30. Suzuki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: A lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) *Selected Areas in Cryptography*, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers. *Lecture Notes in Computer Science*, vol. 7707, pp. 339–354. Springer (2012). https://doi.org/10.1007/978-3-642-35999-6_22
31. Wu, W., Zhang, L.: LBlock: A lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) *ACNS 2011*. *LNCS*, vol. 6715, pp. 327–344. Springer, Berlin, Heidelberg (Jun 2011). https://doi.org/10.1007/978-3-642-21554-4_19
32. Xiang, Z., Wang, X., Yu, B., Sun, B., Zhang, S., Zeng, X., Shen, X., Li, N.: Links between quantum distinguishers based on Simon’s algorithm and truncated differentials. *IACR Trans. Symmetric Cryptol.* **2024**(2), 296–321 (2024). <https://doi.org/10.46586/TOSC.V2024.I2.296-321>
33. Yu, B., Shi, T., Dong, X., Shen, X., Luo, Y., Sun, B.: Quantum attacks: A view of data complexity on offline simon’s algorithm. In: Ge, C., Yung, M. (eds.) *Information Security and Cryptology - 19th International Conference, Inscrypt 2023*, Hangzhou, China, December 9-10, 2023, Revised Selected Papers, Part II. *Lecture Notes in Computer Science*, vol. 14527, pp. 329–342. Springer (2023). https://doi.org/10.1007/978-981-97-0945-8_19
34. Zhang, L., Wu, W., Wang, Y., Wu, S., Zhang, J.: LAC: a lightweight authenticated encryption cipher. Submission to the CAESAR competition (2014), <https://competitions.cr.yp.to/round1/lacv1.pdf>
35. Zhou, Y., Wang, S., Wu, Y., Hu, B.: Improved method of searching for boomerang distinguishers on feistel structures—applications to WARP, TWINE, LBlock, LBlock-S, and ALLPC. *Des. Codes Cryptogr.* **93**(9), 3707–3731 (2025). <https://doi.org/10.1007/s10623-025-01639-3>

Supporting Material

A TWINE S-box and Its DDT

Tables 8 and 9 give the 4-bit S-box of TWINE [30] and its DDT.

Table 8. TWINE 4-bit S-box

x	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
$S(x)$	0xc	0x0	0xf	0xa	0x2	0xb	0x9	0x5	0x8	0x3	0xd	0x7	0x1	0xe	0x6	0x4

Table 9. DDT of the TWINE 4-bit S-box

$\Delta_{in} \backslash \Delta_{out}$	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
0x0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	0	2	0	0	2	0	0	0	2	2	2	4	0	0	2
0x2	0	0	0	2	2	2	0	2	0	0	4	2	0	0	2	0
0x3	0	0	2	0	0	2	2	2	2	0	0	0	0	0	2	4
0x4	0	0	0	2	0	0	2	0	0	2	0	4	0	2	2	2
0x5	0	2	4	2	0	0	2	2	0	2	2	0	0	0	0	0
0x6	0	2	0	0	0	4	0	2	0	2	0	0	2	2	2	0
0x7	0	0	0	2	2	2	0	2	4	0	2	0	0	2	0	0
0x8	0	2	2	4	2	2	0	0	0	0	0	0	0	2	0	2
0x9	0	0	0	2	0	0	0	2	4	0	2	0	2	2	0	2
0xa	0	2	0	0	2	0	0	4	2	2	0	2	0	0	0	2
0xb	0	0	2	0	2	0	2	2	0	0	0	2	2	4	0	0
0xc	0	0	2	0	2	0	0	0	2	2	2	0	0	2	4	0
0xd	0	4	2	2	0	0	0	0	2	0	0	2	2	0	2	0
0xe	0	2	0	0	4	0	2	0	0	0	2	0	2	0	2	2
0xf	0	2	0	0	0	2	4	0	2	0	2	2	0	2	0	0

B SMT Encoding of the Complete Distinguisher Model

This supporting material gives the SMT encoding of the complete distinguisher model from Section 3.3. We instantiate the encoding for the 12-round TWINE example used in that section and keep the same notation throughout.

B.1 Variables and Global Quantities

For each round i and branch j , the complete model keeps the symbolic state variables

$$x_j^i, y_j^i \in \{0, 1\}^6$$

from the symbolic model of Liu et al. [25]. It additionally introduces 4-bit exact-difference variables

$$\Delta x_j^i, \Delta y_j^i \in \{0, 1\}^4.$$

For each internal component in round i and bundle $t \in \{0, \dots, 7\}$, the model also declares symbolic states and exact differences for the nodes named `COPY_IN`, `COPY_OUT1`, `COPY_OUT2`, `ROUNDFUNC_IN`, `ROUNDFUNC_OUT`, `XOR_IN1`, `XOR_IN2`, and `XOR_OUT`. In the generated SMT file, these declarations appear as

$$\mathbf{x_j_i}, \mathbf{y_j_i}, \mathbf{xdiff_j_i}, \mathbf{ydiff_j_i},$$

for the branch states, together with

$$\text{COPY_IN_t_i}, \text{COPY_OUT1_t_i}, \text{COPY_OUT2_t_i},$$

$$\text{ROUNDFUNC_IN_t_i}, \text{ROUNDFUNC_OUT_t_i},$$

$$\text{XOR_IN1_t_i}, \text{XOR_IN2_t_i}, \text{XOR_OUT_t_i},$$

and the corresponding difference variables, where `diff` is inserted after the base name.

The complete model further uses the prescribed-period variable, the per-S-box DDT variables used in the probability constraints, and the total DDT probability exponent:

$$\Delta_{\text{per}} \in \{0, 1\}^4, \quad \text{PossibilitySb}_t^i \in \{0, 1\}^4, \quad p \in \{0, 1\}^{16}.$$

We require $\Delta_{\text{per}} \neq 0$. Here Δ_{per} is the prescribed period difference, PossibilitySb_t^i is the DDT value used by the probability constraints for the t -th call to $\mathbf{R}$ in round i , and p is the total DDT probability exponent. In the code, the variables corresponding to Δ_{per} and p are named `perioddiff` and `P`, respectively. At round 0, the implementation further declares the global DDT array and one 1-bit tag for each input branch. The mathematical description below keeps the variables that affect the nonlinear TWINE constraint system; auxiliary variables used by other solver modes are outside the model stated here.

For every branch or internal node z with exact difference Δ_z , the model enforces

$$\Delta_z = 0 \iff z[1 : 0] = 0\text{bin}00, \quad z[5 : 5] = 0\text{bin}1 \Rightarrow \Delta_z = 0.$$

Moreover, the second least significant bit is fixed to zero for every symbolic state:

$$z[1 : 1] = 0\text{bin}0.$$

Therefore the complete model keeps only the symbolic states that do not contain the $\mathbf{R}(\delta)$ component. This is a genuine modeling change from the symbolic model of Liu et al. [25]: once exact differences are introduced explicitly, the complete model no longer needs a separate symbolic variable for $\mathbf{R}(\delta)$, since the corresponding behavior is determined by the exact-difference variables together with the linking constraints. In particular, it still allows 0 , δ , $\mathbf{x}$, $\mathbf{x} \oplus \delta$, $\mathbf{R}(\mathbf{x})$, $\mathbf{R}(\mathbf{x}) \oplus \mathbf{x}$, 0_s , $0_s \oplus \mathbf{x} \oplus \delta$, and $?$.

B.2 Input and Output Constraints

In the code, the input branches are first restricted by

$$x_j^0[5 : 5] \vee x_j^0[4 : 4] \vee x_j^0[3 : 3] \vee x_j^0[1 : 1] = 0$$

for all j . Equivalently, the input states satisfy

$$x_j^0 \in \{0, \delta, \mathbf{x}, \mathbf{x} \oplus \delta\}.$$

As in the implementation, the model requires

- at least one input branch with state δ ,
- at least one input branch with state $\mathbf{x}$,
- and at most one branch carrying the variable part.

The last condition is enforced by 1-bit tags

$$\text{tag}_j = \begin{cases} 1, & x_j^0 \in \{\mathbf{x}, \mathbf{x} \oplus \delta\}, \\ 0, & \text{otherwise,} \end{cases} \quad \sum_j \text{tag}_j \leq 1.$$

Hence a solver solution must contain at least one pure δ branch and exactly one branch carrying the variable part. In particular, although $\mathbf{x} \oplus \delta$ belongs to the initial symbolic domain, it cannot remain on an input branch once the full input constraints are enforced.

The inherited output condition is imposed on the final round output. With the paper notation Y^r for the output after r rounds, the generated variables use y^{r-1} , because y^i is the output state of round $i + 1$. Thus the code imposes

$$\exists j : y_j^{r-1} \in \{0\text{bin}000110, 0\text{bin}000111\} \text{ or } y_j^{r-1}[5 : 4] = 0\text{bin}01.$$

Equivalently, the generated SMT assertion is

$$\bigvee_{j=0}^{15} \left(y_j^{r-1} = 0\text{bin}000110 \vee y_j^{r-1} = 0\text{bin}000111 \vee y_j^{r-1}[5 : 4] = 0\text{bin}01 \right).$$

Because the complete model enforces $y_j^{r-1}[1 : 1] = 0\text{bin}0$, the first two cases cannot occur. Hence the output condition reduces to

$$\exists j : y_j^{r-1}[5 : 4] = 0\text{bin}01,$$

that is, at least one final branch must carry a $0_{\mathbf{s}}$ -type state.

B.3 Structural Encoding for TWINE

TWINE has 16 branches of 4 bits and uses 8 copies, 8 round-function calls, and 8 XORs in each round. Let $t \in \{0, \dots, 7\}$. The wiring of round i is

$$x_{2t}^i \rightarrow \text{COPY_IN}_t^i \rightarrow (\text{ROUNDFUNC_IN}_t^i, y_{2t}^i),$$

$$(x_{2t+1}^i, \text{ROUNDFUNC_OUT}_t^i) \rightarrow \text{XOR}_t^i \rightarrow y_{2t+1}^i.$$

Across every wire, the model imposes equality on both the symbolic state and the exact difference. In the implementation, this constraint is generated by the helper `equal_connect`. Therefore the same wiring applies simultaneously to $\mathbf{s}$ and to Δ . More explicitly, for each bundle t and round i the model contains

$$\begin{aligned} x_{2t}^i &= \text{COPY_IN}_t^i = \text{COPY_OUT1}_t^i = \text{ROUNDFUNC_IN}_t^i, & y_{2t}^i &= \text{COPY_OUT2}_t^i, \\ x_{2t+1}^i &= \text{XOR_IN1}_t^i, & \text{ROUNDFUNC_OUT}_t^i &= \text{XOR_IN2}_t^i, & y_{2t+1}^i &= \text{XOR_OUT}_t^i, \end{aligned}$$

and the same equalities hold for the corresponding exact differences.

Between rounds, TWINE uses the fixed branch permutation

$$\pi = (1, 2, 11, 6, 3, 0, 9, 4, 7, 10, 13, 14, 5, 8, 15, 12),$$

so that

$$x_j^{i+1} = y_{\pi(j)}^i$$

for $j = 0, \dots, 15$.

B.4 Complete Constraint System

The full model is

$$\mathcal{M}_{\text{diff}} := \mathcal{C}_{\text{input}} \wedge \mathcal{C}_{\text{prop}} \wedge \mathcal{C}_{\text{output}} \wedge \mathcal{C}_{\text{diff}}(\mathbf{s}, \Delta) \wedge \mathcal{C}_{\text{DDT}}(\mathbf{s}, \Delta, p, \tau) \wedge \mathcal{C}_{\text{link}}(\mathbf{s}, \Delta, \Delta_{\text{per}}).$$

The three inherited groups keep the symbolic input, symbolic propagation, and symbolic output rules from Section 3.2. The remaining three groups are written below.

The exact-difference constraints $\mathcal{C}_{\text{diff}}(\mathbf{s}, \Delta)$. Along a wire, the symbolic state and the exact difference are copied unchanged. Under `COPY`, the model imposes

$$\begin{aligned} \text{COPY_IN}_t^i &= \text{COPY_OUT1}_t^i = \text{COPY_OUT2}_t^i, \\ \Delta \text{COPY_IN}_t^i &= \Delta \text{COPY_OUT1}_t^i = \Delta \text{COPY_OUT2}_t^i. \end{aligned}$$

At `XOR`, the output exact difference is set to zero when the output state's most significant bit is 1. Otherwise,

$$\Delta_{\text{out}} = \Delta_{\text{in},1} \oplus \Delta_{\text{in},2}.$$

For the symbolic state, the complete model keeps the inherited rules on the bits corresponding to $?$, $0_{\mathbf{s}}$, $\mathbf{R}(\mathbf{x})$, and $\mathbf{x}$:

$$w_{?} = u_{?} \vee v_{?}, \quad w_{0_{\mathbf{s}}} = u_{0_{\mathbf{s}}} \vee v_{0_{\mathbf{s}}}, \quad w_{\mathbf{R}(\mathbf{x})} = u_{\mathbf{R}(\mathbf{x})} \vee v_{\mathbf{R}(\mathbf{x})}, \quad w_{\mathbf{x}} = u_{\mathbf{x}} \oplus v_{\mathbf{x}}.$$

The bit corresponding to $\mathbf{R}(\delta)$ is fixed to zero throughout the complete model. Hence the remaining role of δ is no longer handled by a separate symbolic collision table as in the symbolic model of Liu et al. [25]. Instead, it is controlled jointly by

the input constraints, the exact differences, and the linking constraints. Equivalently, the complete model replaces the old symbolic treatment of $R(\delta)$ by an exact-difference-based treatment. Each call to R also introduces an instantiated input-output pair

$$(\Delta\text{ROUNDFUNC_IN}_t^i, \Delta\text{ROUNDFUNC_OUT}_t^i),$$

which is later used by $\mathcal{C}_{\text{DDT}}$. These pairs are declared for every round-function call before the DDT constraints and the linking constraints are added.

