Shifting Dark Matter: Two-Query Key Recovery on a Post-Quantum OPRF

Hanbeom Shin*, Insung Kim*, Seokhie Hong† and Dongjae Lee‡
*Korea University, Seoul, Republic of Korea
Email: {newonetiger, cmcom35}@korea.ac.kr
†SmartM2M, Seoul, Republic of Korea
Email: shhong@smartm2m.co.kr
‡Kangwon National University, Chuncheon, Republic of Korea
Email: dongjae.lee@kangwon.ac.kr

***Abstract*—At IEEE S&P 2026, Aranha et al. proposed a maliciously secure post-quantum oblivious pseudorandom function (OPRF) from Crypto Dark Matter. Its strong PRF is $F_k(x) = B \cdot \mathrm{vec}(k \cdot \mathrm{elt}(E(x))) \bmod 3$, where the key $k \in \mathbb{F}_{2^n}$ multiplies a systematic encoding $E(x)$. In this paper, we show that a client that follows the protocol recovers the key from two chosen queries. Multiplication by α^{-2} commutes with the key multiplication and acts on the polynomial basis as a shift, up to a correction from two bits. We prove that input pairs whose encodings differ by the factor α^2 correspond to the binary solutions of six linear systems over $\mathbb{F}_3$. For the proposed parameters, we found such a pair in 146 seconds. After a guess of two bits, the outputs give a full-weight ternary syndrome decoding instance whose 164 equations almost always determine the other 254 bits of $k \cdot \mathrm{elt}(E(x))$. Counting memory in list elements, we estimate time and memory complexities of $2^{67.3}$ and $2^{51.8}$, or $2^{69.9}$ and $2^{33.3}$, both far below the claimed 128-bit security. For a 64-bit security target, the whole attack recovered the key with $2^{34.39}$ operations, matching our cost model. A dense field modulus does not prevent the attack. For 128-bit inputs, a non-systematic code of length at least 160 heuristically removes such pairs, and a public $\mathbb{F}_2$-linear mixing of the output removes the linear structure of the key recovery.**

1. Introduction

An oblivious pseudorandom function (OPRF) lets a client evaluate a pseudorandom function (PRF) on its input under a key held by a server [1]. The server learns nothing about the input, and the client learns only the PRF value. OPRFs are a building block of password-authenticated key exchange such as OPAQUE [2] and of private set intersection. Post-quantum OPRFs include a verifiable OPRF from lattices [3] and the power-residue OPRF of Yang et al. [4]. Only a handful of post-quantum OPRFs are secure against malicious parties [5].

Boneh et al. [6] introduced the crypto dark matter or alternating moduli paradigm, in which PRF candidates mix linear maps modulo 2 and modulo 3. A typical weak PRF computes $B \cdot_3 (A \cdot_2 x)$ for a secret matrix A and a public matrix B, where $\cdot_2$ and $\cdot_3$ denote products modulo 2 and 3 [5], [7]. A weak PRF only needs to be indistinguishable from a random function on random inputs. The OPRF protocols for these PRFs either were secure only against semi-honest adversaries [6]–[8] or, as for the partially oblivious PRF of Albrecht et al. [9], relied on costly fully homomorphic encryption and on heuristic assumptions [5].

At IEEE S&P 2026, Aranha et al. [5] proposed a post-quantum OPRF in this paradigm, based on vector oblivious linear evaluation (VOLE) over $\mathbb{F}_{2^n}$, with simulation-based security against malicious adversaries. Its weak PRF computes $B \cdot_3 \mathrm{vec}(k \cdot y)$ for a key $k \in \mathbb{F}_{2^n}$ and an input $y \in \mathbb{F}_{2^n}$, where $B \in \mathbb{F}_3^{t \times n}$ is public, the output has t coordinates in $\mathbb{F}_3$, and vec writes a field element in the polynomial basis $1, \alpha, \ldots, \alpha^{n-1}$. A strong PRF must also be indistinguishable from a random function on chosen inputs. Hashing the input with a random oracle gives one but is costly when the client may be malicious [5]. Following Boneh et al. [6] and Albrecht et al. [9], Aranha et al. instead encode an input $x \in \mathbb{F}_2^d$ as $E(x) = \mathrm{decomp}(G \cdot_3 x) \,\|\, 1$, where G is a systematic code over $\mathbb{F}_3$ and decomp writes each parity coordinate with two bits. Their strong PRF is

$$F_k(x) = B \cdot_3 \mathrm{vec}\big(k \cdot \mathrm{elt}(E(x))\big),$$

where elt is the inverse of vec. For 128-bit security, they set $(n, t) = (256, 82)$ and use a code of length 191 and dimension $d = 128$ [5].

At these parameters, the encoding equation of Aranha et al. [5] gives 255 bits for a 256-bit field element. Their public implementation[1] instead uses a different bit order, layout and two-bit tail. We analyze separately our completion E_{pad} of the encoding equation E_{spec} and the implementation encoder E_{impl}. Each has *related inputs*, whose encodings differ by a field multiplier that commutes with key multiplication.

Several attacks target weak PRFs in the alternating moduli paradigm. Cheon et al. [10] distinguished a construction with a circulant key matrix and a special public output map.

1. We use `oprf/src/input_encoding.rs` in the authors' public repository at revision `102d98e`.

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

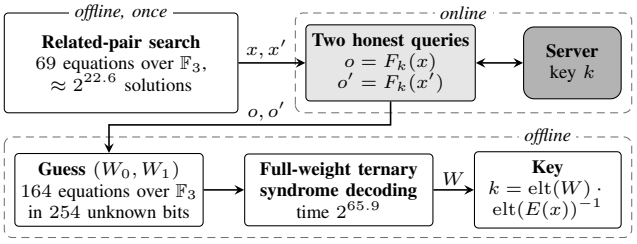


Figure 1. Flow of the key recovery for the proposed parameters $(n, t) = (256, 82)$. Here $E = E_{\mathrm{impl}}$, and both queries pass the implementation's input check (Appendix B). The pair search does not depend on the key. After a guess of (W_0, W_1), the outputs give linear equations over $\mathbb{F}_3$ in the other bits of $W = \mathrm{vec}(k \cdot \mathrm{elt}(E(x)))$. Each of W_0 and W_1 is one bit, so the three-level representation decoder runs on four assignments. Its total modeled time $2^{65.9}$ includes all four runs (Table 1).

Dinur et al. [7] conjectured that random output maps with few coordinates would resist this approach, but Hu et al. [11] later attacked that setting. Ayala and Raddum [12] recovered keys of the variant of Alamati et al. [8] that multiplies each input coordinate by a secret key component, and Sidem and Wang [13] improved that attack. Aranha et al. [5] traced the latter two attacks to positions with a zero key bit.

Aranha et al. argued that their variant resists these attacks [5]. The attack of Hu et al. uses the shift invariance of circulant multiplication and the rank deficiency of random circulant matrices. Reduction modulo an irreducible polynomial instead of $X^n - 1$ "breaks the symmetry of a circulant matrix" [5], and multiplication by a nonzero key is invertible, so the zero-key-bit attacks do not apply. Aranha et al. therefore conjectured that $n = 2\lambda$ suffices for λ-bit security of their weak PRF. Our attack needs chosen related inputs and does not address this conjecture for the weak PRF.

Van Baarsen et al. [14] use a variant of this strong PRF in a maliciously secure shuffled distributed OPRF. They use a non-systematic code, whose effect on related inputs we analyze in Sections 5.1 and 5.2.2.

Our Contributions

In this paper, we show that related inputs exist for the strong PRF of Aranha et al. and enable two-query key recovery. Multiplication by α^{-2} commutes with the key multiplication and acts on the polynomial basis as a shift by two positions, up to a correction from the two lowest bits. Since G is systematic, the condition that a shifted encoding is again valid is linear over $\mathbb{F}_3$ once two parity values are guessed. As in slide attacks [15], the two queries are related by a map that commutes with the keyed function. Figure 1 gives an overview of the attack. Our main contributions are summarized as follows:

- **Related inputs from commuting shifts.** We prove that inputs satisfying $\mathrm{elt}(E(x)) = \alpha^2 \cdot \mathrm{elt}(E(x'))$ are binary solutions of linear systems over $\mathbb{F}_3$. At the proposed parameters, we found such a pair for E_{impl}. Both inputs pass the implementation's input check.
- **Two-query key recovery.** Guessing two bits turns the two outputs into 164 ternary equations in 254 binary unknowns, a full-weight syndrome decoding problem. For the proposed parameters, information set decoding (ISD) with a three-level representation technique [16], [17] has the lowest modeled time in Table 1, at $2^{65.9}$ operations and $2^{48.5}$ stored list elements. The two outputs also give a distinguisher with advantage at least 0.93 (Proposition 2).
- **Experimental validation.** At our reduced test parameters $(n, d, m, t) = (128, 64, 31, 42)$, the attack recovered the unique key from two queries for a completion of E_{spec} and for an encoder with the bit maps, layout, tail and coefficient order of E_{impl}. At the proposed $n = 256$, we found related pairs for completions of E_{spec} and for E_{impl}. For one pair of each kind, the rank-164 key-recovery system contains the true solution.
- **Attack conditions and countermeasures.** When a small-shift pair exists, a dense modulus does not stop the attack. All three $n = 96$ dense-modulus tests recovered the key. For $d = 128$, a non-systematic code of length 160 leaves 0.59 expected related inputs per branch under our model; this does not rule out pairs. General public output mixing changes the relation used by our two-bit affine decoder, without establishing security of the modified PRF.

Organization

The remainder of the paper is organized as follows. Section 2 introduces the strong PRF of Aranha et al., its OPRF protocols and ternary syndrome decoding. Section 3 shows that multiplication by α^{-s} acts as a near shift and characterizes the related inputs by linear systems over $\mathbb{F}_3$. Section 4 recovers the key from two queries and validates the cost model. Section 5 discusses the affected protocols, related constructions and countermeasures. Finally, Section 6 concludes the paper. The source code for the experiments in this paper is available online.[2]

2. Preliminaries

2.1. Notation

Let $f \in \mathbb{F}_2[X]$ be irreducible of degree n with a root $\alpha \in \mathbb{F}_{2^n}$, and write $f = X^n + g(X)$ with $g(X) = \sum_{i=0}^{r} g_i X^i$, where $r = \deg g$ and $g_0 = 1$. The map $\mathrm{vec}\colon \mathbb{F}_{2^n} \to \mathbb{F}_2^n$ sends $\sum_{q=0}^{n-1} Y_q \alpha^q$ to $(Y_0, \ldots, Y_{n-1})$, and elt is its inverse (vec_α and elt_α in [5]). Bit positions are numbered from 0. For $Y \in \mathbb{F}_2^n$, we pad its coordinate sequence with zeros and set $Y_q := 0$ for $q > n - 1$. We write $[u, v]$ for the integer interval $\{u, \ldots, v\}$, which is empty when $u > v$.

2. https://anonymous.4open.science/r/shifting-dark-matter/

For a vector y, $y_{[u,v]} := (y_u, \ldots, y_v)$, $\mathrm{supp}(y)$ is the set of positions of its nonzero coordinates, and $y \,\|\, y'$ is the concatenation of y and y'. We write $\oplus$ for the addition of bits and bit vectors and $+$ for the addition in $\mathbb{F}_{2^n}$, in $\mathbb{F}_3$ and in the integers. We read bits as elements of $\mathbb{F}_3$, call a vector over $\mathbb{F}_3$ *binary* if its coordinates lie in $\{0,1\}$, and write $\mathbf{1}$ for the all-one vector. As in [5], $B \cdot_3 w$ is the product of a matrix B over $\mathbb{F}_3$ and a vector w modulo 3.

2.2. The PRF of Aranha et al.

Aranha et al. [5] use a secret key $k \in \mathbb{F}_{2^n}$ and a random public matrix $B \in \mathbb{F}_3^{t \times n}$, and their weak PRF maps $y \in \mathbb{F}_{2^n}$ to $B \cdot_3 \mathrm{vec}(k \cdot y) \in \mathbb{F}_3^t$. A strong PRF must also be indistinguishable from a random function on inputs that the adversary chooses [5]. Aranha et al. therefore follow Boneh et al. [6] and encode the input with a random systematic code over $\mathbb{F}_3$ and a binary decomposition. As suggested by Albrecht et al. [9], they also append a bit 1. The resulting strong PRF maps $x \in \mathbb{F}_2^d$ to

$$\begin{aligned} F_k(x) &= B \cdot_3 \mathrm{vec}\big(k \cdot \mathrm{elt}(E(x))\big), \\ E(x) &= \mathrm{decomp}(G \cdot_3 x) \,\|\, \tau, \end{aligned} \tag{1}$$

where $G = [I_d \mid G']^\top$ is a systematic code of length $n' = d + m$ over $\mathbb{F}_3$ and τ is a short tail. The encoding equation in [5] has $\tau = (1)$; below we specify the extra padding bit needed at the proposed parameters.

The parity values of x are the coordinates $z_i(x)$ of $z(x) := G'^\top \cdot_3 x \in \mathbb{F}_3^m$. Here d denotes the input length, and the same formula defines the linear forms z_i on $\mathbb{F}_3^d$. The map decomp keeps the d systematic coordinates and writes each parity z_i with two bits $\mathrm{bin}(z_i) = (b_0(z_i), b_1(z_i))$. Aranha et al. [5] compute these bits as $b_0(z) = 2z + 2z^2$ and $b_1(z) = z + 2z^2$, so $\mathrm{bin}(0) = (0,0)$, $\mathrm{bin}(1) = (1,0)$ and $\mathrm{bin}(2) = (0,1)$. Their shared-input encoding places all first bits before all second bits. In this *block layout*, position $j < d$ of $E(x)$ holds x_j, position $d + i$ holds $b_0(z_i)$ and position $d + m + i$ holds $b_1(z_i)$ for $i \in [0, m-1]$. For a word $y \in \mathbb{F}_2^n$, we call (y_{d+i}, y_{d+m+i}) the *cell* i of y.

