Revisiting Automated Quantum Periodic Distinguisher Construction

Jian Guo and Yiran Yao

School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore guojian@ntu.edu.sg, yiran005@e.ntu.edu.sg

Abstract. Simon's algorithm can detect hidden XOR periods in functions derived from symmetric ciphers. Finding such functions becomes difficult when nonlinear layers and diffusion spread the relevant expressions across many branches, so recent work has used symbolic search to automate the construction. We refine the algebraic SMT model of Liu et al. in two ways. Prefix realization checks whether a symbolic starting state can be reached through preceding rounds and records the round-key nibbles needed to produce it. DDT Filtering restricts a local S-box input to a DDT bucket so that the symbolic path can cross an additional nonlinear layer. The latter condition is key-dependent: the target period need not lie in the translation space of the selected bucket, and our results state this condition explicitly. We report the maximum round counts found for GFS-2F, GFS-4F, Skipjack-B, LBlock, TWINE, CRAFT, and SKINNY, with Liu et al.'s automated model as the main comparison. We also combine selected witnesses with partial round-key guesses in the Grover–meet–Simon setting, yielding reduced-round key-recovery candidates below the corresponding comparison budgets.

Keywords: Quantum periodic distinguishers · SMT-based tools

1 Introduction

In discussions of the quantum threat to symmetric cryptography, Grover's algorithm is often taken as the default reference point. It is a quantum algorithm for unstructured search, and in the symmetric setting it gives a quadratic speedup for exhaustive key search. Quantum security is therefore often viewed mainly as a question of key length [12]. This Grover-based baseline, however, does not fully capture the quantum attack surface relevant to symmetric cryptography. Simon's algorithm is designed to recover a hidden period in a structured function. In the present setting, the relevant case is a hidden XOR period induced by a cryptographic construction. Once such a structure is present, Simon's algorithm can test for or recover the period in polynomial time in the stated oracle model [26]. From this viewpoint, quantum symmetric security depends both on the margin against quantum search and on the presence of exploitable structure that a quantum adversary can query and use.

The line of Simon-based symmetric cryptanalysis is by now well established. Early works on reduced-round Feistel and Even–Mansour style constructions

showed how cipher structure can create the hidden periods used by these attacks [18,19], and Santoli and Schaffner later systematized this attack perspective on symmetric primitives [25]. At CRYPTO 2016, Kaplan et al. then demonstrated attacks on widely used symmetric constructions under quantum query access [16]. Subsequent advances broadened both scope and technique, including the Grover–meet–Simon paradigm of Leander and May [20], the Offline Simon refinements of Bonnetain et al. [4], and quantum linearization attacks on parallelizable constructions [5]. Simon-based attacks should therefore be viewed as a structural attack family in their own right.

The central practical difficulty is the construction of a function to which Simon's algorithm can be applied. For small Feistel settings, suitable periodic distinguishers can often be derived directly [18,15]. The situation changes on higher-branch generalized Feistel networks and on more diffusive designs, where the literature contains many carefully engineered, cipher-specific or family-specific constructions [11,10,13,22,8,9,27]. In these richer settings, the difficulty is to preserve the structural constraints that support periodicity as propagation becomes more entangled. As branch interactions accumulate, constants interfere with clean cancellation patterns, and unknown differences propagate rapidly, the symbolic invariants that make low-complexity examples tractable become much harder to preserve. The bottleneck is therefore to turn periodic-function construction into a systematic search procedure rather than a collection of hand-crafted examples. SPN-style block ciphers, represented in our experiments by SKINNY [1] and CRAFT [2], are a complementary test. Their nonlinear and linear layers quickly spread unknown differences to many cells. The coarse symbolic model then loses useful information after only a few rounds.

Existing work addresses this bottleneck from several complementary angles. One line gives structural conditions for Simon-compatible relations, beginning with separability-style viewpoints and their stronger variants [14,31]. A second line tests generalized Simon conditions by enumerating circuits [6]. A third line connects Simon-based distinguishers to truncated differentials, thereby clarifying how quantum periodicity can sometimes be read through more classical propagation arguments [30]; recent simplified searches push this bridge further on selected families [7]. Liu et al. instead use algebraic representations of Feistel branches in an SMT model [21]. These are distinct forms of automation: the algebraic model does not subsume every construction found by circuit enumeration. Our work starts from Liu et al.'s model and adds two specific checks motivated by differential propagation.

This paper takes the recent symbolic model of Liu et al. as the comparison point [21]. Our experiments compare the maximum round counts found by the two automated models. They are not claims of the best known distinguishers or key-recovery attacks on the listed ciphers. We also test whether selected witnesses can be combined with partial round-key guesses while keeping the estimated Grover–meet–Simon work below the stated comparison budget.

1.1 Our Contributions

First, prefix realization checks whether a symbolic suffix boundary B can be produced after p preceding rounds. When it can, the procedure records the input expressions and the required round-key nibbles. Integrating this check into the search makes it possible to compare longer paths under one global complexity limit; the underlying idea of guessing early-round key material is not new.

Second, DDT Filtering restricts a local nonlinear input to a selected DDT bucket. This can preserve a useful symbolic value for one more nonlinear step. The retained bucket supports the intended period only when the key-dependent period s lies in its translation space V. We state this condition and its probability separately instead of assuming that it always holds.

Third, we evaluate the two changes on generalized-Feistel, Skipjack-type, lightweight balanced, and SPN-style targets. The tables report symbolic paths, the required events along those paths, and the maximum round counts found. A path with $W > 0$ is conditional on $s(K) \in V$; it is not by itself a complete distinguisher for every key.

Finally, an automated $p + s + a$ search combines selected witnesses with the round-key nibbles needed before the periodic path and for partial decryption after it. The resulting rows give candidate-search estimates. They do not use the key schedule to recover the full master key.

Table 1 summarizes the main differences from the symbolic model of Liu et al. The Base row is their result. The Prefix-realization row reports connected prefix–suffix paths for $p = 1, 2, 3, 4$. The DDT Filtering row reports symbolic suffix paths with the local filter enabled. For CRAFT, round counts follow the official convention: a searched core trail of length r is reported as $r + 1$ CRAFT rounds. The evaluated targets cover GFS-2F and GFS-4F as generalized-Feistel structures [24], a Skipjack-type structure following the Skipjack analysis of Knudsen and Wagner [17], and the concrete block ciphers LBlock [29], TWINE [28], CRAFT [2], and SKINNY [1]. In all GFS experiments reported here, one cell is instantiated as 4 bits.

1.2 Organization

The remainder of the paper is organized as follows. After the preliminaries in Section 2, Section 3 presents the inherited symbolic model, input-domain extension via prefix realization, DDT Filtering, and the event probabilities used later. Section 4 evaluates the resulting distinguisher-level witnesses through prefix-realization and DDT Filtering experiments. Section 5 combines selected witnesses with partial round-key guesses in Grover–meet–Simon style key-recovery candidates, including the complexity rule, full-search automation, execution procedure, and final rows. Section 6 concludes the paper.

Table 1. Distinguisher-side comparison with the inherited symbolic model of Liu et al. Slash-separated prefix-realization entries correspond to prefix lengths $1, 2, 3, 4$.

Target	Attack	Rounds	Probability of required events	Ref.
GFS-2F	Baseline	6	2^{0}	[21]
	Prefix realization	7/8/9/10	$2^{-4}/2^{-12}/2^{-20}/2^{-28}$	Sec. 4.2
	DDT Filtering	7	2^{-16}	Sec. 4.3
GFS-4F	Baseline	10	2^{0}	[21]
	Prefix realization	11/12/13/14	$2^{-4}/2^{-12}/2^{-24}/2^{-40}$	Sec. 4.2
	DDT Filtering	11	2^{-12}	Sec. 4.3
LBlock	Baseline	8/10	$2^{0}/2^{-16}$	[21]
	Prefix realization	w/o X: 9/10/11/12 with X: 10/11/12/13	$2^{-4}/2^{-16}/2^{-32}/2^{-64}$ $2^{-8}/2^{-16}/2^{-32}/2^{-64}$	Sec. 4.2
	DDT Filtering	10/11	$2^{-12}/2^{-48}$	Sec. 4.3
TWINE	Baseline	8/10	$2^{0}/2^{-16}$	[21]
	Prefix realization	w/o X: 9/10/11/12 with X: 10/11/12/13	$2^{-4}/2^{-16}/2^{-32}/2^{-64}$ $2^{-8}/2^{-16}/2^{-32}/2^{-64}$	Sec. 4.2
	DDT Filtering	10/11	$2^{-12}/2^{-48}$	Sec. 4.3
Skipjack-B	Baseline	16	2^{0}	[21]
	Prefix realization	w/o X: 16/17/18/19 with X: 17/18/19/20	$2^{-16}/2^{-32}/2^{-48}/2^{-64}$ $2^{-32}/2^{-48}/2^{-64}/2^{-80}$	Sec. 4.2
	DDT Filtering	19	2^{-80}	Sec. 4.3
CRAFT	Baseline	8/12	$2^{0}/2^{-40}$	[21]
	Prefix realization	w/o X: 9/10/11/12 with X: 9/10/11/12	$2^{-8}/2^{-28}/2^{-64}/2^{-120}$ $2^{-8}/2^{-28}/2^{-64}/2^{-128}$	Sec. 4.2
SKINNY	Baseline	7/9	$2^{0}/2^{-16}$	[21]
	Prefix realization	w/o X: 8/9/10/11 with X: 10/11/12/13	$2^{-16}/2^{-48}/2^{-80}/2^{-112}$ $2^{-28}/2^{-60}/2^{-92}/2^{-124}$	Sec. 4.2

The table reports symbolic distinguisher results, not complete attack complexities. The probability column accounts for the required prefix-value, collision, and DDT Filtering events along each reported path. For 4-bit-cell targets, charged prefix values, DDT Filtering events, and collision events contribute 4 bits each. For Skipjack-B, required local G-call inputs and collision events contribute 16 bits each, while a DDT Filtering event contributes 32 bits. For every $W > 0$ entry, the displayed value covers these required events only. The row is also conditional on the key-dependent event $s(K) \in V$. The table does not include verification, partial round-key guesses, or full master-key recovery. For CRAFT, rounds are reported in the official convention; the searched core length is one round shorter. The two refinements are orthogonal: prefix realization acts before the suffix boundary, whereas DDT Filtering acts inside the suffix.

2 Preliminaries

2.1 Notation

Let

$$E^r : \{0,1\}^N \to \{0,1\}^N$$

denote the r-round version of the target cipher or public permutation. When the state is partitioned into t branches of n bits, we write $N = tn$ and denote the round function in round i by f_i. For a designated branch, or more generally a designated set of output coordinates, we write $E^r|_u$ for the corresponding projection. This notation is sufficient for all later constructions.

As in the Simon-based distinguisher literature [30,21], inputs will be decomposed as (x, α_b, c), where $b \in \{0,1\}$, $x \in \{0,1\}^n$ is the variable part, $\alpha_0 \neq \alpha_1$ are fixed constants, and c collects the remaining fixed coordinates. We write $\delta = \alpha_0 \oplus \alpha_1$ and reserve $s \neq 0$ for a hidden XOR period. A typical periodic distinguisher is obtained by constructing a function of the form

$$g(x) = E^r(x, \alpha_0, c)|_u \oplus E^r(x, \alpha_1, c)|_u$$

and then showing that g hides a nontrivial period together with suitable control of spurious collisions.

2.2 Simon's Algorithm

Simon showed that a hidden XOR period can be found in polynomial time from quantum oracle access [26]. Concretely, for a function $g : \{0,1\}^n \to \{0,1\}^m$, if there exists $s \neq 0$ such that $g(x) = g(x \oplus s)$ for all x, and if collisions not explained by s remain sufficiently sparse, then $O(n)$ quantum queries yield linear relations orthogonal to s, from which the period is recovered with high probability. For the present paper, this standard consequence is the only property of Simon's algorithm that is needed: once a cipher-induced function is shown to hide a period, Simon turns that structure into a polynomial-time quantum test.

In the cryptanalytic line initiated by Kaplan et al. [16], however, Simon is often used in a weaker, distinguisher-oriented sense. The objective need not be the explicit recovery of s. It is enough that the measurement outcomes produced by Simon's procedure are confined to a proper subspace, since a genuine hidden period forces every sample y to satisfy $y \cdot s = 0$, whereas under a random-permutation view the samples span the full n-dimensional space with overwhelming probability. This viewpoint is already explicit in the Simon-based distinguisher literature, including the Feistel attacks of Ito et al. [15], and it is the interpretation adopted throughout this paper whenever we say that Simon's algorithm applies.

2.3 Grover–Meet–Simon

Leander and May introduced the Grover–meet–Simon method for settings in which periodicity becomes visible only after guessing an auxiliary value [20]. In

its basic form, one considers a function $h(u, x)$ such that only the correct outer guess u^* makes the partial function $x \mapsto h(u^*, x)$ hide a nontrivial period. Grover search then explores the outer guess space, while Simon serves as the inner test that recognizes whether the corresponding partial function is periodic. Bonnetain et al. refined this method and its complexity analysis in later work [4].

We use Grover–meet–Simon only as an outer complexity wrapper. When a periodic relation appears only after guessing some intermediate value or key-dependent quantity in a space of size 2^m, the later key-recovery analysis will account for the attack cost as Grover search over that space, up to polynomial factors coming from oracle evaluation and the inner Simon routine. No further detail is needed here.

The Grover–meet–Simon costs used later are stated in the $Q2$ model. The inner Simon test requires coherent superposition access to $x \mapsto h(u, x)$ for each outer guess u. With only $Q1$ access, the adversary obtains classical oracle data instead; using such data requires a separate offline-Simon conversion [4] and separate data/time accounting. Consequently, the $\widetilde{O}(2^{m/2}\,\mathrm{poly}(n))$ bounds used here do not directly apply in the $Q1$ model. Within $Q2$, our default access model is *qCPA*, namely superposition chosen-plaintext access. We use *qCCA* only when a construction requires superposition access to the inverse oracle; local reversible computation under guessed outer-round values does not by itself require such access.