The DDT constraints $\mathcal{C}_{\text{DDT}}(\mathbf{s}, \Delta, p, \tau)$. The TWINE DDT is listed in Table 9. In the SMT model, the DDT array is encoded by four 4-bit values:

$$16 \mapsto 0\text{bin}0000, \quad 4 \mapsto 0\text{bin}0010, \quad 2 \mapsto 0\text{bin}0011, \quad 0 \mapsto 0\text{bin}1111.$$

Thus the nontrivial DDT entries are encoded as the values 2 and 3 in the probability constraints, while the code $0\text{bin}1111$ marks a forbidden transition.

For every call to R , the model introduces a 4-bit variable PossibilitySb_t^i . If the input symbolic state is δ , then

$$\text{PossibilitySb}_t^i = \text{DDT}[\Delta\text{ROUNDFUNC_IN}_t^i \parallel \Delta\text{ROUNDFUNC_OUT}_t^i],$$

and the forbidden value $0\text{bin}1111$ is excluded. Otherwise,

$$\text{PossibilitySb}_t^i = 0\text{bin}0000.$$

Equivalently, in mathematical form,

$$\text{PossibilitySb}_t^i = \begin{cases} \text{DDT}[\Delta\text{ROUNDFUNC_IN}_t^i \parallel \Delta\text{ROUNDFUNC_OUT}_t^i], & \text{if } \text{ROUNDFUNC_IN}_t^i = \delta, \\ 0, & \text{otherwise,} \end{cases}$$

subject to

$$\text{DDT}[\Delta\text{ROUNDFUNC_IN}_t^i \parallel \Delta\text{ROUNDFUNC_OUT}_t^i] \neq 0\text{bin}1111.$$

in the active case. The total DDT probability exponent is

$$p = \sum_{i,t} \text{PossibilitySb}_t^i, \quad p \leq \tau.$$

In the generated SMT file, this sum is encoded by zero-extending each 4-bit term to 16 bits:

$$p = \text{BVPLUS}\left(16, 0\text{bin}00000000000000@ \text{PossibilitySb}_0^0, \dots, 0\text{bin}00000000000000@ \text{PossibilitySb}_7^{r-1}\right),$$

followed by the bound

$$\text{BVLE}(p, \tau).$$

Hence the complete model keeps only candidates whose instantiated input-output differences through the round function are compatible with the DDT bound.

The linking constraints $\mathcal{C}_{\text{link}}(\mathbf{s}, \Delta, \Delta_{\text{per}})$. At the input, $\mathcal{C}_{\text{diff}}$ links a branch marked by δ to a nonzero exact-difference variable. The model does not prescribe these branch differences before solving; a solver solution fixes them together with the prescribed period. This input-side linking does not create a new symbolic state.

For inputs other than the two shifted states considered below, $\mathbf{R}$ uses the following transitions:

$$0 \rightarrow 0, \quad 0_{\mathbf{s}} \rightarrow 0_{\mathbf{s}}, \quad \mathbf{x}, \mathbf{R}(\mathbf{x}), \mathbf{R}(\mathbf{x}) \oplus \mathbf{x} \rightarrow \mathbf{R}(\mathbf{x}), \quad \delta \rightarrow \delta.$$

Inputs not covered by these transitions or the two shifted cases below are mapped to $?$.

Only the two states $\mathbf{x} \oplus \delta$ and $0_{\mathbf{s}} \oplus \mathbf{x} \oplus \delta$ are resolved by comparing the exact difference with the prescribed period:

$$u \in \{\mathbf{x} \oplus \delta, 0_{\mathbf{s}} \oplus \mathbf{x} \oplus \delta\} \implies v = \begin{cases} 0_{\mathbf{s}}, & \Delta_u = \Delta_{\text{per}}, \\ ?, & \Delta_u \neq \Delta_{\text{per}}. \end{cases}$$

This is the only rule in $\mathcal{C}_{\text{link}}(\mathbf{s}, \Delta, \Delta_{\text{per}})$ where a new symbolic state is determined jointly by a symbolic state and an exact difference. Hence the full case distinction inside $\mathbf{R}$ can be written as

$$v = \begin{cases} 0, & u = 0, \\ 0_{\mathbf{s}}, & u = 0_{\mathbf{s}}, \\ \mathbf{R}(\mathbf{x}), & u \in \{\mathbf{x}, \mathbf{R}(\mathbf{x}), \mathbf{R}(\mathbf{x}) \oplus \mathbf{x}\}, \\ \delta, & u = \delta, \\ 0_{\mathbf{s}}, & u \in \{\mathbf{x} \oplus \delta, 0_{\mathbf{s}} \oplus \mathbf{x} \oplus \delta\} \text{ and } \Delta_u = \Delta_{\text{per}}, \\ ?, & u \in \{\mathbf{x} \oplus \delta, 0_{\mathbf{s}} \oplus \mathbf{x} \oplus \delta\} \text{ and } \Delta_u \neq \Delta_{\text{per}}, \\ ?, & \text{otherwise.} \end{cases}$$

In the generated SMT file, the same rule appears as one nested **IF-THEN-ELSE** over the explicit 6-bit encodings:

$$\begin{aligned} 0\text{bin}000000 &\mapsto 0\text{bin}000000, & 0\text{bin}010000 &\mapsto 0\text{bin}010000, \\ \{0\text{bin}000100, 0\text{bin}001000, 0\text{bin}001100\} &\mapsto 0\text{bin}001000, & 0\text{bin}000001 &\mapsto 0\text{bin}000001, \\ 0\text{bin}000101 &\mapsto \begin{cases} 0\text{bin}010000, & \Delta_u = \Delta_{\text{per}}, \\ 0\text{bin}100000, & \Delta_u \neq \Delta_{\text{per}}, \end{cases} \\ 0\text{bin}010101 &\mapsto \begin{cases} 0\text{bin}010000, & \Delta_u = \Delta_{\text{per}}, \\ 0\text{bin}100000, & \Delta_u \neq \Delta_{\text{per}}, \end{cases} \end{aligned}$$

and all remaining inputs are mapped to $0\text{bin}100000$. This is the symbolic branch of the nested **IF-THEN-ELSE** rule used in the implementation.

B.5 Build and Solve Order

The model generator constructs the TWINE constraint system in the following order.

1. Call `generate_round_state(0, ...)` to declare the round-0 branch states, the exact differences, the DDT array, the prescribed-period variable `perioddiff`, and the input tags.
2. For each round $i = 0, \dots, r - 1$, call `generate_round_state(i+1, ...)` and instantiate the 8 calls to `ROUNDFUNC`, `COPY`, and `XOR`.
3. Generate the wire and permutation equalities during the round loop by calls to `equal_connect`, and append them to the model file after all round-component constraints.
4. After all rounds, call `generate_Possibility` to impose $p \leq \tau$, and call `END` on the last round output, namely `END(r-1, ...)` for an r -round target.
5. Append `QUERY(FALSE);` and `COUNTEREXAMPLE;`, so that any solver solution of $\mathcal{M}_{\text{diff}}$ is returned as a counterexample.

After model generation, `run.py` invokes STP on `model.cvc` and saves its output to `solver_output.txt`. The branch states, exact differences, DDT-based probability constraints, and prescribed period are all declared before the final query and are solved together in one SMT constraint system.

B.6 Returned Solution

A solver solution of $\mathcal{M}_{\text{diff}}$ determines the symbolic states, exact differences, prescribed period Δ_{per} , and total DDT probability exponent p . An output branch satisfying the periodic condition is then identified from the final symbolic states. Thus no separate exact-difference instantiation step is needed after solving. Algorithm 1 searches for auxiliary tuples satisfying the resulting prescribed-period relation.

After the model returns a candidate, we can implement the corresponding construction and test it experimentally. If a rare unintended event causes the candidate to fail this check, we can rerun the model.

C SMT Encoding of the Key-Recovery Model

This supporting material gives the SMT encoding used for the key-recovery model in Section 4.4. The presentation follows the three-part construction in Section 4.3: the model first searches for a paired-shift core, then checks whether the required key-dependent values can be produced from the input side, and finally checks whether an equivalent periodic expression can be obtained from the output side. These three parts are encoded in one SMT formula.

C.1 Search Objective

The model targets the sufficient condition in Theorem 2. The searched expression has the form

$$f_K(x) = F_2(F(x \oplus \psi_1) \oplus F(x \oplus \psi_2)) \oplus F_0(x), \quad (5)$$

where F is the public function used in the target primitive, F_0 is the removable public part, and F_2 is fixed once the key is fixed. The values ψ_1 and ψ_2 are independent of x and may depend on subkeys. After cancelling $F_0(x)$, the resulting function has period

$$s_K = \psi_1 \oplus \psi_2.$$

In the implementation, the two abstract shifts are represented by the 10-bit tag variables

$$\Psi_1, \Psi_2 \in \{0, 1\}^{10},$$

with constraints

$$\Psi_1 \neq 0, \quad \Psi_1 \neq \Psi_2.$$

In the generated CVC files, these variables are named

$$\text{period_value1} \quad \text{and} \quad \text{period_value2}.$$

The condition $\Psi_1 \neq 0$ reserves the zero tag, when used, for Ψ_2 .

The round computation is split into input, core, and output parts. Let r_{in} , r_{core} , and r_{out} be their numbers of rounds. In the build scripts these values are the parameters `num_R_head`, `num_R_core`, and `num_R_linear`; below we write them as h , m , and ℓ . The core part searches for the core in Eq. (5). The input part realizes the values used in ψ_1, ψ_2 from the actual keyed computation. The output part, when present, checks whether the core can still be expressed from the observable output up to an x -independent key-dependent value.

C.2 Symbolic Circuit Representation

Operations. The model uses four operations.

- **F**: application of the public function F , or of F^{-1} in backward wiring when required by the target round equations.
- **XOR**: bitwise XOR of two branches.
- **COPY**: duplication of one branch into two branches.
- **ADK**: XOR of a round subkey into a branch, implemented by **AK** with variables `K_IN` and `K_OUT`.

The target-specific build file specifies the target wiring, the round split, the input-side branch count `head_pr`, and the boundary assertions for the selected entry. The propagation rules for the four operations are generated by the shared key-recovery model.

State alphabet. Each nonlinear state is encoded by a 7-bit vector

$$\sigma = (\sigma_6, \sigma_5, \dots, \sigma_0).$$

The bits correspond to the symbolic terms in Table 4. When the leftmost bit is zero, the lower six bits indicate the symbolic components present in a branch. Their combination and cancellation are governed by the propagation rules below. For example, $(0, 0, 0, 0, 0, 1, 1)$ represents $\mathbf{x} \oplus \mathbf{k}$. If the leftmost bit is one, the symbolic expression is no longer tracked and the remaining bits are ignored.

Key tags. Every nonlinear state also has a 10-bit key tag v . The tag records the x -independent key-dependent value attached to the $\mathbf{k}$ bit. The implementation enforces

$$\sigma_0 = 1 \Rightarrow v \neq 0, \quad \sigma_0 = 0 \Rightarrow v = 0.$$

Thus a branch without the $\mathbf{k}$ component cannot carry a hidden key tag. The ADK operation updates the tag by XOR with the tag assigned to the injected subkey. When the same tag appears twice, it cancels.

Assigned-value propagation uses an auxiliary array

$$\hat{F} : \{0, 1\}^{10} \rightarrow \{0, 1\}^{10}.$$

It is generated as a random permutation before solving and is fixed in the SMT file, together with the subkey tags. These assigned values serve as fingerprints for key-dependent expressions. The array updates tags in expressions such as $F(k)$, while the state alphabet handles the symbolic x -dependent part. In the generated CVC files, the array is named $\mathbf{F}$, distinct from the symbolic $\mathbf{F}$ operation in the build scripts. The 10-bit tags are internal search values, not the concrete periods in the verification experiments. Candidate expressions are checked separately against the target computation.