Aranha et al. [5] give decomposition formulas and an input-check gadget with opposite bit orders. The gadget $(2,1)$ decodes $(1,0)$ as 2, whereas the formulas encode 1 as $(1,0)$. We use the formulas' order for their encoding equation. Their public implementation uses $b_0^{\mathrm{impl}}(z) = z + 2z^2$ and $b_1^{\mathrm{impl}}(z) = 2z + 2z^2$. Thus $b_0^{\mathrm{impl}} = b_1$ and $b_1^{\mathrm{impl}} = b_0$. We analyze the implementation's encoder in Appendix B, where b_0 and b_1 denote its component maps. In either order, bin is injective and its image is $\{0,1\}^2 \setminus \{(1,1)\}$, whose elements we call *valid cells*. A pair (β_0, β_1) of bits is thus a valid cell if and only if $\beta_0\beta_1 = 0$, and $\mathrm{bin}^{-1}(\beta_0, \beta_1)$ is then the parity value that it encodes. For the chosen bit order, write $Z_* := \{z \in \mathbb{F}_3 : b_1(z) = 0\}$. It is $\{0,1\}$ for the encoding equation and $\{0,2\}$ for the implementation's encoder.

Aranha et al. append one bit 1 and write $n = d + 2m + 1$ [5]. Their proposed 128-bit parameters give $(n,t) = (256, 82)$ and $(n', d) = (191, 128)$, hence $m = 63$ and $d + 2m + 1 = 255$. Thus their encoding equation leaves one bit unspecified at $n = 256$. We distinguish the encoding equation of Aranha et al., our completion with a zero before the final bit, and their public implementation:

$$\begin{aligned} E_{\mathrm{spec}}(x) &= x \,\|\, b_0(z(x)) \,\|\, b_1(z(x)) \,\|\, (1) \in \mathbb{F}_2^{255}, \\ E_{\mathrm{pad}}(x) &= x \,\|\, b_0(z(x)) \,\|\, b_1(z(x)) \,\|\, (0,1) \in \mathbb{F}_2^{256}, \\ E_{\mathrm{impl}}(x) &= \mathrm{rev}\big(b_0^{\mathrm{impl}}(z(x)) \,\|\, x \\ &\qquad \|\, b_1^{\mathrm{impl}}(z(x)) \,\|\, (1,1)\big) \in \mathbb{F}_2^{256}. \end{aligned}$$

Here b_j acts coordinatewise on $z(x)$, as does b_j^{impl}, and rev reverses the bit vector before conversion to a field element. Appendix A details the length and bit-order differences between these encoders. When a result states other bit maps or another tail, E_{pad} denotes the systematic-first encoding with those bit maps and that tail. The construction requires an irreducible polynomial but does not specify one for these parameters. Our experiments use the polynomial of the field `F256b` in the authors' public implementation, $f = X^{256} + X^{10} + X^5 + X^2 + 1$, so $r = 10$ and $g_1 = 0$.

2.3. The OPRF Protocols

In the maliciously secure protocol of Aranha et al. [5], the client chooses x and the server holds k. Authenticated VOLE gives the parties shares of $k \cdot \mathrm{elt}(E(x))$ while the client commits to $E(x)$. The input check tests the fixed tail $\tau = (1)$, written as $y_n = 1$ in the one-bit encoding, and the code constraints using a gadget matrix and a parity-check matrix H. Appendix A shows that this check also accepts some words outside the image of E, but our attack uses only valid encodings. The protocol uses n local doubly authenticated bits (laBits) to express the commitment modulo 3 and n doubly authenticated bits (daBits) to convert the product bits from modulo 2 to modulo 3. The parties apply B to their shares, and the client learns $F_k(x)$. A batch uses the same key for every input. The protocol realizes the ideal functionality for (1) against malicious parties [5].

The half-malicious protocol uses the same encoding and input check but assumes a semi-honest server [5]. The semi-honest protocol instead hashes x with a random oracle $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^{n-1}$ and evaluates the weak PRF on $\mathrm{elt}(\mathcal{H}(x) \,\|\, 1)$. The shared-output variants reveal shares rather than $F_k(x)$; the shared-input shared-output variant also shares the inputs and key.

Our attacker chooses its inputs, follows the protocol and obtains both PRF outputs. Section 5 separates protocols that give the client these outputs from variants that reveal only shares.

2.4. Ternary Syndrome Decoding and ISD

We use the following decoding problem in our key-recovery attack.

Definition 1 (Ternary syndrome decoding [18])**.** Let $M \in \mathbb{F}_3^{\kappa \times N}$, $\sigma \in \mathbb{F}_3^\kappa$ and $\omega \in [0, N]$. The ternary syndrome decoding problem asks for a vector $u \in \mathbb{F}_3^N$ of weight

$|\operatorname{supp}(u)| = \omega$ with $Mu = \sigma$. We call the case $\omega = N$ *full weight*.

Here and in Section 4, N denotes the decoding length and κ the number of parity checks. If M has rank κ, it is a parity-check matrix of a code of length N and dimension $N - \kappa$. Bricout et al. [18] studied this problem at large weights, and Karpman and Lefevre [19] gave time–memory trade-offs for it. Our instances have full weight. Over $\mathbb{F}_3$, this case is the restricted syndrome decoding problem of Baldi et al. [20] with errors in $\{1, 2\}^N$, and Bitzer et al. [21] gave decoders for restricted errors based on representations. ISD [22] permutes the coordinates and uses row reduction to determine some coordinates from the others. It searches the remaining coordinates and checks the syndrome and weight. Section 4.4 gives the block decomposition and the search methods used in Table 1.

3. Related Inputs from Commuting Shifts

In this section, we characterize the inputs whose shifted encodings are again valid encodings by small linear systems over $\mathbb{F}_3$, and we find such inputs for the proposed parameters. The field identities below apply to E_{pad} and E_{impl} of Section 2.2. Section 3.2 treats E_{pad}, and Appendix B treats E_{impl}.

3.1. Multiplication by α^{-s}

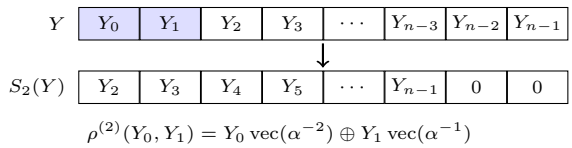
Since the key multiplication in (1) is a field multiplication, it commutes with every other field multiplication. If $\mathrm{elt}(E(x')) = c \cdot \mathrm{elt}(E(x))$ for $c \in \mathbb{F}_{2^n}$ and $W = \mathrm{vec}(k \cdot \mathrm{elt}(E(x)))$, then commutativity gives

$$\begin{aligned} F_k(x) &= B \cdot_3 W, \\ F_k(x') &= B \cdot_3 \mathrm{vec}\big(c \cdot \mathrm{elt}(W)\big). \end{aligned} \tag{2}$$

The relation (2) is useful only when $\mathrm{vec}(c \cdot \mathrm{elt}(W))$ is a simple function of W. We use $c = \alpha^{-s}$ for a small $s \geq 1$, which acts on the polynomial basis as a shift by s positions up to a correction that depends on the s lowest bits. For bits $Y_0, \ldots, Y_{s-1}$ we define this correction as

$$\rho^{(s)}(Y_0, \ldots, Y_{s-1}) := \bigoplus_{p=0}^{s-1} Y_p \cdot \mathrm{vec}(\alpha^{p-s}) \in \mathbb{F}_2^n. \tag{3}$$

Figure 2 illustrates the shift and its correction for $s = 2$.



$$\rho^{(2)}(Y_0, Y_1) = Y_0 \,\mathrm{vec}(\alpha^{-2}) \oplus Y_1 \,\mathrm{vec}(\alpha^{-1})$$
$$\mathrm{vec}(\alpha^{-2} \,\mathrm{elt}(Y)) = S_2(Y) \oplus \rho^{(2)}(Y_0, Y_1)$$

Figure 2. The near shift for $s = 2$, where $S_2(Y) = (Y_2, \ldots, Y_{n-1}, 0, 0)$. The two highlighted bits determine the correction $\rho^{(2)}$.

Lemma 1 (Near shift)**.** *Let $1 \leq s < n$, $Y \in \mathbb{F}_2^n$ and $\rho = \rho^{(s)}(Y_0, \ldots, Y_{s-1})$. Then, for all $q \in [0, n-1]$,*

$$\mathrm{vec}\big(\alpha^{-s} \cdot \mathrm{elt}(Y)\big)_q = Y_{q+s} \oplus \rho_q. \tag{4}$$

Moreover, $\operatorname{supp}(\mathrm{vec}(\alpha^{-j})) \subseteq [0, r-1] \cup [n-j, n-1]$ for every $j \geq 1$, so $\operatorname{supp}(\rho) \subseteq [0, r-1] \cup [n-s, n-1]$. If $r \leq n-3$ and $s = 2$, then $\rho_{n-2} = Y_0$ and $\rho_{n-1} = g_1 Y_0 \oplus Y_1$.

Proof. We have

$$\alpha^{-s} \cdot \mathrm{elt}(Y) = \sum_{q=s}^{n-1} Y_q \alpha^{q-s} + \sum_{p=0}^{s-1} Y_p \alpha^{p-s}.$$

The first sum has coordinate Y_{q+s} at position q. The second equals $\mathrm{elt}(\rho)$ by the $\mathbb{F}_2$-linearity of elt and $\mathrm{elt}(\mathrm{vec}(a)) = a$. This proves (4).

Since $f(\alpha) = 0$ and $g_0 = 1$,

$$\alpha^{-1} = \alpha^{n-1} + \sum_{i=1}^{r} g_i \, \alpha^{i-1}, \tag{5}$$

so the claim about the support of α^{-j} holds for $j = 1$. Suppose it holds for j. Multiplying by α^{-1} moves each positive-degree term down one position; a constant term contributes either 0 or α^{-1}. By (5), the new support lies in $[0, r-1] \cup [n-j-1, n-1]$. Induction proves the claim, and (3) gives the stated support of ρ.

For $s = 2$, multiplying (5) by α^{-1} gives

$$\alpha^{-2} = \alpha^{n-2} + g_1 \alpha^{-1} + \sum_{i=2}^{r} g_i \, \alpha^{i-2}.$$

When $r \leq n-3$, the last two coordinates of $\mathrm{vec}(\alpha^{-1})$ are $(0, 1)$ and those of $\mathrm{vec}(\alpha^{-2})$ are $(1, g_1)$. Equation (3) therefore gives $\rho_{n-2} = Y_0$ and $\rho_{n-1} = g_1 Y_0 \oplus Y_1$. □

For the modulus $f = X^{256} + X^{10} + X^5 + X^2 + 1$, the supports of $\mathrm{vec}(\alpha^{-1})$ and $\mathrm{vec}(\alpha^{-2})$ are $\{1, 4, 9, 255\}$ and $\{0, 3, 8, 254\}$. Thus $\rho^{(2)}$ can change only their union, six low positions and the last two.

The XOR with a known bit is affine over $\mathbb{F}_3$ on binary values. For bits $\beta, \gamma \in \{0, 1\}$ we have, over the integers and hence over $\mathbb{F}_3$,

$$\beta \oplus \gamma = \gamma + (-1)^{\gamma} \, \beta. \tag{6}$$

The next lemma applies the near shift to the second PRF output in (2). Guessing the s low bits of $W = \mathrm{vec}(k \cdot \mathrm{elt}(E(x)))$ fixes $\rho^{(s)}$, so (6) makes that output affine in the remaining bits. Section 4 uses these equations to recover W.

Lemma 2 (Affine after guessing)**.** *Let $B \in \mathbb{F}_3^{t \times n}$ be the public output matrix and $1 \leq s < n$. Choose $h \in \mathbb{F}_2^s$ and $W \in \mathbb{F}_2^n$ with $W_{[0,s-1]} = h$. Set $\rho^h = \rho^{(s)}(h)$ and $v = \mathrm{vec}(\alpha^{-s} \cdot \mathrm{elt}(W)) \in \mathbb{F}_2^n$, and read the bits of v in $\mathbb{F}_3$ for the product $B \cdot_3 v$. Let $B_{*,q} \in \mathbb{F}_3^t$ be column q of B. Then*

$$B \cdot_3 v = \sum_{q=0}^{n-1} \big(\rho_q^h + (-1)^{\rho_q^h} \, W_{q+s}\big) B_{*,q}.$$

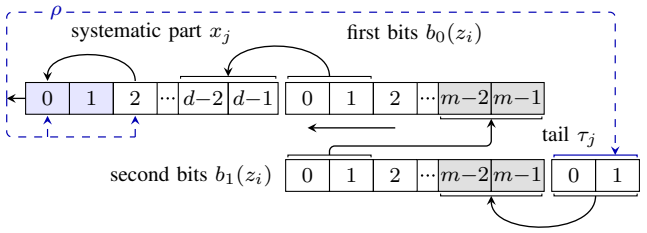


Figure 3. Multiplication by α^{-2} on $E(x)$ in the block layout with $\tau = (0,1)$ and $r \le d$. Boxes are positions, labeled with their index within each block, and the row of second bits continues the row of first bits, so column i of these rows is cell i. Every bit moves from position $q+2$ to q, as the solid arrows show. Hence cell $i \le m-3$ receives cell $i+2$, and the last two systematic positions receive $b_0(z_0)$ and $b_0(z_1)$. The shaded boundary cells $m-2$ and $m-1$ receive $(b_1(z_0), 0)$ and $(b_1(z_1), 1)$. The bits x_0 and x_1 leave the word and enter the correction ρ of (3), which sets the tail positions to $(x_0, g_1 x_0 \oplus x_1)$ and is added to a few positions in $[0, r-1]$.