2.4 Link to Truncated Differentials

An important conceptual step in this area is the observation of Xiang et al. that many Simon-based periodic distinguishers admit a truncated-differential interpretation [30]. In the present notation, one studies two related inputs (x, α_0, c) and (x, α_1, c) and asks whether, after a suitable number of rounds, the projected output difference on u behaves periodically in x. If so, the function

$$g(x) = E^r(x, \alpha_0, c)|_u \oplus E^r(x, \alpha_1, c)|_u$$

inherits a hidden period and becomes a valid target for Simon's algorithm. The truncated-differential viewpoint is useful because it identifies which coordinates must remain fixed, which may vary, and where cancellation occurs in the projected output.

For the purposes of this paper, however, this bridge is only a sufficient mechanism, not a complete characterization of useful periodic constructions. It is most transparent when the relevant propagation can be captured by tractable truncated differences, as in several Feistel and generalized-Feistel examples [30]. The settings considered later are less benign: unknown differences, stronger diffusion, and interactions across several branches may still permit periodic behavior even when no direct truncated-differential formulation is available. The next section therefore moves from this bridge to the more general symbolic-construction methodology that underlies the search model of Liu et al. [21] and the refinements developed in this paper.

3 Baseline Model and Our Modifications

3.1 Periodic-Distinguisher Search Target

Before describing the SMT model, we recall the path property that the model is designed to find. Write the two related inputs as

$$(x, \alpha_0, c), \qquad (x, \alpha_1, c),$$

where x is the query variable, α_0, α_1 are two fixed constants, c collects the remaining fixed coordinates, and $\delta = \alpha_0 \oplus \alpha_1$. For a selected output coordinate u, the periodic-distinguisher function has the form

$$g(x) = E^r(x, \alpha_0, c)|_u \oplus E^r(x, \alpha_1, c)|_u.$$

The useful endpoint condition is that the projected output still contains a separable dependence on x and α_b. Liu et al. call this the *differential separability property*. A typical sufficient form is

$$E^r(x, \alpha_b, c)|_u = F(x \oplus G(\alpha_b, c), c) \oplus P(x, c) \oplus Q(\alpha_b, c).$$

In the difference defining $g(x)$, the term $P(x, c)$ cancels and $Q(\alpha_b, c)$ contributes only a fixed difference. The remaining F-term has period

$$s = G(\alpha_0, c) \oplus G(\alpha_1, c),$$

because shifting x by s swaps the two inputs to F. Hence $g(x \oplus s) = g(x)$.

The symbolic search does not reconstruct this algebraic expression in full. Instead, it tracks whether a path preserves enough structure to certify such an endpoint. In the inherited model, this is represented by an output branch that is either separable, denoted $0s$, or still directly related to the x-family variable. The rest of this section describes the symbolic abstraction and constraints used to search for such paths.

3.2 The Inherited Baseline as a Control Setting

We now describe the inherited SMT model used as the control setting. For a fixed round wiring and round count, the model asks whether there is a symbolic path from an allowed input pattern to one of the endpoint forms described above. The starting point is the algebraic SMT model of Liu et al. [21]. It represents the values of Feistel branches algebraically. Canale et al. [6] instead enumerate circuits that satisfy generalized Simon conditions. The two approaches are related but distinct: the model of Liu et al. is not a simple extension of circuit enumeration and does not contain all of its constructions.

The solver takes three kinds of input. The first is the round wiring, written only with the three generic operations Split, R, and Xor. The second is the symbolic domain allowed at the input branches. The third is a set of global budgets on constrained events. A satisfiable instance returns a valid input assignment, the symbolic state of every intermediate branch, and the event flags activated along the path. An unsatisfiable instance means that no valid path exists under the same wiring, domain, and budgets.

Symbolic states. To keep the search tractable, the baseline does not track full algebraic expressions on each branch. Instead, each branch is assigned a coarse symbolic state drawn from

$$\{0, x, \delta, R(x), R(\delta), 0s, \perp\},$$

together with the corresponding mixed forms encoded in the same state system. Internally, this is a component-wise Boolean encoding rather than a one-hot label system, so mixed forms such as $x \oplus \delta$, $R(x) \oplus x$, and $R(\delta) \oplus \delta$ can be represented within the same abstraction. Here 0 denotes the zero difference, x denotes the query-variable family, δ denotes the input difference $\alpha_0 \oplus \alpha_1$, and $R(x)$ and $R(\delta)$ are buckets for values that remain dependent on the x-family or δ-family after local nonlinear propagation, while their exact local expressions are no longer tracked. The state $0s$ denotes a separable endpoint term of the form $F(x \oplus G(\alpha_b, c), c)$. The failure state $\perp$ marks a branch that no longer preserves usable periodic structure. This abstraction keeps only the information needed for the search: x-dependence, δ-dependence, and separability.

Split. The Split rule copies the input state to each outgoing branch. It introduces no new symbolic effect beyond equality of the duplicated states.

R gate. The baseline R rule is case-based. The states x, $R(x)$, and $R(x) \oplus x$ are all mapped to $R(x)$. The states δ, $R(\delta)$, and $R(\delta) \oplus \delta$ are all mapped to $R(\delta)$. The states 0 and $0s$ remain unchanged. The special family

$$x \oplus \delta, \qquad x \oplus R(\delta), \qquad x \oplus R(\delta) \oplus \delta$$

may produce $0s$ through one special transition; otherwise it is mapped to $\perp$. All remaining inputs are also mapped to $\perp$. The Boolean variable s_i records whether this special transition is used at the i-th R gate.

XOR. If either input of Xor is $\perp$, the output is $\perp$. Otherwise, the baseline updates the symbolic components conservatively. The $0s$ and $R(x)$ parts propagate by OR-style accumulation, while identical symbolic components cancel deterministically, for example $x \oplus x = 0$ and $\delta \oplus \delta = 0$. For delta-family nonlinear terms whose equality is not forced by the symbolic state, the model uses a small set of ambiguous collision cases. At the concrete level, when two such delta-family differences entering an Xor happen to collide, the output may collapse to a simpler state, typically 0 and in some cases δ. Each such potential event is marked by a Boolean variable ξ_i, and the total collision count is

$$X = \sum_i \xi_i.$$

Thus $X = 0$ is the deterministic-path setting, while $X > 0$ records how many probabilistic collision events are required by the path.

Input, output, and global constraints. A candidate path is accepted only if it satisfies the baseline input, output, and global constraints. The baseline input domain is

$$\{0, x, \delta, x \oplus \delta\}.$$

At least one input branch must contain δ, and an x-like branch may appear at most once. At the output, at least one branch must either be separable ($0s$) or remain directly related to x. This is the baseline success condition, because such a branch can later serve as the endpoint from which a periodic function is constructed. In addition, the special transition is allowed at most once, so $\sum_i s_i \leq 1$, while the collision count X is compared against the chosen collision budget. The second output form is the special case in which F is the identity function.

Meaning of a solution. Under these definitions, the baseline becomes an SMT feasibility problem. A SAT solution is a complete symbolic path that satisfies all propagation rules and constraints above. An UNSAT answer means that no such path exists in the inherited model under the chosen setting.

The states are not merely truncated differences. In particular, $0s$ certifies the value-level term $F(x \oplus G(\alpha_b, c), c)$; the other accepted terms depend only on x or only on (α_b, c). Consequently, an accepted branch has the form

$$F(x \oplus G(\alpha_b, c), c) \oplus P(x, c) \oplus Q(\alpha_b, c).$$

Taking the difference between $b = 0$ and $b = 1$ cancels P, and the shift

$$s = G(\alpha_0, c) \oplus G(\alpha_1, c)$$

swaps the two inputs to F. The resulting function therefore satisfies $g(x) = g(x \oplus s)$. The transition that first creates $0s$ is used at most once. Any later occurrence of $0s$ can only be copied or combined from that term and therefore keeps the same period; an accepted path cannot combine unrelated separable terms with different periods. This conclusion follows from the model of Liu et al.; our two modifications change how a boundary is reached and how one nonlinear step is conditioned, but not this accepting condition.

3.3 Forward Extension: Adding Prefix Rounds with Accounted Cost

The inherited abstraction is designed to search symbolic suffix paths. It starts from an allowed input pattern, propagates the coarse states through the round wiring, and checks whether the final state still contains a usable x-related or separable branch. Its input alphabet is only

$$\{0, x, \delta, x \oplus \delta\}.$$

This is sufficient when the state at the beginning of the path can already be written in this alphabet. Prefix realization asks an additional question: can this

suffix starting state be obtained after one or more preceding rounds? If so, the distinguisher gains those prefix rounds, and the search records which round-key cells are needed to produce the required input-side expressions.

The idea is already visible in GFS-2F. Suppose the symbolic suffix path starts from

$$(x, \delta, 0, 0).$$

This state can be reached after one preceding GFS-2F round from the input-side expression

$$(F_0(x, \delta), 0, 0, x).$$

Here $F_0(x, \delta)$ is the expression forced by the one-round wiring and by the key-mixed S-box input. After one forward round, this expression gives exactly the suffix starting state above. The path can therefore be extended by one preceding round if the input side contains this expression. Figure 1 shows this instance and marks the required round-key nibble.

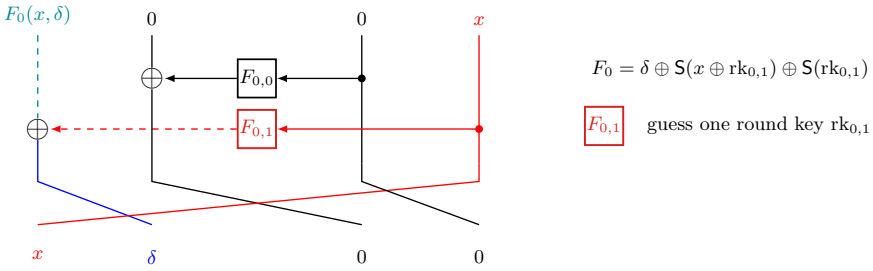

Fig. 1. One-round prefix realization for the GFS-2F predecessor in the running example. The red local function $F_{0,1}$ depends on the round key $\mathrm{rk}_{0,1}$, so producing the predecessor requires fixing this round-key nibble.

This example motivates the boundary terminology used below. The *prefix-suffix boundary* is the state where the added prefix rounds meet the symbolic suffix path. For a $p + s$ split, it is the state after the p prefix rounds and before the s suffix rounds. In the one-round GFS-2F example above, the boundary is $(x, \delta, 0, 0)$, and $(F_0(x, \delta), 0, 0, x)$ is the input-side expression that reaches it.

For a one-round prefix, this idea can be described by first-pass symbols

$$F_i(x), \qquad F_i(x) \oplus \delta.$$

The symbol $F_i(x)$ says that the x-dependent value has passed through a specific local round-function instance before the suffix starts. In this view, the input-side language is enlarged from

$$\{0, x, \delta, x \oplus \delta\}$$

to expressions such as

$$\{F_i(x), F_i(x) \oplus \delta\}.$$

This notation explains why the old input alphabet is too small, but it is not a good representation for longer prefixes. After two or more preceding rounds, the input-side expressions can contain nested local functions, XORs of several branches, and several key-mixed values. Treating all of these as new coarse symbols would hide the required round-key cells. Therefore, the prefix is not folded into the suffix alphabet. The suffix remains symbolic, while the prefix is handled as an expression-construction problem.

For a fixed prefix length p, the construction starts from a selected prefix-suffix boundary and walks backward through the p preceding rounds. Copy edges simply copy the required expression to the predecessor wire. For an XOR update, the rule is the usual algebraic rearrangement: if the forward update contains

$$v' = a \oplus F_{r,i}(b),$$

and the boundary requires a particular expression for v', then one predecessor expression can be chosen by setting

$$a = v' \oplus F_{r,i}(b).$$

The local nonlinear function is not inverted; it is kept as a named key-dependent function inside the expression. Wires that are not needed by the chosen path are fixed to constants. The constructed input-side expressions are then forwarded through the same p rounds to check that they reach the selected prefix-suffix boundary.

The prefix can be used in two ways. In a prefix-only scan, the search enumerates possible prefix-suffix boundaries and asks how many preceding rounds can produce them, together with the minimum number of round-key cells needed. In the key-recovery candidate search, the suffix solver first determines the actual suffix starting state used by a $p+s+a$ decomposition; the same prefix construction is then applied to that state. Thus the prefix changes which suffix starting states are reachable from chosen plaintexts, but it does not change the symbolic propagation rules inside the suffix.

The input-side expressions commonly contain S-box or local round-function inputs masked by round keys. To impose the required plaintext-side values, the attack must know the corresponding round-key components, or satisfy the matching data condition. We record their number by

$$K_{\mathrm{fwd}},$$

the number of distinct required round-key components used by the prefix expressions. Its bit contribution is obtained from the cipher-specific key granularity. In Figure 1, the red $F_{0,1}$ box accounts for one such component, namely $\mathrm{rk}_{0,1}$.

One extra guard is needed for the enlarged input side. If the prefix-suffix boundary has no pure δ anchor and every nonzero branch is only of the form $x \oplus \delta$, the apparent period can collapse to the trivial shift $s = \delta$. Indeed, let the two inputs used in the period check be $P_0(x)$ and $P_1(x)$. In this degenerate case, shifting x by δ does not create a new pair of inputs; it only swaps the old pair:

$$P_0(x \oplus \delta) = P_1(x), \qquad P_1(x \oplus \delta) = P_0(x).$$

Therefore, for any output branch u,

$$E(P_0(x \oplus \delta))|_u \oplus E(P_1(x \oplus \delta))|_u = E(P_1(x))|_u \oplus E(P_0(x))|_u.$$

The equality holds by commutativity of XOR, independently of the cipher propagation. Thus the four evaluations normally involved in checking $g(x) = g(x \oplus s)$ collapse to the same two plaintexts read in the opposite order. The full searches therefore reject the bare $x \oplus \delta$-only family. Boundaries such as $(x, \delta, 0, \ldots)$ or $(x \oplus \delta, \delta, 0, \ldots)$ remain allowed because they include an independent δ anchor.

With this extension, a valid witness consists of a symbolic suffix path together with input-side expressions that reach its prefix-suffix boundary through p preceding rounds. The suffix endpoint condition is unchanged: the path must still end with a branch that is separable or directly related to x. The new information is the prefix length and K_{fwd}, reported with the later experimental results.