C.3 Core Part: Core Search

The core part searches for an internal branch satisfying the paired-shift condition. Its input is the state x^h , where $h = r_{\text{in}}$ in the generated file. The linking constraints first remove states that cannot start the core search:

$$x_j^h[6 : 5] = 0\text{bin}00.$$

If a shifted-call bit is already present at this boundary, the corresponding shift must be zero:

$$x_j^h[4 : 4] = 1 \Rightarrow \Psi_1 = 0, \quad x_j^h[2 : 2] = 1 \Rightarrow \Psi_2 = 0.$$

The state x^h must contain at least one useful x -dependent component:

$$\bigvee_j (x_j^h[1 : 1] \vee x_j^h[2 : 2] \vee x_j^h[3 : 3] \vee x_j^h[4 : 4]) = 1.$$

Rules of COPY, XOR, and ADK. The COPY operation copies both the state and the key tag. For XOR, if the two inputs are (σ, v) and (ρ, w) , and the output is (ω, z) , then

$$\begin{aligned}\omega_6 &= \sigma_6 \vee \rho_6, & \omega_5 &= \sigma_5 \vee \rho_5, & \omega_4 &= \sigma_4 \oplus \rho_4, \\ \omega_3 &= \sigma_3 \vee \rho_3, & \omega_2 &= \sigma_2 \oplus \rho_2, & \omega_1 &= \sigma_1 \oplus \rho_1.\end{aligned}$$

The k bit depends on the two key tags:

$$v = w \Rightarrow \omega_0 = \sigma_0 \oplus \rho_0, \quad v \neq w \Rightarrow \omega_0 = \sigma_0 \vee \rho_0,$$

and the output tag is

$$z = v \oplus w.$$

An ADK operation only changes the key-dependent part. If the injected subkey tag is a , then

$$v = a \Rightarrow \omega_0 = 0, \quad v \neq a \Rightarrow (z = v \oplus a, \omega_0 = 1).$$

Rules of F. The F operation is where the model recognizes the two shifted calls. For the zero state and x -independent key-dependent states, the rules are

$$(0, 0) \mapsto (0, 0), \quad (k, a) \mapsto (k, \widehat{F}(a)).$$

For the public variable, the rule depends on whether one of the two shifts is zero:

$$x \mapsto \begin{cases} F_0(x), & \Psi_1 \neq 0 \text{ and } \Psi_2 \neq 0, \\ F(x \oplus \psi_1), & \Psi_1 = 0, \\ F(x \oplus \psi_2), & \Psi_2 = 0. \end{cases}$$

For a state $x \oplus k$ with assigned value a , the tag decides whether the branch is one of the two shifted inputs:

$$(x \oplus k, a) \mapsto \begin{cases} F(x \oplus \psi_1), & a = \Psi_1, \\ F(x \oplus \psi_2), & a = \Psi_2, \\ ?, & a \notin \{\Psi_1, \Psi_2\}. \end{cases}$$

The paired-shift core is accepted by the rule

$$F(x \oplus \psi_1) \oplus F(x \oplus \psi_2) \mapsto F_2(T_K(x)).$$

The states $F_0(x)$ and $F_2(T_K(x))$ are preserved by another application of F. A state labelled $F(x \oplus \psi_i)$ with $\Psi_i = 0$ is rewritten as $F_0(x)$ after another application of F. A single shifted-call state with $\Psi_i \neq 0$ is mapped to ?. The state $x \oplus F_0(x)$ maps to $F_0(x)$. When $\Psi_2 = 0$, the XOR of any two or all three of x , $F_0(x)$, and $F(x \oplus \psi_2)$ also maps to $F_0(x)$. The implemented case table also maps $F_2(T_K(x)) \oplus k$ back to $F_2(T_K(x))$, and whenever the output contains the k bit, the output tag is updated by $\widehat{F}$. The XOR of the two shifted calls, optionally combined with $F_2(T_K(x))$, k ,

or both, maps to $F_2(T_k(x))$. Inputs not covered by these rules are mapped to ?, and a discarded input keeps the discarded bit set.

Accepted core patterns. When $\ell = 0$, the **END** constraint accepts the nonlinear part if at least one branch has a 7-bit state $\sigma = (\sigma_6, \dots, \sigma_0)$ satisfying one of the following patterns:

$$\begin{aligned} \sigma_6 = 0, \quad \sigma_5 = 1, \quad \sigma_4 = \sigma_2, \\ \sigma_6 = 0, \quad \sigma_4 = 1, \quad \sigma_2 = 1, \\ \sigma_6 = 0, \quad \sigma_4 = 1, \quad \Psi_1 \neq 0. \end{aligned}$$

These predicates select candidate output states. The corresponding expression, removable public term, and period are reconstructed and checked against the conditions in Definition 3.

C.4 Input Part: Input Realization

The input part links candidate symbolic initial states to the core boundary through h rounds of the target computation. The states and tags at the end of the input part are linked to the initial core states by

$$xhead_j^h = x_j^h, \quad xhead_k_j^h = x_k_j^h.$$

The same linking routine also fixes the number of input-side branches carrying the k bit at round 0:

$$\sum_j xhead_j^0[0 : 0] = head_pr.$$

The value **head_pr** is target-specific and is set separately for each target. The reconstructed initial-state map must be checked for public computability before the candidate is used.

The input part uses the same **COPY**, **XOR**, and **ADK** rules as the core part. The rule for **F** is stricter, because the input part is used only to construct admissible input-side values. If the input to **F** contains an unknown component, an x -independent key-dependent component, the core bit, or an incompatible shifted-call bit, then the model sets

$$end = 0.$$

Otherwise,

$$0 \mapsto 0, \quad \sigma \neq 0 \mapsto F_0(x).$$

Since the global constraint fixes **end** = 1, any path using one of the forbidden input-part transitions is rejected.

For the MISTY L-FK build, the default parameters are two input-side rounds, four core rounds, and no output-side rounds. The target-specific assertions

$$x_0^2 = 0000001, \quad x_1^2 = 0000011$$

force the core input to contain k and $x \oplus k$. The input-part constraints then decide whether these states are actually produced from the plaintext side and the preceding subkey additions.

C.5 Output Part: Output Realization

The output part checks whether the internal core can be expressed from output-side branches. If there is no output part, this is exactly the accepted-core condition above. If an output part is present, the model changes to a 2-bit state

$$\lambda = (\lambda_1, \lambda_0) \in \{0, 1\}^2.$$

The three meaningful states are

$$00, \quad 01, \quad 11.$$

The generated constraints enforce this alphabet by $\lambda_1 = 1 \Rightarrow \lambda_0 = 1$ and $\lambda_0 = 0 \Rightarrow \lambda_1 = 0$. The state 11 means that the branch is directly computable from the output. The state 01 means that the branch is computable up to an x -independent key-dependent value, which is sufficient for an equivalent periodic function. The state 00 means that the branch is unusable for the output oracle. When a subkey is added to a usable branch, the resulting branch is assigned state 01, since the added value is independent of x . At the ciphertext boundary, the solver assigns each observable branch a label in $\{00, 01, 11\}$. The label 11 retains exact computability, while lower labels discard information not needed by the selected expression. The output-side constraints propagate these labels backward to the core boundary.

At the transition from the core part to the output part, the generated formula introduces one mask bit $\mathbf{xlinearfirstmask}_j^{h+m}$ for each branch and imposes $\mathbf{MASKSUM} = 1$. Thus exactly one candidate branch is selected. A mask bit can be one only if the corresponding nonlinear state satisfies one of the following four predicates:

$$\begin{aligned} \sigma_6 = 0, \quad \sigma_5 = 1, \quad \sigma_4 = \sigma_2, \\ \sigma_6 = 0, \quad \sigma_4 = 1, \quad \sigma_2 = 1, \\ \sigma_6 = 0, \quad \sigma_4 = 1, \quad \Psi_1 \neq 0, \\ \sigma_6 = 0, \quad \sigma_2 = 1, \quad \Psi_2 \neq 0. \end{aligned}$$

The last predicate is the symmetric shifted-call case that is used only by the output-part mask. If a mask bit is one, the implementation asserts $\lambda_0 = 1$ for the selected branch.

The output-part rules are

$$\mathbf{COPY} : \lambda' = \lambda'' = \lambda,$$

$$\mathbf{F} : 00 \mapsto 00, \quad 01 \mapsto 00, \quad 11 \mapsto 11,$$

$$\mathbf{ADK} : 00 \mapsto 00, \quad 01 \mapsto 01, \quad 11 \mapsto 01.$$

For **XOR**, let $\lambda^{(1)} = (\lambda_1^{(1)}, \lambda_0^{(1)})$ and $\lambda^{(2)} = (\lambda_1^{(2)}, \lambda_0^{(2)})$ be the labels of the two input branches, and let $\lambda_{\text{out}} = (\lambda_{\text{out},1}, \lambda_{\text{out},0})$ be the output label. The superscript

distinguishes the two input branches, and the subscript denotes the bit in the 2-bit label. The rule is

$$\lambda_{\text{out},0} = \lambda_0^{(1)} \wedge \lambda_0^{(2)}, \quad \lambda_{\text{out},1} = \lambda_1^{(1)} \wedge \lambda_1^{(2)}.$$

These rules encode the following output-side reasoning. Applying **F** to a branch known only up to an x -independent value destroys output usability, because the unknown key-dependent value enters the input of the public function. The XOR output is usable only when both input branches are usable.

C.6 Complete SMT Formula

Although the construction is described as three parts, the implementation does not solve them separately. It builds one formula

$$\mathcal{M}_{\text{KR}} = \mathcal{C}_{\text{core}} \wedge \mathcal{C}_{\text{input}} \wedge \mathcal{C}_{\text{output}} \wedge \mathcal{C}_{\text{wire}} \wedge \mathcal{C}_{\text{tag}} \wedge \mathcal{C}_{\text{period}} \wedge \mathcal{C}_{\text{target}}.$$

Here $\mathcal{C}_{\text{core}}$ contains the core-part rules, $\mathcal{C}_{\text{input}}$ contains the input-part realization rules, and $\mathcal{C}_{\text{output}}$ contains the output-side equivalent function rules. The group $\mathcal{C}_{\text{wire}}$ connects operation nodes according to the target structure. The group $\mathcal{C}_{\text{tag}}$ enforces the key-tag discipline and subkey-tag updates. The group $\mathcal{C}_{\text{period}}$ fixes $\Psi_1 \neq 0$ and $\Psi_1 \neq \Psi_2$, implemented as assertions on `period_value1` and `period_value2`. The group $\mathcal{C}_{\text{target}}$ contains the assertions written directly by the target-specific build file, including fixed boundary states and, for the entries that use one, a fixed relation on $\Psi_1 \oplus \Psi_2$.

A satisfying assignment gives the symbolic states and key tags. The candidate input pattern, paired-shift core, and output expression are reconstructed from these assignments and checked against the target computation. The separate experiments instantiate the candidate expressions with randomly chosen keys and check the resulting period relation. An unsatisfiable formula excludes candidates only for the given round split, target constraints, assigned tags, auxiliary permutation, and encoded construction class.

After the model returns a candidate, we can implement the corresponding construction and test it experimentally. If a rare unintended event causes the candidate to fail this check, we can rerun the model.

D Distinguisher State Information

This supporting material lists additional round-state tables for the periodic distinguishers in Section 5.1.

D.1 LBlock branch convention

The LBlock and LBlock-S tables and paths use a 4-bit-branch notation for the official round of LBlock. In the specification, the 32-bit words satisfy

$$X_i = F(X_{i-1}, K_{i-1}) \oplus (X_{i-2} \ll 8).$$

A 32-bit word $U = U_7 \parallel \dots \parallel U_0$ is written in branch order $U[0], \dots, U[7]$, where $U[j] = U_j$. The input state of the displayed round is ordered as

$$X^i = (X_{i-1}[0], \dots, X_{i-1}[7], X_{i-2}[0], \dots, X_{i-2}[7]).$$

The model first forms the internal output state Y^i . Its first eight branches are copied from X_{i-1} . Its last eight branches form X_i and therefore include both the «8 term and the permutation P in the LBlock specification. If $(Z_0, \dots, Z_7)$ denotes the eight S-box outputs in our branch order, then the specified permutation gives

$$(U_0, \dots, U_7) = (Z_1, Z_3, Z_0, Z_2, Z_5, Z_7, Z_4, Z_6).$$

Thus the last eight branches of Y^i are

$$Y^i[8+j] = X_{i-2}[(j-2) \bmod 8] \oplus U_j, \quad 0 \leq j \leq 7.$$

The SMT model implements the same order by feeding round-function outputs 1, 3, 0, 2, 5, 7, 4, 6 into $Y^i[8], \dots, Y^i[15]$. The next input state is ordered as $X^{i+1} = (X_i, X_{i-1})$, so the round connection used in the model and in the figures is

$$X^{i+1}[j] = Y^i[(j+8) \bmod 16].$$

The round-state tables list nonzero entries of X^i , while output conditions such as $Y^{11}[6] = 0_s$ refer to the internal output state before this half swap.

D.2 ALLPC branch convention

The ALLPC table and path also use a 4-bit-branch notation. Each 8-bit branch X_j in the ALLPC specification is split into two 4-bit branches,

$$X_j = (X^i[2(7-j)], X^i[2(7-j)+1]), \quad 0 \leq j \leq 7.$$

The two 4-bit S-box inputs of the round function are therefore branches 14 and 15, which correspond to X_0 . In the specified F function, the two S-box outputs are crossed before being xored into the two halves of X_7 . The round-key additions are fixed XORs and do not change branch differences. For all non-final rounds, the model applies the branch permutation

$$X^{i+1}[\pi(j)] = Y^i[j], \quad \pi = (2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0, 1, 4, 5),$$

which is the 4-bit-branch form of the ALLPC branch order

$$(X_1, X_7, X_0, X_6, X_5, X_4, X_3, X_2).$$

In the final round, we follow the ALLPC specification and stop at Y^{25} , before applying the branch permutation. Hence the output condition $Y^{25}[13] = 0_s \oplus \mathbf{x} \oplus \delta$ is imposed on this internal state.