For a fixed h, the right-hand side is an affine function of $(W_s, \dots, W_{n-1})$ over $\mathbb{F}_3$.

Proof. By the definition of matrix multiplication, $B \cdot_3 v = \sum_{q=0}^{n-1} v_q B_{*,q}$. Lemma 1 gives $v_q = W_{q+s} \oplus \rho_q^h$, which equals $\rho_q^h + (-1)^{\rho_q^h} W_{q+s}$ in $\mathbb{F}_3$ by (6). Substitution proves the identity. For fixed h, each ρ_q^h is constant, so the right-hand side is affine in the remaining bits of W. □

3.2. The Pair Condition as a Linear System

Here $E = E_{\mathrm{pad}}$ is the systematic-first completion with the bit order of the encoding equation and $\tau = (0,1)$. A word $\hat{y} \in \mathbb{F}_2^n$ is a *valid encoding* if $\hat{y} = E(\hat{y}_{[0,d-1]})$. For $s \ge 1$ we call $x \in \{0,1\}^d$ *s-good* if the word

$$\hat{y} = \mathrm{vec}\big(\alpha^{-s} \cdot \mathrm{elt}(E(x))\big)$$

is a valid encoding. Then $x' := \hat{y}_{[0,d-1]}$ satisfies

$$\mathrm{elt}(E(x)) = \alpha^s \cdot \mathrm{elt}(E(x')),$$

and we call (x, x') a *related pair* with shift s. The same derivation applies to the opposite bit order; Appendix B analyzes the encoder in the authors' public implementation.

We fix the block layout with $n = d+2m+2$, $m \ge 2$ and $\tau = (0,1)$, which places the specified appended 1 at the final coordinate after padding, and assume $r \le d$. Appendix C covers the tail $\tau = (1,0)$, other shifts and the one-bit tail of the specification [5]. Let $\bar{\rho} := \rho^{(2)}(0,1) = \mathrm{vec}(\alpha^{-1})$ be the correction (3) for the low bits $(x_0, x_1) = (0,1)$. For a candidate binary input x, let $a := z_0(x)$ and $c := z_1(x)$. We fix their values in each system below. For fixed $(a,c) \in \mathbb{F}_3^2$ we define the affine map $\pi_{a,c} \colon \mathbb{F}_3^d \to \mathbb{F}_3^d$ by

$$\pi_{a,c}(x)_j = \begin{cases} \bar{\rho}_j + (-1)^{\bar{\rho}_j} x_{j+2}, & j \in [0, d-3], \\ \bar{\rho}_j + (-1)^{\bar{\rho}_j} b_0(a), & j = d-2, \\ \bar{\rho}_j + (-1)^{\bar{\rho}_j} b_0(c), & j = d-1. \end{cases} \tag{7}$$

For $a \in \mathbb{F}_3$ and $c \in Z_*$, let $S_{a,c}$ be the following system of $m+4$ affine equations over $\mathbb{F}_3$ in the unknown $x \in \mathbb{F}_3^d$:

$$x_0 = 0, \qquad x_1 = 1, \tag{8a}$$
$$z_0(x) = a, \qquad z_1(x) = c, \tag{8b}$$
$$z_i\big(\pi_{a,c}(x)\big) = z_{i+2}(x) \qquad \text{for } i \in [0, m-3], \tag{8c}$$
$$\begin{aligned} z_{m-2}\big(\pi_{a,c}(x)\big) &= \mathrm{bin}^{-1}\big(b_1(a), 0\big), \\ z_{m-1}\big(\pi_{a,c}(x)\big) &= \mathrm{bin}^{-1}\big(b_1(c), 1\big). \end{aligned} \tag{8d}$$

In the shifted encoding, the last parity cell is $(b_1(c), 1)$. In this bit order, $b_1(0) = b_1(1) = 0$ and $b_1(2) = 1$, so the cell is valid exactly when $c \in Z_* = \{0,1\}$. The preceding cell $(b_1(a), 0)$ is always valid. Thus both right-hand sides of (8d) are defined. Figure 3 shows the bit layout of $E(x)$ and where the shift by two positions sends each cell. The branch (a,c) guesses the two parities whose first bits move into the systematic part, and by (8a) the two low bits that the reduction moves into the tail are fixed. The following theorem shows that these guesses are the only non-linear part of the pair condition.

Theorem 1 (Related inputs as binary solutions)**.** *Let $x \in \{0,1\}^d$ and $(a,c) = (z_0(x), z_1(x))$. Then x is 2-good if and only if $c \in Z_*$ and x is a solution of $S_{a,c}$. In this case $x' = \pi_{a,c}(x) \in \{0,1\}^d$ and $\mathrm{elt}(E(x)) = \alpha^2 \cdot \mathrm{elt}(E(x'))$. Consequently, the set $\mathcal{X}_2$ of 2-good inputs is the disjoint union*

$$\mathcal{X}_2 = \bigcup_{(a,c) \in \mathbb{F}_3 \times Z_*} \big\{ x \in \{0,1\}^d : x \text{ solves } S_{a,c} \big\} \tag{9}$$

over $|\mathbb{F}_3 \times Z_*| = 6$ branches.

Proof. Let $y = E(x)$ and $\hat{y} = \mathrm{vec}(\alpha^{-2} \cdot \mathrm{elt}(y))$. Since $n - 3 = d + 2m - 1 \ge d \ge r$, Lemma 1 applies with $s = 2$ and gives $\hat{y}_q = y_{q+2} \oplus \rho_q$ with $\rho = \rho^{(2)}(x_0, x_1)$, $\mathrm{supp}(\rho) \subseteq [0, d-1] \cup \{n-2, n-1\}$, $\rho_{n-2} = x_0$ and $\rho_{n-1} = g_1 x_0 \oplus x_1$. The word $\hat{y}$ is a valid encoding if and only if its tail equals τ and each cell i of $\hat{y}$ is a valid cell that encodes $z_i(\hat{y}_{[0,d-1]})$.

Tail. The positions $n-2$ and $n-1$ receive no shifted bit, so $(\hat{y}_{n-2}, \hat{y}_{n-1}) = (x_0, g_1 x_0 \oplus x_1)$, which equals $\tau = (0,1)$ if and only if (8a) holds. From now on we assume (8a), so that $\rho = \bar{\rho}$.

Systematic part. For $j \in [0, d-3]$, we have

$$\hat{y}_j = x_{j+2} \oplus \bar{\rho}_j = \bar{\rho}_j + (-1)^{\bar{\rho}_j} x_{j+2}$$

by (6). Positions $d-2$ and $d-1$ receive $y_d = b_0(a)$ and $y_{d+1} = b_0(c)$, so $\hat{y}_{d-2} = b_0(a) \oplus \bar{\rho}_{d-2}$ and $\hat{y}_{d-1} = b_0(c) \oplus \bar{\rho}_{d-1}$. Hence $\hat{y}_{[0,d-1]} = \pi_{a,c}(x)$, and this vector is binary because x is binary.

Aligned cells. The parity region $[d, n-3]$ lies outside $\mathrm{supp}(\rho)$, so for $i \in [0, m-3]$, cell i of $\hat{y}$ is $(y_{d+i+2}, y_{d+m+i+2}) = \mathrm{bin}(z_{i+2}(x))$, which is a valid cell. Since bin is injective, it encodes $z_i(\hat{y}_{[0,d-1]}) = z_i(\pi_{a,c}(x))$ if and only if (8c) holds for i.

Boundary cells. Cell $m-2$ of $\hat{y}$ is $(y_{d+m}, y_{d+2m}) = (b_1(a), \tau_0) = (b_1(a), 0)$, which is always a valid cell. Cell $m-1$ is $(y_{d+m+1}, y_{d+2m+1}) = (b_1(c), 1)$, which is valid

if and only if $c \in Z_*$. They encode $z_{m-2}(\pi_{a,c}(x))$ and $z_{m-1}(\pi_{a,c}(x))$ if and only if (8d) holds.

By the choice of (a, c), x satisfies (8b). Hence $\hat{y}$ is a valid encoding if and only if $c \in Z_*$ and x satisfies $S_{a,c}$, and then $\hat{y} = E(x')$ with $x' = \pi_{a,c}(x)$, which gives $\mathrm{elt}(E(x)) = \alpha^2 \cdot \mathrm{elt}(E(x'))$. Conversely, a binary solution of $S_{a,c}$ satisfies $(z_0(x), z_1(x)) = (a, c)$ by (8b), so the argument above applies to it with the same branch. Every binary x therefore solves at most one system $S_{a,c}$, and the union (9) is disjoint. □

3.3. Number of Related Inputs

Each 2-good input x gives one related pair (x, x'), so we count related pairs by their first input. The two fixed bits in (8a) leave 2^{d-2} binary assignments of x. To estimate how many solve $S_{a,c}$, we model the values of its other $m+2$ affine forms on these assignments as uniform in $\mathbb{F}_3^{m+2}$. Under this model, each assignment satisfies the equations with probability $3^{-(m+2)}$. Linearity of expectation gives

$$N_{\mathrm{pair}} = 2^{d-2} \cdot 3^{-(m+2)}. \tag{10}$$

This estimate is heuristic for a fixed public code matrix, since full rank over $\mathbb{F}_3$ alone does not establish uniformity on binary inputs.

For the parameters $(n, d, m) = (256, 128, 63)$ proposed by Aranha et al. [5], each branch has about

$$N_{\mathrm{pair}} = 2^{126} \cdot 3^{-65} = 2^{23.0}$$

binary solutions by (10).

3.4. Cost of Finding a Related Pair

At $(n, d, m) = (256, 128, 63)$, the reduced row echelon form of a full-rank branch system gives 67 pivot variables and 61 free variables. Two of the pivot rows are the equations (8a), so a binary solution is an assignment of the free variables for which the other 65 pivot values are binary. We visit these assignments in Gray-code order, so that each step adds or subtracts one column of the reduced system to the vector of pivot values. With each value of $\mathbb{F}_3$ stored in two bit planes, one step updates two 64-bit words that hold 64 of the 65 pivot values, and the last value is computed only when these 64 are binary. Under the model of Section 3.3, a step finds a solution with probability $(2/3)^{65} = 2^{-38.0}$. Thus one related pair takes about $2^{38.0}$ Gray-code steps on average under this model. Since a branch has $2^{23.0}$ solutions, independent random starting points per thread work as well as one systematic walk.

The search is a one-time offline computation and does not depend on the key.

3.5. Experimental Validation

3.5.1. Exhaustive Counts at Reduced Size. We checked the heuristic for $(d, m) = (24, 11)$, $n = 48$, $f = X^{48} + X^5 + X^3 + X^2 + 1$ and the tail $\tau = (1, 0)$ of Corollary 1, where its six branches predict $6 \cdot 2^{22} \cdot 3^{-13} = 15.8$ related inputs with shift 2. Testing all 2^{24} inputs for three random choices of G' gave 10, 17 and 16 related inputs, and their mean 14.3 is close to the predicted 15.8. A relation that maps each input to an independent random word would give $2^d \cdot 2^{d-n} = 1$ related input on average. The shift structure therefore increases the number of related inputs by a factor of about 14 at this size and about $2^{25.6}$ at $n = 256$.

3.5.2. Pairs in the Systematic-First Layout. The specification does not fix G', so we sampled G' uniformly from $\mathbb{F}_3^{128\times 63}$. All six systems $S_{a,c}$ had rank $67 = m + 4$ for E_{pad} and for the same systematic-first layout with b_0 and b_1 interchanged. We found a related pair for each encoding. For the alternative completion of E_{spec} with $\tau = (1, 0)$, a search with 10 threads on an Intel Core i7-12700K found a related pair in 146 seconds.

Table 3 in Appendix D records these three pairs, at $(n, d, m) = (256, 128, 63)$ and with the modulus of Section 2.2. For each pair, we verified in $\mathbb{F}_{2^{256}}$ that both encodings are valid and that the first is α^2 times the second.

3.5.3. Pair for E_{impl} at the Proposed Size. For E_{impl} and the fixed code matrix generated by the authors' public implementation, we found a pair by solving the rank-69 system of Theorem 2. Exact field arithmetic verified the relation, and both encoded inputs passed the implementation's gadget and final-bit checks. Appendix B gives the system and its proof.

4. Key Recovery

In this section, we turn the outputs on a related pair (x, x') with shift 2 into linear equations over $\mathbb{F}_3$ in the bits of $W = \mathrm{vec}(k \cdot \mathrm{elt}(E(x)))$ and estimate the cost of solving them.

4.1. Equations in the Unknown W

Let $o = F_k(x)$ and $o' = F_k(x')$. By the definition of W,

$$k \cdot \mathrm{elt}(E(x)) = \mathrm{elt}(W).$$

The related pair satisfies

$$\mathrm{elt}(E(x')) = \alpha^{-2} \cdot \mathrm{elt}(E(x)).$$

Since field multiplication commutes,

$$k \cdot \mathrm{elt}(E(x')) = \alpha^{-2} \cdot \mathrm{elt}(W).$$

Substitution into (1) gives

$$o = B \cdot_3 W,$$
$$o' = B \cdot_3 \mathrm{vec}(\alpha^{-2} \cdot \mathrm{elt}(W)).$$

The first identity gives one equation per row of B, such as $o_0 = B_{0,0}W_0 + \cdots + B_{0,n-1}W_{n-1}$, and hence 82 equations at $t = 82$.

