Quantum Algorithms for Fast Correlation Attacks on LFSR-Based Stream Ciphers[⋆]

Akinori Hosoyamada

NTT Social Informatics Laboratories, Tokyo, Japan
NTT Research Center for Theoretical Quantum Information, Atsugi, Japan
akinori.hosoyamada@ntt.com

Abstract. This paper presents quantum algorithms for fast correlation attacks, one of the most powerful techniques for cryptanalysis on LFSR-based stream ciphers in the classical setting. Typical fast correlation attacks recover a value related to the initial state of the underlying LFSR by solving a decoding problem on a binary linear code with the Fast Walsh-Hadamard Transform (FWHT). Applying the FWHT on a function in the classical setting is mathematically equivalent to applying the Hadamard transform on the corresponding state in quantum computation. While the classical FWHT on a function with ℓ-bit inputs requires $O(\ell 2^{\ell})$ operations, the Hadamard transform on ℓ-qubit states requires only a parallel application of $O(\ell)$ basic gates. This difference leads to the exponential speed-up by some quantum algorithms, including Simon's period finding algorithm.

Given these facts, the question naturally arises of whether a quantum speedup can also be achieved for fast correlations by replacing the classical FWHT with the quantum Hadamard transform. We show quantum algorithms achieving speed-up in such a way, introducing a new attack model in the Q2 setting. The new model endows adversaries with a quite strong power, but we demonstrate its feasibility by showing that certain members of the ChaCha and Salsa20 families will likely be secure in the new model. Our attack exploits the link between LFSRs' state update and multiplication in a fine field to apply Shor's algorithm for the discrete logarithm problem. We apply our attacks on SNOW 2.0, SNOW 3G, and Sosemanuk, observing a large speed-up from classical attacks.

Keywords: symmetric-key cryptography · quantum cryptanalysis · fast correlation attacks · LFSR-based stream ciphers

1 Introduction

While research and standardization of post-quantum public-key cryptosystems have been steadily progressing in the past decade [68], research on quantum security of symmetric-key cryptography has also been advancing. Starting with the

[⋆] This article is the full version of the paper with the same title accepted to Asiacrypt 2024, ©IACR 2024.

early results of Grover's algorithm [43] for speeding up the exhaustive key search and the BHT algorithm [19] for speeding up collision search, a wide variety of attack techniques have been proposed, including those breaking some schemes in polynomial time with Simon's algorithm pioneered by Kuwakado and Morii [54, 55], Kaplan et al. [51], and Santoli and Scaffner [74]. A more recent research [16] has revealed that, even in the most conservative attack model, simply doubling the size of a secret key does not necessarily ensure the same level of security as in the classical setting. Another line of research started in [47, 28] has shown more rounds of some hash functions are broken in the quantum setting than in the classical setting, which underscores the importance of studying quantum attacks on symmetric key cryptography.

Whereas some quantum attacks are based on ideas completely different from classical attacks, others attempt to speed up classical attacks through quantum computation (e.g., [52, 15]). A large speed-up is sometimes obtained, while in other cases, an attack classically faster than the generic attack turns out to be slower than the quantum generic attack. To better understand security in the quantum setting, it is important to investigate how the efficiency and the validity of each classical attack change.

There are two attack models in the quantum setting, which are called Q1 and Q2 [52]. Q1 assumes that an adversary has a quantum computer, but oracles remain unchanged from the classical setting. In contrast, Q2 assumes both are quantum and that an oracle allows quantum superposition queries. The assumption of Q2 is strong, but Q2 attacks are still quite worth studying. If the key length of a target scheme is sufficiently long, Q2 attacks can be converted into Q1 by emulating the quantum oracle after getting all the outputs of the classical oracle (full codebook). Quite powerful Q1 attacks are sometimes developed from Q2 attacks [13, 16].

Quantum Fourier transforms (QFTs) play a crucial role in achieving exponential speedups in specific quantum algorithms, such as Shor's [79] and Simon's [81]. There are various types of QFTs, depending on the base group. For example, Shor's algorithm utilizes QFT over the cyclic group of a large order. Meanwhile, Simon's algorithm uses a QFT over $(\mathbb{Z}/2\mathbb{Z})^{\oplus n}$ for some n, which is referred to as the Hadamard transform. This transform is mathematically equivalent to the Walsh-Hadamard Transform (WHT) in classical computation.

The WHT has strong relationships with several classical attack techniques, particularly linear cryptanalysis [60]. Linear correlations of block ciphers can be obtained by applying the WHT, and the Fast-Walsh Hadamard Transform (FWHT) is commonly employed to accelerate key recovery [26]. It naturally raises the question of whether these traditional methods can be combined with the Hadamard transform to achieve quantum speedups. In fact, a recent work showed a framework to combine the quantum Hadamard transform and the classical linear key recovery attack with FHWT [76].

An important class of attacks closely linked with linear cryptanalysis is (fast) correlation attacks on LFSR-based stream ciphers. Correlation attacks, initially proposed by Siegenthaler [80], exploit linear correlations between keystreams output by a target cipher and the underlying LFSR's output sequence. An enhanced version, today known as the fast correlation attack, was later given by Meier and Staffelbach [62]. Having been continually improved ever since [89, 64, 23, 49, 50, 20, 22, 63, 25, 92, 83, 41, 91], the fast correlation attack is currently the most effective method for attacking various LFSR-based ciphers. For major ciphers such as SNOW 3G [34], research is being done to see how efficient the fast correlation attack can be, even if it is slower than the generic attack [72, 88, 40, 41, 39].

Roughly speaking, fast correlation attacks aim to recover the initial state of the underlying LFSR (or a related value) by solving the decoding problem of a linear code. Typical attacks perform the decoding quickly by applying the FWHT [25]. Given the aforementioned result on linear cryptanalysis in the quantum setting, the question naturally arises whether an interesting quantum attack can also be obtained for fast correlation attacks by replacing the classical FWHT with the Hadamard transform.

Based on the above motivation, this paper studies the quantum speedup of fast correlation attacks on LFSR-based stream ciphers. We focus on the setting where the decoding problem is defined over a binary code and a one-pass decoding algorithm with FWHT is applied, as this has been widely applied to various ciphers.

Technical Overview and Our Contributions. Before outlining our contributions, we briefly overview the basics of classical attacks.

Classical Fast Correlation Attacks on LFSR-Based Stream Ciphers. Typical LFSR-based stream ciphers are composed of an *initialization phase* and a *keystream generation phase*. The initialization phase takes a secret key K and an IV as input, non-linearly mixing them and loading the resulting values into internal registers. Following that, the keystream generation phase computes keystream bits, updating the internal states at each clock. Encryption and decryption are performed by XORing the keystream bits to a message or a ciphertext, as done in the counter mode. By the *initial state*, we denote the state right after the initialization phase.

As mentioned earlier, fast correlation attacks recover a value related to the LFSR's initial state by solving a decoding problem. In the simplest case, when there is a linear approximation with correlation c between the key stream output by the cipher and the output sequence of the underlying binary LFSR of length ℓ, a binary linear code is defined such that a message of ℓ bits (the initial state of the LFSR) is encoded to a codeword of length $N \gg \ell/c^2$. The basic idea is that if we regard the keystream as the encoded message with some noise added and decode it, correcting the errors, then we obtain the original message, namely the initial state of the LFSR.

Decoding is performed in the following manner. First, a certain function $\Psi(\boldsymbol{x})$ with ℓ-bit inputs (determined according to the binary code and keystream bits) is computed for all $\boldsymbol{x}$. Second, the WHT of the function Ψ, denoted by $\mathcal{W}(\Psi)$, is computed using FWHT with $O(\ell 2^{\ell})$ operations. For each decoded message candidate $\boldsymbol{x}$, the larger the value $((\mathcal{W})(\Psi))(\boldsymbol{x})^2$ is, the more likely it is that $\boldsymbol{x}$ is the correct result. In particular, the decoding result is identified to be the $\boldsymbol{x}$ that gives the maximum value of $((\mathcal{W})(\Psi))(\boldsymbol{x})^2$. The computational complexity is $O(N + \ell 2^{\ell})$ in total.