For this ALLPC instance, $E_j^{25}(x, \alpha_b, c)$ uses a public input mapping in which branch 3 receives $x \oplus \alpha_b[3]$, with $\alpha_1[3] = \alpha_0[3] \oplus 0\mathbf{x}2$. The other input branches carry the corresponding components of α_b or c . Algorithm 1 applies its quartet test to these inputs with prescribed period $0\mathbf{x}1$.

Table 10. Round states of the 11-round TWINE distinguisher.

Round i	Nonzero entries of X^i
1	$1 : \delta(0x5), 2 : \delta(0x2), 3 : \delta(0xa), 6 : \delta(0x6), 7 : \delta(0x5), 13 : \delta(0xf), 14 : \delta(0x6), 15 : x(0x0)$
2	$0 : \delta(0x5), 1 : \delta(0x2), 3 : \delta(0x6), 10 : \delta(0xf), 11 : \delta(0x6), 14 : x \oplus \delta(0x5)$
3	$4 : \delta(0x6), 5 : \delta(0x5), 9 : \delta(0xf), 11 : x \oplus \delta(0x5), 14 : 0_s(0x0)$
4	$2 : x \oplus \delta(0x5), 6 : \delta(0xf), 7 : \delta(0x6), 11 : 0_s(0x0), 14 : 0_s(0x0)$
5	$1 : x \oplus \delta(0x5), 2 : 0_s(0x0), 3 : \delta(0xf), 4 : 0_s(0x0), 11 : 0_s(0x0), 14 : 0_s(0x0)$
6	$0 : x \oplus \delta(0x5), 1 : 0_s(0x0), 2 : 0_s(0x0), 4 : 0_s \oplus \delta(0xf), 7 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 14 : 0_s(0x0)$
7	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 4 : 0_s(0x0), 5 : x \oplus \delta(0x5), 7 : 0_s \oplus \delta(0xf), 8 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : ?(0x0), 14 : 0_s(0x0), 15 : 0_s(0x0)$
8	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 4 : 0_s(0x0), 5 : 0_s(0x0), 6 : 0_s(0x0), 7 : 0_s(0x0), 8 : 0_s \oplus \delta(0xf), 9 : 0_s(0x0), 10 : ?(0x0), 11 : 0_s(0x0), 12 : 0_s \oplus x \oplus \delta(0x5), 13 : 0_s(0x0), 14 : 0_s(0x0), 15 : ?(0x0)$
9	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : ?(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 5 : 0_s(0x0), 6 : ?(0x0), 7 : 0_s(0x0), 8 : 0_s(0x0), 9 : ?(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : 0_s \oplus \delta(0xf), 14 : ?(0x0), 15 : 0_s \oplus x \oplus \delta(0x5)$
10	$0 : 0_s(0x0), 1 : ?(0x0), 2 : 0_s(0x0), 3 : ?(0x0), 4 : ?(0x0), 5 : 0_s(0x0), 6 : ?(0x0), 7 : 0_s(0x0), 8 : ?(0x0), 9 : 0_s(0x0), 10 : 0_s \oplus \delta(0xf), 11 : ?(0x0), 12 : 0_s(0x0), 13 : 0_s(0x0), 14 : ?(0x0), 15 : 0_s(0x0)$
11	$0 : ?(0x0), 1 : 0_s(0x0), 2 : ?(0x0), 3 : ?(0x0), 4 : ?(0x0), 5 : 0_s(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : ?(0x0), 9 : 0_s \oplus \delta(0xf), 10 : 0_s(0x0), 11 : ?(0x0), 12 : ?(0x0), 13 : ?(0x0), 14 : ?(0x0), 15 : 0_s(0x0)$
Prescribed period DDT-path probability estimate $\Delta_{\text{per}} = 0x5$ 2^{-14} Output condition Experimental probability $Y^{11}[10] = 0_s$ $2^{-13.98}$	

Table 11. Round states of the 11-round LBlock distinguisher.

Round i	Nonzero entries of X^i
1	$0 : \delta(0x1), 6 : \delta(0x4), 7 : \delta(0xa), 8 : x(0x0), 9 : \delta(0xb), 11 : \delta(0x2), 13 : \delta(0x4), 15 : \delta(0x6)$
2	$1 : \delta(0x6), 2 : x \oplus \delta(0x1), 3 : \delta(0xb), 8 : \delta(0x1), 14 : \delta(0x4), 15 : \delta(0xa)$
3	$2 : \delta(0x1), 3 : 0_s(0x0), 9 : \delta(0x6), 10 : x \oplus \delta(0x1), 11 : \delta(0xb)$
4	$1 : 0_s(0x0), 4 : x \oplus \delta(0x1), 5 : \delta(0xb), 10 : \delta(0x1), 11 : 0_s(0x0)$
5	$0 : 0_s(0x0), 5 : 0_s(0x0), 6 : 0_s(0x0), 9 : 0_s(0x0), 12 : x \oplus \delta(0x1), 13 : \delta(0xb)$
6	$2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 6 : x \oplus \delta(0x1), 7 : 0_s \oplus \delta(0xb), 8 : 0_s(0x0), 13 : 0_s(0x0), 14 : 0_s(0x0)$
7	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 5 : ?(0x0), 6 : 0_s(0x0), 7 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 14 : x \oplus \delta(0x1), 15 : 0_s \oplus \delta(0xb)$
8	$0 : 0_s \oplus x \oplus \delta(0x1), 1 : 0_s \oplus \delta(0xb), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : ?(0x0), 5 : 0_s(0x0), 6 : 0_s(0x0), 7 : 0_s(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 13 : ?(0x0), 14 : 0_s(0x0), 15 : 0_s(0x0)$
9	$0 : ?(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 5 : 0_s(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : 0_s \oplus x \oplus \delta(0x1), 9 : 0_s \oplus \delta(0xb), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : ?(0x0), 13 : 0_s(0x0), 14 : 0_s(0x0), 15 : 0_s(0x0)$
10	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : ?(0x0), 3 : 0_s \oplus \delta(0xb), 4 : 0_s(0x0), 5 : ?(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : ?(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : 0_s(0x0), 14 : ?(0x0), 15 : ?(0x0)$
11	$0 : ?(0x0), 1 : ?(0x0), 2 : ?(0x0), 3 : ?(0x0), 4 : ?(0x0), 5 : ?(0x0), 6 : 0_s(0x0), 7 : ?(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 10 : ?(0x0), 11 : 0_s \oplus \delta(0xb), 12 : 0_s(0x0), 13 : ?(0x0), 14 : ?(0x0), 15 : ?(0x0)$
Prescribed period DDT-path probability estimate $\Delta_{\text{per}} = 0x1$ 2^{-14} Output condition Experimental probability $Y^{11}[6] = 0_s$ $2^{-14.62}$	

Table 12. Round states of the 12-round LBlock distinguisher.

Round i	Nonzero entries of X^i
1	$1 : \delta(0x2), 3 : \delta(0x2), 8 : \delta(0xa), 10 : \delta(0x5), 11 : \delta(0xa), 13 : \delta(0xb), 14 : x(0x0), 15 : \delta(0x1)$
2	$0 : x \oplus \delta(0x2), 2 : \delta(0xa), 4 : \delta(0x5), 5 : \delta(0xa), 7 : \delta(0xb), 9 : \delta(0x2), 11 : \delta(0x2)$
3	$2 : 0_s(0x0), 4 : \delta(0x1), 6 : \delta(0x2), 8 : x \oplus \delta(0x2), 10 : \delta(0xa), 12 : \delta(0x5), 13 : \delta(0xa), 15 : \delta(0xb)$
4	$1 : \delta(0xb), 2 : x \oplus \delta(0x2), 3 : 0_s(0x0), 4 : \delta(0xa), 10 : 0_s(0x0), 12 : \delta(0x1), 14 : \delta(0x2)$
5	$1 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 9 : \delta(0xb), 10 : x \oplus \delta(0x2), 11 : 0_s(0x0), 12 : \delta(0xa)$
6	$0 : 0_s(0x0), 1 : 0_s(0x0), 3 : \delta(0xb), 4 : x \oplus \delta(0x2), 5 : 0_s(0x0), 6 : 0_s \oplus \delta(0xa), 9 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0)$
7	$0 : 0_s(0x0), 1 : \delta(0x1), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 5 : 0_s(0x0), 6 : 0_s(0x0), 7 : ?(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 11 : \delta(0xb), 12 : x \oplus \delta(0x2), 13 : 0_s(0x0), 14 : 0_s \oplus \delta(0xa)$
8	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 5 : ?(0x0), 6 : 0_s \oplus x \oplus \delta(0x2), 7 : 0_s(0x0), 8 : 0_s(0x0), 9 : \delta(0x1), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : 0_s(0x0), 14 : 0_s(0x0), 15 : ?(0x0)$
9	$0 : 0_s(0x0), 1 : ?(0x0), 2 : 0_s(0x0), 3 : 0_s \oplus \delta(0x1), 4 : ?(0x0), 5 : 0_s(0x0), 6 : 0_s(0x0), 7 : 0_s(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : ?(0x0), 14 : 0_s \oplus x \oplus \delta(0x2), 15 : 0_s(0x0)$
10	$0 : ?(0x0), 1 : ?(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 5 : 0_s(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : 0_s(0x0), 9 : ?(0x0), 10 : 0_s(0x0), 11 : 0_s \oplus \delta(0x1), 12 : ?(0x0), 13 : 0_s(0x0), 14 : 0_s(0x0), 15 : 0_s(0x0)$
11	$0 : ?(0x0), 1 : 0_s(0x0), 2 : ?(0x0), 3 : ?(0x0), 4 : 0_s(0x0), 5 : ?(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : ?(0x0), 9 : ?(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : 0_s(0x0), 14 : ?(0x0), 15 : ?(0x0)$
12	$0 : ?(0x0), 1 : ?(0x0), 2 : ?(0x0), 3 : ?(0x0), 4 : ?(0x0), 5 : ?(0x0), 6 : 0_s(0x0), 7 : ?(0x0), 8 : ?(0x0), 9 : 0_s(0x0), 10 : ?(0x0), 11 : ?(0x0), 12 : 0_s(0x0), 13 : ?(0x0), 14 : ?(0x0), 15 : ?(0x0)$
Prescribed period DDT-path probability estimate $\Delta_{\text{per}} = 0x2$ 2^{-24} Output condition Experimental probability $Y^{12}[6] = 0_s$ $2^{-22.77}$	

Table 13. Round states of the 11-round LBlock-S distinguisher.

Round i	Nonzero entries of X^i
1	$0 : \delta(0x1), 5 : \delta(0x9), 7 : \delta(0x1), 8 : x(0x0), 10 : \delta(0xc), 11 : \delta(0x1), 12 : \delta(0x1), 14 : \delta(0xb)$
2	$0 : \delta(0xb), 2 : x \oplus \delta(0x9), 6 : \delta(0x1), 8 : \delta(0x1), 13 : \delta(0x9), 15 : \delta(0x1)$
3	$1 : \delta(0x1), 3 : 0_s(0x0), 8 : \delta(0xb), 10 : x \oplus \delta(0x9), 14 : \delta(0x1)$
4	$1 : 0_s(0x0), 2 : \delta(0xb), 4 : x \oplus \delta(0x9), 9 : \delta(0x1), 11 : 0_s(0x0)$
5	$0 : 0_s(0x0), 5 : 0_s(0x0), 6 : 0_s(0x0), 9 : 0_s(0x0), 10 : \delta(0xb), 12 : x \oplus \delta(0x9)$
6	$2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s \oplus \delta(0xb), 6 : x \oplus \delta(0x9), 7 : 0_s(0x0), 8 : 0_s(0x0), 13 : 0_s(0x0), 14 : 0_s(0x0)$
7	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 5 : 0_s(0x0), 6 : ?(0x0), 7 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s \oplus \delta(0xb), 14 : x \oplus \delta(0x9), 15 : 0_s(0x0)$
8	$0 : 0_s \oplus x \oplus \delta(0x9), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 5 : 0_s(0x0), 6 : 0_s \oplus \delta(0xb), 7 : ?(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 13 : 0_s(0x0), 14 : ?(0x0), 15 : 0_s(0x0)$
9	$0 : ?(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 5 : ?(0x0), 6 : 0_s(0x0), 7 : ?(0x0), 8 : 0_s \oplus x \oplus \delta(0x9), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : 0_s(0x0), 14 : 0_s \oplus \delta(0xb), 15 : ?(0x0)$
10	$0 : 0_s \oplus \delta(0xb), 1 : ?(0x0), 2 : ?(0x0), 3 : 0_s(0x0), 4 : ?(0x0), 5 : ?(0x0), 6 : 0_s(0x0), 7 : 0_s(0x0), 8 : ?(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : ?(0x0), 14 : 0_s(0x0), 15 : ?(0x0)$
11	$0 : ?(0x0), 1 : ?(0x0), 2 : ?(0x0), 3 : ?(0x0), 4 : ?(0x0), 5 : 0_s(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : 0_s \oplus \delta(0xb), 9 : ?(0x0), 10 : ?(0x0), 11 : 0_s(0x0), 12 : ?(0x0), 13 : ?(0x0), 14 : 0_s(0x0), 15 : 0_s(0x0)$
Prescribed period DDT-path probability estimate $\Delta_{\text{per}} = 0x9$ 2^{-14} Output condition Experimental probability $Y^{11}[5] = 0_s$ $2^{-13.97}$	

Table 14. Round states of the 12-round LBlock-S distinguisher.