For a guess $h = (W_0, W_1) \in \{0,1\}^2$ and $\rho^h = \rho^{(2)}(h)$, Lemma 2 gives another t equations from the second output. Together they are

$$\begin{aligned} o_i &= \textstyle\sum_{q=0}^{n-1} B_{i,q}\, W_q, \\ o'_i &= \textstyle\sum_{q=0}^{n-1} B_{i,q}\big(\rho_q^h + (-1)^{\rho_q^h}\, W_{q+2}\big), \end{aligned} \tag{11}$$

over $\mathbb{F}_3$ for $i \in [0, t-1]$, where W_0 and W_1 take the guessed values.

Let $N = n-2$ and $w = (W_2, \ldots, W_{n-1})^\top \in \mathbb{F}_2^N$. Once h is guessed, ρ^h is known. In the first t equations, we move $B_{i,0}h_0 + B_{i,1}h_1$ to the right-hand side, and in the other t equations, we move $\sum_q B_{i,q}\rho_q^h$ there. The coefficients of w_j in these two groups are $B_{i,j+2}$ and $(-1)^{\rho_j^h} B_{i,j}$, respectively. They form $M_h \in \mathbb{F}_3^{2t\times N}$, while the right-hand sides form $c_h \in \mathbb{F}_3^{2t}$. Thus $M_h w = c_h$ for the correct guess. At $(n,t) = (256, 82)$, this system has 164 equations in 254 unknown bits. The bit 1 in the tail gives $\mathrm{elt}(E(x)) \neq 0$, so the key follows as

$$k = \mathrm{elt}(W) \cdot \mathrm{elt}(E(x))^{-1}.$$

Conversely, let w be a binary solution for a guess h and let $\widehat{W} = (h, w) \in \mathbb{F}_2^n$. The first t equations of (11), and Lemma 2 applied with $(\widehat{W}_0, \widehat{W}_1) = h$ to the other t equations, give

$$\begin{aligned} B \cdot_3 \widehat{W} &= o, \\ B \cdot_3 \mathrm{vec}(\alpha^{-2} \cdot \mathrm{elt}(\widehat{W})) &= o'. \end{aligned}$$

The candidate key is

$$\widehat{k} = \mathrm{elt}(\widehat{W}) \cdot \mathrm{elt}(E(x))^{-1}.$$

By (2), it satisfies $F_{\widehat{k}}(x) = o$ and $F_{\widehat{k}}(x') = o'$. The derivation of (11) applies to every key with these two outputs, and both directions hold for any right-hand sides (o, o'), not only for the outputs of F_k. The binary solutions of the four systems therefore correspond one-to-one to the keys that agree with F_k on x and x'. Algorithm 1 returns these keys from the two outputs.

Algorithm 1 Two-query key recovery on the strong PRF (1)

Require: oracle access to F_k and a related pair (x, x') with shift 2 (Section 3.2)
Ensure: the set $\mathcal{K}$ of keys that agree with both queried outputs

```
o ← F_k(x), o' ← F_k(x')                          ▷ two queries
K ← ∅
for h ∈ {0,1}^2 do
    build M_h w = c_h from (11)
    for each binary solution w of M_h w = c_h do
        Ŵ ← (h, w), k̂ ← elt(Ŵ) · elt(E(x))^{-1}
        K ← K ∪ {k̂}
    end for
end for
return K
```

4.2. Uniqueness of the Solution

We assume that M_h has rank $2t$. The matrix M_h depends only on B, f and h, since the key and the pair enter only c_h. We also assume that $M_h w$ behaves like a uniform vector of $\mathbb{F}_3^{2t}$ for every binary w, except the solution when h is the correct guess.

Proposition 1 (False solutions)**.** *Under the uniformity assumption above, for a fixed guess h, the expected number of binary solutions of $M_h w = c_h$ other than the correct one is about $2^{n-2} \cdot 3^{-2t}$.*

Proof. For the correct guess, $2^{n-2} - 1$ binary vectors w differ from the correct solution, and for a wrong guess all 2^{n-2} binary vectors are wrong candidates. By the second assumption above, each wrong candidate satisfies the $2t$ equations with probability 3^{-2t}. The expected number of false solutions is therefore $(2^{n-2} - 1) \cdot 3^{-2t} \approx 2^{n-2} \cdot 3^{-2t}$ for the correct guess and $2^{n-2} \cdot 3^{-2t}$ for a wrong guess. □

The set $\mathcal{K}$ of Algorithm 1 always contains k, and its other elements come from false solutions. For $(n,t) = (256, 82)$, their expected number is $2^{254} \cdot 3^{-164} = 2^{-5.9}$ per guess. Over the four guesses for $(256, 82)$, the expected number of false solutions is $4 \cdot 2^{-5.9} = 2^{-3.9}$. Under the heuristic of Proposition 1, Markov's inequality shows that the correct key is the only candidate with probability at least $1 - 2^{-3.9} \approx 0.93$.

4.3. A Two-Query Distinguisher

Without the uniformity assumption, the same value $2^{-3.9}$ bounds the probability that a random function passes the following test on two outputs.

Proposition 2 (Two-query distinguisher)**.** *Let (x, x') be a related pair with shift 2. For a function $F\colon \mathbb{F}_2^d \to \mathbb{F}_3^t$, the test accepts F if and only if one of the four systems $M_h w = c_h$ built from $o = F(x)$ and $o' = F(x')$ by* (11) *has a binary solution. It accepts F_k for every key $k \in \mathbb{F}_{2^n}$, and it accepts a uniformly random function with probability at most $2^n \cdot 3^{-2t}$, which is below $2^{-3.9}$ for $(n,t) = (256, 82)$.*

Proof. Let $\mathcal{A} \subseteq \mathbb{F}_3^{2t}$ be the set of output pairs (o, o') that the test accepts. As shown in Section 4.1, the binary solutions of the four systems correspond one-to-one to the keys $\widehat{k}$ with $F_{\widehat{k}}(x) = o$ and $F_{\widehat{k}}(x') = o'$. Hence the test accepts F_k for every k, and $\mathcal{A}$ is the set of pairs $(F_{\widehat{k}}(x), F_{\widehat{k}}(x'))$ for $\widehat{k} \in \mathbb{F}_{2^n}$, so $|\mathcal{A}| \leq 2^n$. Since $\mathrm{elt}(E(x)) = \alpha^2 \cdot \mathrm{elt}(E(x'))$ with $\mathrm{elt}(E(x)) \neq 0$ and $\alpha^2 \neq 1$, the inputs x and x' differ. For a uniformly random function, (o, o') is therefore uniform over $\mathbb{F}_3^{2t}$, and the test accepts with probability $|\mathcal{A}| \cdot 3^{-2t} \leq 2^n \cdot 3^{-2t}$. For $(n,t) = (256, 82)$, this bound is $2^{256} \cdot 3^{-164} = 2^{-3.93}$. □

The test accepts F_k with probability 1 and a random function with probability at most $2^{-3.9}$. Two queries therefore distinguish F_k from a random function with an advantage of at least $1 - 2^{-3.9} \approx 0.93$. We decide the test with

the same search as the key recovery. When all four M_h have rank 164, testing all $4 \cdot 2^{90} = 2^{92}$ binary assignments of the free unknowns decides it exactly. This is the first row of Table 1, which omits the polynomial cost of each assignment.

4.4. Full-Weight Decoding Methods and Costs

For $w \in \{0,1\}^N$, set $u = 2w + 2 \cdot \mathbf{1} \in \mathbb{F}_3^N$. Each coordinate u_i is 2 when $w_i = 0$ and 1 when $w_i = 1$, so u has weight N. Conversely, $w = 2(u + \mathbf{1})$, and the system $M_h w = c_h$ holds if and only if

$$M_h u = 2c_h - M_h \mathbf{1}.$$

The binary solutions therefore correspond exactly to the full-weight solutions of this ternary syndrome decoding instance (Definition 1). Our instances have length $N = 254$ and dimension $N - 2t = 90$, and by Proposition 1 the four instances have an expected total of $2^{-3.9}$ false solutions. The methods in Table 1 use this same system with different window searches.

For a fixed guess h, set $\sigma := 2c_h - M_h \mathbf{1}$ and $\kappa := 2t$. Fix $0 \le \ell \le \kappa$ to leave ℓ equations for window matching. The first $\kappa - \ell$ columns of M_h may be dependent, and row operations cannot make them an identity block.

Since M_h has rank κ, we choose an independent set of $\kappa - \ell$ columns and move it to the front with a permutation matrix $\Pi \in \mathbb{F}_3^{N \times N}$. If the leading columns are already independent, $\Pi = I_N$ suffices. Permuting the coordinates of u accordingly preserves its weight. Row reduction using these columns as pivots gives an invertible $T \in \mathbb{F}_3^{\kappa \times \kappa}$ with

$$T M_h \Pi = \begin{pmatrix} I_{\kappa-\ell} & M_1 \\ 0 & M_2 \end{pmatrix}, \qquad T\sigma = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}.$$

The first $\kappa - \ell$ coordinates are pivots, and the remaining $n_{\mathrm{w}} = N - \kappa + \ell$ coordinates form the window. The identity block lets the first $\kappa - \ell$ rows determine the pivots once the window is fixed, and the zero block leaves the remaining ℓ equations in the window variables alone. We split the permuted solution as $\Pi^{-1} u = (u_{\mathrm{piv}}, u_{\mathrm{win}})^\top$, where $u_{\mathrm{piv}} \in \mathbb{F}_3^{\kappa-\ell}$ and $u_{\mathrm{win}} \in \mathbb{F}_3^{n_{\mathrm{w}}}$. The equation $M_h u = \sigma$ is equivalent to

$$\begin{pmatrix} I_{\kappa-\ell} & M_1 \\ 0 & M_2 \end{pmatrix} \begin{pmatrix} u_{\mathrm{piv}} \\ u_{\mathrm{win}} \end{pmatrix} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}.$$

The lower ℓ rows test $M_2 u_{\mathrm{win}} = \sigma_2$, and the upper rows fix $u_{\mathrm{piv}} = \sigma_1 - M_1 u_{\mathrm{win}}$. At full weight, all window entries and the computed pivots must be nonzero. For the proposed parameters, $n_{\mathrm{w}} = 90 + \ell$. Writing the binary preimage as $w_{\mathrm{win}} \in \{0,1\}^{n_{\mathrm{w}}}$, we have $u_{\mathrm{win}} = 2w_{\mathrm{win}} + 2 \cdot \mathbf{1}_{n_{\mathrm{w}}}$. Thus the binary window equation has the known right-hand side

$$M_2 w_{\mathrm{win}} = 2\sigma_2 - M_2 \mathbf{1}_{n_{\mathrm{w}}}.$$

The four searches differ in how they generate and test window candidates. Time counts generated list entries and checked matches, or assignments for free-bit search; we call these operations. Memory counts the largest stored list, up to the constant number of simultaneous lists. We omit polynomial costs such as the Gaussian elimination of each trial. The four guesses of h give a factor 4. Figure 4 compares the three list searches.

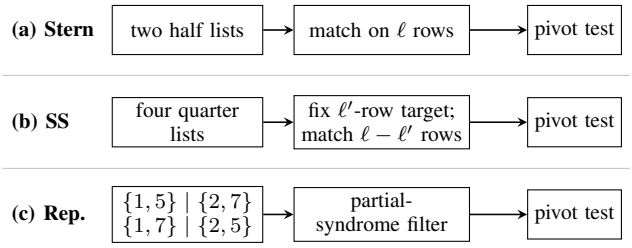


Figure 4. Window searches of (a) Stern's algorithm, (b) Stern with the Schroeppel–Shamir technique (SS) and (c) the representation technique (Rep.). Stern matches two half lists on ℓ rows. Schroeppel–Shamir forms two half lists from four quarter lists for each target on ℓ' rows, then matches on the remaining rows. In (c), $\{1,5\} \mid \{2,7\}$ and $\{1,7\} \mid \{2,5\}$ are two representations of the window vector with support $S = \{1,2,5,7\}$, and the decoder finds this vector if one of them passes every partial-syndrome filter. All three methods test the resulting window against the pivot equations.

4.4.1. Free-Bit Search. For $\ell = 0$, complete elimination gives

$$T M_h \Pi = (I_{164}\ A), \qquad A \in \mathbb{F}_3^{164 \times 90}.$$

The window has 90 free coordinates in the binary w-system. We test their 2^{90} assignments and keep those whose 164 pivots are binary. Across four guesses, this takes 2^{92} assignments and no exponential-size list.