3.4 Backward Extension: DDT Filtering on Known Differences

The backward extension is used when suffix propagation reaches a local S-box input pair of the form

$$S(x \oplus m), \qquad S(x \oplus m \oplus \delta).$$

The input difference δ is known, but the actual S-box input still contains the query variable x. In the inherited symbolic model, this local step is useful when it matches the special transition to a separable value $0s$. Otherwise the nonlinear output no longer fits one of the tracked useful forms and is treated as an unknown state $\perp$, so the path cannot continue through that S-box while retaining the x-family branch.

The idea of our improvement comes from the standard way of studying an S-box through its difference distribution table (DDT): for a fixed input difference, one may restrict the allowed input values so that the output difference falls into a chosen value or set of values [3,23]. This suggests the question used here: can we restrict the local input $x \oplus m$ to a DDT bucket so that the output difference is fixed, while x remains a query variable over a smaller domain? If this is possible, a suffix path can pass through one more nonlinear layer without losing the x-family branch; the price is an explicit local filtering condition.

Table 2 gives the local picture for the TWINE S-box [28]. Suppose a search reaches a selected S-box whose two inputs have the form

$$z_0 = x \oplus m, \qquad z_1 = x \oplus m \oplus \delta,$$

where q denotes the part of this S-box input other than the query variable x, and k is the round-key nibble XORed into this S-box input. Thus $m = q \oplus k$ is the local key-mixed offset that shifts x before the S-box. The known input difference is δ, but the actual S-box input variable is

$$y = x \oplus m.$$

For the concrete row $\delta = \mathtt{1}$, the table lists

$$D_1S(y) = S(y) \oplus S(y \oplus \mathtt{1})$$

for all 4-bit inputs y. In this row, the largest output-difference bucket is obtained at $\Delta_{\text{out}} = \mathtt{C}$ and retains four values of y.

Table 2. Toy DDT Filtering example using the TWINE S-box. For the known input difference $\Delta_{\text{in}} = \mathtt{1}$, the largest DDT bucket has output difference $\Delta_{\text{out}} = \mathtt{C}$ and retains $y = x \oplus m \in \{\mathtt{0}, \mathtt{1}, \mathtt{6}, \mathtt{7}\}$.

y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(y)$	C	0	F	A	2	B	9	5	8	3	D	7	1	E	6	4
$S(y \oplus 1)$	0	C	A	F	B	2	5	9	3	8	7	D	E	1	4	6
$D_1S(y)$	C	C	5	5	9	9	C	C	B	B	A	A	F	F	2	2

The retained set in the example is

$$X_{1,\mathtt{C}} = \{y : D_1S(y) = \mathtt{C}\} = \{\mathtt{0}, \mathtt{1}, \mathtt{6}, \mathtt{7}\}.$$

Restricting $y = x \oplus m$ to this set fixes the S-box output difference to $\mathtt{C}$. The output value still depends on x through $S(x \oplus m)$, but the pairwise output difference is now controlled. This is the local operation that we call *DDT Filtering*. In general, for a known input difference δ, the filter chooses an output difference from a largest DDT bucket

$$X_{\delta,\beta} = \{y : S(y) \oplus S(y \oplus \delta) = \beta\}, \qquad \beta \in \arg\max_{\gamma} |X_{\delta,\gamma}|.$$

This fixes the local output difference while discarding as little of the S-box input domain as possible.

This restriction is useful for a periodic distinguisher only if the retained input set still supports a Simon-style query domain. After the local offset m is fixed, the query variable is restricted to

$$x \in m \oplus X_{\delta,\beta}.$$

A Simon oracle cannot be queried on an arbitrary subset. The retained domain must be stable under the target period: if x is allowed, then $x \oplus s$ must also be allowed. A convenient sufficient form is an affine domain $A = a + V$ with $s \in V$. In that case, if B is a basis matrix for V, one can write

$$x = a + B\lambda, \qquad \lambda \in \mathbb{F}_2^{\dim V}.$$

When $s = B\tau$, the restricted oracle $H(\lambda) = G(a + B\lambda)$ still satisfies a Simon relation:

$$H(\lambda \oplus \tau) = G(a + B\lambda \oplus s) = H(\lambda).$$

Thus the same Simon procedure applies on the lower-dimensional variable λ.

The DDT bucket must therefore be compatible with this affine-domain requirement. For every S-box and every input difference δ,

$$D_\delta S(y \oplus \delta) = S(y \oplus \delta) \oplus S(y) = D_\delta S(y).$$

Hence every DDT bucket is stable under translation by δ. In the TWINE example, the largest bucket is also a two-dimensional affine subspace:

$$X_{1,\mathtt{C}} = \{\mathtt{0}, \mathtt{1}, \mathtt{6}, \mathtt{7}\} = \mathrm{span}\{\mathtt{1}, \mathtt{6}\}.$$

The shift by m changes the coset but not the translation space. It follows that the filtered query set is a valid reduced Simon domain only when the target period lies in that space.

This last condition is not automatic. The space V is fixed once δ and the DDT bucket have been selected, but the period $s(K)$ depends on the key. If $v_1, \ldots, v_d$ is a basis of V, the concrete check is the linear-system test

$$s(K) \in V \quad \Longleftrightarrow \quad s(K) = \sum_{i=1}^{d} \lambda_i v_i \quad \text{for some } \lambda_i \in \mathbb{F}_2.$$

A symbolic witness with $W > 0$ is therefore conditional on this membership test and need not apply to every key. The numerical $W > 0$ rows below are therefore reported as conditional symbolic results. No such condition occurs when $W = 0$.

The bucket-size loss is not the only condition. To know which shifted bucket $m \oplus X_{\delta,\beta}$ the query variable must lie in, the attack must guess or condition on the key-mixed offset $m = q \oplus k$. In the TWINE setting this offset is one 4-bit nibble, so the later key-recovery estimate includes one 4-bit unit for a W event rather than using only the two-bit size loss of the largest TWINE DDT bucket.

In the symbolic suffix model, DDT Filtering is represented by a Boolean switch w_i attached to a local nonlinear gate. The retained-x rule used in the searches is

$$x \oplus \delta \xrightarrow{w_i=1} x \oplus R(\delta).$$

Here δ is a known concrete input difference, which determines the DDT row and the retained bucket. The switch is mutually exclusive with the special-transition flag s_i at the same gate, so the model chooses one local interpretation there: either the special transition or the DDT Filtering event. Its total use is limited by

$$W = \sum_i w_i, \qquad W \le W_{\max}.$$

The toy example already shows why this budget is kept small. One constrained TWINE S-box step keeps only 4 of the 16 local inputs. If a comparable restriction is imposed again later, the surviving set shrinks again, and it becomes harder to maintain a period-compatible query domain. In the actual search these restrictions need not be identical or independent, but the general effect is the same: each extra DDT Filtering event adds another input restriction and quickly

reduces the remaining freedom. For that reason, the default setting in this work is $W_{\max} = 1$: the model is allowed to use one explicit DDT Filtering event, and no more.

A SAT solution with $W > 0$ is a symbolic path with one explicit DDT Filtering event. Its concrete interpretation records the retained local input set, the resulting output-difference value, and the input restriction needed to realize the event in the CPA setting. The backward-side mechanism is orthogonal to the forward-side prefix realization. A later candidate search combines W, X, the prefix round-key nibbles, the partial-decryption round-key nibbles, and the $s(K) \in V$ condition.

3.5 Search Axes and Result Interpretation

The two extensions above affect different parts of the search. Prefix realization extends the input side by constructing symbolic expressions that reach an internal prefix-suffix boundary. DDT Filtering extends the suffix side by allowing one controlled nonlinear transition with a known input difference. To use these mechanisms in a key-recovery candidate, the path is divided into three parts.

The full search fixes a decomposition

$$R = p + s + a.$$

Here p is the number of prefix-realization rounds before the symbolic boundary, s is the number of symbolic suffix rounds, and a is the number of output-side rounds appended after the suffix endpoint. The prefix contributes the set $\mathcal{K}_{\text{fwd}}$ of required input-side key components, with $K_{\text{fwd}} = |\mathcal{K}_{\text{fwd}}|$. The suffix contributes the event counts X and W, where X counts collision events and W counts DDT Filtering events. The appended output side contributes the set $\mathcal{K}_{\text{out}}$ needed to evaluate the chosen output branch through the final a rounds, with $K_{\text{out}} = |\mathcal{K}_{\text{out}}|$.

The accepted row is one combined witness, so these quantities are not optimized separately. For 4-bit cells, the exponent of the required suffix events is

$$P = 4X + 4W,$$

for the 4-bit-cell setting used in the main experiments. The special $0s$ transition determines the path shape and contributes no probability term. The guessed-bit count is computed from the union of the input-side and output-side supports, so a shared key component is counted only once:

$$G = \text{bits}(\mathcal{K}_{\text{fwd}} \cup \mathcal{K}_{\text{out}}) .$$

For a comparison budget κ, a candidate-search template is retained only when

$$C = G + P < \kappa.$$

If $W > 0$, this estimate is conditional on $s(K) \in V$. The concrete definition of an effective key component follows the cipher. In TWINE, LBlock, GFS, CRAFT, and SKINNY this is the natural 4-bit key-mixed cell.

The next two sections use these quantities at different levels. Section 4 first compares the distinguisher rounds reachable after applying the practical techniques above; this gives a direct view of how prefix realization and DDT Filtering change the maximum round count. Section 5 then combines selected paths with the round-key nibbles required for partial decryption and applies the budget test above.

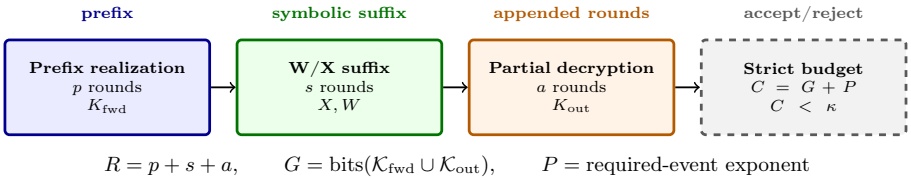

Fig. 2. Suffix-first decomposition of the key-recovery candidate search. A candidate fixes $R = p + s + a$, realizes the input-side prefix, propagates the suffix with collision and DDT Filtering events, expands the dependencies needed for partial decryption, and retains the row only if the combined estimate is strictly below the stated comparison budget.

4 Distinguisher Results

4.1 Evaluation Setting and Baseline

This section evaluates the two refinements from Section 3 at the distinguisher level. The comparison point is the symbolic search of Liu et al. [21]; we do not claim that these are the best known distinguishers on the listed ciphers. We first measure how many additional rounds become reachable when internal suffix boundaries are realized by prefix-round expressions. We then measure how the maximum suffix length changes when DDT Filtering is added to the inherited collision mechanism. Reporting these results at the distinguisher level gives a direct view of the effect of each technique before partial round-key guesses are added.

The inherited search starts from its original symbolic input language, propagates symbolic states through the round wiring, and records the collision counter X. The experiments below ask two questions. First, which internal suffix boundaries can be reached after adding prefix rounds, and at what number of required round-key cells? Second, how far can the suffix search be extended by DDT Filtering, either alone or together with collision events?

The tables report three views of these results. Standalone prefix results count the round-key cells needed to realize internal boundaries before any suffix path is chosen. Connected prefix–suffix results show which realized boundaries also support long suffix paths. DDT Filtering results compare maximum suffix lengths under different event choices.

The evaluated targets follow the cipher set analyzed by Liu et al. [21], so the reported rows can be compared directly with the inherited symbolic-search baselines. At the same time, they represent several common cipher families. GFS-2F and GFS-4F are generalized-Feistel benchmarks in the sense of Nyberg's generalized Feistel networks [24] and provide compact synthetic tests for multi-branch diffusion. In every reported GFS experiment, one cell is 4 bits; the generic GFS name does not imply a standard key size. Skipjack-B is a Skipjack-type structure following the structural analysis of Knudsen and Wagner [17], where the local nonlinear operation is a larger G call rather than a 4-bit S-box; the corresponding guessed key-mixed local value is 16 bits. LBlock [29] and TWINE [28] provide lightweight 4-bit-cell Feistel-like targets. CRAFT [2] and SKINNY [1] provide SPN-style targets, where stronger diffusion makes prefix realization and suffix extension more expensive. For CRAFT, we follow the official round-counting convention also used by Liu et al.: the search counts the repeated core layers, while the reported CRAFT round number additionally includes the terminal round. Thus a CRAFT core trail of length r is reported as $r+1$ official CRAFT rounds; the event probabilities and key-recovery estimates use the searched core layers and are unchanged.

4.2 Input-Domain Extension via Prefix Realization

We split a path into p prefix rounds followed by a symbolic suffix. The state at this split is the prefix–suffix boundary: it is the starting state of the suffix search and the ending state that the prefix rounds must realize. For a prefix length p and a suffix boundary B, the prefix-realization problem asks whether there are round-0 expressions P satisfying

$$\mathrm{Round}^p(P) = B.$$

When such expressions exist, B can be used as an internal suffix starting point while still corresponding to valid inputs before the prefix rounds. The suffix itself is searched in the inherited symbolic alphabet; the input-domain extension lies in the round-0 realization of the suffix boundary.

Realization procedure. The realization search checks a candidate prefix–suffix boundary for a fixed round wiring and a fixed prefix length. It first propagates a fixed reference assignment through the p prefix rounds. A candidate boundary B is then converted into required values at the prefix–suffix boundary, relative to this reference execution. These required values are propagated backward through the prefix wiring to synthesize round-0 expressions P. When the backward traversal meets a nonlinear local call, the required round-key cell is recorded instead of inverting the call with a new symbolic rule. The synthesized expressions are then replayed forward; the boundary is accepted only if the replay reaches B. Algorithm 1 summarizes the procedure. For the 4-bit-cell targets, K_{fwd} is the number of distinct required round-key nibbles. For Skipjack-B, it counts required 16-bit local G-call inputs.

Algorithm 1: Prefix realization for a suffix boundary.