The decoding complexity $\ell 2^{\ell}$ too large in most cases, making the attack slower than the exhaustive key search. To address the issue, a preprocessing procedure is usually performed in advance to reduce the dimension of the code (i.e., the parameter ℓ), thereby reducing the decoding complexity. However, the preprocessing procedure is usually heavy, and reducing the dimension increases the data complexity N. Hence, the preprocessing procedure and the number of dimensions to reduce are carefully adjusted to balance the computational complexity of preprocessing, the amount of data N, and the decoding complexity.

Next, we explain our results in the quantum setting.

Attempt in Q1. First, we try to obtain a quantum speed-up in the Q1 model by naturally extending classical attacks.

At the beginning of the attack, we obtain N bits of a keystream segment required to mount the attack (for some N). Next, we prepare a quantum state $|\psi\rangle$ corresponding to the function $\Psi(\boldsymbol{x})$. We show that the state $|\psi\rangle$ can be prepared with a complexity $\tilde{O}(N)$ in typical cases. Then, we apply the Hadamard transform to $|\psi\rangle$. The resulting quantum state $H^{\otimes \ell} |\psi\rangle$ is a quantum superposition of the message candidates $|\boldsymbol{x}\rangle$, among which the one with the largest quantum amplitude is the correct message. To find the correct $\boldsymbol{x}$, we apply the Quantum Amplitude Amplification (QAA) technique. The Boolean function required to apply QAA, denoted by f, can be chosen depending on the structure of the attack target. If the decoding problem is defined from a linear approximation with correlation c, the bit length of LFSR is ℓ, and the computational complexity to compute f is T_f, then the attack complexity becomes $O(N + 2^{\ell/2} T_f / \sqrt{Nc^2})$.

This is a quite natural extension of the classical attacks. However, we observe that applying the above algorithm to speed up existing classical attacks does not yield a quantum attack faster than the Grover search. Rather, we suspect it is quite hard to mount a fast correlation attack that is faster than the generic attack in the Q1 setting, or more fairly non-trivial techniques will be required. So, we focus on Q2 attacks.

Attack in Q2. An important feature of stream ciphers is that they can generate an exponentially long keystream from a single IV. In the Q2 setting, we first introduce a new attack model and security notion that reflects this feature well. Very roughly, the model allows an adversary to query the positions of keystream bits as well as IVs in quantum superposition. Although this attack model is very strong, we show that it is feasible in that some stream ciphers, such as some members of Chacha and Salsa20 families [9, 10], are likely to achieve the security notion.

We then show a quantum decoding algorithm in the Q2 model. The underlying idea is the same as in Q1, but we make a non-trivial observation that the preparation of the quantum state $|\psi\rangle$ can be performed very efficiently using Shor's algorithm.

Recall that, in the Q1 attack, we first prepared the state $|\psi\rangle$ with both data and time complexity about N, which is exponentially large in usual scenarios. In the Q2 setting, our strong attack model allows us to reduce the data complexity (i.e., the number of queries) from $O(N)$ to $O(1)$.

Our key observation is that even the time complexity can be reduced from $O(N)$ to polynomial time using Shor's algorithm. In preparing the state $|\psi\rangle$, we need to solve the following problem: Given an arbitrary $\boldsymbol{x}$, find an index i such that $\boldsymbol{x}$ equals to the i-th column of the generating matrix G of the code used in the attack. G is typically determined from the LFSR's state update matrix and linear correlation masks. By leveraging the fact that the LFSR's state update corresponds to the multiplication of an element generating $(\mathbb{F}_{2^\ell})^\times$, we find that the problem is reduced to a discrete logarithm problem in a typical case and can be efficiently solved with Shor's algorithm.

As in the Q1 attack, QAA is applied to the state $H^{\otimes \ell}|\psi\rangle$ to amplify the quantum amplitude of the correct message. For all attack targets, we utilize the quantum counting algorithm to implement a Boolean function for QAA, similar to Kaplan et al.'s approach for quantum linear distinguishers [52]. As a result, the computational complexity of the attack is $O(\ell^4/c^2)$ when the attack is based on a code defined from a linear approximation with absolute correlation c. The value of c is large enough for some ciphers (around 2^{-20} in some cases) to achieve faster attacks than the exhaustive key recovery with Grover's algorithm.

As applications, we show attacks on the ISO/IEC standard SNOW 2.0 [31, 48], SNOW 3G specified by 3GPP [34], and Sosemanuk in the eSTREAM portfolio [6, 29]. For SNOW 2.0, our attack works with time and query complexity $2^{59.3}$ and $2^{89.3}$, respectively. This is the first attack on the 256-bit key version of SNOW 2.0 faster than the Grover search[1]. When our technique is applied to SNOW 3G, the resulting time and query complexity become $2^{102.9}$ and $2^{72.9}$, respectively. This is slower than the Grover search but significantly faster than classical attacks. (As mentioned earlier, research of attacks on SNOW 3G has actively been continued to determine how efficient fast correlation attacks can be [72, 88, 40, 41, 39], even though they are slower than the exhaustive key search.) About Sosemanuk, the time and query complexity of our attack becomes $2^{101.11}$ and $2^{73.15}$, respectively. This is slower than the quantum guess-and-determine attack in the Q1 model by Ding et al. [27], but faster than the Grover search when the key length is long (e.g., 256-bit). See also Table 1.

The quantum state $H^{\otimes \ell}|\psi\rangle$ appearing our attacks can be regarded as an analogy of the *correlation state* in the quantum linear key recovery attack by Schrottenloher [76], as the amplitude of each basis state $|\boldsymbol{x}\rangle$ in $H^{\otimes \ell}|\psi\rangle$ is, in fact, proportional to the correlation between a binary sequence derived from keystream and the codeword corresponding to $\boldsymbol{x}$. Still, the techniques used in

[1] A previous work [27] showed an attack on SNOW 2.0 running in time about 2^{88}, but it requires exponentially many qubits (as large as 2^{88}) and in fact slower than the generic attack by the parallelized Grover search. More details are given in Remark 5.

Target	Key Length	Attack Model	Time	Data/Query	Ref./Note
SNOW 2.0	128/256	Classical	$2^{162.86}$	$2^{159.62}$	[41]
		Q2	$2^{89.3}$	$2^{59.3}$	Section 6
SNOW 3G	128	Classical	$2^{174.95}$	$2^{170.81}$	[39]
		Q2	$2^{102.9}$	$2^{72.9}$	Section 6
Sosemanuk	$0 \leq \kappa \leq 256$	Classical	$2^{134.8}$	2^{135}	[91]
		Q2	$2^{101.11}$	$2^{73.15}$	Section 6
		Q1	$\approx 2^{88}$	≈ 176	[27]
Any	κ	Q1/Q2	$\approx 2^{\kappa/2}$	$\approx \kappa$	Generic attack (Grover's algorithm [43])

Table 1. Comparison of attack complexity. The previous works define the time complexity unit as the time to perform arithmetic operations such as modular additions and finite field multiplications or, more ambiguously, as the time required to run the targeted cipher once. We regard the time complexity of an attack as the depth of the quantum circuit implementing it, where the depth is measured in T-gates and oracle gates.

the two attacks are quite different. The linear key recovery attack utilizes the Hadamard operator to compute some functions' convolutions, performing (classical and) manual computation on the Walsh-Hadamard transform of a public function and exploiting a target cipher's structure in which a subkey is XORed into a state. On the other hand, such convolution and manual computation of the Walsh-Hadamard transform do not appear in our attack, and Shor's algorithm for discrete logarithms is utilized to prepare the state, exploiting the relationship between LFSRs' state update and multiplication in a finite field.