4.4.2. Stern's Algorithm. For $\ell = 41$, with T and Π chosen again, the block form is

$$T M_h \Pi = \begin{pmatrix} I_{123} & M_1 \\ 0 & M_2 \end{pmatrix}.$$

Here $M_1 \in \mathbb{F}_3^{123 \times 131}$ and $M_2 \in \mathbb{F}_3^{41 \times 131}$. The 41 lower rows test a 131-coordinate window, and the 123 pivots must be nonzero in the u-system. Stern's algorithm [23], in the partial-Gaussian-elimination form that Bricout et al. [18] use for ternary codes, forms two lists of binary window assignments, of sizes $L_1 = 2^{\lfloor n_{\mathrm{w}}/2 \rfloor}$ and $L_2 = 2^{\lceil n_{\mathrm{w}}/2 \rceil}$. Under the uniformity assumption of Section 4.2, a left-right pair matches on the ℓ rows with probability $3^{-\ell}$. Let N_{list} count the assignments generated for both halves, and let N_{match} be the expected number of matching pairs:

$$N_{\mathrm{list}} := L_1 + L_2 \approx 2^{1 + n_{\mathrm{w}}/2},$$
$$N_{\mathrm{match}} := L_1 L_2\, 3^{-\ell} = 2^{n_{\mathrm{w}}} 3^{-\ell}.$$

The count N_{list} includes all entries generated in both lists. To bound peak storage, we keep the smaller list and stream the larger one. Including the four guesses, the time and memory estimates are

$$C_{\mathrm{time}} := 4(N_{\mathrm{list}} + N_{\mathrm{match}}),$$
$$C_{\mathrm{memory}} := L_1 = 2^{\lfloor n_{\mathrm{w}}/2 \rfloor}.$$

Time is measured in operations and memory in stored list elements. At $\ell = 41$, the two lists have 2^{65} and 2^{66} entries, so the streaming strategy stores 2^{65} entries. Since $n_{\mathrm{w}} = 90 + \ell$, the list-generation cost grows with ℓ while the expected matching cost falls. Evaluating C_{time} for all integers $0 \le \ell \le 164$ gives its minimum at $\ell = 41$, with $2^{69.3}$ time and 2^{65} memory.

4.4.3. Stern with Schroeppel–Shamir. The Schroeppel–Shamir technique [24] splits the window into four quarters. For one value of the left half-syndrome on $\ell' \le \ell$ rows at a time, it builds both half lists from the quarter lists. Its modeled memory is

$$C_{\mathrm{memory}} \approx \max\{2^{n_{\mathrm{w}}/4},\, 2^{n_{\mathrm{w}}/2} 3^{-\ell'}\}$$

list elements. The time model counts repeated quarter- and half-list generation and the final matches. To give a lower-memory option, we minimize the modeled time over integers $0 \le \ell \le \kappa$ and $0 \le \ell' \le \ell$, subject to $C_{\mathrm{memory}} \le 2^{34}$. This selects $(\ell, \ell') = (40, 20)$ and gives $2^{69.9}$ time and $2^{33.3}$ memory.

4.4.4. Representation Technique. Figure 4(c) shows two splits of the support of one binary window vector into two halves. The representation technique [16], [17] recursively splits weight-ν window vectors and matches their partial ternary syndromes. At each split, a partial-syndrome target is chosen uniformly for one child and the complementary target for the other. The filters keep only pairs that meet these targets on the selected rows and the parent target on the remaining rows. A trial succeeds only when the true binary window has weight ν and at least one of its representations passes every filter.

For $R \in \{2, 3\}$ levels, we evaluated every integer $0 \le \ell \le \kappa$, $0 \le \nu \le n_{\mathrm{w}}$ and filter lengths $0 \le r_R \le \cdots \le r_1 \le \ell$ in the balanced-split cost model below. The minimum for two levels occurs at $(\ell, \nu; r_1, r_2) = (71, 70; 43, 20)$, with $2^{66.1}$ time and $2^{49.3}$ memory. For three levels, it occurs at $(70, 68; 42, 19, 9)$, with $2^{65.9}$ time and $2^{48.5}$ memory. Table 1 reports these model minima.

At the two-level minimum, the 161 window positions split into two halves, and the target weight splits as $70 = 35 + 35$, then $35 = 17 + 18$. At three levels, the window has 160 positions and the weights split as $68 = 34 + 34$, $34 = 17 + 17$, and $17 = 8 + 9$. Before building either tree, we mix the ℓ matching rows by a random invertible matrix over $\mathbb{F}_3$, because row reduction otherwise leaves identity columns that correlate the partial syndromes.

Let θ_v be the probability that a fixed weight-v vector satisfying its assigned partial-syndrome target is generated by its subtree. At a leaf, θ_v is the fraction covered when the half-window split weight lies within two standard deviations of its mean. If a parent of weight v has children of weights a and b, a Poisson approximation gives

$$\theta_v = 1 - \exp\left(-\binom{v}{a} 3^{-r_{\mathrm{child}}} \theta_a \theta_b\right),$$

TABLE 1. COST OF RECOVERING THE KEY FOR $(n, t) = (256, 82)$ WITH THE FACTOR 4 FOR THE GUESS OF (W_0, W_1). TIME COUNTS GENERATED LIST ENTRIES AND CHECKED MATCHES, OR ASSIGNMENTS FOR FREE-BIT SEARCH; MEMORY COUNTS THE LARGEST STORED LIST UP TO A CONSTANT FACTOR. SS IS THE SCHROEPPEL–SHAMIR TECHNIQUE, AND REP. IS THE REPRESENTATION TECHNIQUE. STERN STORES THE SMALLER HALF LIST AND STREAMS THE LARGER ONE, AND EACH LEAF OF REP. DOES SO FOR ONE HALF-WINDOW SPLIT WEIGHT AT A TIME. HERE ℓ IS THE NUMBER OF MATCHING ROWS, AND ℓ' IS THE NUMBER OF ROWS ON WHICH SS FIXES THE LEFT HALF-SYNDROME. FOR REP., THE TUPLE LISTS ℓ, THE WINDOW WEIGHT ν, AND THE FILTER LENGTHS AT SUCCESSIVE LEVELS; THESE ARE THE MINIMA WITHIN OUR BALANCED-SPLIT MODEL. THE SS TIME COUNTS REGENERATION OF ALL FOUR QUARTER LISTS IN EACH LOOP.

Method	Parameters	Time	Memory
Free-bit search	90 free bits	2^{92}	negligible
Stern	$\ell = 41$	$2^{69.3}$	2^{65}
Stern with SS	$\ell = 40$, $\ell' = 20$	$2^{69.9}$	$2^{33.3}$
Rep., 2 levels	$(71, 70; 43, 20)$	$2^{66.1}$	$2^{49.3}$
Rep., 3 levels	$(70, 68; 42, 19, 9)$	$2^{65.9}$	$2^{48.5}$

where r_{child} is the filter length for the child lists. A true window of weight ν is thus generated with probability $p_{\mathrm{surv}} = \theta_\nu$. Each trial negates the columns of M_h in a uniformly random set of positions. The resulting system has the solution u with these coordinates negated, whose binary preimage is w with the same bits flipped. The binary window is therefore uniform for every key, and the selected weight has probability

$$p_\nu = \binom{n_{\mathrm{w}}}{\nu} 2^{-n_{\mathrm{w}}}.$$

If N_{trial} counts generated list entries and checked matches in one trial, and the trials alternate among the four guesses until a trial returns a binary solution, the expected time is

$$C_{\mathrm{time}} = 4\, \frac{N_{\mathrm{trial}}}{p_\nu p_{\mathrm{surv}}}.$$

For the two-level minimum, the model gives $N_{\mathrm{trial}} = 2^{55.474}$, $p_\nu = 2^{-5.963}$, and $p_{\mathrm{surv}} = 2^{-2.622}$. Hence the time is $2^{66.06}$ and the largest stored list has $2^{49.27}$ elements. At three levels, the corresponding values are $2^{53.957}$, $2^{-6.579}$, and $2^{-3.341}$, giving $2^{65.88}$ time and $2^{48.47}$ stored list elements.

4.4.5. Cost Comparison and Scaling. Table 1 gives the lowest modeled time, $2^{65.9}$ with $2^{48.5}$ stored list elements, for the three-level representation technique. The $2^{67.3}$ time and $2^{51.8}$ memory in the abstract instead use the two-level setting $(\ell, \nu; r_1, r_2) = (79, 84; 51, 24)$, whose weight $\nu = 84 = \lfloor n_{\mathrm{w}}/2 \rfloor$ is the central weight of its 169-position window. The table minimizes over the window weight and filter lengths. Stern with the Schroeppel–Shamir technique needs $2^{69.9}$ operations and only $2^{33.3}$ stored list elements. These costs are far below the claimed 128-bit security [5]. Even the first row, which tests all 2^{90} binary values of the free unknowns for each guess and needs no cost model, stays far below 2^{128}. Stern's algorithm, with or without the Schroeppel–Shamir technique, also finds every binary

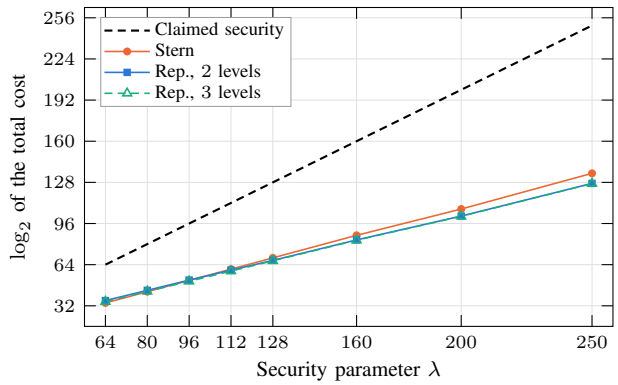


Figure 5. Extrapolated key-recovery cost for $n = 2\lambda$ and our rule $t = \lceil \lambda / \log_2 3 \rceil + 1$, which gives the proposed $(n, t) = (256, 82)$ at $\lambda = 128$. The cost counts generated list entries and checked matches, including the four guesses of (W_0, W_1). The Rep. curves fix the window weight at $\lfloor n_w/2 \rfloor$ or $\lceil n_w/2 \rceil$, and the $\lambda = 128$ two-level point is the $2^{67.3}$ estimate in the abstract. Table 1 reports separately optimized costs. The dashed line marks the claimed security of λ bits.

solution, so it decides the test of Proposition 2 at the costs in Table 1. The representation times are expected costs for every key, and they already include the window-weight and representation-survival probabilities.

A Gröbner basis approach is not competitive. The binary constraints on the 164 pivots and the 90 free unknowns give 254 quadratic equations in the free unknowns. Under the semi-regularity assumption of Bardet et al. [25], their degree of regularity is 9, with about $2^{39.5}$ multilinear monomials up to this degree, so the linear algebra alone needs at least 2^{79} arithmetic operations.

Aranha et al. [5] state the general choice $(n, t) = (2\lambda, \lambda/\log_2 3)$ and give concrete values only for $\lambda = 128$, namely $(256, 82)$. Figure 5 explores our extrapolation of that choice, using $t = \lceil \lambda / \log_2 3 \rceil + 1$ to recover $t = 82$ at $\lambda = 128$. Under this extrapolation, the central-weight two-level representation estimate remains below 2^{128} at $(n, t) = (500, 159)$, where it is $2^{127.2}$. Minimizing over the window weight and filter lengths as in Table 1 gives $2^{125.1}$ there.

4.5. Experimental Validation

4.5.1. Rank and Correctness of the Linear System. For 17 randomly generated matrices B and for the public output matrix of the implementation, all four matrices M_h had rank $164 = 2t$. The true W solved its system for the first two pairs of Table 3, for E_{pad} and for E_{pad} with interchanged bit maps, with three random keys each, and for the E_{impl} pair of Appendix B. These checks confirm the linear system used in the attack. We test the cost model of Section 4.4 below.

4.5.2. Full Key Recovery at Reduced Size. We chose $(n, d, m, t) = (128, 64, 31, 42)$ as a reduced test setting in which full key recovery is feasible. Its (n, t) is the $\lambda = 64$ point of Figure 5. The target proposes no degree-128 field modulus, so we selected the irreducible polynomial $f = X^{128} + X^7 + X^2 + X + 1$ for these tests. One run used a completion of E_{spec} with $\tau = (1, 0)$. The other used the bit maps, parity/input/parity layout, $(1, 1)$ tail and coefficient reversal of E_{impl}, with code and output matrices generated at the reduced dimensions. With $(\ell, \ell') = (19, 8)$, Stern with the Schroeppel–Shamir technique solved the four systems of 84 equations in 126 unknown bits in each run.

For this low-memory test point, the decoder forms four quarter lists of lengths $(16, 15, 15, 15)$. It indexes the first and third lists once, then queries them with the second and fourth lists for each of the 3^8 partial syndromes. Counting generated list entries, repeated lookups and expected matches over the four guesses gives

$$C_{\mathrm{model}} = 4\big[(2^{16} + 3 \cdot 2^{15}) + 3^8(2^{15} + 2^{15}) + 2^{31} + 2^{30} + 2^{61}3^{-19}\big] = 2^{34.39}.$$

Table 1 instead counts the regeneration of all four quarter lists in each loop, which gives $2^{34.49}$ at this size. The model predicts $2^{30.886}$ matching pairs per guess and a largest mean half list of $2^{18.32}$ entries. The observed matching-pair counts agree with the prediction to three decimal places in the exponent. The decoder's count of lookups, generated half-list entries and checked matches was $2^{34.39}$ over the four guesses of each run. Each run left only the correct key after the two PRF outputs.

For completions of E_{spec}, we also ran 20 instances at $n = 48$, three at $n = 64$, and three at $n = 96$; all gave only the correct key.

4.5.3. Stern and Schroeppel–Shamir Matching Counts. At $(n, t) = (96, 32)$, the model of Stern with $\ell = 13$ predicts $2^{43} \cdot 3^{-13} = 5.517 \cdot 10^6$ matching pairs of half-list entries per guess. In the first guess of three runs, the observed counts ranged from $5.517 \cdot 10^6$ to $5.518 \cdot 10^6$. For Stern with Schroeppel–Shamir and $(\ell, \ell') = (12, 4)$, the prediction is $2^{42} \cdot 3^{-12} = 8.276 \cdot 10^6$, and the four guesses of one run gave $8.272 \cdot 10^6$ to $8.279 \cdot 10^6$. A matching pair agrees on the ℓ ternary equations and is then tested against the pivot equations.