Input: Round wiring, prefix length p, and candidate suffix-boundary family $\mathcal{B}$.

1: For each $B \in \mathcal{B}$, compute the reference state after the p prefix rounds.

2: Translate B into the required boundary values relative to the reference state.

3: Propagate the required values backward through the prefix wiring and synthesize round-0 expressions P.

4: Record the round-key cell required at every nonlinear local call encountered in this traversal.

5: Replay P forward through the p prefix rounds.

6: If the replay reaches B, record B, P, and the required round-key cells; compute K_{fwd} from these cells.

Output: Prefix-realizable boundaries, round-0 expressions, and input-side realization round-key counts.

The tables use K_{fwd} for this input-side count, matching the notation used by the implementation. For the 4-bit-cell targets this is a count of required round-key nibbles; for Skipjack-B it is a count of required local G-call values, each corresponding to a 16-bit key-mixed value rather than a round-key cell.

Round-key cells for prefix realization. The standalone experiment isolates the prefix before any suffix path is selected. It asks how few round-key cells are needed before a usable internal boundary can be realized for each prefix length. The results in Table 3 report the corresponding values used by the corresponding scans.

Table 3. Round-key cells required by standalone prefix realization. Entries are K_{fwd} values for prefix lengths $p = 1, 2, 3, 4$.

Target	K_{fwd} for $p = 1, 2, 3, 4$
GFS-2F	1/3/5/7
GFS-4F	1/3/6/10
Skipjack-B	1/2/3/4
LBlock	0/2/4/8
TWINE	0/2/4/8
CRAFT	2/5/12/24
SKINNY	4/12/20/28

Connected prefix–suffix paths. A standalone boundary requiring the fewest cells is not necessarily the boundary that gives the longest suffix. Suffix propagation depends on the boundary shape: where the x-family term is placed, where the independent δ anchor is placed, which branches are zero, and how the next rounds propagate these branches through nonlinear calls and XOR gates. The combined experiment therefore asks a different question: among forward-realizable boundaries, which ones support the longest suffix paths?

Tables 4 and 5 give these results. DDT Filtering is not enabled in these rows. For each prefix length $p = 1, 2, 3, 4$, the search selects a forward-realizable boundary and connects it to a suffix path. The two tables separate the deterministic $X = 0$ setting from the X-enabled setting, because changing the suffix path can also change the boundary shape and hence the required K_{fwd}.

Table 4. Connected prefix–suffix paths with $X = 0$. Each slash-separated list gives the values for prefix lengths $p = 1, 2, 3, 4$. For CRAFT, total rounds are reported in the official convention; the searched core length is one round shorter.

Target	suffix rounds	total rounds	K_{fwd}
GFS-2F	6/6/6/6	7/8/9/10	1/3/5/7
GFS-4F	10/10/10/10	11/12/13/14	1/3/6/10
Skipjack-B	15/15/15/15	16/17/18/19	1/2/3/4
LBlock	8/8/8/8	9/10/11/12	1/4/8/16
TWINE	8/8/8/8	9/10/11/12	1/4/8/16
CRAFT	7/7/7/7	9/10/11/12	2/7/16/30
SKINNY	7/7/7/7	8/9/10/11	4/12/20/28

Table 5. Connected prefix–suffix paths with X-enabled suffix search. Each slash-separated list gives the values for prefix lengths $p = 1, 2, 3, 4$. For CRAFT, total rounds are reported in the official convention; the searched core length is one round shorter.

Target	suffix rounds	total rounds	X	K_{fwd}
GFS-2F	6/6/6/6	7/8/9/10	0/0/0/0	1/3/5/7
GFS-4F	10/10/10/10	11/12/13/14	0/0/0/0	1/3/6/10
Skipjack-B	16/16/16/16	17/18/19/20	1/1/1/1	1/2/3/4
LBlock	9/9/9/9	10/11/12/13	1/1/1/1	1/3/7/15
TWINE	9/9/9/9	10/11/12/13	1/1/1/1	1/3/7/15
CRAFT	7/7/7/7	9/10/11/12	0/0/0/3	2/7/16/29
SKINNY	9/9/9/9	10/11/12/13	4/4/4/4	3/11/19/27

Family-level reading. The GFS rows are the cleanest demonstration of input-domain extension. The same 6-round GFS-2F suffix and 10-round GFS-4F suffix remain available after one to four prefix rounds, so each additional prefix round increases the connected distinguisher length by one. The required cell counts grow moderately: $1, 3, 5, 7$ for GFS-2F and $1, 3, 6, 10$ for GFS-4F.

LBlock and TWINE form a useful 4-bit Feistel-like benchmark. Their standalone separate-boundary minima are $0, 2, 4, 8$, but the no-X connected witnesses use $1, 4, 8, 16$. This gap is the boundary shape effect: the cheapest forward-realizable boundary is not always the boundary that supports the longest suffix.

When one collision is enabled, both ciphers reach one extra suffix round with $X = 1$, and the cell counts become $1, 3, 7, 15$.

CRAFT and SKINNY require more cells because of their SPN structure. CRAFT's standalone counts already grow as $2, 5, 12, 24$, and the connected 4-round prefix requires 30 cells without X or 29 with three collision events. SKINNY has the same qualitative issue: diffusion makes longer-prefix connected witnesses expensive.

Skipjack-B reaches many rounds in this symbolic experiment. With no collisions, the connected rows reach $16, 17, 18, 19$ total rounds for $p = 1, 2, 3, 4$; with one collision they reach $17, 18, 19, 20$. The reported units, however, are local G-call values rather than independent 4-bit S-box nibbles. Each such value is a 16-bit key-mixed quantity to be guessed, so this table should not be read as 4-bit nibbles.

4.3 DDT Filtering in the Suffix Search

Search objective. The second refinement acts inside the suffix search. The inherited suffix solver propagates symbolic states and records collision events in X. DDT Filtering adds a local event at an eligible nonlinear gate, using the filtering transition from Section 3. The experiment asks whether this local event increases the maximum suffix length or reduces the event burden needed to reach a fixed suffix length.

Event counts and procedure. The variable w_i records whether the i-th eligible nonlinear gate uses DDT Filtering, and

$$W = \sum_i w_i$$

is the number of DDT Filtering events on the path. Collision events are still recorded as ξ_i, with $X = \sum_i \xi_i$. For the 4-bit-cell targets, one W event requires one 4-bit key-mixed S-box input; for Skipjack-B, the corresponding quantity is a 16-bit G-call input. Algorithm 2 summarizes the suffix-search procedure.

Algorithm 2: DDT-filtered suffix search.

Input: Round wiring, suffix length s, symbolic input boundary, and budgets for W and X.

1: Initialize the symbolic state from the chosen suffix boundary.
2: Propagate for s rounds using the inherited Split, R, and Xor rules.
3: At each eligible nonlinear gate, either use the ordinary R-transition or apply the DDT Filtering transition and increment W.
4: At each eligible Xor gate, optionally use an inherited collision event and increment X.
5: Reject any path that exceeds the chosen W or X budget.
6: Accept a path when the propagated state satisfies the periodic-output condition.

Output: Target branch, symbolic path, DDT Filtering events, collision events, and event counts W and X.

Table 6. Maximum suffix lengths with DDT Filtering. The entries give the maximum reported suffix rounds and the required events used by the displayed symbolic path. Every $W > 0$ entry is conditional on the row-specific membership event $s(K) \in V$. For CRAFT, rounds are reported in the official convention; the searched core length is one round shorter.

Cipher	baseline	X-collision search	DDT Filtering	DDT Filtering+X
GFS-2F	6R	6R, $X = 0$	5R, $W = 1$	7R, $W = 1, X = 3$
GFS-4F	10R	10R, $X = 0$	9R, $W = 1$	11R, $W = 1, X = 2$
Skipjack-B	16R	16R, $X = 0$	13R, $W = 1$	19R, $W = 1, X = 3$
LBlock	8R	10R, $X = 4$	8R, $W = 1$	11R, $W = 1, X = 11$
TWINE	8R	10R, $X = 4$	8R, $W = 1$	11R, $W = 1, X = 11$
CRAFT	8R	12R, $X = 10$	9R, $W = 1$	9R, $W = 1, X = 0$ †
SKINNY	7R	9R, $X = 4$	7R, $W = 1$	9R, $W = 1, X = 4$ †

† These rows force one DDT Filtering event ($W = 1$). For these SPN targets, the best path found under this constraint is shorter or more expensive than the best X-only result.

Maximum suffix lengths. Table 6 compares four suffix-search regimes: the baseline suffix propagation, the inherited X-collision search, DDT Filtering alone, and DDT Filtering combined with X. The entries report maximum suffix lengths and the event counts used by the displayed path. A DDT Filtering event is a constrained local choice, so the DDT-only result is not expected to dominate the inherited baseline in every target.

Every entry with $W > 0$ is a symbolic result conditional on $s(K) \in V$. The table does not claim that this condition holds for every key.

Family-level reading. The results separate into three regimes. In generalized-Feistel and Skipjack-type suffixes, DDT Filtering increases the conditional symbolic round count when combined with collision events: GFS-2F moves from 6 to 7 suffix rounds, GFS-4F from 10 to 11, and Skipjack-B from 16 to 19. These wirings update only a small part of the state in each round, so the filtered x-family value can remain on a narrow branch path long enough to combine with later X-collision events. For the GFS rows, the 4-bit-cell interpretation is direct: each DDT Filtering event requires one key-mixed 4-bit S-box input. Skipjack-B requires separate interpretation because its local function is a 16-bit G call.

For LBlock and TWINE, DDT Filtering is most informative as an event-mix change at a fixed suffix length. Their sparse nibble-oriented Feistel structure gives local nonlinear gates where one filtered propagation can replace part of the collision burden. At 10 rounds, the X-only path uses $X = 4$, while a nonmaximal DDT Filtering path reaches the same length with $W = 1, X = 2$. Under the 4-bit-cell interpretation, this changes the event cost from 16 bits to 12 bits. The row remains separately conditional on $s(K) \in V$. The 11-round DDT Filtering+X rows reach one more conditional symbolic round, but with a much larger collision count.

The SPN rows show a different interaction pattern. In CRAFT and SKINNY, full-state linear and S-box layers spread special-family values quickly across many cells. A single DDT Filtering event selects one local transition, while the remaining special-family inputs are conservatively treated as unknown and then diffuse through the state. For CRAFT, the longer X-only path is driven by a cancellation-heavy $R(\delta)$ cascade; requiring $W = 1$ changes the timing of the x-family path and leads to a different path. For SKINNY, the X-only 9-round path already uses the available cancellation structure efficiently, so adding one DDT Filtering event reaches the same round count with a higher event exponent. Taken together, these rows show that DDT Filtering interacts with the cipher wiring; it does not uniformly increase the maximum round count.

4.4 Distinguisher-Level Synthesis

The two refinements expose different sources of additional distinguisher rounds. Prefix realization enlarges the effective input domain by realizing an internal suffix boundary through a short keyed prefix. It therefore converts some paths that are invisible from the inherited plaintext-level input language into valid distinguisher witnesses. DDT Filtering acts later, inside the suffix, by paying for one local DDT-controlled propagation when the ordinary symbolic transition would lose the useful x-family structure.

The tables also show why the two refinements should be evaluated by cipher family rather than by a single aggregate number. Generalized-Feistel and Skipjack-type wirings preserve narrow branch paths, so prefix realization and local filtering translate directly into longer paths. LBlock and TWINE show a second effect: even when the maximum symbolic suffix is expensive, DDT Filtering can reduce the collision burden at a fixed round count. CRAFT and SKINNY show the SPN-side behavior, where full-state diffusion changes the required conditions because one local filtered transition is quickly mixed with many other active cells.

Relative to the baseline symbolic search of Liu et al. [21], the refined model therefore does more than report a larger maximum round count. It records which additional witnesses come from input-domain extension, which come from suffix-side DDT Filtering, and what required events are introduced by each mechanism. The next section uses selected witnesses from this evaluation in a key-recovery candidate search with input-side and partial-decryption round-key guesses.

5 From Distinguishers to Key-Recovery Candidates

Section 4 reports periodic-distinguisher witnesses produced by the refined symbolic search. This section explains how selected witnesses are combined with partial round-key guesses in a Grover–meet–Simon search. A distinguisher row contains a symbolic path and the probabilities assigned to its required events. A key-recovery candidate also specifies the round-key nibbles needed before that path and for partial decryption after it. The search returns candidates only for

this listed round-key material. Full master-key recovery using the key schedule is not included.

5.1 Combining the Three Parts

A periodic-distinguisher witness specifies an internal boundary, a suffix path, a target output coordinate, and the required events used by the path. A key-recovery candidate connects this witness to plaintext and ciphertext. We write the total number of rounds as

$$R = p + s + a,$$

where p prefix rounds realize the suffix boundary from input-side expressions, s rounds are covered by the symbolic suffix path, and a output-side rounds are peeled from the ciphertext under guessed key material.

The result is not monotone in the suffix length. A path that gives a longer periodic distinguisher can still fail as a key-recovery candidate if the prefix realization or partial decryption requires too many round-key nibbles. For the latter, the implementation starts from the internal target coordinate and expands its dependencies forward through the a appended rounds toward the ciphertext. It records every branch and round-key nibble needed to compute that target from the ciphertext and the guessed values. The full search therefore considers the whole $p + s + a$ split, not the suffix length alone.

5.2 Recorded Data and Complexity

A searched row has the form

$$M = (p, s, a, B, \tau, \mathcal{R}, E_{\text{in}}, \mathcal{C}_{\text{out}}).$$

Here B is the suffix input boundary, τ is the target output coordinate, and $\mathcal{R}$ is the symbolic suffix path, including its collision events and DDT Filtering events. The object E_{in} records the input-side expressions that realize B after the first p rounds. The object $\mathcal{C}_{\text{out}}$ records the dependencies and round-key nibbles needed to partially decrypt the last a rounds to τ. Let $\mathcal{K}_{\text{fwd}}$ and $\mathcal{K}_{\text{out}}$ be the two corresponding key-component supports, and let K_{fwd} and K_{out} denote their individual sizes.