Related Works. Independently and concurrently, Einsele and Semira also mentioned studies on quantum speed-up of fast correlation attacks [30], but only a short abstract is publicly available. In particular, concrete attack models or attack algorithms are not explained.

Paper Organization. Section 2 describes the notation, promises, and well-known basic results used in later chapters. Section 3 reviews classical fast correlation attacks. Section 4 discusses attacks in Q1. Section 5 introduces a new attack model and security notion, and Section 6 shows attacks in Q2.

2 Preliminaries

Unless otherwise noted, we assume all the vectors are row vectors. For any m and n, we naturally identify elements in $\mathbb{F}_{2^n}^m$ with those in $\mathbb{F}_2^{mn}$, and mn-bit strings. For $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{F}_2^m$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathbb{F}_2}$ denotes their formal inner product. The linear correlation between two binary sequences $\boldsymbol{x} = (x_0, \ldots, x_{N-1})$ and $\boldsymbol{y} =$

$(y_0, \ldots, y_{N-1})$ is defined by

$$\mathrm{Cor}(\boldsymbol{x}, \boldsymbol{y}) := \frac{\#\{i : x_i = y_i\} - \#\{i : x_i \neq y_i\}}{N}.$$

We identify Boolean functions $f : \{0, \ldots, N-1\} \to \mathbb{F}_2$ with binary sequences $(f(0), \ldots, f(N-1))$. The linear correlation between two Boolean functions $\mathrm{Cor}(f, g)$ is naturally defined through this identification. The Walsh-Hadamrd transform of a function $F : \mathbb{F}_2^n \to \mathbb{C}$, denoted by $\mathcal{W}(F)$, is the function from $\mathbb{F}_2^n$ to $\mathbb{C}$ defined as[2]

$$\left(\mathcal{W}(F)\right)(\boldsymbol{z}) = \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x} \in \mathbb{F}_2^n} (-1)^{\langle \boldsymbol{x}, \boldsymbol{z} \rangle_{\mathbb{F}_2}} F(\boldsymbol{x}).$$

2.1 Quantum Computation

This paper assumes the readers are familiar with quantum computation (refer to, e.g., [71] for the basics). We adapt the quantum circuit model as a model for quantum computation, assuming arbitrary circuits composed of a finite number of Clifford+T gates, quantum oracle gates (only in the Q2 model), and quantum Random Access Memory (qRAM) gates. Here, the quantum oracle gate of a function $f : \mathbb{F}_2^m \to \mathbb{F}_2^n$ is the $(m+n)$-qubit gate such that, given a quantum state of the form $\sum_{x,y} \alpha_{x,y} |x, y\rangle$ as an input, outputs the state $\sum_{x,y} \alpha_{x,y} |x, y \oplus f(x)\rangle$. About qRAM, this paper assumes a quantum-accessible *classical* memory is available to an adversary. Namely, for an arbitrarily created list of classical data $(x_1, \ldots, x_n)$, the adversary is given quantum oracle access to the function $i \mapsto x_i$. CNOT gates are assumed to operate on an arbitrary pair of qubits in a circuit. Quantum error correction is assumed to be perfectly performed with its cost being ignored. All the measurements are performed in the computational basis.

How to Evaluate Attack Costs. We always measure the depth of quantum circuits by calculating the depth in T gates, oracle gates, and qRAM gates. We regard the running time of a quantum circuit as its depth in this measure. When considering an attack on an LFSR-based stream cipher, we assume that the number of qubits available for an adversary is in a small polynomial of the underlying LFSR's bit length to enable a fair comparison with the generic key-recovery attack using Grover's algorithm [43] without parallelization.

Quantum Amplitude Amplification. Let U be a unitary operator acting on n-qubit quantum states, $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function, and p be the probability that an x satisfying $f(x) = 1$ is obtained when the quantum state $U|0^n\rangle$ is measured. The Quantum Amplitude Amplification (QAA) technique [18] amplifies the probability p by making $O(p^{-1/2})$ quantum queries to f with $O(p^{-1/2})$ applications of U and $U^\dagger$ as follows.

[2] We adopt the definition with the coefficient $1/\sqrt{2^n}$ to make it consistent with the Hadamard operators in the quantum setting.

Proposition 1 (Plain QAA). *Let $\mathcal{S}_f$ (resp., $\mathcal{S}_0$) be the unitary operators that multiplies the basis state $|x\rangle$ by $(-1)^{f(x)}$ (by (-1) iff $x = 0^n$), and define a unitary operator $Q(U, f) := -U\mathcal{S}_0 U^\dagger \mathcal{S}_f$. When the quantum state $(Q(U, V))^i\, U\, |0^n\rangle$ is measured, an x satisfying $f(x) = 1$ is obtained with probability* $\sin^2((2i + 1)$ $\arcsin(\sqrt{p}))$*, which is at least* $\max(1 - p, p)$ *when* $i := \left\lfloor \frac{\pi}{4\arcsin(\sqrt{p})} \right\rfloor$.

QAA with Certainty. If an adversary knows the exact value of p, then the QAA can be modified in such a way to obtain a good state with certainty, by modifying U to slightly lower the probability p to make $(\pi/(4\arcsin(\sqrt{p})) - (1/2))$ be an integer [18]. In this paper, we assume the cost of this modification is negligible compared to implementing U and $U^\dagger$ themselves, and QAA returns an x satisfying $f(x) = 1$ by applying U, $U^\dagger$, and $\mathcal{S}_f$ at most $\arcsin(\sqrt{p}) \leq p^{-1/2}$ times (if an adversary knows the exact value of p).

QAA without Knowing p. When applying the plain QAA in Proposition 1, the success probability does not become large enough not only if i is too small but also if i is too large. For instance, if $i \approx 2 \cdot \left\lfloor \frac{\pi}{4\arcsin(\sqrt{p})} \right\rfloor$, then the success probability may be as small as p.

However, it is not necessarily easy to find the exact value of p, when it is practically too hard to compute the value $\left\lfloor \frac{\pi}{4\arcsin(\sqrt{p})} \right\rfloor$ exactly. Even in such a case, an x satisfying $f(x) = 1$ can be found by running the plain QAA multiple times with random i as follows [18, 17].

Algorithm QAAw/oKp.

```
1. Let $\alpha := 1$ and $\lambda := 6/5$.
2. Choose $i$ from $\{0, 1, \ldots, \alpha - 1\}$ uniformly at random.
3. Run the plain QAA with $i$ iterations and measure the entire state, i.e.,
   $(Q(U, f))^i U |0^n\rangle$. Let $x$ be the measurement result.
4. If $f(x) = 1$, return $x$ as the output. Otherwise, set $\alpha := \min\{\lambda \cdot \alpha, \sqrt{2^n}\}$
   and go to Step 2.
```

Proposition 2 (QAA without knowing p [18, 17]). *Suppose* $p \leq 3/4$. *Then, the algorithm* QAAw/oKp *returns x satisfying $f(x) = 1$ with an expected number of applications of $Q(U, f)$ at most* $(9/2)p^{-1/2}$.

Grover's algorithm [43] is the special case of QAA when $U = H^{\otimes n}$.

Quantum Counting Algorithm. Let QFT_q denote the quantum Fourier transform over $\mathbb{Z}/q\mathbb{Z}$. For any unitary operator W acting on n-qubit states and any positive integer q that is a power of 2, let $\Lambda_q(W)$ be the operator acting on $(\log q + n)$-qubit states such that $\Lambda_q(W)\,|i\rangle\,|x\rangle = |i\rangle\,(W^i\,|x\rangle)$. Here, $0 \leq i \leq q - 1$ and $x \in \mathbb{F}_2^n$. For a unitary operator U and a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, define the probability p and the operator $Q(U, f)$ as in Proposition 1. In addition, let $\mathsf{Calc}_{n,q}$ be the unitary operator that, given a (classical) value θ, computes

$2^n \cdot \sin^2(\pi\theta/q)$ and write the result into an additional register. Now, consider running the following algorithm without measurement.