Round i	Nonzero entries of X^i
1	$5 : \delta(0x4), 7 : \delta(0x4), 9 : \delta(0x6), 10 : \delta(0x6), 11 : x(0x0), 12 : \delta(0x4), 14 : \delta(0x4), 15 : \delta(0x4)$
2	$0 : \delta(0x4), 1 : \delta(0x4), 3 : \delta(0x6), 5 : x \oplus \delta(0x6), 6 : \delta(0x4), 13 : \delta(0x4), 15 : \delta(0x4)$
3	$0 : \delta(0x6), 2 : \delta(0x4), 4 : 0_s(0x0), 8 : \delta(0x4), 9 : \delta(0x4), 11 : \delta(0x6), 13 : x \oplus \delta(0x6), 14 : \delta(0x4)$
4	$0 : \delta(0x4), 5 : \delta(0x6), 6 : 0_s(0x0), 7 : x \oplus \delta(0x6), 8 : \delta(0x6), 10 : \delta(0x4), 12 : 0_s(0x0)$
5	$5 : 0_s(0x0), 6 : 0_s(0x0), 7 : 0_s(0x0), 8 : \delta(0x4), 13 : \delta(0x6), 14 : 0_s(0x0), 15 : x \oplus \delta(0x6)$
6	$0 : 0_s(0x0), 1 : x \oplus \delta(0x6), 2 : \delta(0x4), 4 : 0_s(0x0), 5 : 0_s(0x0), 7 : 0_s \oplus \delta(0x6), 13 : 0_s(0x0), 14 : 0_s(0x0), 15 : 0_s(0x0)$
7	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : \delta(0x4), 4 : 0_s(0x0), 5 : ?(0x0), 6 : 0_s(0x0), 7 : 0_s(0x0), 8 : 0_s(0x0), 9 : x \oplus \delta(0x6), 10 : \delta(0x4), 12 : 0_s(0x0), 13 : 0_s(0x0), 15 : 0_s \oplus \delta(0x6)$
8	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s \oplus x \oplus \delta(0x6), 4 : ?(0x0), 5 : 0_s(0x0), 6 : 0_s(0x0), 7 : 0_s(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : \delta(0x4), 12 : 0_s(0x0), 13 : ?(0x0), 14 : 0_s(0x0), 15 : 0_s(0x0)$
9	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : 0_s(0x0), 5 : 0_s \oplus \delta(0x4), 6 : ?(0x0), 7 : ?(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s \oplus x \oplus \delta(0x6), 12 : ?(0x0), 13 : 0_s(0x0), 14 : 0_s(0x0), 15 : 0_s(0x0)$
10	$0 : 0_s(0x0), 1 : 0_s(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : ?(0x0), 5 : ?(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : 0_s \oplus \delta(0x4), 14 : ?(0x0), 15 : ?(0x0)$
11	$0 : ?(0x0), 1 : ?(0x0), 2 : 0_s(0x0), 3 : 0_s(0x0), 4 : ?(0x0), 5 : ?(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : 0_s(0x0), 9 : 0_s(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : ?(0x0), 13 : ?(0x0), 14 : ?(0x0), 15 : ?(0x0)$
12	$0 : ?(0x0), 1 : ?(0x0), 2 : ?(0x0), 3 : 0_s(0x0), 4 : ?(0x0), 5 : ?(0x0), 6 : ?(0x0), 7 : ?(0x0), 8 : ?(0x0), 9 : ?(0x0), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : ?(0x0), 13 : ?(0x0), 14 : ?(0x0), 15 : ?(0x0)$
Prescribed period DDT-path probability estimate $\Delta_{\text{per}} = 0x6$ 2^{-24} Output condition Experimental probability $Y^{12}[3] = 0_s$ $2^{-21.34}$	

Table 15. Round states of the full-round ALLPC distinguisher.

Round i	Nonzero entries of X^i
1	$0 : \delta(0x2), 1 : \delta(0x3), 3 : x \oplus \delta(0x2), 5 : \delta(0x3), 7 : \delta(0x3), 8 : \delta(0x2), 12 : \delta(0x2)$
2	$0 : \delta(0x2), 2 : \delta(0x2), 3 : \delta(0x3), 6 : \delta(0x2), 7 : x \oplus \delta(0x2), 8 : \delta(0x2), 9 : \delta(0x3), 10 : \delta(0x2), 11 : \delta(0x3), 12 : \delta(0x2)$
3	$0 : \delta(0x2), 8 : \delta(0x2), 9 : \delta(0x3), 11 : x \oplus \delta(0x1), 12 : \delta(0x2), 13 : \delta(0x3), 14 : \delta(0x2), 15 : \delta(0x3)$
4	$0 : \delta(0x2), 1 : \delta(0x3), 3 : x \oplus \delta(0x2), 4 : \delta(0x2), 5 : \delta(0x3), 6 : \delta(0x2), 7 : \delta(0x3), 8 : \delta(0x2), 9 : \delta(0x3), 10 : \delta(0x2), 11 : \delta(0x3), 12 : \delta(0x2), 13 : \delta(0x3), 15 : x \oplus \delta(0x1)$
5	$0 : \delta(0x2), 1 : \delta(0x3), 2 : 0_s(0x0), 5 : x \oplus \delta(0x1), 6 : \delta(0x2), 7 : x \oplus \delta(0x1), 12 : \delta(0x2), 13 : \delta(0x3), 14 : \delta(0x2), 15 : \delta(0x3)$
6	$0 : \delta(0x2), 1 : \delta(0x3), 2 : \delta(0x2), 3 : \delta(0x3), 4 : \delta(0x2), 5 : \delta(0x3), 6 : 0_s(0x0), 9 : x \oplus \delta(0x1), 10 : \delta(0x2), 11 : x \oplus \delta(0x1)$
7	$3 : \delta(0x3), 6 : \delta(0x2), 7 : x \oplus \delta(0x2), 8 : \delta(0x2), 9 : x \oplus \delta(0x2), 10 : 0_s(0x0), 11 : x \oplus \delta(0x1), 13 : x \oplus \delta(0x1), 14 : \delta(0x2), 15 : x \oplus \delta(0x1)$
8	$1 : x \oplus \delta(0x1), 2 : 0_s \oplus \delta(0x2), 3 : x \oplus \delta(0x1), 4 : \delta(0x2), 5 : x \oplus \delta(0x1), 6 : \delta(0x2), 7 : x \oplus \delta(0x1), 8 : \delta(0x2), 9 : x \oplus \delta(0x2), 12 : \delta(0x2), 13 : x \oplus \delta(0x2), 14 : 0_s(0x0), 15 : x \oplus \delta(0x1)$
9	$0 : \delta(0x2), 1 : x \oplus \delta(0x2), 2 : 0_s(0x0), 3 : 0_s \oplus x \oplus \delta(0x1), 4 : 0_s(0x0), 5 : x \oplus \delta(0x1), 6 : 0_s(0x0), 7 : \delta(0x3), 9 : \delta(0x3), 11 : \delta(0x3), 12 : \delta(0x2), 13 : x \oplus \delta(0x2)$
10	$0 : \delta(0x2), 1 : x \oplus \delta(0x2), 6 : 0_s(0x0), 7 : 0_s \oplus x \oplus \delta(0x2), 8 : 0_s(0x0), 9 : x \oplus \delta(0x2), 10 : 0_s(0x0), 13 : \delta(0x3), 15 : \delta(0x3)$
11	$1 : \delta(0x3), 2 : 0_s(0x0), 3 : \delta(0x3), 5 : \delta(0x3), 6 : 0_s(0x0), 7 : x \oplus \delta(0x2), 8 : 0_s(0x0), 9 : x \oplus \delta(0x2), 10 : 0_s(0x0), 11 : 0_s(0x0), 12 : 0_s(0x0), 13 : x \oplus \delta(0x2), 14 : 0_s(0x0)$
12	$0 : 0_s(0x0), 1 : x \oplus \delta(0x2), 2 : 0_s(0x0), 3 : 0_s \oplus \delta(0x3), 4 : 0_s(0x0), 6 : 0_s(0x0), 7 : x \oplus \delta(0x1), 8 : 0_s(0x0), 9 : x \oplus \delta(0x1), 10 : 0_s(0x0), 12 : 0_s(0x0), 13 : x \oplus \delta(0x2), 14 : 0_s(0x0), 15 : 0_s(0x0)$
13	$0 : 0_s(0x0), 1 : x \oplus \delta(0x2), 2 : 0_s(0x0), 3 : 0_s \oplus x \oplus \delta(0x1), 4 : 0_s(0x0), 5 : 0_s(0x0), 6 : 0_s(0x0), 7 : 0_s \oplus x \oplus \delta(0x2), 8 : 0_s(0x0), 9 : x \oplus \delta(0x1), 10 : 0_s(0x0), 12 : 0_s(0x0), 13 : x \oplus \delta(0x1), 14 : 0_s(0x0)$

Table 16. Round states of the full-round ALLPC distinguisher, rounds 14–25.