For the 4-bit-cell searches, the exponent of the required suffix events is

$$P = 4X + 4W,$$

where X is the number of charged collision events and W is the number of DDT Filtering events. Events that force a useful zero output shape determine the symbolic path but add no probability term. The number of guessed round-key bits is computed from the union of the two supports:

$$G = \text{bits}(\mathcal{K}_{\text{fwd}} \cup \mathcal{K}_{\text{out}}) .$$

The candidate-search exponent is

$$C = G + P.$$

A row is below the comparison budget only when $C < \kappa$. For $W > 0$, the row is additionally conditional on $s(K) \in V$.

Several examples induce 4-bit functions, so a single function can be evaluated exhaustively at low classical cost; Grover–meet–Simon instead searches the required events and partial round-key values. When the required events have total probability 2^{-P}, P represents this outer search rather than the cost of checking one fixed 4-bit function. To rule out an accidental period, a selected candidate should be checked under multiple independent data settings that preserve the predicted relation. The rows below report the base candidate-search exponent C without this verification work.

Across the 4-bit-cell rows, required round-key cells are measured in 4-bit units. Thus one required key-mixed S-box input contributes 4 guessed bits, one charged X-collision contributes a 4-bit condition, and one DDT Filtering event contributes a 4-bit condition. The columns K_{fwd} and K_{out} report the input-side and partial-decryption support sizes selected by the search, and G counts the distinct guessed bits in their union. For example, the TWINE-80 supports are disjoint: the row has $K_{\mathrm{fwd}} = 9$ and $K_{\mathrm{out}} = 7$, giving $G = 4(9 + 7) = 64$. In the CRAFT row the two supports have sizes 19 and 18 but overlap in 13 master-key nibbles, so their union contains 24 nibbles and $G = 96$. The CRAFT round-key mapping is used here only to identify and deduplicate these guessed nibbles; full master-key recovery is not performed. Skipjack-B uses 16-bit branch units instead: a required local G-call input or X-collision contributes 16 bits, while a DDT Filtering event contributes 32 bits.

5.3 Automated Candidate Search

The search combines the prefix, suffix, and partial-decryption analyses in one loop. The suffix path is the structural core of the candidate: it fixes the boundary B, the target coordinate τ, and the event counts X, W. The prefix and partial-decryption checks then decide whether this path can be connected to plaintext and ciphertext under the comparison budget. The prefix check determines whether B is reachable and records the required round-key nibbles. Starting at τ, the partial-decryption check expands dependencies toward the ciphertext through the appended rounds and records the additional nibbles. Only then is $G + P < \kappa$ tested. A $W > 0$ template is marked conditional.

The implementation is suffix-first because the periodic structure is created inside the suffix. Once a suffix witness is found, the same boundary is passed to the prefix-realization routine, which reconstructs input-side expressions and records the round-key nibbles needed to instantiate them. The target coordinate is passed to the partial-decryption routine, which expands its forward dependencies through the appended rounds and records the nibbles needed to evaluate the target from the ciphertext. The two supports are merged before the guessed-bit

count is computed. The procedure does not extend this partial support into a full master-key recovery.

Algorithm 3: Automated key-recovery candidate search.

Input: Round wiring, allowed suffix-boundary family, comparison budget κ, and ranges for R, p, s, a.

1: For each total round count R, enumerate decompositions $R = p + s + a$.
2: Search the s-round suffix for a boundary B, target coordinate τ, and path $\mathcal{R}$ under the symbolic rules of Section 3.
3: Check the p-round prefix realization and record the input-side expressions E_{in} and their required round-key nibbles.
4: Starting at τ, expand its dependencies through the a appended rounds toward the ciphertext; record $\mathcal{C}_{\text{out}}$ and the required partial-decryption round-key nibbles.
5: Compute $G = \text{bits}(\mathcal{K}_{\text{fwd}} \cup \mathcal{K}_{\text{out}})$, add the suffix-event exponent P, and retain the row when $G + P < \kappa$. Mark every $W > 0$ row as conditional on $s(K) \in V$.

Output: Retained rows with their decomposition, target, required events, required round-key nibbles, and total exponent.

The experiments use this procedure in two passes. First, they scan upward in the total number of rounds and stop once the next round has no retained row in the configured search range. This pass identifies the longest candidate found in the current path family. Second, at that round count, they find the lowest accepted value of C. The reported rows therefore record both the longest candidate found and its lowest exponent under the stated ranges. This is why the best suffix-only distinguisher is not automatically the best key-recovery row: a longer suffix can lose to a shorter one if its prefix or partial-decryption guess is too large. In practice, a row may fail because the suffix path is absent, the boundary cannot be realized within the allowed prefix range, or the appended rounds make $G + P$ exceed the selected comparison budget.

5.4 Executing a Selected Candidate

The search outputs a symbolic path and its required round-key nibbles. For a trial value θ, the row defines an induced function h_θ by realizing the input boundary, querying the cipher, and peeling the appended output rounds. The attack marks θ when h_θ passes the predicted Simon-period test.

The value θ is a candidate for the attacked round-key nibbles. Its restrictions to $\mathcal{K}_{\text{fwd}}$ and $\mathcal{K}_{\text{out}}$ are denoted by θ_{in} and θ_{out}, respectively; a component shared by the two supports has the same value in both restrictions. With θ_{in}, the attacker prepares a query family whose first p rounds reach the selected boundary B. With θ_{out}, the attacker performs the required partial decryption after the encryption query and obtains the projected function h_θ. For the correct support value, this projected function is expected to expose the period certified by the suffix path.

Algorithm 4: Executing a selected key-recovery candidate.

Input: Quantum chosen-plaintext access to E_K, a selected row M, prefix expressions E_{in}, required suffix events, and partial-decryption data $\mathcal{C}_{\text{out}}$.

1: Let Θ be the G-bit search space for the attacked round-key value θ.
2: Use amplitude amplification over $\theta \in \Theta$.
3: For a trial θ, assign its bits to $\mathcal{K}_{\text{fwd}} \cup \mathcal{K}_{\text{out}}$; any component shared by the two supports uses the same guessed value.
4: Instantiate E_{in} with θ_{in} and prepare the plaintext superposition whose prefix reaches the selected boundary B.
5: Apply the filters associated with the charged collision and DDT Filtering events. A $W > 0$ row is applicable only when the resulting period lies in V.
6: Query E_K on the retained plaintext superposition.
7: Use θ_{out} and $\mathcal{C}_{\text{out}}$ to peel the last a rounds and project the target coordinate τ, thereby obtaining the induced function h_θ.
8: Confirm the predicted relation for h_θ under multiple independent data settings.

Output: Candidate values for the attacked round-key nibbles.

The G-bit value θ accounts only for the guessed round-key cells. The charged suffix events are imposed as filters on the prepared query family: an X-event asks for the selected collision to occur, while a DDT Filtering event fixes the corresponding key-mixed S-box input so that the required difference transition is retained. Amplitude amplification is applied to the combined key guess and filter conditions, giving the base estimate $\widetilde{O}(2^{(G+P)/2})$. For $W > 0$, this estimate applies only to keys for which $s(K) \in V$.

The output values are candidates only for the round-key nibbles listed by the selected row. We do not analyze how the key schedule might combine them with the remaining material to recover the master key. The row instead specifies the partially evaluated function h_θ, its predicted periodic structure, and the candidate-search estimate under the stated assumptions.

5.5 Candidate Rows and a Representative Witness

Table 7 gives the rows obtained by the automated search. It is intended as a compact index: the column $p + s + a$ records the prefix, suffix, and appended output-side lengths; G is the number of guessed round-key bits; P is the suffix-event exponent; and $C = G + P$ is the base candidate-search exponent before division by two. Only the fully instantiated $W = 0$ rows are included. The GFS experiments use 4-bit cells and an illustrative 80-bit comparison budget aligned with TWINE-80; this is not a standardized GFS key size. CRAFT uses its official 128-bit key baseline; its 64-bit block size is not a 64-bit brute-force baseline. For CRAFT, R follows the official round-counting convention, while the $p + s + a$ split records the searched core-layer decomposition.

Table 7. Key-recovery candidate rows from the automated search. K_{fwd} and K_{out} are the individual input-side and partial-decryption support sizes reported by the implementation; G counts the distinct guessed bits in their union, P is the suffix-event exponent, and $C = G + P$. All rows have $W = 0$ and are strictly below their stated budgets. Full master-key recovery is not included.

Target	κ	R	$p+s+a$	out	K_{fwd}	K_{out}	X	W	G	P	C	$C/2$	Fig.
TWINE-80	80	17	$4+9+4$	u_5	9	7	1	0	64	4	68	34	3
TWINE-128	128	19	$5+9+5$	u_7	15	12	1	0	108	4	112	56	4
LBlock-80	80	17	$4+9+4$	u_{13}	9	7	1	0	64	4	68	34	5
LBlock-128	128	19	$5+9+5$	u_{10}	15	12	1	0	108	4	112	56	6
GFS-2F	80	16	$0+6+10$	u_1	0	16	2	0	64	8	72	36	7
CRAFT-128	128	17	$2+10+4$	u_6	19	18	6	0	96	24	120	60	10
SKINNY-64	64	13	$1+8+4$	u_9	3	7	4	0	40	16	56	28	11
SKINNY-64-128	128	15	$3+8+4$	u_{10}	19	7	4	0	104	16	120	60	12

The search also found a conditional GFS-4F template with $R = 18$, $p+s+a = 4+7+7$, target u_3, $K_{\mathrm{fwd}} = 6$, $K_{\mathrm{out}} = 7$, $X = 2$, and $W = 1$. It has $G = 52$, $P = 12$, and base exponent $C = 64$, but it is not included as an instantiated row in Table 7. Its use requires $s(K) \in V$. The corresponding conditional symbolic witness is shown in Figure 8.

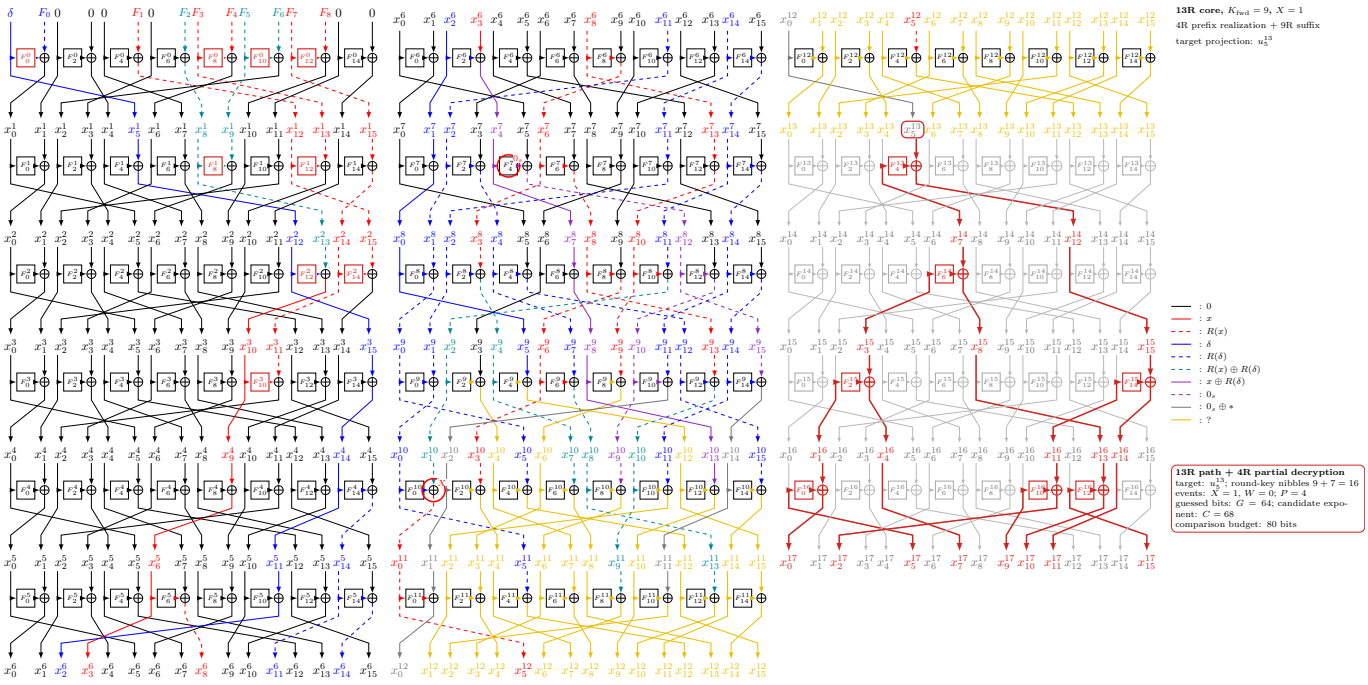

Fig. 3. Representative TWINE-80 17-round key-recovery witness with a 4-round prefix realization, a 9-round suffix, and 4 appended output-side rounds. Colors in rounds 0–13 denote symbolic states; red paths in rounds 13–17 denote the dependencies used for partial decryption.

Figure 3 shows one representative TWINE-80 row from Table 7.

The witness is read from left to right, with the attack stages determined by the round ranges rather than by a single coloring scheme. The first four rounds prepare the boundary $u_9^4 = x$, $u_{14}^4 = \delta$, with all other branches zero, and require nine round-key nibbles. During the nine-round suffix, the transition from $x \oplus R(\delta)$ to $0s$ at the boundary between rounds 7 and 8 fixes the symbolic path but adds no probability term. At the boundary between rounds 10 and 11, the charged collision identifies the two $R(\delta)$-family terms and reduces $R(\delta) \oplus (R(x) \oplus R(\delta))$ to $R(x)$. This is the only X-event, so $X = 1$, $W = 0$, and $P = 4$. The suffix reaches $\tau = u_5^{13} = 0s \oplus R(x) \oplus R(\delta)$.

The red dependencies in the last four rounds are expanded from u_5^{13} toward the ciphertext. They identify the ciphertext branches and seven additional round-key nibbles needed to evaluate this target. The dependency analysis and the subsequent evaluation run in opposite directions: the support is constructed from u_5^{13} toward the ciphertext, whereas partial decryption uses the ciphertext and the guessed nibbles to compute u_5^{13}. Thus $K_{\mathrm{fwd}} = 9$, $K_{\mathrm{out}} = 7$, $G = 64$, and $C = 68$, giving a base candidate-search estimate $\widetilde{O}(2^{34})$. This figure is

one accepted witness, not a uniqueness statement, and it does not include full TWINE master-key recovery.