Algorithm QC. Prepare $|0^{\log_2 q}\rangle\,|0^n\rangle$ as the initial state. Apply $(QFT_q \otimes H^{\otimes n})$, $\Lambda_q(Q(H^{\otimes n}, f))$, and then $(QFT_q^\dagger \otimes I_n)$ in sequential order. Finally, apply $\mathsf{Calc}_{n,q}$, taking input from the left $\log q$-bit register and writing the output into an auxiliary register.

Proposition 3 ([18]). *Let $Z := |f^{-1}(1)|$. If the above algorithm* QC *is run and the auxiliary register is measured, then a value $\tilde{Z}$ satisfying*

$$\left|Z - \tilde{Z}\right| \leq 2\pi\sqrt{Z(2^n - Z)}/q + (2^n \cdot \pi^2)/q^2 \tag{1}$$

is obtained with probability at least 0.8.

The depth to implement QC is typically dominated by that of $\Lambda_q(Q(H^{\otimes n}, f))$, which makes exactly q queries to f. We can show that $\Lambda_q(Q(H^{\otimes n}, f))$ can be implemented on a quantum circuit of depth at most about $q \cdot D_f$ by using n auxiliary qubits, where D_f is the depth to implement the quantum oracle of f. Hence, the depth of QC is also at most about $q \cdot D_f$, and the amount of the auxiliary qubits needed is at most the number of qubits required to compute $\mathsf{Calq}_{n,q}$. See Section B in the appendix for more details.

2.2 LFSR Basics

Let $\mathbb{F}_q$ be a finite field of order q. The LFSR on $\mathbb{F}_q$ of length L with a feedback polynomial $f(x) := c_L x^L + c_{L-1}x^{L-1} + \cdots c_1 x + 1 \in \mathbb{F}_q[x]$ generates an infinite sequence $(s_t)_{t\geq 0}$ in $\mathbb{F}_q$ from an initial state $\boldsymbol{s}^{(0)} = (s_0, \ldots, s_{L-1}) \in \mathbb{F}_q^L$ as

$$s_{t+L} := \sum_{1\leq i\leq L} c_i s_{t+L-i} \text{ for } t \geq 0,$$

maintaining the internal state $\boldsymbol{s}^{(t)} := (s_t, \ldots, s_{t+L-1})$ at time t. The state update can be regarded as a linear map over $\mathbb{F}_q$, and $\boldsymbol{s}^{(t+1)} = \boldsymbol{s}^{(t)} \cdot M$ holds for

$$M := \begin{pmatrix} 0\,0 \cdots 0 & c_L \\ 1\,0 \cdots 0 & c_{L-1} \\ \vdots \quad \ddots \;\vdots & \vdots \\ 0\,0 \cdots 1 & c_1 \end{pmatrix}. \tag{2}$$

A well-known fact is that the period of LFSR sequences and internal states becomes the longest (i.e., $q^L - 1$) when f is a primitive polynomial.

Throughout the paper, we only consider LFSRs whose feedback polynomial f is primitive and assume that q is a power of 2.

The reciprocal polynomial of f is called the *characteristic polynomial* of the LFSR, which we denote by $f^*(x)$ (that is, $f^*(x) = x^L f(1/x)$). As we assume that f is primitive (and thus irreducible), so is f^*. Hence, the quotient ring $\mathfrak{F} := \mathbb{F}_q[x]/(f^*(x))$ becomes a field, which is isomorphic to $\mathbb{F}_q^L$ as vector spaces over $\mathbb{F}_q$. Let $\xi : \mathbb{F}_q^L \to \mathfrak{F}$ be the isomorphism defined by

$$\xi(\boldsymbol{a}) = \sum_{0 \le i \le L-1} a_i \cdot \alpha^i \tag{3}$$

for $\boldsymbol{a} = (a_0, \ldots, a_{L-1}) \in \mathbb{F}_q^L$, where $\alpha := x + (f^*(x)) \in \mathfrak{F}$ is a generator element of $\mathfrak{F}$ over $\mathbb{F}_q$. Since q is assumed to be a power of 2, some straightforward calculations show

$$\xi(\boldsymbol{a} \cdot M^\top) = \xi(\boldsymbol{a}) \cdot \alpha. \tag{4}$$

Since f^* is not only irreducible but also primitive, α is a generator of the multiplicative group $\mathfrak{F}^\times \cong \mathbb{Z}/(q^L - 1)\mathbb{Z}$, and so $\beta \cdot \alpha^i \neq \beta$ holds for arbitrary $\beta \in \mathfrak{F} \setminus \{0\}$ and $i = 1, \ldots, q^L - 2$. From this fact and Eq. (4),

$$\boldsymbol{a} \cdot \left(M^\top\right)^i \neq \boldsymbol{a} \quad \text{for} \quad i = 1, \ldots, q^L - 2 \tag{5}$$

follows for $\boldsymbol{a} \in \mathbb{F}_q^L \setminus \{\boldsymbol{0}\}$.

3 Classical Fast Correlation Attack

This section briefly reviews classical fast correlation attacks related to our results. We focus on so-called one-pass algorithms working using FWHT [25] that can be regarded as a decoding procedure for a binary linear code, as it has been most widely applied. First, we explain the simplest case where LFSR sequences themselves are correlated with keystreams in Section 3.1. Then, Section 3.2 explains how the attack idea is extended to more general cases. Section 3.3 gives a brief summary and a note on the amount of necessary data. Throughout the section, we assume an adversary is given a keystream segment produced from a single pair of a key and an IV. See, e.g., [2, 61, 21], for more details on classical fast correlation attacks.

3.1 Simplest Case

Suppose a stream cipher is built upon a single LFSR of length L over $\mathbb{F}_2$ and we have an N-bit keystream segment $\boldsymbol{z} = (z_0, z_1, \ldots, z_{N-1}) \in \mathbb{F}_2^N$. Our goal is to recover the initial state $\boldsymbol{s}^{(0)} \in \mathbb{F}_2^L$ of the LFSR. Once $\boldsymbol{s}^{(0)}$ is recovered, it is often easy to determine the entire initial state, and even the master secret key is recovered in some cases.

In the simplest case, the fast correlation attack models that the keystream $\boldsymbol{z}$ is obtained by transmitting the LFSR sequence $\boldsymbol{s} = (s_0, \ldots, s_{N-1}) \in \mathbb{F}_2^N$ generated from $\boldsymbol{s}^{(0)}$ through a Binary Symmetric Channel (BSC). Namely, it regards as if $e_i := z_i \oplus s_i$ were an independent random error bit for each i (see Fig. 1), expecting that the error bit sequence $\boldsymbol{e} = (e_0, \ldots, e_{N-1})$ is highly biased. Note that $\boldsymbol{e}$ is biased iff the squared linear correlation $\mathrm{Cor}(\boldsymbol{s}, \boldsymbol{z})^2$ is large. For

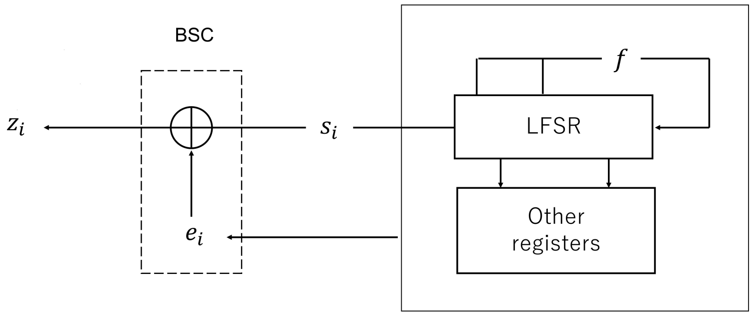

Fig. 1. LFSR-based cipher modeled as a BSC.

ease of explanation, we assume e_i is biased to 0 and (the expected value of) the correlation $c := \mathbf{Ex}_{K,IV}\left[\mathrm{Cor}(\boldsymbol{s}, \boldsymbol{z})\right]$ is close to 1.