Round i	Nonzero entries of X^i
14	0 : $0_s(0x0)$, 1 : $x \oplus \delta(0x1)$, 2 : $0_s(0x0)$, 3 : $0_s \oplus x \oplus \delta(0x2)$, 4 : $0_s(0x0)$, 6 : $0_s(0x0)$, 7 : $0_s(0x0)$, 8 : $0_s(0x0)$, 9 : $0_s \oplus x \oplus \delta(0x1)$, 10 : $0_s(0x0)$, 11 : $0_s \oplus \delta(0x3)$, 12 : $0_s(0x0)$, 13 : $x \oplus \delta(0x1)$, 14 : $0_s(0x0)$
15	0 : $0_s(0x0)$, 1 : $x \oplus \delta(0x1)$, 2 : $0_s(0x0)$, 3 : $0_s \oplus x \oplus \delta(0x2)$, 4 : $0_s(0x0)$, 6 : $0_s(0x0)$, 7 : $0_s \oplus \delta(0x3)$, 8 : $0_s(0x0)$, 9 : $0_s \oplus x \oplus \delta(0x1)$, 10 : $0_s(0x0)$, 11 : $0_s \oplus x \oplus \delta(0x1)$, 12 : $0_s(0x0)$, 13 : $0_s \oplus x \oplus \delta(0x1)$, 14 : $0_s(0x0)$, 15 : $0_s \oplus \delta(0x3)$
16	0 : $0_s(0x0)$, 1 : $0_s \oplus x \oplus \delta(0x1)$, 2 : $? (0x0)$, 3 : $0_s(0x0)$, 4 : $0_s(0x0)$, 5 : $0_s \oplus \delta(0x3)$, 6 : $0_s(0x0)$, 7 : $0_s \oplus \delta(0x3)$, 8 : $0_s(0x0)$, 9 : $0_s \oplus x \oplus \delta(0x1)$, 10 : $0_s(0x0)$, 11 : $0_s \oplus x \oplus \delta(0x2)$, 12 : $0_s(0x0)$, 13 : $0_s \oplus x \oplus \delta(0x1)$, 14 : $0_s(0x0)$, 15 : $0_s \oplus x \oplus \delta(0x1)$
17	0 : $0_s(0x0)$, 1 : $0_s \oplus x \oplus \delta(0x1)$, 2 : $0_s(0x0)$, 3 : $0_s \oplus \delta(0x3)$, 4 : $0_s(0x0)$, 5 : $0_s \oplus x \oplus \delta(0x1)$, 6 : $? (0x0)$, 7 : $0_s \oplus x \oplus \delta(0x1)$, 8 : $0_s(0x0)$, 9 : $0_s \oplus x \oplus \delta(0x2)$, 10 : $0_s(0x0)$, 11 : $0_s \oplus x \oplus \delta(0x2)$, 12 : $0_s(0x0)$, 13 : $0_s \oplus x \oplus \delta(0x1)$, 14 : $0_s(0x0)$, 15 : $0_s \oplus x \oplus \delta(0x2)$
18	0 : $0_s(0x0)$, 1 : $0_s \oplus x \oplus \delta(0x1)$, 2 : $? (0x0)$, 3 : $0_s(0x0)$, 4 : $0_s(0x0)$, 5 : $0_s \oplus x \oplus \delta(0x2)$, 6 : $0_s(0x0)$, 7 : $0_s \oplus x \oplus \delta(0x1)$, 8 : $0_s(0x0)$, 9 : $0_s \oplus \delta(0x3)$, 10 : $? (0x0)$, 11 : $0_s \oplus \delta(0x3)$, 12 : $0_s(0x0)$, 13 : $0_s \oplus x \oplus \delta(0x2)$, 14 : $0_s(0x0)$, 15 : $0_s \oplus x \oplus \delta(0x2)$
19	0 : $0_s(0x0)$, 1 : $0_s \oplus x \oplus \delta(0x2)$, 2 : $? (0x0)$, 3 : $0_s \oplus \delta(0x3)$, 4 : $0_s(0x0)$, 5 : $0_s \oplus x \oplus \delta(0x2)$, 6 : $? (0x0)$, 7 : $0_s \oplus \delta(0x3)$, 8 : $0_s(0x0)$, 9 : $0_s \oplus x \oplus \delta(0x1)$, 10 : $0_s(0x0)$, 11 : $0_s \oplus x \oplus \delta(0x2)$, 12 : $0_s(0x0)$, 13 : $0_s \oplus \delta(0x3)$, 14 : $? (0x0)$, 15 : $0_s \oplus \delta(0x3)$
20	0 : $0_s(0x0)$, 1 : $0_s \oplus \delta(0x3)$, 2 : $? (0x0)$, 3 : $? (0x0)$, 4 : $? (0x0)$, 5 : $0_s \oplus \delta(0x3)$, 6 : $? (0x0)$, 7 : $0_s \oplus x \oplus \delta(0x2)$, 8 : $0_s(0x0)$, 9 : $0_s \oplus \delta(0x3)$, 10 : $? (0x0)$, 11 : $0_s \oplus x \oplus \delta(0x2)$, 12 : $0_s(0x0)$, 13 : $0_s \oplus x \oplus \delta(0x1)$, 14 : $0_s(0x0)$, 15 : $0_s \oplus x \oplus \delta(0x2)$
21	0 : $0_s(0x0)$, 1 : $0_s \oplus x \oplus \delta(0x1)$, 2 : $? (0x0)$, 3 : $0_s \oplus \delta(0x3)$, 4 : $0_s(0x0)$, 5 : $0_s \oplus x \oplus \delta(0x2)$, 6 : $? (0x0)$, 7 : $? (0x0)$, 8 : $? (0x0)$, 9 : $0_s(0x0)$, 10 : $? (0x0)$, 11 : $0_s \oplus x \oplus \delta(0x1)$, 12 : $0_s(0x0)$, 13 : $0_s \oplus \delta(0x3)$, 14 : $? (0x0)$, 15 : $0_s \oplus x \oplus \delta(0x2)$
22	0 : $0_s(0x0)$, 1 : $0_s \oplus \delta(0x3)$, 2 : $? (0x0)$, 3 : $? (0x0)$, 4 : $? (0x0)$, 5 : $0_s \oplus x \oplus \delta(0x2)$, 6 : $? (0x0)$, 7 : $0_s \oplus \delta(0x3)$, 8 : $? (0x0)$, 9 : $0_s \oplus x \oplus \delta(0x2)$, 10 : $? (0x0)$, 11 : $? (0x0)$, 12 : $? (0x0)$, 13 : $0_s(0x0)$, 14 : $? (0x0)$, 15 : $0_s \oplus x \oplus \delta(0x1)$
23	0 : $? (0x0)$, 1 : $0_s(0x0)$, 2 : $? (0x0)$, 3 : $? (0x0)$, 4 : $? (0x0)$, 5 : $0_s \oplus x \oplus \delta(0x1)$, 6 : $? (0x0)$, 7 : $? (0x0)$, 8 : $? (0x0)$, 9 : $0_s(0x0)$, 10 : $? (0x0)$, 11 : $0_s \oplus x \oplus \delta(0x1)$, 12 : $? (0x0)$, 13 : $0_s \oplus x \oplus \delta(0x2)$, 14 : $? (0x0)$, 15 : $? (0x0)$
24	0 : $? (0x0)$, 1 : $0_s \oplus x \oplus \delta(0x2)$, 2 : $? (0x0)$, 3 : $? (0x0)$, 4 : $? (0x0)$, 5 : $? (0x0)$, 6 : $? (0x0)$, 7 : $? (0x0)$, 8 : $? (0x0)$, 9 : $0_s \oplus x \oplus \delta(0x1)$, 10 : $? (0x0)$, 11 : $? (0x0)$, 12 : $? (0x0)$, 13 : $0_s(0x0)$, 14 : $? (0x0)$, 15 : $0_s \oplus x \oplus \delta(0x1)$
25	0 : $? (0x0)$, 1 : $0_s(0x0)$, 2 : $? (0x0)$, 3 : $? (0x0)$, 4 : $? (0x0)$, 5 : $0_s \oplus x \oplus \delta(0x1)$, 6 : $? (0x0)$, 7 : $? (0x0)$, 8 : $? (0x0)$, 9 : $? (0x0)$, 10 : $? (0x0)$, 11 : $? (0x0)$, 12 : $? (0x0)$, 13 : $0_s \oplus x \oplus \delta(0x1)$, 14 : $? (0x0)$, 15 : $? (0x0)$
Prescribed period $\Delta_{\text{per}} = 0x1$ Output condition $Y^{25}[13] = 0_s \oplus x \oplus \delta$	
DDT-path probability estimate 2^{-12} Experimental probability $2^{-9.06}$	

E Distinguisher Path Information

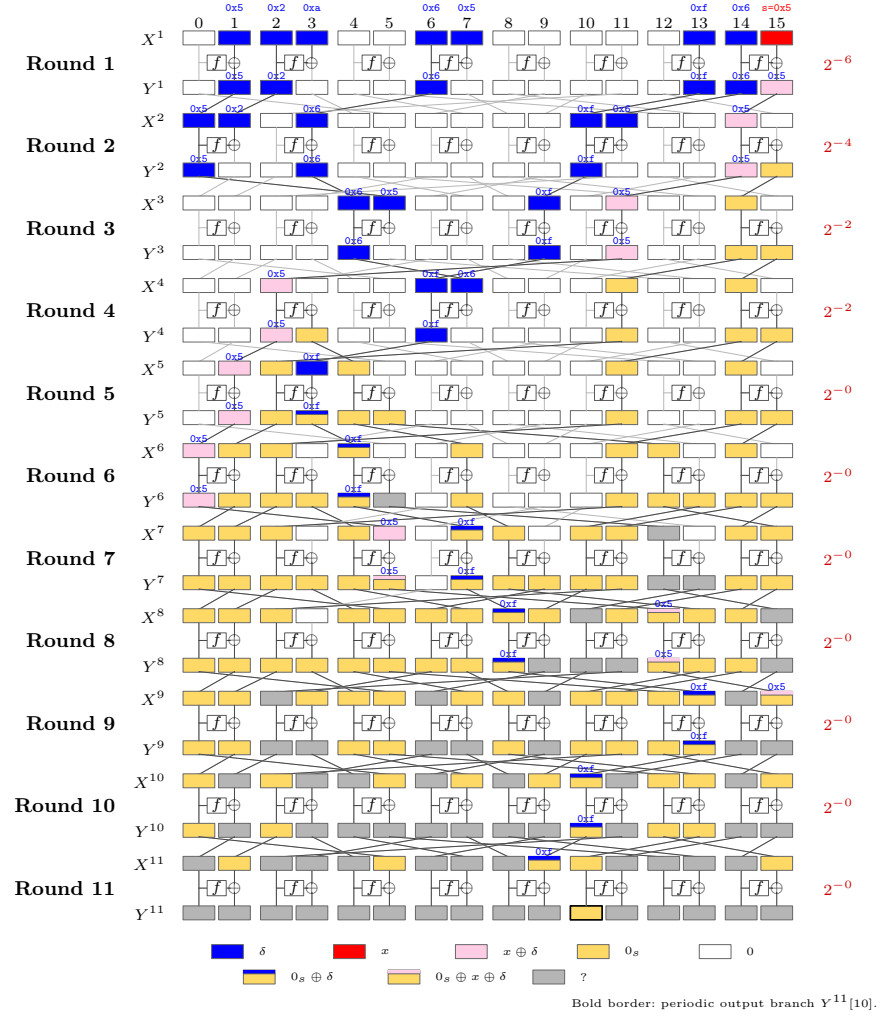


Fig. 2. 11-round periodic distinguisher for TWINE.

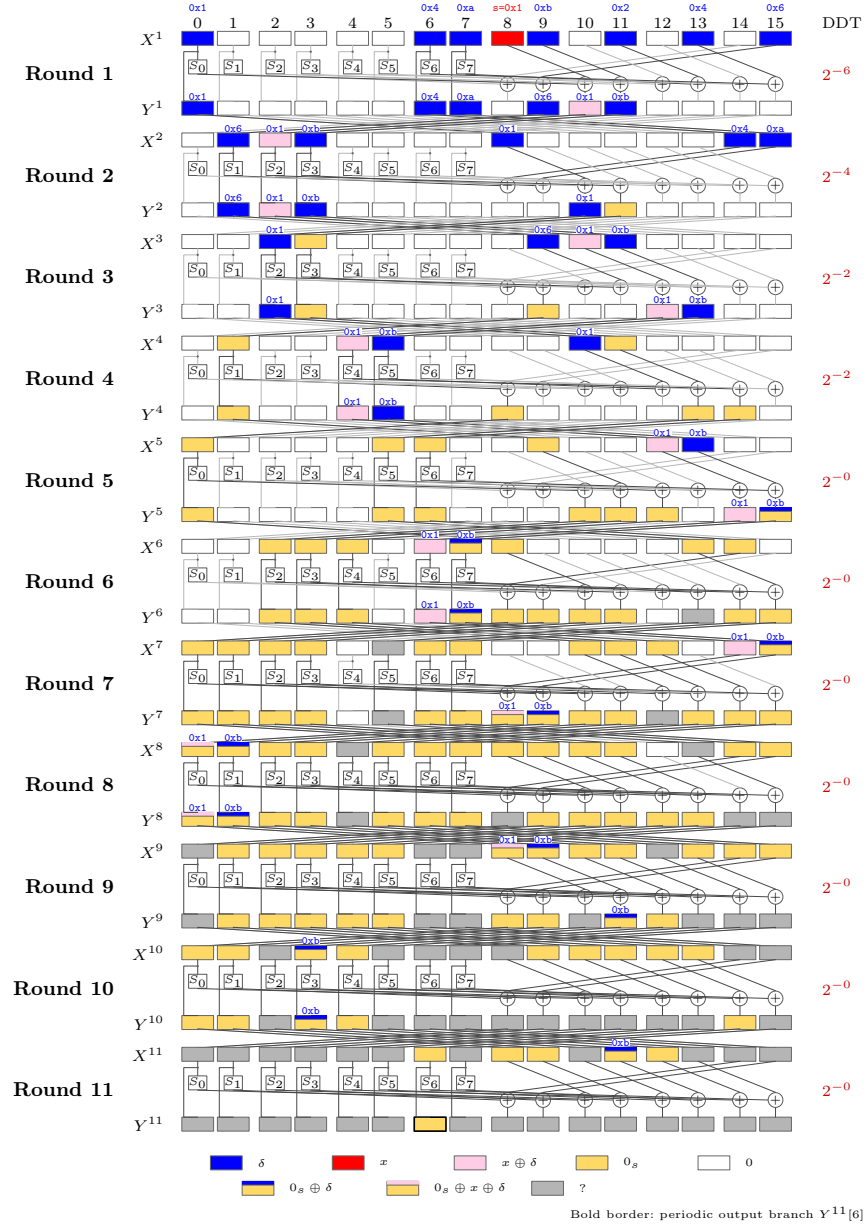


Fig. 3. 11-round periodic distinguisher for LBlock.