6 Conclusion

This paper studied the automated construction of quantum periodic distinguishers for block-cipher structures. Starting from the algebraic SMT model of Liu et al., we added checks that connect an internal symbolic path to preceding rounds and that condition one nonlinear step on a DDT bucket. Our main comparison is with that automated model; the results are not claims of the best known attacks on the listed ciphers.

The main methodological additions are input-domain extension via prefix realization and DDT Filtering. Prefix realization determines whether an internal suffix boundary can be produced from input-side expressions, and records the round-key nibbles needed by that realization. DDT Filtering adds a controlled local suffix event based on a DDT-compatible transition through a nonlinear component. The target period does not necessarily lie in the translation space of the selected bucket. We therefore state every $W > 0$ numerical result as conditional on the membership event $s(K) \in V$; $W = 0$ rows are unaffected.

At the distinguisher level, the experiments show how these refinements behave across several cipher families. For the generalized-Feistel and Skipjack-type structures, the refined model finds longer symbolic periodic paths. For TWINE and LBlock, the results give parallel 4-bit Feistel-like profiles under the same corrected symbolic rules. For CRAFT and SKINNY, the experiments show how stronger SPN-style diffusion changes the balance between longer paths and the conditions needed to realize or filter them. The experiments separate input realization, local filtering, and collision conditions so that the source of each change can be checked directly.

We also combined selected witnesses with the round-key nibbles required before the periodic path and for partial decryption after it. The candidate-search estimate includes collision and filtering conditions. When $W > 0$, it is conditional on $s(K) \in V$. These rows search only for candidates for the listed round-key material; full master-key recovery through the key schedule is outside the paper.

Acknowledgements

The authors used generative AI tools as assistance for language polishing and code organization. All technical content and claims were checked and validated by the authors. The authors take full responsibility for the correctness and integrity of the submitted work.

References

1. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. pp. 123–153. Lecture Notes in Computer Science, Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_5, https://doi.org/10.1007/978-3-662-53008-5_5
2. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symmetric Cryptol. **2019**(1), 5–45 (2019). https://doi.org/10.13154/TOSC.V2019.I1.5-45, https://doi.org/10.13154/tosc.v2019.i1.5-45
3. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991). https://doi.org/10.1007/BF00630563, https://doi.org/10.1007/BF00630563
4. 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.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. pp. 552–583. Lecture Notes in Computer Science, Springer (2019). https://doi.org/10.1007/978-3-030-34578-5_20, https://doi.org/10.1007/978-3-030-34578-5_20
5. Bonnetain, X., Leurent, G., Naya-Plasencia, M., Schrottenloher, A.: Quantum linearization attacks. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I. pp. 422–452. Lecture Notes in Computer Science, Springer (2021). https://doi.org/10.1007/978-3-030-92062-3_15, https://doi.org/10.1007/978-3-030-92062-3_15
6. Canale, F., Leander, G., Stennes, L.: Simon's algorithm and symmetric crypto: Generalizations and automatized applications. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III. pp. 779–808. Lecture Notes in Computer Science, Springer (2022). https://doi.org/10.1007/978-3-031-15982-4_26, 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 - 20th International Conference, Inscrypt 2024, Kunming, China, December 14-16, 2024, Revised Selected Papers, Part II. pp. 425–443. Lecture Notes in Computer Science, Springer (2024). https://doi.org/10.1007/978-981-96-4734-7_21, https://doi.org/10.1007/978-981-96-4734-7_21
8. 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.) Progress in Cryptology - INDOCRYPT 2020 - 21st International Conference on Cryptology in India, Bangalore, India,

December 13-16, 2020, Proceedings. pp. 373–394. Lecture Notes in Computer Science, Springer (2020). https://doi.org/10.1007/978-3-030-65277-7_17, https://doi.org/10.1007/978-3-030-65277-7_17
9. 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, https://doi.org/10.1007/s11128-021-03027-x
10. Dong, X., Dong, B., Wang, X.: Quantum attacks on some feistel block ciphers. Des. Codes Cryptogr. **88**(6), 1179–1203 (2020). https://doi.org/10.1007/S10623-020-00741-Y, https://doi.org/10.1007/s10623-020-00741-y
11. 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, https://doi.org/10.1007/s11432-017-9436-7
12. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 212–219. ACM (1996). https://doi.org/10.1145/237814.237866, https://doi.org/10.1145/237814.237866
13. 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, https://doi.org/10.1007/s11128-020-02929-6
14. 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. pp. 461–480. Lecture Notes in Computer Science, Springer (2020). https://doi.org/10.1007/978-3-030-44223-1_25, https://doi.org/10.1007/978-3-030-44223-1_25
15. Ito, G., Hosoyamada, A., Matsumoto, R., Sasaki, Y., Iwata, T.: Quantum chosen-ciphertext attacks against feistel ciphers. In: Matsui, M. (ed.) Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings. pp. 391–411. Lecture Notes in Computer Science, Springer (2019). https://doi.org/10.1007/978-3-030-12612-4_20, https://doi.org/10.1007/978-3-030-12612-4_20
16. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. pp. 207–237. Lecture Notes in Computer Science, Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_8, https://doi.org/10.1007/978-3-662-53008-5_8
17. Knudsen, L.R., Wagner, D.A.: On the structure of skipjack. Discret. Appl. Math. **111**(1-2), 103–116 (2001). https://doi.org/10.1016/S0166-218X(00)00347-4, https://doi.org/10.1016/S0166-218X(00)00347-4
18. 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, https://doi.org/10.1109/ISIT.2010.5513654
19. 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/

20. Leander, G., May, A.: Grover meets simon - quantumly attacking the fx-construction. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II. pp. 161–178. Lecture Notes in Computer Science, Springer (2017). https://doi.org/10.1007/978-3-319-70697-9_6, https://doi.org/10.1007/978-3-319-70697-9_6
21. Liu, Q., Wang, H., Wang, J., Li, B., Wang, M.: Quantum periodic distinguisher construction: Symbolization method and automated tool. In: Hanaoka, G., Yang, B. (eds.) Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part VIII. pp. 224–255. Lecture Notes in Computer Science, Springer (2025). https://doi.org/10.1007/978-981-95-5125-5_8, https://doi.org/10.1007/978-981-95-5125-5_8
22. 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., Gupta, S.S. (eds.) Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings. pp. 433–455. Lecture Notes in Computer Science, Springer (2019). https://doi.org/10.1007/978-3-030-35423-7_22, https://doi.org/10.1007/978-3-030-35423-7_22
23. Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. pp. 55–64. Lecture Notes in Computer Science, Springer (1993). https://doi.org/10.1007/3-540-48285-7_6, https://doi.org/10.1007/3-540-48285-7_6
24. Nyberg, K.: Generalized feistel networks. In: Kim, K., Matsumoto, T. (eds.) Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings. pp. 91–104. Lecture Notes in Computer Science, Springer (1996). https://doi.org/10.1007/BFB0034838, https://doi.org/10.1007/BFb0034838
25. Santoli, T., Schaffner, C.: Using simon's algorithm to attack symmetric-key cryptographic primitives. Quantum Inf. Comput. **17**(1&2), 65–78 (2017). https://doi.org/10.26421/QIC17.1-2-4, https://doi.org/10.26421/QIC17.1-2-4
26. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(5), 1474–1483 (1997). https://doi.org/10.1137/S0097539796298637, https://doi.org/10.1137/S0097539796298637
27. Sun, H., Cai, B., Qin, S., Wen, Q., Gao, F.: Quantum attacks on type-1 generalized feistel schemes. IACR Cryptol. ePrint Arch. **2023**, 776 (2023), https://eprint.iacr.org/2023/776
28. Suzaki, 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. pp. 339–354. Lecture Notes in Computer Science, Springer (2012). https://doi.org/10.1007/978-3-642-35999-6_22, https://doi.org/10.1007/978-3-642-35999-6_22
29. Wu, W., Zhang, L.: Lblock: A lightweight block cipher. In: López, J., Tsudik, G. (eds.) Applied Cryptography and Network Security - 9th International Conference,

ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings. pp. 327–344. Lecture Notes in Computer Science, Springer (2011). https://doi.org/10.1007/978-3-642-21554-4_19, https://doi.org/10.1007/978-3-642-21554-4_19

30. 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, https://doi.org/10.46586/tosc.v2024.i2.296-321
31. Xu, Y., Du, X., Jia, M., Wang, X., Zou, J.: Quantum attacks on generalized feistel networks based on the strong-weak separability. Quantum Inf. Process. **22**(10), 375 (2023). https://doi.org/10.1007/S11128-023-04135-6, https://doi.org/10.1007/s11128-023-04135-6

A Supplementary TWINE Key-Recovery Figures

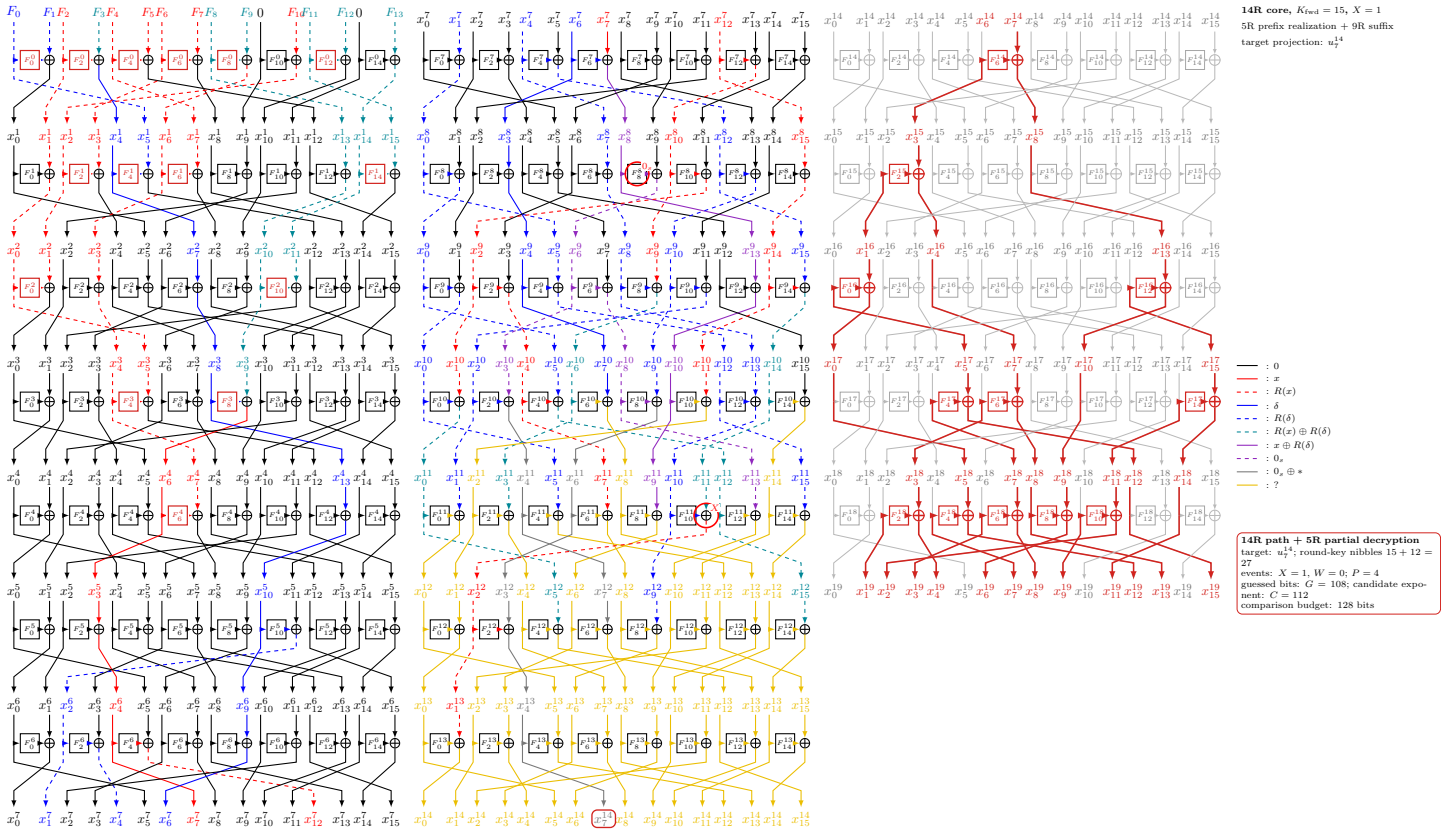

Fig. 4. TWINE-128 19-round key-recovery witness with a 5-round prefix realization, a 9-round suffix, and 5 appended output-side rounds.