In this model, the problem of recovering $\boldsymbol{s}^{(0)}$ from $\boldsymbol{z}$ can be regarded as a decoding problem with respect to a binary linear code. Let G be the binary $L \times N$ matrix of which the i-th column vector is $M^{i-1} \cdot (1, 0, \ldots, 0)^\top$. Then $\boldsymbol{s} = \boldsymbol{s}^{(0)}G$ holds by definition of LFSR. (Multiplication by M corresponds to clocking the LFSR once, and so multiplying by G generates the sequence $\boldsymbol{s} = (s_0, \ldots, s_{N-1})$.) In addition, G is full-rank if N is sufficiently large. Especially, G can be regarded as a generating matrix of an $[N, L]$ binary linear code $\mathcal{C}$, where encoding a message vector corresponds to multiplying G from right. The initial state $\boldsymbol{s}^{(0)}$ corresponds to an original message before the encoding, and the LFSR sequence $\boldsymbol{s} = \boldsymbol{s}^{(0)}G$ to the codeword of $\mathcal{C}$ after the encoding. From this perspective, recovering $\boldsymbol{s}^{(0)}$ from $\boldsymbol{z} = \boldsymbol{s} \oplus \boldsymbol{e}$ is equivalent to correcting errors and recovering the original message.

Concretely, $\boldsymbol{s}^{(0)}$ is recovered by maximum likelihood decoding, which can be realized roughly as follows.

1. For each candidate message $\boldsymbol{x} \in \mathbb{F}_2^L$, compute and store the squared linear correlation $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{z})^2$ between the codeword $\boldsymbol{x}G \in \mathcal{C}$ $(\subset \mathbb{F}_2^N)$ and $\boldsymbol{z}$.
2. If $\boldsymbol{x} = \boldsymbol{s}^{(0)}$, the value $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{z})^2 = \mathrm{Cor}(\boldsymbol{s}, \boldsymbol{z})^2$ will be large by assumption. On the other hand, it will be small for a random $\boldsymbol{x} \neq \boldsymbol{s}^{(0)}$. With this in mind, output $\boldsymbol{x}$ with the largest $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{z})^2$ as the decoding result.

For each $\boldsymbol{x}$, computing $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{z})^2$ requires $O(N)$ operations because we have to check whether $(\boldsymbol{x}G)_i = z_i$ for $i = 0, \ldots, N-1$. Hence this procedure requires $O(2^L \cdot N)$ operations in total. To achieve a high success probability, $N \geq \Omega(L/c^2)$ is necessary due to Shannon's noisy-channel coding theorem, and some statistical analysis shows that $N = O(L/c^2)$ is indeed sufficient (we will elaborate this later in Section 3.3).

By applying the Fast Walsh-Hadamard Transform (FWHT), the decoding complexity drops from $O(N \cdot 2^L)$ to $O(N + L2^L)$. Define a function $\Psi : \mathbb{F}_2^L \to \mathbb{C}$

by

$$\Psi(\boldsymbol{w}) := \sum_{\substack{0 \le i \le N-1: \\ \boldsymbol{w} = \text{the } (i+1)\text{-th column of } G}} (-1)^{z_i}. \tag{6}$$

Compute and store $\Psi(\boldsymbol{w})$ for all $\boldsymbol{w}$, which can be done with $O(N)$ operations and $O(2^L)$ memory. Then, apply the FWHT to compute and store the value $(\mathcal{W}(\Psi))(\boldsymbol{x})$ for all $\boldsymbol{x}$, which requires $O(L2^L)$ operations. Now, some straightforward calculations[3] show

$$(\mathcal{W}(\Psi))(\boldsymbol{x}) = \frac{N}{2^{L/2}} \cdot \mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{z}).$$

Hence, the first step of the aforementioned decoding procedure can be performed with $O(N + L2^L)$ operations.

3.2 More General Cases

Modern stream ciphers are well-designed so that keystreams themselves are not strongly correlated with LFSR sequences, and the above attack does not work. Yet, almost the same idea is applicable if there is another code relating initial states and keystreams.

For instance, suppose

- there are (1) the generating matrix G of an $[N, \ell]$ binary code for some ℓ, (2) a binary sequence $\boldsymbol{\zeta} := (\zeta_0, \dots, \zeta_{N-1})$ computed from a keystream segment, and (3) a value $\boldsymbol{\sigma}^{(0)} \in \mathbb{F}_2^{\ell}$ that is related to the initial value $\boldsymbol{s}^{(0)}$, such that
- (the absolute value of the expected value of) the correlation $c := \big|\mathbf{Ex}_{K,IV}\big[\mathrm{Cor}\left(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta}\right)\big]\big|$ is large.

Then, the aforementioned decoding algorithm with FWHT works in exactly the same way, except that now the decoding algorithm recovers $\boldsymbol{\sigma}^{(0)}$ and the parameters and variables such as L and z_i are replaced with ℓ and ζ_i, etc. The decoding complexity with FWHT becomes $O(N + \ell 2^{\ell})$, and $N \ge \Omega(\ell/c^2)$ is required for a sufficiently high success probability. Once $\boldsymbol{\sigma}^{(0)}$ is recovered, at least the keystream is distinguished from random, and sometimes it is possible to recover the entire initial state and even the master secret key of the cipher.

A typical way to find such an alternative code is to search for a linear approximation between internal states of an LFSR and keystreams. Suppose that an attack target is based on an LFSR of length L over $\mathbb{F}_{2^n}$ for some n, and that the LFSR sequence (resp., keystream) is denoted by $s_0, s_1, \dots \in \mathbb{F}_{2^n}$ (resp., $z_0, z_1, \dots \in \mathbb{F}_{2^n}$). As before, let $\boldsymbol{s}^{(i)} := (s_i, \dots, s_{i+L-1}) \in \mathbb{F}_{2^n}^L$ be the internal state of the LFSR at time i. Assume there are an index set $\mathbf{I}_{\mathrm{lfsr}} \subset \mathbb{Z}_{\ge 0}$ and linear masks $\{\boldsymbol{\Gamma}_j\}_{j \in \mathbf{I}_{\mathrm{lfsr}}} \subset \mathbb{F}_{2^n}^L$ for the LFSR's internal states (resp., an index

[3] By definition of Ψ, $\mathcal{W}$, and Cor, it immediately follows that both sides are equal to $\frac{1}{2^{L/2}} \sum_{w,i} (-1)^{\langle \boldsymbol{x}, \boldsymbol{g}_i \rangle_{\mathbb{F}_2} \oplus z_i} \delta_{w, \boldsymbol{g}_i}$, where $\boldsymbol{g}_i$ is the $(i+1)$-th column of G.

set $\mathbf{I}_{\mathrm{ks}} \subset \mathbb{Z}_{\geq 0}$ and linear masks $\{\boldsymbol{\Lambda}_j\}_{j \in \mathbf{I}_{\mathrm{ks}}} \subset \mathbb{F}_{2^n}$ for keystreams) such that the linear approximation

$$\bigoplus_{j \in \mathbf{I}_{\mathrm{lfsr}}} \langle \boldsymbol{s}^{(i+j)}, \boldsymbol{\Gamma}_j \rangle_{\mathbb{F}_2} \approx \bigoplus_{j \in \mathbf{I}_{\mathrm{ks}}} \langle z_{i+j}, \boldsymbol{\Lambda}_j \rangle_{\mathbb{F}_2} \tag{7}$$

holds with an absolute correlation $c \gg 0$ for every i. Below, we explain how to define G, $\boldsymbol{\zeta}$, and $\boldsymbol{\sigma}^{(0)}$ such that $\left|\mathbf{Ex}_{K,IV}\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta})\right]\right| \approx c$ from the above linear approximation.