4.5.4. Representation Decoder List Sizes. We ran four experiments at $n = 96$, each with a separately generated key, related pair and public matrices. In each experiment, we built the key-recovery linear system $M_h w = c_h$ for the correct guess $h = (W_0, W_1)$ and ran 50 trials of the two-level representation decoder with $(\ell, \nu; r_1, r_2) = (19, 24; 13, 6)$. Each trial draws a fresh column permutation, mixing matrix and partial-syndrome targets, and is one iteration of the decoder in Section 4.4.4. It returns the solution with probability $p_\nu p_{\mathrm{surv}}$, and the attack repeats trials until one does. The 200 trials therefore measure these per-trial probabilities, which determine C_{time}, rather than the success rate of complete attacks. These runs used the central weight $\nu = 24$ of the 49-position window and no column negation. Without negation, the number of ones of the solution in the window is hypergeometric for its total number of ones, so the selected weight has probability 0.119 on average

over the four solutions, instead of $p_\nu = 0.112$. Together with $p_{\mathrm{surv}} = 0.583$, the model expects about 14 successful trials out of 200. The window had the selected weight in 21 trials, against about 24 expected. The decoder returned the true solution in 16 of these 21 trials, against about 12 expected from p_{surv}, and in no other trial. Each of the four systems was solved in at least two of its 50 trials, and no trial returned a false solution.

Across these trials, the mean top-list size was $2^{14.94}$ entries, against $2^{14.95}$ predicted. Divided by the expected number $\binom{n_{\mathrm{w}}}{\nu}3^{-\ell}$ of weight-ν window vectors that meet the ℓ rows, this size gives a retained fraction of 0.578, compared with 0.579 on random systems of the same size and 0.583 in the model. The other list sizes differed from the model by at most 0.02 bits. The four tested key-recovery systems gave list sizes close to those of random systems at this size. In a separate three-level experiment with $(\ell, \nu; r_1, r_2, r_3) = (19, 24; 12, 6, 3)$ on random systems, the mean top-list size was $2^{14.10}$ rather than the modeled $2^{14.23}$, a 0.13-bit difference.

5. Impact and Countermeasures

In this section, we discuss which protocols and constructions the attack affects and how the strong PRF (1) can be changed to resist it. Unless stated otherwise, all costs refer to $(n, t, d, m) = (256, 82, 128, 63)$.

5.1. Affected Protocols and Constructions

The maliciously secure and half-malicious protocols of Aranha et al. [5] let a client submit a batch of chosen inputs and receive their F_k outputs under one server key. A client can submit a related pair (x, x') in one batch. For E_{impl}, we verified a pair that passes the implementation's input checks (Appendix B). The pair depends only on public parameters and can be reused against any server key.

The protocol proofs of Aranha et al. [5] concern the evaluation of F_k; our attack shows that this function does not reach the claimed 128-bit security as a strong PRF. After two valid queries, the client can recover k and evaluate F_k offline on inputs that it did not submit. In private set intersection based on an OPRF [1], for example, the server sends the PRF values of its elements, and a client that knows k can test any candidate element against them. The variants with secret-shared outputs [5], including the shared-input shared-output protocol, are affected only in applications in which one party chooses two related inputs and later learns both outputs. The semi-honest protocol [5] encodes x as $\mathrm{elt}(\mathcal{H}(x) \,\|\, 1)$ with a random oracle $\mathcal{H}$, so a client cannot choose a related pair, and the attack does not apply.

The partially oblivious PRF of Albrecht et al. [9] uses an encoded-input PRF of the same form. Its key is a uniform matrix over $\mathbb{F}_2$, from which a random oracle derives the key of each tag [9]. Such a key has no public structure that commutes with the key multiplication, so the relation (2) has no counterpart, and our attack does not apply to the partially oblivious PRF as defined by Albrecht et al. [9].

Van Baarsen et al. [14] use a variant of (1) with a non-systematic code $G \in \mathbb{F}_3^{r' \times d}$ in a maliciously secure shuffled distributed OPRF. The encoding applies bin to all r' coordinates of $G \cdot_3 x$ and appends the bit 1, so $n = 2r' + 1$, which is 385 for $d = \lambda = 128$ and $r' = 3\lambda/2$ [14]. The key multiplication still commutes with α^{-s}, but the encoding no longer contains the input, so x' becomes a second binary unknown that enters only through $G \cdot_3 x'$. For $s = 2$, the pair condition is then a system of about $r' + 2$ affine equations over $\mathbb{F}_3$ in the $2d$ binary unknowns (x, x'). Each of the r' cells of the shifted word gives one equation, and the two guessed parities give two more. Under the model of Section 3.3, a branch therefore has about

$$2^{2d} \cdot 3^{-(r'+2)} \tag{12}$$

related inputs, which is $2^{-51.5}$ for $d = 128$ and $r' = 192$.

We tested (12) with $d = 16$, $s = 2$, a sparse modulus, the block layout, the tail $(1, 0)$ and three random codes for each $r' \in \{16, 17, 18, 19, 20, 22, 24\}$. The tail fixes the first bits of both guessed parities, so only one or two branches remain. A test of all 2^{16} inputs found on average 15.7, 3.0, 0.33, 0.33, 0.33, 0 and 0 related inputs over these branches. Summed over the same branches and multiplied by $2/3$ for each of the at most three parity cells that $\rho^{(2)}$ flips (Section 5.2.1), the values of (12) are 14.8, 4.9, 0.55, 0.24, 0.18, 0.015 and 0.001. We also tested all inputs for the encoding of [14] with $d = \lambda \in \{16, 20\}$, $r' = 3\lambda/2$ and three random codes each. No related input appeared for any shift $s \in [1, n-1]$, while systematic codes with the same d and code length gave 16 to 49 related inputs over all shifts.

5.2. Countermeasures

The attack needs a multiplier c with two properties. (1) Many pairs of inputs satisfy

$$\mathrm{elt}(E(x')) = c \cdot \mathrm{elt}(E(x))$$

and are the binary solutions of linear systems. (2) The vector $\mathrm{vec}(c \cdot \mathrm{elt}(W))$ is affine over $\mathbb{F}_3$ in W after a guess of a few bits. A dense modulus, a non-systematic code and input mixing target property (1), and output mixing targets property (2). Hashing the input removes the choice of the encoding, while larger parameters raise the attack cost without removing either property. We examine each change below; protocol costs are estimates per evaluation of the maliciously secure protocol [5].

5.2.1. Dense Modulus. A dense irreducible polynomial f makes the shift correction $\rho^{(s)}$ flip parity cells. For any nonzero two-bit flip, exactly one of the three encoded symbols becomes the invalid cell $(1, 1)$. The other two symbols have the form $z = \delta + \varepsilon v$ with constants $\delta, \varepsilon \in \mathbb{F}_3$ and $v \in \{0, 1\}$, and the decoded value after the flip is affine in z. Replacing the original cell equation therefore adds one binary unknown and one $\mathbb{F}_3$ equation. If F aligned cells are flipped, the model of Section 3.3 multiplies the expected number of related inputs by $(2/3)^F$. We sampled

four degree-256 irreducible polynomials with random middle coefficients; they contain 127 to 139 nonzero terms. For $s = 2$, the expected number of related inputs falls from $2^{25.6}$ with the sparse modulus to between $2^{-6.4}$ and $2^{-0.2}$ with these dense moduli.

Fewer pairs for $s = 2$ do not suffice to stop the attack. The pair condition remains a linear system in binary unknowns. The attacker can also try a larger shift. At $n = 48$, the extended systems agreed with exhaustive pair counts in all nine tests. At $n = 96$, three dense moduli gave no pair for $s = 2$ but gave 2, 1 and 1 pairs for $s = 3$. In each case, a pair led to unique key recovery from two queries and the $2^3 = 8$ guesses of (W_0, W_1, W_2).

For $n = 256$, larger shifts can raise the expected pair count. Over $2 \le s \le 6$, the counts sum to 1.8 to 4.4 per tested modulus. Searching one shift with Stern's algorithm [23] costs an estimated $2^{46.1}$ to $2^{51.6}$ operations. If a pair is found, key recovery needs 2^s instead of four guesses, which raises the central-weight two-level estimate $2^{67.3}$ to at most $2^{71.3}$ for these shifts, below the claimed 2^{128} security level. To justify a dense modulus as a countermeasure, one must therefore rule out usable pairs or show that finding one costs at least 2^{128}. The counts are heuristic and do not guarantee a pair for any given modulus.

At unchanged n and t, a dense modulus adds no daBits, laBits or protocol communication per evaluation; its local field-arithmetic cost is implementation dependent.

5.2.2. Long Non-Systematic Code. By (12), the expected number of related inputs per branch falls below 1 if and only if $r' + 2 > 2d/\log_2 3$, that is, $r' \ge 160$ for $d = 128$. For a non-systematic code, this is a condition on its length, and the non-systematic form alone does not suffice. An expected number below 1 per branch does not exclude related pairs. At the boundary $r' = 160$, a branch still has 0.59 related inputs on average, and other branches and shifts add to this number. A non-systematic code with $r' = 128$ would still leave $2^{50.0}$ related inputs per branch, and we did not study whether they can be found efficiently. The choice $r' = 192$ of van Baarsen et al. [14] meets the condition with a margin of 32 and gives $2^{-51.5}$ related inputs per branch. It costs a larger field, with $n = 385$ instead of 256, so the n daBits and n laBits of each evaluation of the maliciously secure protocol [5] grow by a factor of about 1.5. The bit length of the three transmitted $\mathbb{F}_{2^n}$ elements per evaluation also grows by this factor. Without systematic coordinates, the input check of Appendix A, with H a parity-check matrix of G, also accepts $\mathrm{bin}(G \cdot_3 x) \,\|\, 1$ for every $x \in \mathbb{F}_3^d$. A malicious client could then solve the pair condition over $\mathbb{F}_3$ without binary constraints, so the protocol must also prove that x is binary. We do not estimate this cost. Van Baarsen et al. [14] compute the encoding inside their protocol from a commitment to x over $\mathbb{F}_2$, so their input is binary by construction.

5.2.3. Output Mixing. Output mixing places a uniform public matrix $\Psi \in \mathrm{GL}_n(\mathbb{F}_2)$ before B:

$$F_k(x) = B \cdot_3 \big(\Psi \cdot_2 \mathrm{vec}(k \cdot \mathrm{elt}(E(x)))\big). \tag{13}$$

Here $\cdot_2$ denotes binary matrix multiplication. Let $A_c Y := \mathrm{vec}(c \cdot \mathrm{elt}(Y))$. For a pair with $\mathrm{elt}(E(x')) = c \cdot \mathrm{elt}(E(x))$, the mixed vectors $V = \Psi W$ and $V' = \Psi A_c W$ satisfy $V' = \Psi A_c \Psi^{-1} V$.

A permutation of the output bits does not help, because it preserves the near-shift relation of Lemma 1. A general Ψ instead replaces A_c by $\Psi A_c \Psi^{-1}$, so our two-bit affine system need not apply. This does not rule out a decoder that guesses more bits or uses a different relation. We therefore do not claim 128-bit security for the output-mixed PRF.

In the maliciously secure protocol [5], the $\mathbb{F}_{2^n}$-linear authentication of $k \cdot \mathrm{elt}(E(x))$ does not commute with Ψ. The $\mathbb{F}_2$ parts of the n daBits $\zeta \in \mathbb{F}_2^n$, however, are authenticated bit by bit, so the parties can compute $\Psi^{-1}\zeta$ locally and open

$$e = k \cdot \mathrm{elt}(E(x)) + \mathrm{elt}(\Psi^{-1}\zeta)$$

instead of the value masked by $\mathrm{elt}(\zeta)$. Then

$$\Psi \,\mathrm{vec}(e) = \Psi \,\mathrm{vec}\big(k \cdot \mathrm{elt}(E(x))\big) \oplus \zeta,$$

and the conversion modulo 3 is unchanged. We therefore estimate no additional daBits or openings and $O(n^2)$ local operations per evaluation. The original protocol consumes n daBits and n laBits and transmits three $\mathbb{F}_{2^n}$ and t $\mathbb{F}_3$ elements per evaluation [5].

5.2.4. Input Mixing. Input mixing computes $k \cdot \mathrm{elt}(PE(x))$ for a public random matrix $P \in \mathrm{GL}_n(\mathbb{F}_2)$. A related pair must then satisfy $E(x') = P^{-1} A_{\alpha^{-s}} PE(x)$, which is no longer a near shift on the encoding. The pair condition no longer gives the $\mathbb{F}_3$ systems of Section 3.2, but this does not prove a search lower bound. To resist our attack at the 128-bit level, finding a usable pair must cost at least 2^{128}, or no such pair must exist.

For any fixed nontrivial multiplier c, a uniform P maps two distinct nonzero encodings to a uniform pair of distinct nonzero vectors. An injective encoder therefore has on average

$$\frac{2^d(2^d - 1)}{2^n - 2}$$

related pairs for this multiplier. The proposed parameters satisfy $d = n/2$, giving approximately one pair. The same expectation remains near one if n and d grow with this ratio. Thus random input mixing alone does not give a pair-absence guarantee or establish 128-bit security. This count does not determine the cost of finding a pair.

For $(n, d, m) = (48, 24, 11)$, three random choices of P and $s = \pm 2$, a test of all inputs found on average 0.7 related inputs per shift with the sparse modulus and 1.7 with a dense one. These counts are close to the value 1 of a random relation, while the original design gives 14.3. If a related pair is found, it still satisfies

$$k \cdot \mathrm{elt}(PE(x')) = \alpha^{-s} \cdot k \cdot \mathrm{elt}(PE(x)),$$

so the decoder of Section 4 remains applicable; input mixing changes the search for the pair, not the subsequent decoding. In the maliciously secure protocol, the vector $PE(x)$ cannot

be checked directly over $\mathbb{F}_3$. The n laBits $\zeta \in \mathbb{F}_2^n$ that mask the committed input are authenticated bit by bit, so the parties can open $e = \mathrm{elt}(PE(x)) + \mathrm{elt}(P\zeta)$ instead of the value masked by $\mathrm{elt}(\zeta)$. Then $P^{-1}\,\mathrm{vec}(e) = E(x) \oplus \zeta$, and the parties check $E(x)$ modulo 3 as before. We therefore estimate no further laBits or openings and $O(n^2)$ local operations per evaluation.