B Supplementary LBlock Key-Recovery Figures

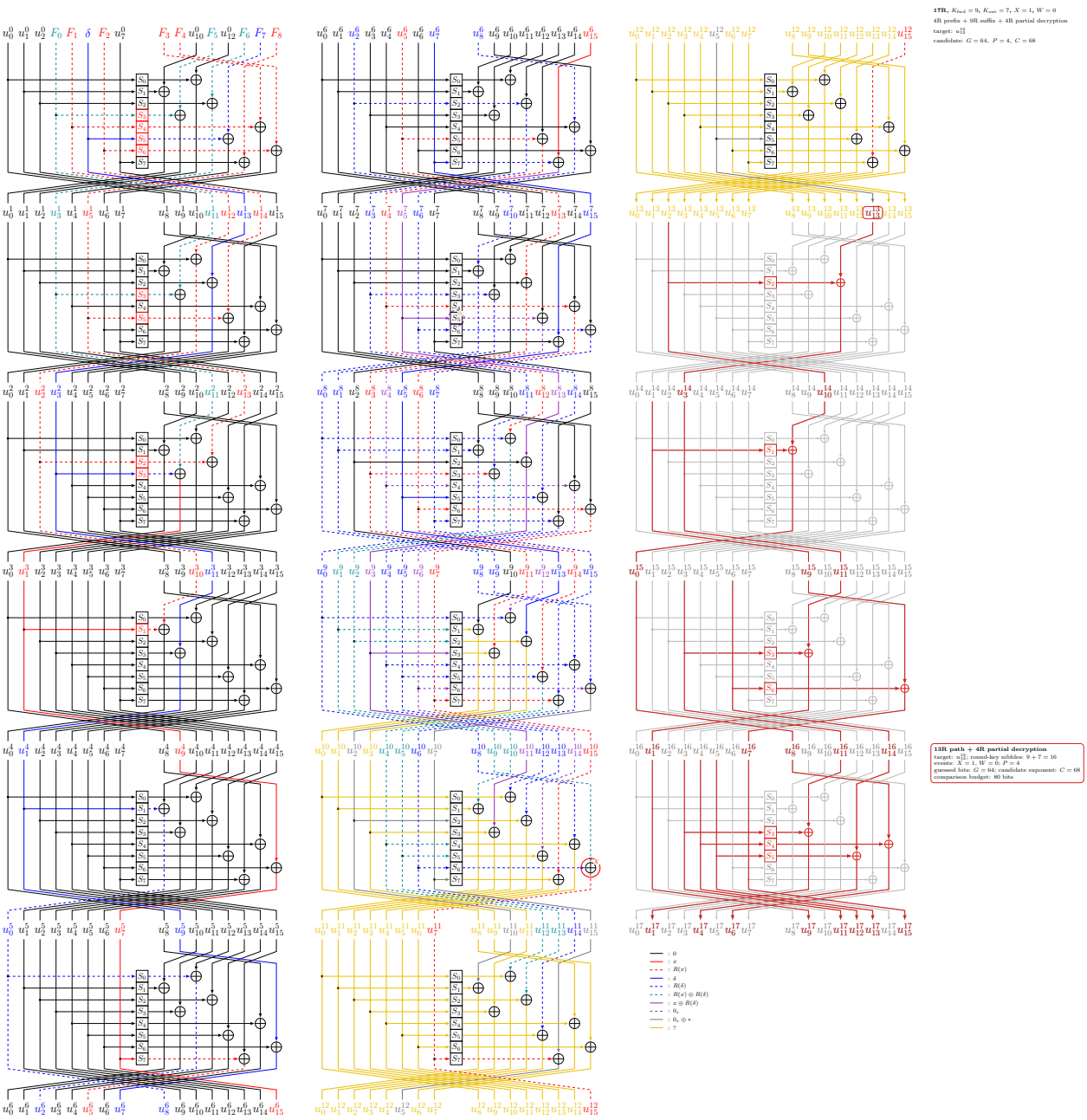

Fig. 5. LBlock-80 17-round key-recovery witness with a 4-round prefix realization, a 9-round suffix, and 4 appended output-side rounds.

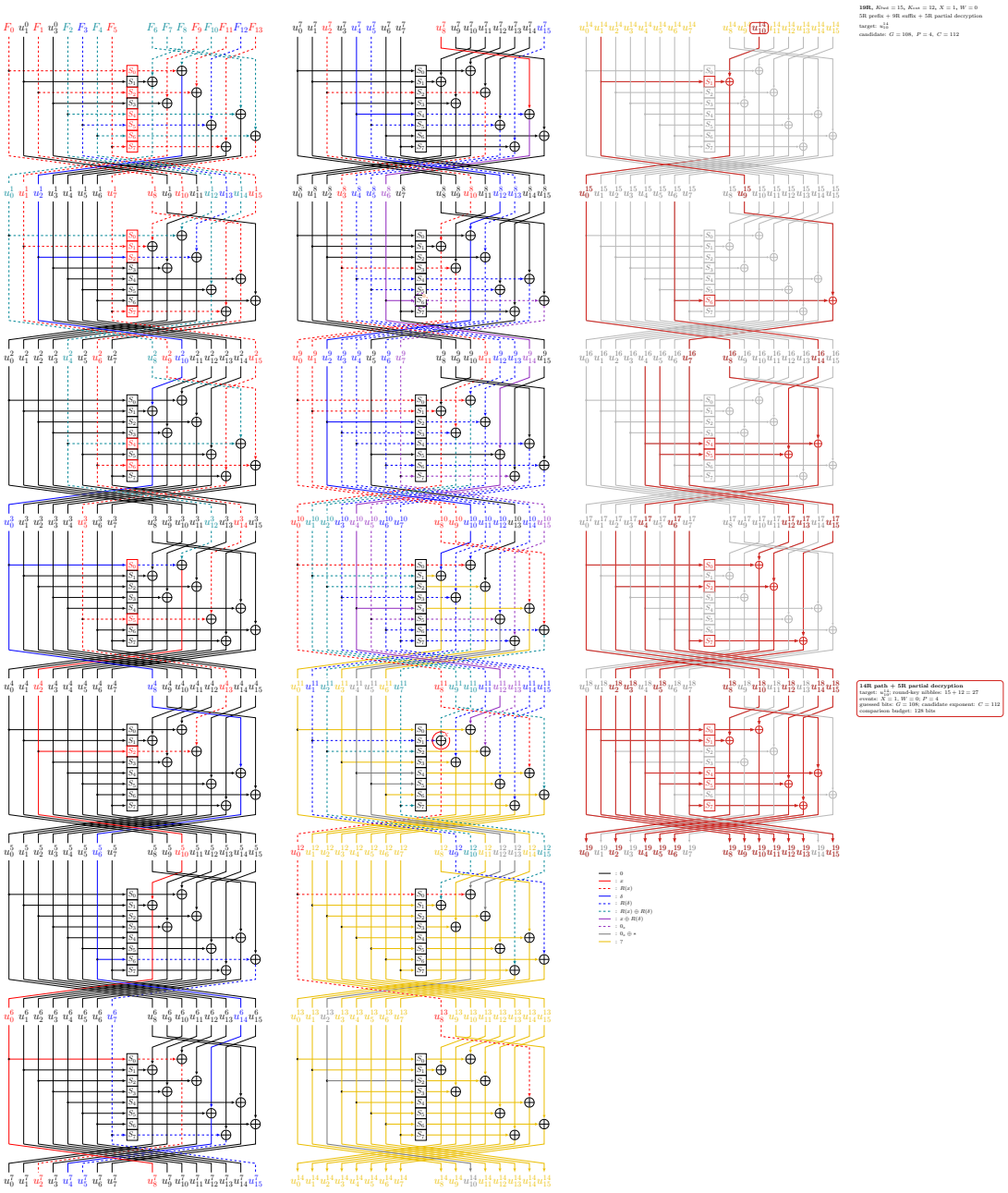

Fig. 6. LBlock-128 19-round key-recovery witness with a 5-round prefix realization, a 9-round suffix, and 5 appended output-side rounds.

C Supplementary GFS Key-Recovery Figures

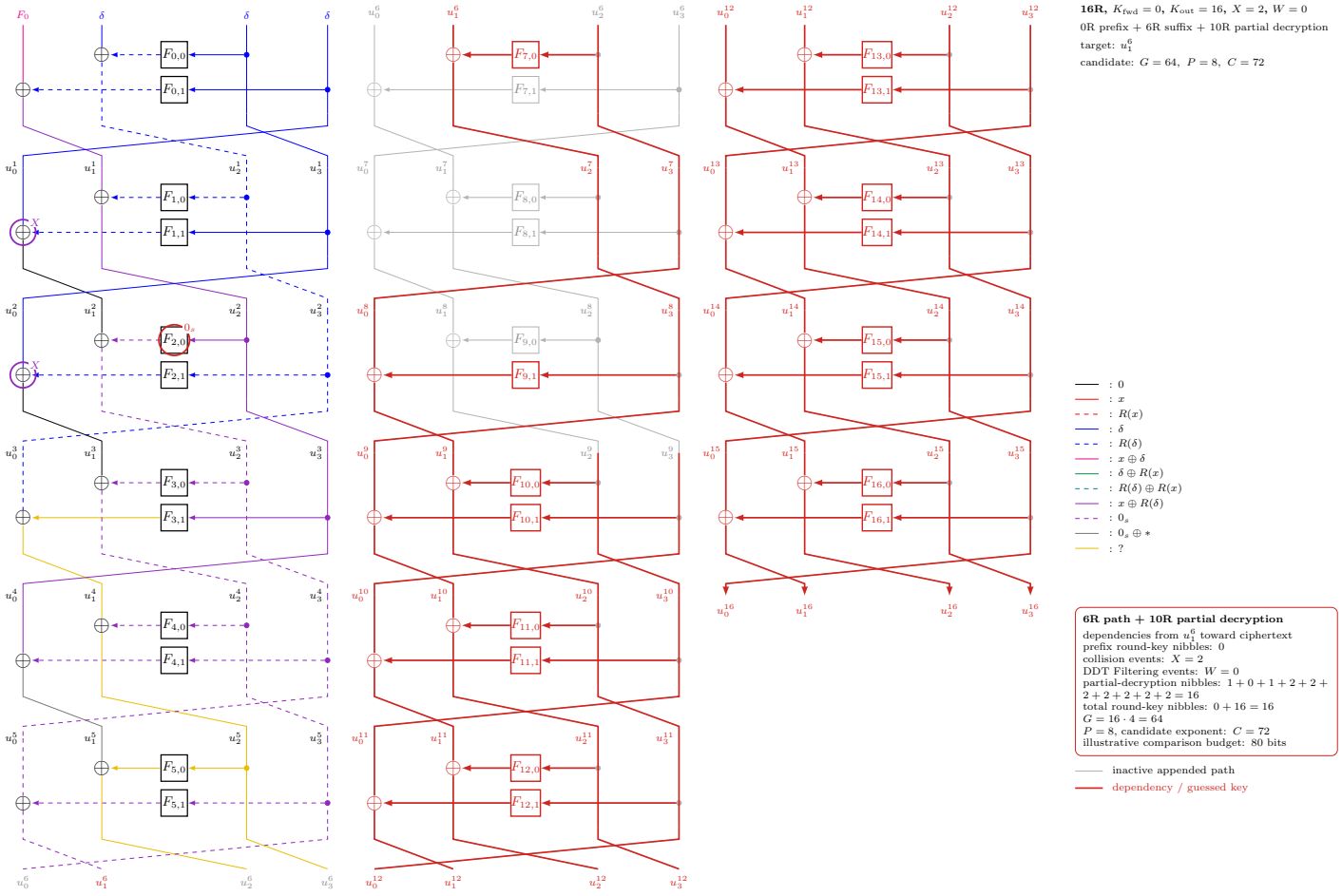

Fig. 7. GFS-2F 16-round key-recovery witness under the 80-bit comparison budget, with no prefix realization, a 6-round suffix, and 10 appended output-side rounds.

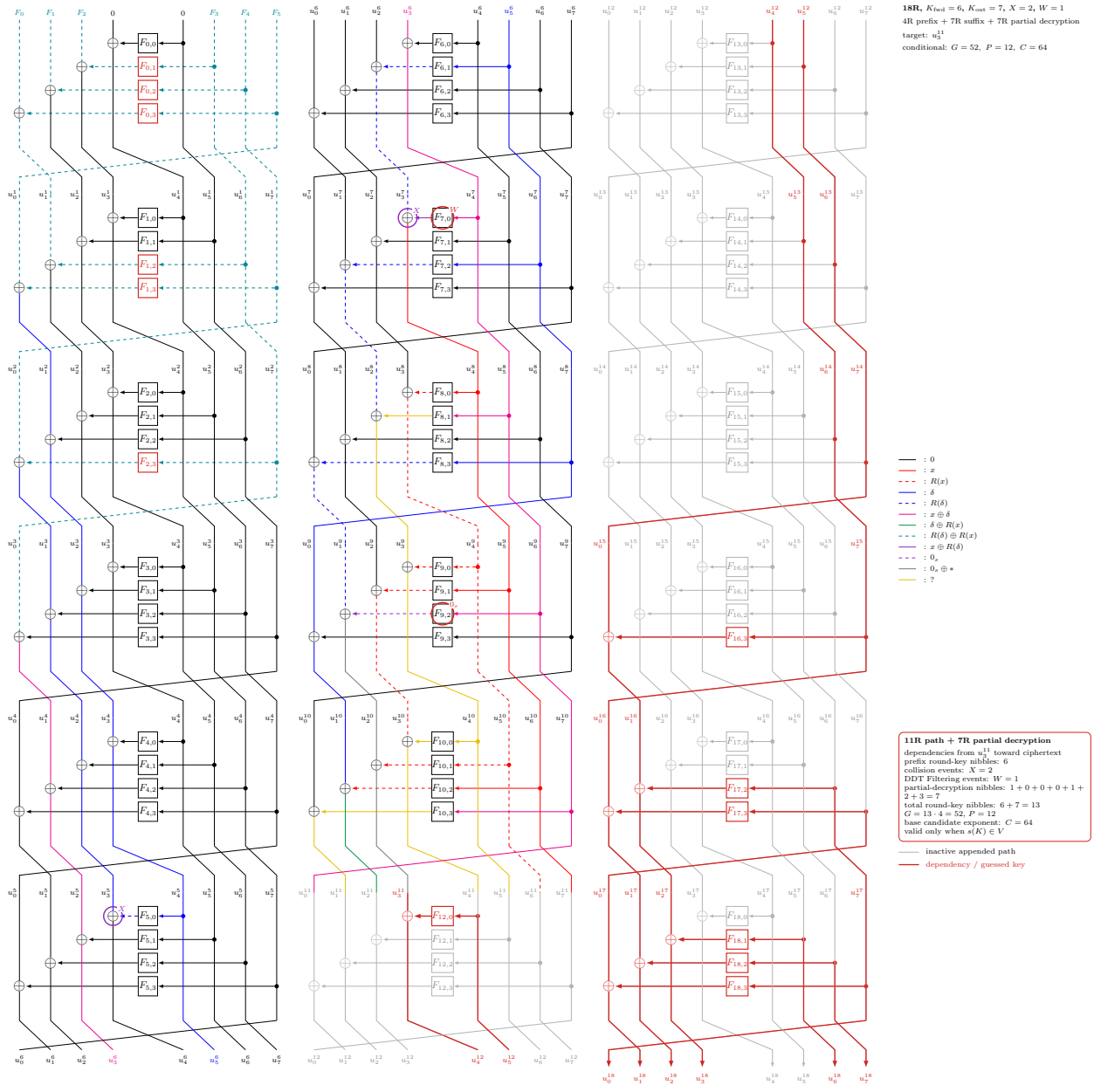

Fig. 8. Conditional GFS-4F 18-round symbolic template under the 80-bit comparison budget, with a 4-round prefix realization, a 7-round suffix, and 7 appended output-side rounds. It applies only when $s(K) \in V$.