Define $\boldsymbol{\Gamma} \in \mathbb{F}_{2^n}^L$ by $\boldsymbol{\Gamma} := \bigoplus_{j \in \mathbf{I}_{\mathrm{lfsr}}} \left(\boldsymbol{\Gamma}_j \cdot \left(M^\top\right)^j\right)$. Then we have

$$\langle \boldsymbol{s}^{(0)}, \boldsymbol{\Gamma} \cdot (M^\top)^i \rangle_{\mathbb{F}_2} = (\text{the left hand side of (7)}).$$

With this in mind, setting $\ell := L \cdot n$ (and identifying $\mathbb{F}_{2^n}^L$ with $\mathbb{F}_2^\ell$), define

- G as the $\ell \times N$ binary matrix of which the i-th column is $M^{i-1} \cdot \boldsymbol{\Gamma}^\top$,
- $\boldsymbol{\sigma}^{(0)} := \boldsymbol{s}^{(0)}$, and
- $\boldsymbol{\zeta} = (\zeta_0, \ldots, \zeta_{N-1})$ by $\zeta_i :=$ (the right hand side of (7)).

Then, Eq. (7) can be rewritten as

$$(\boldsymbol{\sigma}^{(0)}G)_i \approx \zeta_i,$$

which implies $\left|\mathbf{Ex}_{K,IV}\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta})\right]\right| \approx c$.

Usual attacks further convert the above G into another matrix G' to reduce the code's dimension and the decoding complexity, at the cost of a decrease in the (squared) correlation and an increase in the data complexity. This is usually done by solving some k-sum problems with Wagner's k-tree algorithm [85].

3.3 Summary and Note on the Size of N

To mount fast correlation attacks, an attacker first looks for an $\ell \times N$ matrix G, along with a binary sequence $\boldsymbol{\zeta} = (\zeta_0, \ldots, \zeta_{N-1})$ that can be computed from a keystream segment, such that $c := \left|\mathbf{Ex}_{K,IV}\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta})\right]\right|$ is large for some $\boldsymbol{\sigma}^{(0)} \in \mathbb{F}_2^\ell$ that depends on the (secret) initial value $\boldsymbol{s}^{(0)}$ of the LFSR. Here, G is public and the adversary can compute it offline.

Once finding such G, $\boldsymbol{\zeta}$, and $\boldsymbol{\sigma}^{(0)}$, the adversary performs maximum likelihood decoding of $\boldsymbol{\zeta}$ with respect to the $[N, \ell]$ binary linear code of which the generating matrix is G. The decoding can be realized with $O(N + \ell 2^\ell)$ operations as follows.

1. Compute all the values of the function $\Psi(\boldsymbol{z}) := \sum_{\substack{0 \leq i \leq N-1: \\ \boldsymbol{z} = \text{the } (i+1)\text{-th column of } G}} (-1)^{\zeta_i}$ and store them into a memory.
2. Apply the FWHT on $\Psi(\boldsymbol{z})$. Now, the values $(\mathcal{W}(\Psi))(\boldsymbol{x}) = \frac{N}{2^{\ell/2}} \cdot \mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})$ are stored in the memory for all $\boldsymbol{x}$.
3. Output $\boldsymbol{x}$ such that $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2$ is significantly larger than others.

G and $\boldsymbol{\zeta}$ are typically derived from linear approximations between LFSR sequences and keystreams.

Very roughly and intuitively, $\boldsymbol{\sigma}^{(0)}$ corresponds to (a linear transformation of) the initial state of LFSR, $\boldsymbol{\sigma}^{(0)}G$ to the output sequence of LFSR, and $\boldsymbol{\zeta}$ to the key stream. If $\boldsymbol{\zeta}$ is linearly approximated by $\boldsymbol{\sigma}^{(0)}G$, then $\boldsymbol{\zeta}$ can be regarded as the result of encoding $\boldsymbol{\sigma}^{(0)}$ with a code corresponding to G and sending through a noisy channel. Hence, $\boldsymbol{\sigma}^{(0)}$ (and thus the initial state of the LFSR) can be recovered by the most likelihood decoding, using FWHT as above.

About the Size of $\boldsymbol{N}$. Here we explain why $N = O(\ell/c^2)$ is sufficient to achieve a large success probability. Let us call $\boldsymbol{\sigma}^{(0)}$ the correct decoding results, and $\boldsymbol{x} \in \mathbb{F}_2^\ell$ such that $\boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)}$ incorrect decoding results. We heuristically assume that, for an incorrect $\boldsymbol{x}$, the correlation $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})$ is approximated by the linear correlation of two random binary sequences of length N, as done in classical attacks. Then the following claim holds.

Claim. Suppose $N \geq 8\ell/c^2$ and $\ell \geq 1$. Then we have

$$\Pr_{K,IV}\left[\text{There is } \boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)} \text{ such that } \mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \geq c^2/4\right] \lessapprox (2/e)^\ell, \tag{8}$$

$$\Pr_{K,IV}\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta})^2 \geq c^2/2\right] \gtrapprox 0.95. \tag{9}$$

Especially, the decoding algorithm succeeds with a sufficiently high probability.

See Section C in the appendix for why it is plausible to regard that this claim holds.

4 Quantam Fast Correlation Attack in the Q1 Model

This section studies quantum speed-up of the decoding procedure of fast correlation attacks with the FWHT in the Q1 model. In fact, it later turns out that it seems hard to achieve a fast correlation attack that is faster than the Grover search in the Q1 model by speeding-up existing classical attacks. Yet, we show a Q1 algorithm here to make it the starting point of a more complex Q2 attack in Section 6, and to see why achieving a meaningful speed-up of existing classical fast correlation attacks seems hard in Q1.

As in the classical setting, we assume that a keystream segment generated from a single key and IV pair is given to an adversary. We consider the general cases reviewed in Section 3.2, and use the same notations.

Below, we first describe a rough idea of the quantum attack in Section 4.1, and then provide the formal details in Section 4.2. Section 4.3 provides discussions on applications and some observations.

4.1 Overview and Rough Idea

Our idea is to perform quantum analogue of the operations in the classical decoding procedure in a natural way.

- We first prepare the quantum counter part of the function $\varPsi$, namely the quantum state
$$|\psi\rangle := \sum_{\boldsymbol{w}\in\mathbb{F}_2^\ell} \frac{\varPsi(\boldsymbol{w})}{\sqrt{\sum_{\boldsymbol{w}} |\varPsi(\boldsymbol{w})|^2}} |\boldsymbol{w}\rangle . \tag{10}$$
How we can prepare $|\psi\rangle$ is a non-trivial question, but we show that a unitary operator U satisfying $U|0^\ell\rangle = |\psi\rangle$ can be realized as an efficient quantum algorithm, given that some data are precomputed and stored in qRAM in advance.
- Second, we apply the Hadamard tranform on the entire state. Since the Walsh-Hadamard transform on classical functions is mathematically the same as the Hadamard transform on quantum states, we get
$$H^{\otimes\ell}|\psi\rangle = \sum_{\boldsymbol{x}\in\mathbb{F}_2^\ell} \frac{(\mathcal{W}(\varPsi))(\boldsymbol{x})}{\sqrt{\sum_{\boldsymbol{w}} |\varPsi(\boldsymbol{w})|^2}} |\boldsymbol{x}\rangle = \sum_{\boldsymbol{x}\in\mathbb{F}_2^\ell} \frac{N\cdot \mathrm{Cor}(\boldsymbol{x}G,\boldsymbol{\zeta})}{\sqrt{\sum_{\boldsymbol{w}} |\varPsi(\boldsymbol{w})|^2}\cdot 2^{\ell/2}} |\boldsymbol{x}\rangle . \tag{11}$$
Measuring this state, we obtain an $\boldsymbol{x}$ with a probability proportional to the squared correlation $\mathrm{Cor}(\boldsymbol{x}G,\boldsymbol{\zeta})^2$. Namely, we will obtain the correct decoding result $\boldsymbol{\sigma}^{(0)}$ with a higher probability than incorrect results. However, the probability to obtain $\boldsymbol{\sigma}^{(0)}$ is usually still too small.
- Thus, we amplify the probability of obtaining a correct result with QAA. To apply QAA, we must implement a unitary operator computing the Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ such that $f(\boldsymbol{x}) = 1$ iff $\boldsymbol{x} = \boldsymbol{\sigma}^{(0)}$. How to choose and implement f can depend on the internal structure of the target cipher.