5.2.5. Hashed Inputs. Hashing the input with a random oracle as in the semi-honest protocol [5] removes the choice of the encoding. With a malicious client, however, the protocol must check that the client computed the hash correctly, and Aranha et al. [5] avoid this approach because of its large overhead in that setting. This check requires a protocol for the chosen hash, so we give no numerical cost estimate.

5.2.6. Larger Parameters. Larger n and t preserve the attack's structure. Under our extrapolation $n = 2\lambda$ and $t = \lceil \lambda / \log_2 3 \rceil + 1$, the central-weight two-level representation estimate has a time exponent close to $\lambda/2$ (Figure 5). It is still $2^{127.2} < 2^{128}$ at $\lambda = 250$, or $(n, t) = (500, 159)$. Thus raising the modeled decoding cost above 2^{128} requires parameters around twice the proposed field and output sizes. This also roughly doubles the daBits and laBits per evaluation and the bit length of transmitted field elements.

6. Conclusion

We presented a two-query key-recovery attack on the strong PRF of Aranha et al. [5], covering E_{pad} and E_{impl}. A field shift supplies related inputs, reducing key recovery to ternary syndrome decoding. At the proposed parameters, the three-level representation model gives $2^{65.9}$ time and $2^{48.5}$ stored list elements, below the claimed 128-bit security. We did not run the decoder at $n = 256$. Its costs there are model estimates. Section 4.5 tests the representation model at $n = 96$ and full key recovery up to $n = 128$.

A dense modulus did not prevent the attack in our reduced-size experiments. Under the model of Section 3.3, a non-systematic code of length 192 leaves only $2^{-51.5}$ related inputs per branch on average, suppressing this attack path if the protocol also proves that x is binary. Output mixing changes the relation used by our two-bit decoder, but we establish no security bound for the modified PRF. Input mixing changes the pair search, but at the proposed ratio $d = n/2$ it leaves about one related pair per fixed multiplier on average and gives no 128-bit security bound. Hashed inputs prevent chosen encodings, while larger parameters raise both the attack and protocol costs.

Acknowledgments

We used Anthropic Claude (Opus 5.5) and OpenAI Codex (GPT-6 sol) to write and run the experiment and cost-model code and to draft and edit the text from our notes and results. This use covers the abstract, all sections and appendices, the figures and the tables. Every experimental result and cost estimate in the paper comes from this code and its logged outputs in the artifact, and every proof is written out in the paper. The authors directed the work, checked the results and the text, and take full responsibility for the paper.

References

[1] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *Theory of Cryptography – TCC 2005*, ser. Lecture Notes in Computer Science, vol. 3378. Springer, 2005, pp. 303–324.

[2] S. Jarecki, H. Krawczyk, and J. Xu, "OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks," in *Advances in Cryptology – EUROCRYPT 2018*, ser. Lecture Notes in Computer Science, vol. 10822. Springer, 2018, pp. 456–486.

[3] M. R. Albrecht, A. Davidson, A. Deo, and N. P. Smart, "Round-optimal verifiable oblivious pseudorandom functions from ideal lattices," in *Public-Key Cryptography – PKC 2021*, ser. Lecture Notes in Computer Science, vol. 12711. Springer, 2021, pp. 261–289.

[4] Y. Yang, F. Benhamouda, S. Halevi, H. Krawczyk, and T. Rabin, "Gold OPRF: Post-quantum oblivious power-residue PRF," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 259–278.

[5] D. F. Aranha, A. van Baarsen, A. Blatchley Hansen, K. Nielsen, and P. Scholl, "A maliciously-secure post-quantum OPRF from crypto dark matter," in *2026 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2026, pp. 1195–1214, Cryptology ePrint Archive, Paper 2026/533.

[6] D. Boneh, Y. Ishai, A. Passelègue, A. Sahai, and D. J. Wu, "Exploring crypto dark matter: New simple PRF candidates and their applications," in *Theory of Cryptography – TCC 2018*, ser. Lecture Notes in Computer Science, vol. 11240. Springer, 2018, pp. 699–729, Cryptology ePrint Archive, Paper 2018/1218.

[7] I. Dinur, S. Goldfeder, T. Halevi, Y. Ishai, M. Kelkar, V. Sharma, and G. Zaverucha, "MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications," in *Advances in Cryptology – CRYPTO 2021*, ser. Lecture Notes in Computer Science, vol. 12828. Springer, 2021, pp. 517–547.

[8] N. Alamati, G.-V. Policharla, S. Raghuraman, and P. Rindal, "Improved alternating-moduli PRFs and post-quantum signatures," in *Advances in Cryptology – CRYPTO 2024*, ser. Lecture Notes in Computer Science, vol. 14927. Springer, 2024, pp. 274–308.

[9] M. R. Albrecht, A. Davidson, A. Deo, and D. Gardham, "Crypto dark matter on the torus: Oblivious PRFs from shallow PRFs and TFHE," in *Advances in Cryptology – EUROCRYPT 2024*, ser. Lecture Notes in Computer Science, vol. 14656. Springer, 2024, pp. 447–476, Cryptology ePrint Archive, Paper 2023/232.

[10] J. H. Cheon, W. Cho, J. H. Kim, and J. Kim, "Adventures in crypto dark matter: Attacks and fixes for weak pseudorandom functions," in *Public-Key Cryptography – PKC 2021*, ser. Lecture Notes in Computer Science, vol. 12711. Springer, 2021, pp. 739–760, Cryptology ePrint Archive, Paper 2020/783.

[11] K. Hu, G. Leander, H. Raddum, A. Sandrib, and A. Udovenko, "Cryptanalysis of two alternating moduli weak PRFs," *IACR Transactions on Symmetric Cryptology*, vol. 2026, no. 1, pp. 95–118, 2026, Cryptology ePrint Archive, Paper 2026/482.

[12] I. M. Ayala and H. Raddum, "Zeroed out: Cryptanalysis of weak PRFs in alternating moduli," *IACR Transactions on Symmetric Cryptology*, vol. 2025, no. 2, pp. 1–15, 2025, Cryptology ePrint Archive, Paper 2024/2055.

[13] A. Sidem and Q. Wang, "General key recovery attack on pointwise-keyed functions: Application to alternating moduli weak PRFs," in *Advances in Cryptology – ASIACRYPT 2025*, ser. Lecture Notes in Computer Science, vol. 16245. Springer, 2025, pp. 131–154.

[14] A. van Baarsen, A. Goel, L. Kohl, P. Miao, P. V. L. Pham, P. Scholl, and S. Singh, "Maliciously secure shuffled distributed OPRF with applications to private set operations," in *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2026, pp. 107–121, Cryptology ePrint Archive, Paper 2026/2035.

[15] A. Biryukov and D. Wagner, "Slide attacks," in *Fast Software Encryption – FSE '99*, ser. Lecture Notes in Computer Science, vol. 1636. Springer, 1999, pp. 245–259.

[16] N. Howgrave-Graham and A. Joux, "New generic algorithms for hard knapsacks," in *Advances in Cryptology – EUROCRYPT 2010*, ser. Lecture Notes in Computer Science, vol. 6110. Springer, 2010, pp. 235–256.

[17] A. May, A. Meurer, and E. Thomae, "Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$," in *Advances in Cryptology – ASIACRYPT 2011*, ser. Lecture Notes in Computer Science, vol. 7073. Springer, 2011, pp. 107–124.

[18] R. Bricout, A. Chailloux, T. Debris-Alazard, and M. Lequesne, "Ternary syndrome decoding with large weight," in *Selected Areas in Cryptography – SAC 2019*, ser. Lecture Notes in Computer Science, vol. 11959. Springer, 2019, pp. 437–466, Cryptology ePrint Archive, Paper 2019/304.

[19] P. Karpman and C. Lefevre, "Time-memory tradeoffs for large-weight syndrome decoding in ternary codes," in *Public-Key Cryptography – PKC 2022*, ser. Lecture Notes in Computer Science, vol. 13177. Springer, 2022, pp. 82–111.

[20] M. Baldi, M. Battaglioni, F. Chiaraluce, A.-L. Horlemann, E. Persichetti, P. Santini, and V. Weger, "A new path to code-based signatures via identification schemes with restricted errors," *Advances in Mathematics of Communications*, vol. 19, no. 5, pp. 1360–1381, 2025.

[21] S. Bitzer, A. Pavoni, V. Weger, P. Santini, M. Baldi, and A. Wachter-Zeh, "Generic decoding of restricted errors," in *2023 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2023, pp. 246–251.

[22] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962.

[23] J. Stern, "A method for finding codewords of small weight," in *Coding Theory and Applications, 3rd International Colloquium*, ser. Lecture Notes in Computer Science, vol. 388. Springer, 1988, pp. 106–113.

[24] R. Schroeppel and A. Shamir, "A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems," *SIAM Journal on Computing*, vol. 10, no. 3, pp. 456–464, 1981.

[25] M. Bardet, J.-C. Faugère, and B. Salvy, "On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations," in *Proceedings of the International Conference on Polynomial System Solving*, 2004, pp. 71–74.

Appendix A. Points of the Specification with Two Readings

The encoding equation E_{spec} and the implementation encoder E_{impl} of Aranha et al. [5] differ in several details. Theorem 1 treats the completion E_{pad}, while Theorem 2 treats E_{impl}.

Length of the encoding. The PRF definition and security theorem of Aranha et al. [5] assume $n = d + 2(n' - d) + 1$, while their parameter set gives $(n, t) = (256, 82)$ and $(n', d) = (191, 128)$, for which $d + 2(n' - d) + 1 = 255$. The encoding therefore needs one padding bit. The original protocol checks the prescribed one-bit suffix 1 as $y_n = 1$, with positions numbered from 1. We analyze both placements of the missing zero: $\tau = (0, 1)$ keeps that 1 at position n, while $\tau = (1, 0)$ places the zero after it. The latter also completes the encoding equation, but Aranha et al. define no padded protocol check for it. We also analyze a single tail bit with $n = d + 2m + 1$ for $(d, m) = (129, 63)$ or $(127, 64)$ at $n = 256$.

Order of the two bits. Aranha et al. [5] write the shared-input decomposition as $(x, 2z + 2z^2, z + 2z^2)$, which gives $\mathrm{bin}(1) = (1, 0)$ and $\mathrm{bin}(2) = (0, 1)$. The gadget matrix $(2, 1) \otimes I$ of their input check reconstructs $z = 2\beta_0 + \beta_1$ and thus uses the opposite order. Theorem 1 holds for both orders.

Input check. The maliciously secure and half-malicious protocols check only $H \cdot G_{\mathrm{gadget}} \cdot y_{[1,n-1]} = 0$ and $y_n = 1$ for a committed input y, again with positions numbered from 1 [5]. The gadget matrix maps a cell (β_0, β_1) to $2\beta_0 + \beta_1$, so the invalid cell $(1, 1)$ is mapped to $3 \equiv 0$ and passes the check wherever the parity value is 0. The equivalence between the check and a valid encoding stated by Aranha et al. [5] therefore does not hold, and a malicious client can query words outside the image of E.

Appendix B. The Implementation Encoder E_{impl}

The encoder E_{impl} in the authors' public implementation differs from their encoding equation in bit maps, layout and tail. In this appendix, b_0 and b_1 denote its maps b_0^{impl} and b_1^{impl} from Section 2.2. The encoder forms

$$\begin{aligned} q(x) &= (b_0(z_i(x)))_{i=0}^{62} \,\|\, x \\ &\quad \|\, (b_1(z_i(x)))_{i=0}^{62} \,\|\, (1, 1), \\ E_{\mathrm{impl}}(x) &= \mathrm{rev}(q(x)), \end{aligned}$$

where rev places the first bit of q at coefficient 255 of the polynomial basis. The actual gadget checks the parity cells after 128 zeros are prepended to the first 254 bits of q. For $i < 128$ it reads the systematic bit x_i; for $i \geq 128$ it reads $2b_0(z_{i-128}) + b_1(z_{i-128}) = z_{i-128}$. Both trailing bits are 1. Thus every $q(x)$ passes the gadget and final-bit checks.