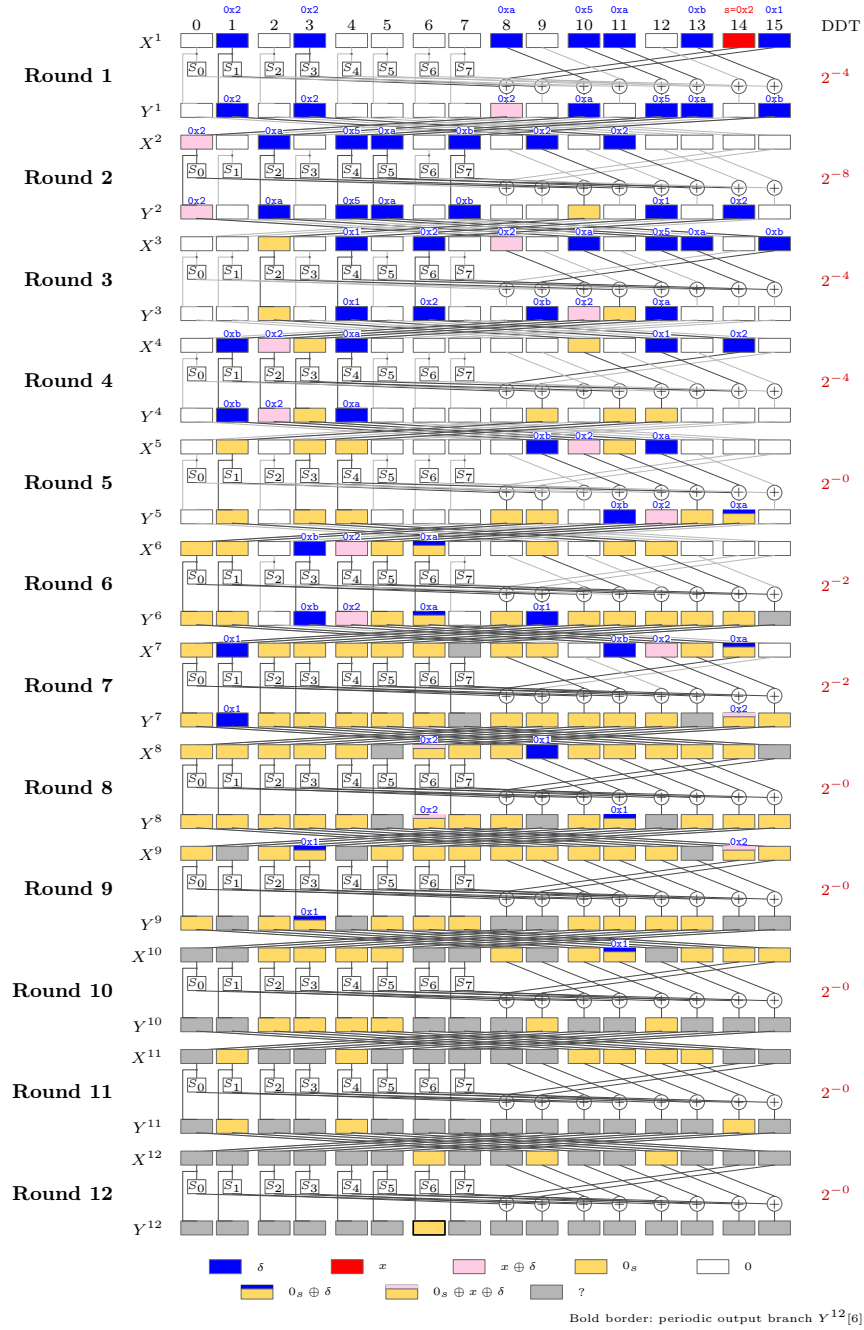


Fig. 4. 12-round periodic distinguisher for LBlock.

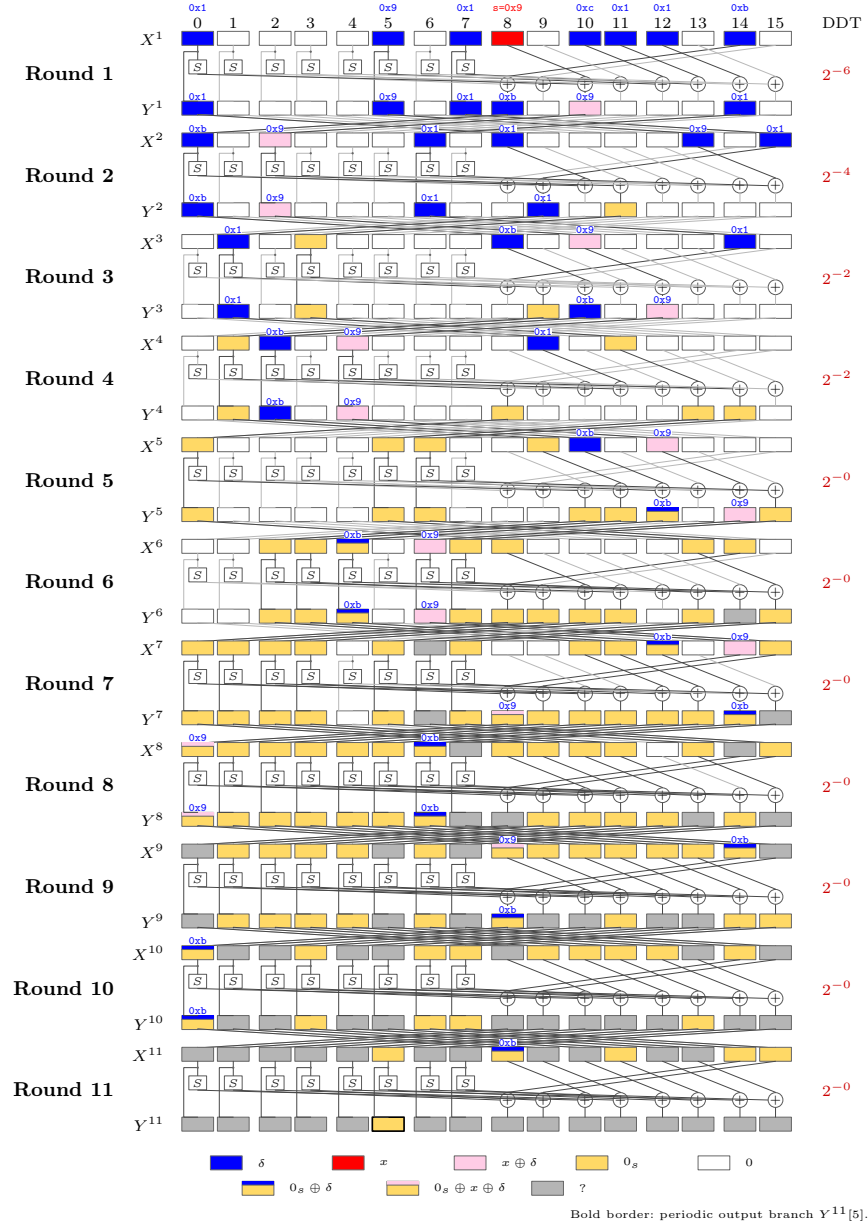


Fig. 5. 11-round periodic distinguisher for LBlock-S.

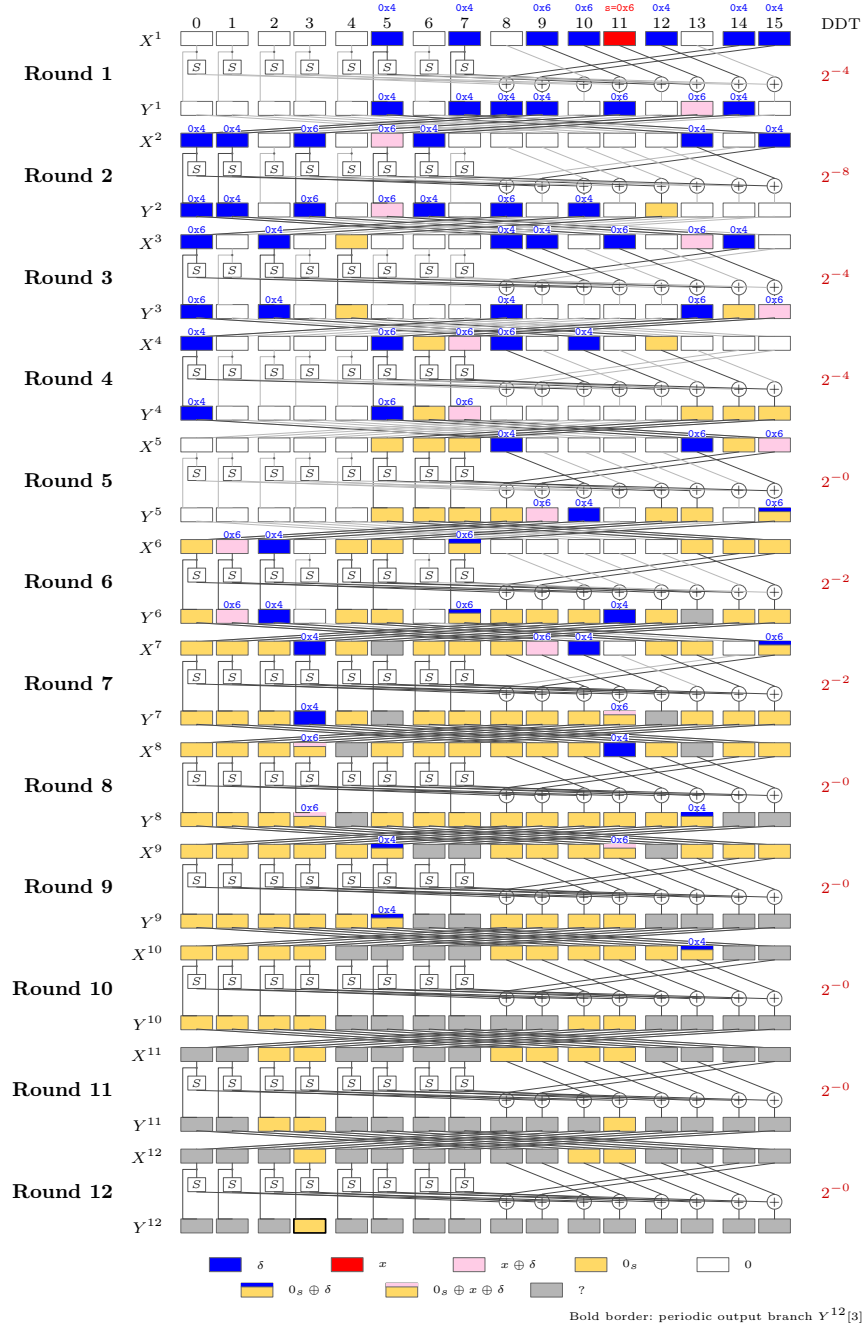


Fig. 6. 12-round periodic distinguisher for LBlock-S.

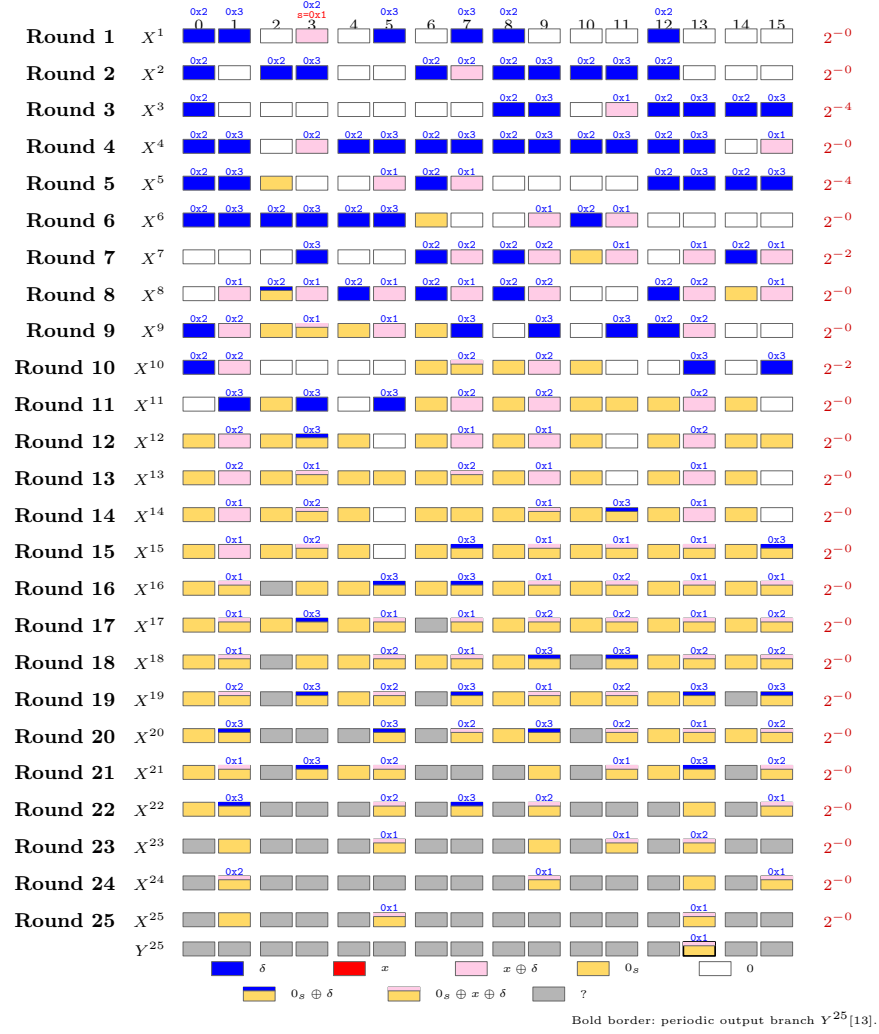


Fig. 7. Full-round periodic distinguisher for ALLPC.

F Key-Recovery Attacks

This supporting material records the remaining key-dependent periodic functions not detailed in Section 5.2 and explains how they are used in the corresponding polynomial-time key-recovery attacks. We use the same notation as in Section 5.2. In particular, E_j^r denotes branch j after r rounds, and F^i and F^{-i} denote i -fold iteration of F and F^{-1} .