4.2 Formal Details

First, we explain some precomputaiton required for later steps. Second, we show how to prepare the state $|\psi\rangle$ in Eq. (10). Third, we provide a formal description and analysis of the entire attack algorithm.

We denote the i-th column vector of G by $\boldsymbol{g}_i$, and define $\mu := \max_{\boldsymbol{x}\in\mathbb{F}_2^\ell} \#\{i : \boldsymbol{g}_i = \boldsymbol{x}\}$. We suppose that $\left|\mathbf{Ex}_{K,IV}\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G,\boldsymbol{\zeta})\right]\right| = c$ for some $c \gg 0$ and $8\ell/c^2 \leq N \leq 2^\ell$.

Precomputation. Given sufficient amount of keystream bits, we first compute $\boldsymbol{\zeta} = (\zeta_0,\ldots,\zeta_{N-1})$ and store them into qRAM. Then, we compute $\boldsymbol{g}_i$ and store the data $(i,\boldsymbol{g}_i)$ into a list in a sequential order for all i. Along with $\boldsymbol{g}_i$, store the information of how many times the value $\boldsymbol{g}_i$ has appeared before. That is, store a counter set to be 0 if $\boldsymbol{g}_j \neq \boldsymbol{g}_i$ for all $j < i$, and increment it to 1 if there is unique $j < i$ such that $\boldsymbol{g}_j = \boldsymbol{g}_i$, and so on. (Eventually, each entry of the list has the form $(i,\boldsymbol{g}_i,ctr_i)$.) Then, store the list into qRAM.

Note that $0 \leq ctr_i \leq \mu - 1$ for all i, and the value ctr_i is represented as a $\log \mu$ bit string. If $\boldsymbol{g}_i$ can be computed with $O(1)$ operations for each i, this precomputation can be completed with $O(N \log N)$ operations.

How to Prepare $|\psi\rangle$. We implement a unitary U satisfying $U |0^n\rangle = |\psi\rangle$ as the following algorithm[4].

Algorithm PREP1.

1. Create the superposition $\sum_{0 \leq i \leq N-1} \sqrt{1/N} |i\rangle$.
2. Access qRAM to obtain $\sum_{0 \leq i \leq N-1} \sqrt{1/N} |i\rangle |\boldsymbol{g}_i\rangle |ctr_i\rangle$. (By abuse of notation, we denote $\boldsymbol{g}_N$ by $\boldsymbol{g}_0$.)
3. Multiply each basis state by the phase $(-1)^{\zeta_i}$ by accessing qRAM. Now the state is $\sum_{0 \leq i \leq N-1} \sqrt{1/N}(-1)^{\zeta_i} |i\rangle |\boldsymbol{g}_i\rangle |ctr_i\rangle$.
4. Set the leftmost register to $|0^\ell\rangle$. This is done by searching for the tuple $(\boldsymbol{g}_i, ctr_i)$ in qRAM and adding the corresponding index to the first register. The resulting state is $|0^\ell\rangle \sum_{0 \leq i \leq N-1} \sqrt{1/N}(-1)^{\zeta_i} |\boldsymbol{g}_i\rangle |ctr_i\rangle$.
5. Apply the Hadamard gates to the rightmost register. Some calculations show that the resulting state is

$$\frac{1}{\sqrt{N}\sqrt{\mu}} |0^\ell\rangle \left(\sum_{\boldsymbol{w}} \Psi(\boldsymbol{w}) |\boldsymbol{w}\rangle \right) |0^{\log \mu}\rangle + |\varepsilon\rangle , \tag{12}$$

   where the third register of $|\varepsilon\rangle$ is orthogonal to $|0^{\log \mu}\rangle$.
6. Apply QAA (that returns a correct answer with certainty) on (12) to amplify the state of which the third register is $0^{\log \mu}$. This can be done by performing Steps 1-5 and their uncomputations at most $p_{\text{init}}^{-1/2}$ times each in total, where $p_{\text{init}} := \sum_{\boldsymbol{w}} |\Psi(\boldsymbol{w})|^2 / N\mu$.

The complexity of PREP1 depends on G and $\boldsymbol{\zeta}$ but it is small in most cases.

For example, suppose $\boldsymbol{g}_i \neq \boldsymbol{g}_j$ holds for $i \neq j$. Then $\mu = 1$ and $\sum_{\boldsymbol{w}} |\Psi(\boldsymbol{w})|^2 = N$ hold, which implies $p_{\text{init}} = 1$. In particular, QAA is actually not necessary and a single execution of Steps 1-5 is sufficient to prepare $|\psi\rangle$.

Even if $\boldsymbol{\zeta}$ and each $\boldsymbol{g}_i$ are random, the number of QAA iterations in Step 6 is at most $O(\ell / \log \ell)$ on average: Since $\boldsymbol{g}_i$ is random, $\mu \leq \ell / \log \ell$ holds with an overwhelming probability by the standard balls-into-bins arguments [66, Lem. 5.12]. In addition, the value $|\Psi(\boldsymbol{w})|^2 = \left| \sum_{i : \boldsymbol{g}_i = \boldsymbol{w}} (-1)^{\zeta_i} \right|^2$ is always a non-negative integer, and $\Pr\left[|\Psi(\boldsymbol{w})|^2 \neq 0 \right] \geq 1/2$ holds for each $\boldsymbol{w}$ because $\boldsymbol{\zeta}$ is random. Hence, the expected value $\sum_{\boldsymbol{w}} |\Psi(\boldsymbol{w})|^2$ is at least $\#\{\boldsymbol{w} : \Psi(\boldsymbol{w}) \neq 0\} \times (1/2) \geq N/2\mu$, and $p_{\text{init}} = \sum_{\boldsymbol{w}} |\Psi(\boldsymbol{w})|^2 / N\mu \geq 1/(2\mu^2) \gtrapprox (\log \ell)^2 / (2\ell^2)$ on average.

[4] The idea of applying Hadamrd to the rightmost register in Step 4 and then using QAA is inspired from the state preparation technique by Sanders et al [73].

The Entire Algorithm and Analysis. Our Q1 attack runs as follows.

Algorithm QFCA1.

1. Get a keystream segment long enough to mount the attack.
2. Perform the precomputation described on p.15.
3. Run QAAw/oKp on p.8 with $U := H^{\otimes \ell} \cdot \mathsf{PREP1}$ on the Boolean function $f : \mathbb{F}_2^\ell \to \mathbb{F}_2$ such that $f(\boldsymbol{x}) = 1$ iff $\boldsymbol{x} = \boldsymbol{\sigma}^{(0)}$. (How to compute f depends on attack targets.)

Let p be the probability that we obtain an $\boldsymbol{x}$ satisfying $f(\boldsymbol{x}) = 1$ when we measure the state $H^{\otimes \ell} \cdot \mathsf{PREP1}\,|0^\ell\rangle$ $(= H^{\otimes \ell}\,|\psi\rangle)$. That is,

$$p = \frac{N^2 \cdot \mathrm{Cor}(\boldsymbol{\sigma}^{(0)} G, \boldsymbol{\zeta})^2}{2^\ell \cdot \sum_{\boldsymbol{w}} |\Psi(\boldsymbol{w})|^2}. \tag{13}$$

By the claim at the end of Section 3.3, with probability at least 0.9 (when K and IV are randomly chosen and ℓ is sufficiently large, e.g., $\ell \geq 10$), $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \leq c^2/4$ holds for all $\boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)}$ and $\mathrm{Cor}(\boldsymbol{\sigma}^{(0)} G, \boldsymbol{\zeta})^2 \geq c^2/2$ holds. Provided these inequalities really hold,

$$p \geq \frac{N^2 c^2}{2^{\ell+1} \cdot \sum_{\boldsymbol{w}} |\Psi(\boldsymbol{w})|^2} \tag{14}$$

holds, and the attack finds the correct decoding result $\boldsymbol{\sigma}^{(0)}$ in expected time complexity at most about

$$T_{\mathrm{total}} = T_{\mathrm{precomp}} + (9/2)p^{-1/2} \left(2 \cdot T_{\mathrm{prepare}} + T_f\right), \tag{15}$$

where T_{prepare} (resp., T_f) is the running time of the algorithm PREP1 (resp., the time complexity required to compute f). In addition, T_{precomp} is the time complexity to collect necessary data and perform the precomputation. How to compute f depends on the internal structure of the target cipher.