D Supplementary Skipjack-B Key-Recovery Figure

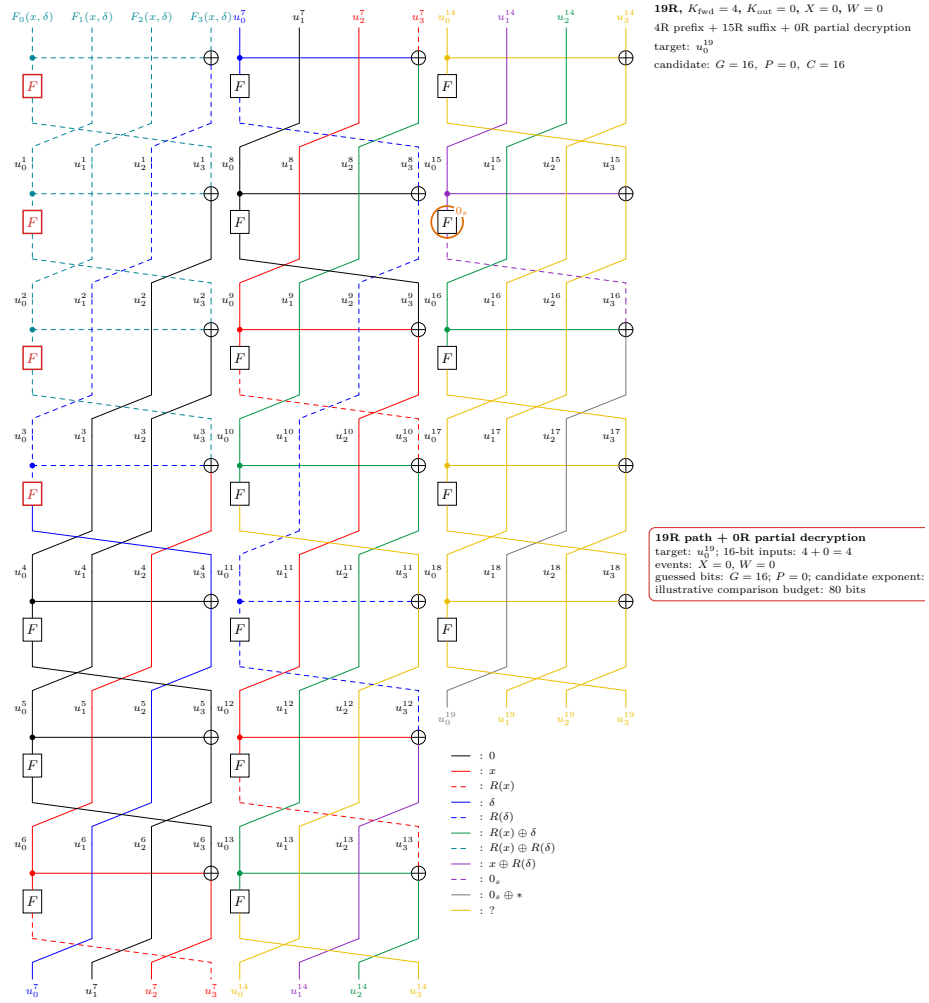

Fig. 9. Skipjack-B 19-round witness with a 4-round prefix realization, a 15-round suffix, and no appended output-side rounds.

E Supplementary CRAFT Key-Recovery Figure

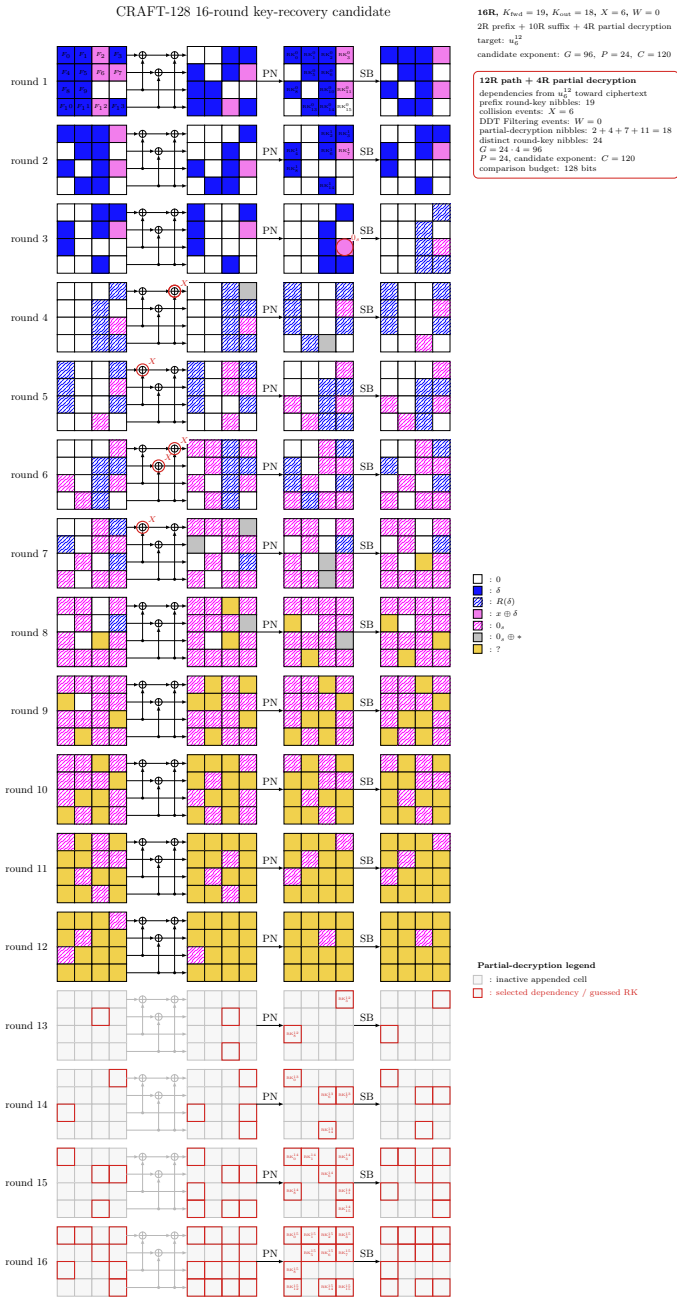

Fig. 10. CRAFT-128 official 17-round key-recovery witness with core split $2 + 10 + 4$: a 2-round prefix realization, a 10-round suffix, and 4 appended output-side rounds.

F Supplementary SKINNY Key-Recovery Figure

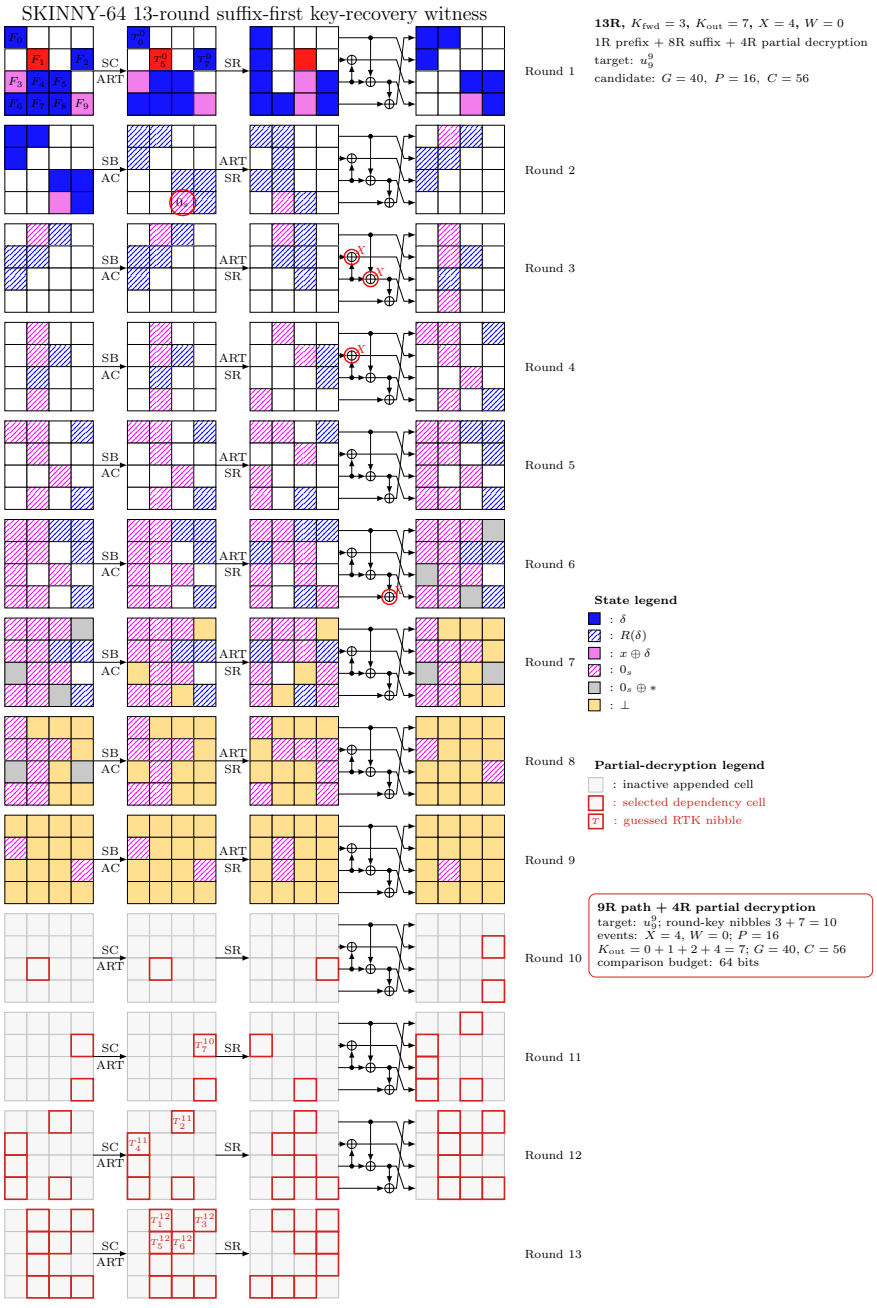

Fig. 11. SKINNY-64 13-round key-recovery witness with a 1-round prefix realization, an 8-round suffix, and 4 appended output-side rounds.

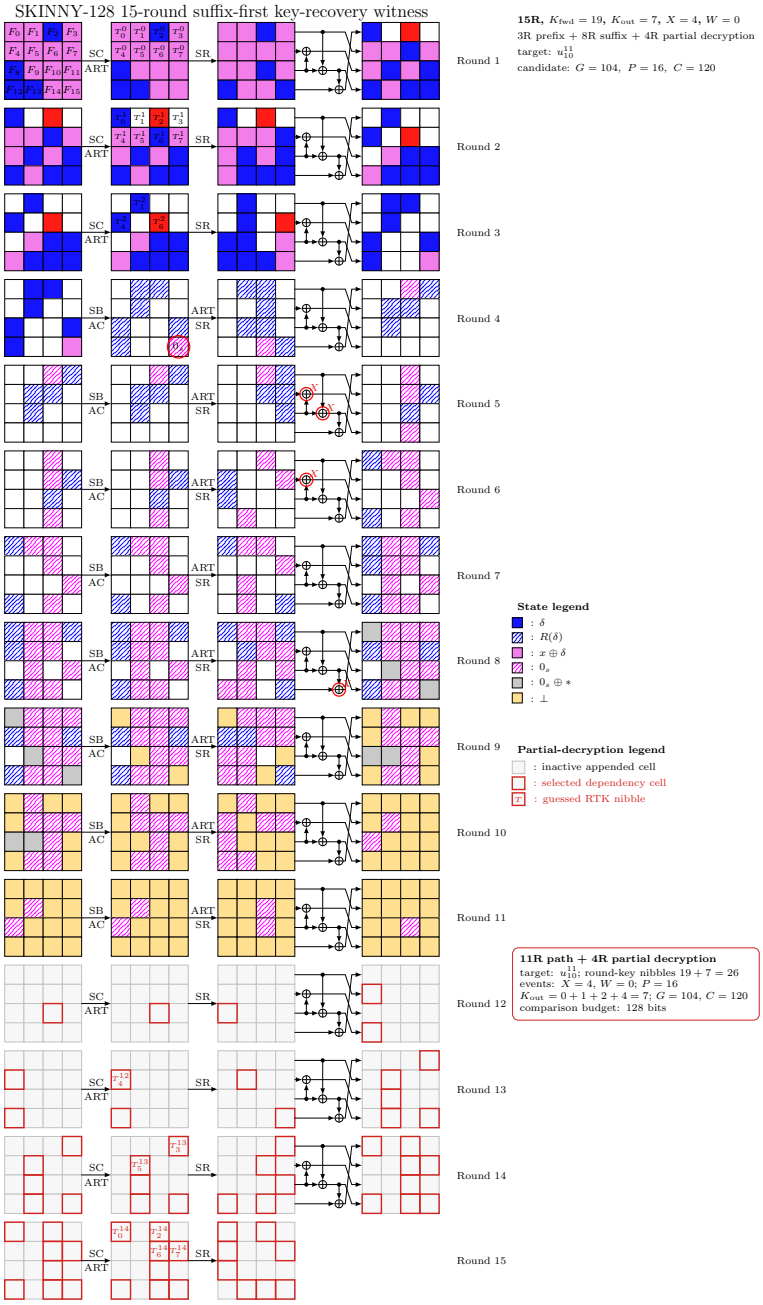

Fig. 12. SKINNY-64-128 15-round key-recovery witness with a 3-round prefix realization, an 8-round suffix, and 4 appended output-side rounds.