F.1 MISTY L-KF

For 5-round MISTY L-KF, let the input be $(0, x)$. Define

$$f(x) = x_L^4 \oplus k^4 = F^{-1}(x_L^5 \oplus x_R^5).$$

Then

$$f(x) = F(x \oplus F(k^0) \oplus k^2) \oplus F(x \oplus k^1) \oplus x \oplus F(k^0) \oplus k^4.$$

Thus $G(x) = f(x) \oplus x$ has period

$$s = F(k^0) \oplus k^1 \oplus k^2.$$

This period is again a subkey relation. To recover k^0 , set the public input to (y, x) and define

$$f_y(x) = F(x \oplus F(y \oplus k^0) \oplus k^2) \oplus F(x \oplus k^1) \oplus x \oplus F(y \oplus k^0) \oplus k^4.$$

For each fixed y , $G_y(x) = f_y(x) \oplus x$ has period

$$g(y) = F(y \oplus k^0) \oplus k^1 \oplus k^2.$$

When $g(y) = 0$, the inner oracle G_y is constant. Since F is a permutation, the equation $g(y) = 0$ has exactly one solution. The inner procedure returns zero for this input. As in the L-FK attack, the inner procedure evaluates $g(y)$ coherently for the outer Simon call. Thus,

$$H(y) = g(y) \oplus F(y) = F(y \oplus k^0) \oplus F(y) \oplus k^1 \oplus k^2$$

has period k^0 . The nested Simon procedure therefore recovers the first subkey, except for the usual outer degeneracies for which the Simon promise fails.

After k^0 is recovered, the first round can be removed. The first MISTY L-KF round maps an input (a, b) to

$$(b, b \oplus F(a \oplus k^0)).$$

Thus, to query the remaining suffix on a chosen state (z_L, z_R) , one queries the original oracle on

$$(F^{-1}(z_L \oplus z_R) \oplus k^0, z_L).$$

This gives oracle access to the 4-round suffix with round keys $k^1, \dots, k^4$. To reuse the same 5-round periodic function, append one public MISTY L-KF round with zero key to this suffix. The effective 5-round oracle has round keys $k^1, \dots, k^4, 0$, so the nested Simon procedure recovers k^1 . Repeating this step, appending enough zero-key public rounds to keep an effective 5-round oracle, recovers k^2, k^3, k^4 in turn.

F.2 MISTY R-KF

The 5-round MISTY R-KF construction gives a direct period equal to the first round key. For a public input $(x, F(x))$, let

$$y_L = E_L^5(x, F(x)), \quad y_R = E_R^5(x, F(x)).$$

The attacker can compute

$$G(x) = F^{-1}(y_L \oplus y_R) \oplus y_L \oplus y_R \oplus F^{-1}(y_R)$$

from the two output branches. Expanding the rounds gives

$$G(x) = F(A(x) \oplus k^1) \oplus k^3 \oplus k^4, \quad A(x) = F(x \oplus k^0) \oplus F(x).$$

Since $A(x \oplus k^0) = A(x)$, the function G has period k^0 . Simon's algorithm therefore recovers k^0 , except for degenerate keys for which the Simon promise fails.

After k^0 is recovered, the first round can be removed. Indeed, the first MISTY R-KF round maps an input (a, b) to

$$(b \oplus F(a \oplus k^0), F(a \oplus k^0)).$$

Thus, to query the remaining suffix on a chosen state (z_L, z_R) , one queries the original oracle on

$$(F^{-1}(z_R) \oplus k^0, z_L \oplus z_R).$$

This gives oracle access to the 4-round suffix with round keys $k^1, \dots, k^4$. To reuse the same 5-round periodic function, append one public MISTY R-KF round with zero key to this suffix. The effective 5-round oracle has round keys $k^1, \dots, k^4, 0$, so the same Simon call recovers k^1 . Repeating this step, appending enough zero-key public rounds to keep an effective 5-round oracle, recovers k^2, k^3, k^4 in turn.

F.3 Type-1 GFS-FK

The Type-1 GFS-FK result uses the decryption structure as the search target, because this direction gives a longer periodic function than the corresponding encryption structure. For the 17-round target, the round update is

$$(x_0^{i+1}, x_1^{i+1}, x_2^{i+1}, x_3^{i+1}) = (x_3^i, x_0^i \oplus F(x_3^i) \oplus k^i, x_1^i, x_2^i).$$

The public input used by the periodic function is

$$u(x) = (F^4(x) \oplus x, F(x), F^2(x), F^3(x)).$$

Let

$$(Y_0, Y_1, Y_2, Y_3) = E^{17}(u(x)).$$

The attacker computes the output expression

$$G(x) = F(Y_0) \oplus Y_1 \oplus x.$$

To see its period, set

$$A(x) = F(x \oplus k^0) \oplus F(x) \oplus k^3.$$

Expanding the 17 rounds gives

$$\begin{aligned} G(x) = & F(F(F(A(x)) \oplus k^2 \oplus k^6) \oplus F(k^2) \oplus k^1 \oplus k^5 \oplus k^9) \\ & \oplus F(F(k^2) \oplus k^1 \oplus k^5) \oplus F(k^1) \oplus k^0 \oplus k^4 \oplus k^8 \oplus k^{12} \oplus k^{16}. \end{aligned}$$

Since $A(x \oplus k^0) = A(x)$, the function G has period k^0 . Simon's algorithm recovers k^0 , again excluding degenerate keys.

The first round can then be removed explicitly. Given a desired input

$$(z_0, z_1, z_2, z_3)$$

to the remaining 16-round suffix, query the original oracle on

$$(z_1 \oplus F(z_0) \oplus k^0, z_2, z_3, z_0).$$

After the first round, this state becomes exactly (z_0, z_1, z_2, z_3) . Hence the original oracle gives oracle access to the 16-round suffix with round keys $k^1, \dots, k^{16}$. To reuse the same 17-round periodic function, append one public Type-1 GFS-FK round with zero key to this suffix. The effective 17-round oracle has round keys $k^1, \dots, k^{16}, 0$, so the same Simon call recovers k^1 . Repeating this step, appending enough zero-key public rounds to keep an effective 17-round oracle, recovers $k^2, \dots, k^{16}$ in turn.

F.4 Other Targets

The 5-round Feistel-FK construction is the example used in Section 4.3 to illustrate the realizable criterion construction. The resulting function has period

$$s = F(k^0) \oplus k^1,$$

which matches the key relation checked by the verification experiment. This reproduces the first periodic function used in the key-recovery attack of Canale et al. [6].

For MISTY R-FK, the model reproduces the 4-round periodic function of Canale et al. [6], whose period is k^0 . The 5-round MISTY R-KF construction is detailed in Supporting Material F.2.

The same model also applies to multi-branch constructions. For Type-1 GFS, the 4-branch FK and KF settings give 17- and 13-round key-recovery attacks in the qCCA setting. The 17-round FK construction is detailed in Supporting Material F.3.

First 13-round periodic function for Type-1 GFS-KF. For 13-round Type-1 GFS-KF, the round update is

$$(x_0^{i+1}, x_1^{i+1}, x_2^{i+1}, x_3^{i+1}) = (x_3^i, x_0^i \oplus F(x_3^i \oplus k^i), x_1^i, x_2^i).$$

Let

$$v(x) = (F(x), 0, 0, x).$$

For

$$(Y_0, Y_1, Y_2, Y_3) = E^{13}(v(x)),$$

the output-computable function

$$G(x) = Y_0 \oplus x$$

has period k^0 . Simon's algorithm therefore recovers k^0 when the constructed oracle satisfies the standard unique nonzero-period promise; the zero-key degenerate case is excluded from this Simon step.

The first round can then be peeled off. Given a desired input

$$(z_0, z_1, z_2, z_3)$$

to the remaining 12-round suffix, query the original oracle on

$$(z_1 \oplus F(z_0 \oplus k^0), z_2, z_3, z_0).$$

After the first round, this state becomes exactly (z_0, z_1, z_2, z_3) . Hence the original oracle gives oracle access to the 12-round suffix with round keys $k^1, \dots, k^{12}$. To reuse the same 13-round periodic function, append one public Type-1 GFS-KF round with zero key to this suffix. The effective 13-round oracle has round keys $k^1, \dots, k^{12}, 0$, so the same Simon call recovers k^1 . Repeating this step, appending enough zero-key public rounds to keep an effective 13-round oracle, recovers $k^2, \dots, k^{12}$ in turn.

For Type-2 GFS, the corresponding FK and KF settings give 7 and 5 rounds in the qCPA setting.

First 7-round periodic function for Type-2 GFS-FK. For 7-round Type-2 GFS-FK, the round update is

$$(x_0^{i+1}, x_1^{i+1}, x_2^{i+1}, x_3^{i+1}) = (x_1^i \oplus F(x_0^i) \oplus k_0^i, x_2^i, x_3^i \oplus F(x_2^i) \oplus k_1^i, x_0^i).$$

Let

$$u(x) = (0, x, F(x), F^2(x)).$$

Then

$$g_{\text{FK}}(x) = F(E_1^7(u(x))) \oplus E_2^7(u(x)) \oplus x$$

has period $k_0^0 \oplus F(0)$. Since $F(0)$ is public, the subkey k_0^0 is obtained from the period, except for the zero-period degenerate case $k_0^0 = F(0)$. By applying the same construction to the pair-swapped input

$$u'(x) = (F(x), F^2(x), 0, x),$$

the output-computable function

$$g'_{\text{FK}}(x) = F(E_3^7(u'(x))) \oplus E_0^7(u'(x)) \oplus x$$

has period $k_1^0 \oplus F(0)$, and hence gives k_1^0 .

After k_0^0 and k_1^0 are known, the first round can be removed. Given a desired input (z_0, z_1, z_2, z_3) to the remaining 6-round suffix, query the original oracle on

$$(z_3, z_0 \oplus F(z_3) \oplus k_0^0, z_1, z_2 \oplus F(z_1) \oplus k_1^0).$$

The state after the first round is exactly (z_0, z_1, z_2, z_3) . Appending one public zero-key Type-2 GFS-FK round gives an effective 7-round oracle with round-key pairs $(k_0^1, k_1^1), \dots, (k_0^6, k_1^6), (0, 0)$. Repeating the two Simon calls and the peeling step recovers all seven round-key pairs.

First 5-round periodic function for Type-2 GFS-KF. For 5-round Type-2 GFS-KF, the round update is

$$(x_0^{i+1}, x_1^{i+1}, x_2^{i+1}, x_3^{i+1}) = (x_1^i \oplus F(x_0^i \oplus k_0^i), x_2^i, x_3^i \oplus F(x_2^i \oplus k_1^i), x_0^i).$$

Let

$$v(x) = (0, 0, x, F(x)).$$

Then

$$g_{\text{KF}}(x) = E_1^5(v(x)) \oplus x$$

has period k_1^0 . The pair-swapped input

$$v'(x) = (x, F(x), 0, 0)$$

gives the output-computable function

$$g'_{\text{KF}}(x) = E_3^5(v'(x)) \oplus x,$$

which has period k_0^0 . These two Simon calls recover the first round-key pair, excluding the zero-key degenerate case for the current target period.

The first round is then peeled off by querying, for a desired suffix input (z_0, z_1, z_2, z_3) ,

$$(z_3, z_0 \oplus F(z_3 \oplus k_0^0), z_1, z_2 \oplus F(z_1 \oplus k_1^0)).$$

Appending one public zero-key Type-2 GFS-KF round keeps an effective 5-round oracle. Repeating the two Simon calls recovers $(k_0^1, k_1^1), \dots, (k_0^4, k_1^4)$.

Skipjack-B is the variant of Skipjack [27] that repeatedly uses Rule B. The model finds 17-round FK and 14-round KF constructions.

First 17-round periodic function for Skipjack-B-FK. For 17-round Skipjack-B-FK, the round update is

$$(x_0^{i+1}, x_1^{i+1}, x_2^{i+1}, x_3^{i+1}) = (x_1^i, x_2^i, x_3^i \oplus x_0^i, F(x_0^i) \oplus k^i).$$

Let

$$u(x) = (F^{-1}(F^{-3}(x) \oplus x), F^{-3}(x), F^{-2}(x), F^{-1}(F^{-3}(x) \oplus x) \oplus F^{-1}(x)).$$

Then

$$g_{\text{FK}}(x) = E_0^{17}(u(x)) \oplus F(x)$$

has period k^0 . Simon's algorithm recovers k^0 under the usual nonzero-period promise. Given a desired suffix input (z_0, z_1, z_2, z_3) , the first round is peeled off by querying the original oracle on

$$(F^{-1}(z_3 \oplus k^0), z_0, z_1, z_2 \oplus F^{-1}(z_3 \oplus k^0)).$$

Appending one public zero-key Skipjack-B-FK round gives another effective 17-round oracle, so applying Simon's algorithm to the same function recovers k^1 . Repeating this step recovers $k^2, \dots, k^{16}$.

First 14-round periodic function for Skipjack-B-KF. For 14-round Skipjack-B-KF, the round update is

$$(x_0^{i+1}, x_1^{i+1}, x_2^{i+1}, x_3^{i+1}) = (x_1^i, x_2^i, x_3^i \oplus x_0^i, F(x_0^i \oplus k^i)).$$

Let

$$v(x) = (x, 0, 0, x).$$

Then

$$g_{\text{KF}}(x) = F^{-1}(E_3^{14}(v(x))) \oplus F(x)$$

has period k^0 . After recovering k^0 , query the original oracle on

$$(F^{-1}(z_3) \oplus k^0, z_0, z_1, z_2 \oplus F^{-1}(z_3) \oplus k^0)$$

to realize any desired suffix input (z_0, z_1, z_2, z_3) . Appending one public zero-key Skipjack-B-KF round keeps a 14-round effective oracle. Repeating the same Simon call recovers $k^1, \dots, k^{13}$, again excluding the degenerate zero-period case for the current target key.