Let $x^\star = (x_2, \ldots, x_{127}, 0, 0)$ and $J = \{53, 54, 58, 59\}$. The following system has 69 equations over $\mathbb{F}_3$ in the 132 binary unknowns $(x, (v_i)_{i \in J})$:

$$z_0(x) = z_1(x) = 2, \tag{14}$$

$$z_i(x^\star) = z_{i+2}(x) \qquad (i \in [0, 60] \setminus J), \tag{15}$$

$$z_{i+2}(x) = v_i, \quad z_i(x^\star) = 1 - v_i \qquad (i \in J), \tag{16}$$

$$z_{61}(x^\star) = 2x_0, \quad z_{62}(x^\star) = 2x_1. \tag{17}$$

Theorem 2 (Related inputs for the implementation's encoder)**.** *For $f = X^{256} + X^{10} + X^5 + X^2 + 1$ and the encoding E_{impl}, a binary x satisfies $\mathrm{elt}(E_{\mathrm{impl}}(x^\star)) = \alpha^2 \,\mathrm{elt}(E_{\mathrm{impl}}(x))$ if and only if there is a binary $v \in \{0, 1\}^J$ satisfying (14)–(17). The vector v is unique. In this case, $(x^\star, x)$ is a related pair usable in the two-query key recovery of Section 4.*

Proof. Reversing the coefficient order turns multiplication by α^2 into a shift of q by two positions. Reduction by f adds q_1 at positions $245, 250, 253, 255$ and q_0 at positions $244, 249, 252, 254$ to $(q_2, \dots, q_{255}, 0, 0)$. This gives the word q' with $\mathrm{elt}(\mathrm{rev}(q')) = \alpha^2 \,\mathrm{elt}(E_{\mathrm{impl}}(x))$, so the relation holds if and only if $q' = q(x^\star)$. The final bits of q' are (q_0, q_1), which equal $(1,1)$ if and only if $z_0(x) = z_1(x) = 2$, that is, (14). Then the systematic block of q' is $(x_2, \dots, x_{127}, b_1(2), b_1(2)) = x^\star$. For $i \in [0, 60]$, cell i of q' is $(q'_i, q'_{191+i}) = \mathrm{bin}(z_{i+2}(x))$, with its second bit flipped if $i \in J$. Without the flip, it encodes $z_{i+2}(x)$, which gives (15). With the flip, it is valid if and only if $v_i = z_{i+2}(x) \in \{0,1\}$, and it then encodes $1 - v_i$, which gives (16). Since $q'_{252} = q'_{253} = 1 \oplus 1 = 0$, cells 61 and 62 are $(x_0, 0)$ and $(x_1, 0)$ and encode $2x_0$ and $2x_1$, which gives (17). These conditions are necessary and sufficient, and v is unique. □

If the parity equations behave like independent uniform constraints, this system has $2^{128}3^{-65}(2/3)^4 = 2^{22.6}$ binary solutions on average. We reconstructed G' from the zero-seeded generator in the authors' public implementation. For this matrix, the system has rank 69 and 63 free variables. None of its 69 pivot values is constant, so under the same assumption one related pair takes about $(3/2)^{69} = 2^{40.4}$ Gray-code steps on average. Our Gray-code search found a solution, and exact field arithmetic verified the relation. Both encoded inputs pass the implementation's gadget and final-bit checks. We also reconstructed the public output matrix B used by the implementation. The corresponding 164×254 key-recovery system has rank 164 and contains the true W. Appendix D records the pair. The full-size decoding cost remains a model estimate.

Appendix C. Related Inputs for Other Shifts and Tails

In this appendix, we extend Theorem 1 to the tail $(1,0)$, to other shifts and to the one-bit tail. Each proof gives only the differences from the proof of Theorem 1. For each case, let E be the n-bit encoder with the stated layout and tail, and put $y = E(x)$, $\hat{y} = \mathrm{vec}(\alpha^{-s} \cdot \mathrm{elt}(y))$ and $\rho^h := \rho^{(s)}(h)$ for $h \in \{0,1\}^s$. For the low bits $h = (x_0, \dots, x_{s-1})$ of x, Lemma 1 gives $\hat{y}_q = y_{q+s} \oplus \rho^h_q$.

Putting the zero after the appended 1 changes only the tail condition and the two boundary cells.

Corollary 1 (Tail $(1,0)$)**.** *Under the assumptions of Theorem 1 with $\tau = (1,0)$, let $\bar{\rho} := \rho^{(2)}(1, g_1) = \mathrm{vec}(\alpha^{-2}) \oplus g_1 \,\mathrm{vec}(\alpha^{-1})$ and define $\pi_{a,c}$ by* (7)*. Replace* (8a) *by $x_0 = 1$, $x_1 = g_1$, and the right-hand sides of* (8d) *by $\mathrm{bin}^{-1}(b_1(a), 1)$ and $\mathrm{bin}^{-1}(b_1(c), 0)$. Then $x \in \{0,1\}^d$ is 2-good if and only if $a = z_0(x) \in Z_*$ and x solves the modified system $S_{a,c}$ with $c = z_1(x)$. The 2-good inputs are the disjoint union of the binary solutions over the six branches $(a, c) \in Z_* \times \mathbb{F}_3$.*

Proof. The tail of $\hat{y}$ is $(x_0, g_1 x_0 \oplus x_1)$, which equals $(1,0)$ if and only if $x_0 = 1$ and $x_1 = g_1$, and then $\rho = \bar{\rho}$. The boundary cells become $(b_1(a), 1)$ and $(b_1(c), 0)$, so the validity condition moves from c to a. The other parts of the proof of Theorem 1 do not use the tail. □

TABLE 2. BRANCHES (BR.) OF THE PAIR CONDITION IN THE BLOCK LAYOUT FOR $n = 256$, $f = X^{256} + X^{10} + X^5 + X^2 + 1$ AND A UNIFORMLY SAMPLED G'. THE NUMBER OF EQUATIONS AND $\log_2 N_{\mathrm{pair}}$ FOR THE EXPECTED NUMBER N_{pair} OF RELATED INPUTS PER BRANCH COME FROM THE STATEMENTS (FORMULA) AND FROM THE SYSTEMS THAT OUR CODE BUILDS (CODE). INTERCHANGING b_0 AND b_1 GIVES THE SAME NUMBERS.

				Equations		$\log_2 N_{\mathrm{pair}}$	
τ	(d, m)	s	Br.	formula	code	formula	code
$(1,0)$	$(128, 63)$	1	0	–	–	–	–
$(1,0)$	$(128, 63)$	2	6	67	67	23.0	23.0
$(1,0)$	$(128, 63)$	3	30	69	69	20.4	20.4
$(1,0)$	$(128, 63)$	4	150	71	71	17.8	17.8
$(0,1)$	$(128, 63)$	2	6	67	67	23.0	23.0
(1)	$(129, 63)$	1	2	65	65	26.6	26.6
(1)	$(129, 63)$	2	10	67	67	24.0	24.0
(1)	$(127, 64)$	1	2	66	66	23.0	23.0

Table 2 compares the number of equations and the expected number of related inputs from the statements with the systems that our code builds for $n = 256$.

C.1. Other Shifts in the Block Layout

Let $n = d + 2m + 2$, $\tau \in \{(1,0), (0,1)\}$, $r \le d$ and $2 \le s \le m$. We call $h \in \{0,1\}^s$ *tail-compatible* if $(\rho^h_{n-2}, \rho^h_{n-1}) = \tau$. For $j \in [0, s-1]$ let $\eta_j(h) := \tau_j$ if $j \le 1$ and $\eta_j(h) := \rho^h_{n-2-s+j}$ if $j \ge 2$. A *branch* is a pair (h, a) with h tail-compatible and $a \in \mathbb{F}_3^s$ such that $a_j \in Z_*$ whenever $\eta_j(h) = 1$. For a branch (h, a) let

$$\pi_{h,a}(x)_j = \begin{cases} \rho^h_j + (-1)^{\rho^h_j} x_{j+s}, & j \in [0, d-s-1], \\ b_0(a_{j-d+s}) \oplus \rho^h_j, & j \in [d-s, d-1], \end{cases}$$

and let $S_{h,a}$ be the system of the $m + 2s$ equations

$$\begin{aligned} &x_p = h_p, \qquad z_p(x) = a_p, \\ &z_i(\pi_{h,a}(x)) = z_{i+s}(x), \\ &z_{m-s+j}(\pi_{h,a}(x)) = \mathrm{bin}^{-1}\big(b_1(a_j), \eta_j(h)\big) \end{aligned}$$

for $p \in [0, s-1]$, $i \in [0, m-s-1]$ and $j \in [0, s-1]$.

Theorem 3 (Block layout, shift s)**.** *Let $x \in \{0,1\}^d$, let $h = (x_0, \dots, x_{s-1})$ be its low bits, and let $a = (z_0(x), \dots, z_{s-1}(x))$ be its first s parities. Then x is s-good if and only if (h, a) is a branch and x is a solution of $S_{h,a}$, and then $x' = \pi_{h,a}(x)$. Under the model of Section 3.3, each branch has $2^{d-s} \cdot 3^{-(m+s)}$ binary solutions on average.*

Proof. By Lemma 1, $\mathrm{supp}(\rho^h) \subseteq [0, d-1] \cup [n-s, n-1]$, and the positions $q \ge n - s$ receive no shifted bit, so $\hat{y}_q = \rho^h_q$ there. The tail positions $n-2$ and $n-1$ give the tail-compatibility of h, and the positions $[n-s, n-3]$ are the second bits of the cells $m-s+2, \dots, m-1$, which gives $\eta_j(h)$ for $j \ge 2$. For $j \in [0, s-1]$, the first bit of cell $m-s+j$ is $y_{d+m+j} = b_1(a_j)$, and for $j \le 1$ its second bit is $y_{d+2m+j} = \tau_j$. Cell $m-s+j$ is therefore $(b_1(a_j), \eta_j(h))$,

which is valid if and only if $a_j \in Z_*$ or $\eta_j(h) = 0$. The cells $i \in [0, m-s-1]$ lie in $[d, n-s-1]$, where ρ^h vanishes, so they are $\mathrm{bin}(z_{i+s}(x))$. The systematic part of $\hat{y}$ is $\pi_{h,a}(x)$ by (6), and the rest is the proof of Theorem 1 with s in place of 2. □

C.2. Shift One and the One-Bit Tail

For both two-bit tails there is no 1-good input. Position $n-2$ of $\hat{y}$ is $y_{n-1} \oplus \rho^{(1)}(x_0)_{n-2} = \tau_1$, because the coordinate of $\mathrm{vec}(\alpha^{-1})$ at $n-2$ is $g_{n-1} = 0$ for $r \le d$. A valid encoding has τ_0 at this position, and $\tau_0 \neq \tau_1$ for both tails.

With a single tail bit and $n = d + 2m + 1$, the shift $s = 1$ works in the block layout. Position $n-1$ of $\hat{y}$ is $\rho^{(1)}(x_0)_{n-1} = x_0$, so $x_0 = 1$, and the last cell of $\hat{y}$ is $(b_1(z_0(x)), 1)$, so $z_0(x) \in Z_*$. The two branches give $m+2$ equations each and $2^{d-1} \cdot 3^{-(m+1)}$ binary solutions on average. For $s = 2$, the tail gives $x_1 = 1 \oplus g_1 x_0$, and cells $m-2$ and $m-1$ of $\hat{y}$ are $(b_1(z_0(x)), 1)$ and $(b_1(z_1(x)), x_0)$. Thus $z_0(x) \in Z_*$, and also $z_1(x) \in Z_*$ when $x_0 = 1$, which gives $6+4 = 10$ branches with $m+4$ equations and $2^{d-2} \cdot 3^{-(m+2)}$ binary solutions each on average (Table 2).

Appendix D. Related Pairs for the Proposed Parameters

Table 3 gives one related pair for E_{pad}, one for the form of E_{pad} with b_0 and b_1 interchanged, one for the completion of E_{spec} with $\tau = (1, 0)$, which our search found in 146 seconds, and one for E_{impl}. All four use $(n, t, n', d) = (256, 82, 191, 128)$ and $f = X^{256} + X^{10} + X^5 + X^2 + 1$. The first three pairs use the same uniformly sampled $G' \in \mathbb{F}_3^{128\times 63}$, and the fourth uses the public implementation's generated G'. The artifact regenerates both matrices and contains all four pairs. We write a vector v as the integer $\sum_i v_i 2^i$ in hexadecimal, so that bit i of $E(x)$ is the coefficient of α^i. For E_{impl}, the inputs x and x' in Table 3 are $x^\star$ and the solution x of Theorem 2.

Appendix B describes how we found and verified the E_{impl} pair.

TABLE 3. FOUR RELATED PAIRS AT $(n, d) = (256, 128)$. EACH SATISFIES $\mathrm{elt}(E(x)) = \alpha^2\,\mathrm{elt}(E(x'))$ FOR THE ENCODER INDICATED BELOW. ENCODINGS ARE SHOWN IN HIGH AND LOW 128-BIT HALVES.

E_{pad}	
x	0xd51eee1bdb426c90cb1aea6a0be20a82
x'	0x3547bb86f6d09b2432c6ba9a82f880b2
$E(x)$	0x80168561162012647300213051 2cdb10 d51eee1bdb426c90cb1aea6a0be20a82
$E(x')$	0xa005a158458804991cc0084c144b36c4 3547bb86f6d09b2432c6ba9a82f880b2
E_{pad} with b_0, b_1 interchanged	
x	0xa5e52b735c2ea8ce5da3dadbd957601a
x'	0x29794adcd70baa339768f6b6f655da14
$E(x)$	0xb2c016d1288c210e0003c0400e218dc0 a5e52b735c2ea8ce5da3dadbd957601a
$E(x')$	0xacb005b44a2308438000f01003886370 29794adcd70baa339768f6b6f655da14
E_{spec} completed with $\tau = (1, 0)$	
x	0xd51eee0763674887dcb481982784d175
x'	0x3547bb81d8d9d221f72d206609e13554
$E(x)$	0x4246822090d1b00071027b3a1e0c17c8 d51eee0763674887dcb481982784d175
$E(x')$	0x5091a08824346c001c409ece878305f2 3547bb81d8d9d221f72d206609e13554
Public implementation: E_{impl}	
x	0x0bc753116933e32a0a9f0a05e95fa4a3
x'	0x2f1d4c45a4cf8ca82a7c2817a57e928f
$E_{\mathrm{impl}}(x)$	0x212682911aab600f8a4bf52f40a1f2a0 a98f992d1195c7a18420209088203043
$E_{\mathrm{impl}}(x')$	0xc849a0a446aad803e292fd4bd0287ca8 2a63e64b446571e86108082422080f0b