Typically, we have $T_{\mathrm{prepare}} \ll T_f$, $T_{\mathrm{precomp}} = O(N)$, and $\sum_{\boldsymbol{w}} |\Psi(\boldsymbol{w})|^2 = O(N)$, when the complexity (15) becomes roughly about $(N + T_f \cdot \frac{2^{\ell/2}}{\sqrt{Nc^2}})$. Balancing the two terms, we obtain

$$N = 2^{\ell/3} \cdot (T_f/c)^{2/3}. \tag{16}$$

In summary, with probability at least 0.9 (on the randomness of K and IV), the attack recovers $\boldsymbol{\sigma}^{(0)}$ in expected time complexity $2^{\ell/3+1} \cdot (T_f/c)^{2/3}$.

4.3 Discussions and Observations

The above algorithm QFCA1 is a very natural extension of classical fast correlation attacks. By applying QFCA1 to speed up existing classical fast correlation attacks, we expected to achieve quantum attacks faster than the Grover search However, we have not obtained a meaningful speedup so far with this approach.

One reason is that the absolute correlations in some classical attacks are too small. For instance, the attack on Grain v1 by Todo et al. [83] utilizes linear approximations of absolute correlation $c = 2^{-36}$, while both the LFSR and key lengths of Grain v1 are 80. If a single linear approximation is used, we need the data complexity at least $c^{-2} \geq 2^{72}$, which is much larger than the exhaustive key search with Grover's algorithm. The data complexity (and time complexity possibly also) decreases to some extent by using multiple approximations, but we find it still hard to achieve an attack faster than the Grover search.

Another reason is that the LFSR length is quite large in some ciphers. For instance, SNOW 2.0 [31] is based on a 512-bit LFSR. As explained around Eq. (16), we will have a factor of order $2^{\ell/2}$ or $2^{\ell/3}$ in the time complexity, which is too large when $\ell = 512$. Classical attacks reduce the dimension of the code (i.e., the parameter ℓ) by solving k-sum problems. However, in the Q1 setting, we observe that the cost to solve k-sum problems sufficiently reducing the dimension is too heavy (even with the dedicated quantum algorithms [42, 69, 75]) compared to the quantum exhaustive key search with Grover's algorithm when k is small (e.g., $k = 2$), and the correlations after dimension reduction become too small when k is large (e.g., $k = 4$).

Due to these reasons, we suspect it is quite hard to mount a fast correlation attack that is faster than the generic attack in the Q1 setting, or more fairly non-trivial techniques will be required. Given this situation, we next focus on Q2 attacks.

Remark 1. The state $H^{\otimes \ell} \left|\psi\right\rangle$ in Eq. (11) is a superposition of candidate messages $\left|\boldsymbol{x}\right\rangle$ and the quantum amplitude is proportional to the correlation $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})$ and so can be regarded as an analogy of the *correlation state* in Schrottenloher's quantum linear key recovery. Still, our technique and Schrottenloher's are quite different. Unlike the preparation of the correlation state, convolutions are not computed in preparing $H^{\otimes \ell} \left|\psi\right\rangle$. Moreover, in the Q2 attack in Section 6, we will use Shor's algorithm to efficiently prepare a state corresponding to $H^{\otimes \ell} \left|\psi\right\rangle$.

Remark 2. Measuring the state (11), we obtain $\boldsymbol{x}$ with a probability proportional to $\left|\mathcal{W}(\Psi)(\boldsymbol{x})\right|^2$. This can be regarded as a random sampling according to the distribution induced by $\mathcal{W}(\Psi)$. A possible alternative approach could be to iteratively perform this sampling and estimate the values $\left|\mathcal{W}(\Psi)(\boldsymbol{x})\right|^2$ instead of applying QAA, but so far, we have not found more efficient attacks with this idea. Leveraging such random samplings in a better way could be a possible future research to investigate.

Remark 3. The attacks in this section essentially rely on linear approximations of which the linear masks cover the entire state of LFSR. If one were to use masks that only cover part of LFSR (as done in, e.g., [8]), the attacks would proceed by applying our method for the part covered by the linear masks and guessing the remaining part with the Grover search. The same thing holds true for the Q2 attacks shown later.

5 New Attack Model and Security Definition in Q2

This section introduces a new attack model and a security definition for stream ciphers in the Q2 setting.

When studying Q2 attacks, we must carefully consider which attack breaks what security notion. This is because the Q2 setting allows adversaries to perform operations that were never anticipated when some classical security notions were defined, e.g., querying all the messages at once in superposition. Let us briefly illustrate this with attacks on MACs as an example. A classical attack on a deterministic MAC is considered meaningful if it forges a valid tag for a message that has not been queried by the adversary. Meanwhile, Q2 attacks are typically allowed to query all messages simultaneously in quantum superposition. This makes it unclear how we should interpret the meaning of a Q2 attack on a MAC if it produces several valid message-tag pairs after making a single quantum query consisting of exponentially many messages in superposition. The seminal work by Kaplan et al. [51] carefully addresses this issue and demonstrates that (some of) their attacks on MACs are valid in that they break Boneh and Zhandry's EUF-qCMA security [12].

As we will see below, there is also a subtle issue regarding Q2 attacks on stream ciphers. In what follows, We denote a random function by RF, of which the domain and range will be clear from the context.

Classical Security Notion: IV-Based Stream Ciphers as PRFs. As shown by Berbain and Gilbert [7], the classical security definition appropriate for IV-based stream ciphers is the Pseudo-Random Function (PRF) security. Here, stream ciphers are regarded as keyed functions $\mathsf{SC} : \mathbb{F}_2^{\kappa} \times \mathbb{F}_2^{iv} \to \mathbb{F}_2^{D}$ that take key and IV as input and return a keystream of length D for some $D \gg 1$. Recall that the PRF advantage of an oracle-aided algorithm $\mathcal{A}$ for SC is defined as

$$\mathsf{Adv}_{\mathsf{SC}}^{\mathrm{PRF}}(\mathcal{A}) := \left|\Pr\left[\mathcal{A}^{\mathsf{SC}_K} \text{ outputs } 1\right] - \Pr\left[\mathcal{A}^{\mathsf{RF}} \text{ outputs } 1\right]\right|, \tag{17}$$

where the probability is taken over both the randomness of $\mathcal{A}$ and the choice of the secret key K or the random function RF. The ciphers are considered secure iff no adversary $\mathcal{A}$ with reasonable computational resources can distinguish SC and RF with a non-negligible advantage.

qPRF Security and Some Issues. The counterpart of the PRF security in the Q2 setting is the quantum pseudo-random function (PRF) security by Zhandry [90], where the oracle of the keyed function and the random function are replaced with the corresponding quantum oracles that accept inputs and returns outputs in quantum superposition. Thus, to choose a security definition for stream ciphers in the Q2 setting, the easiest way is simply to adapt the qPRF security.

However, the qPRF security does not mesh well with stream ciphers. Typical Q2 attacks assume a moderate (polynomial) size quantum computer with qRAM, whereas the quantum oracle of stream ciphers returns an exponentially long output in quantum superposition all at once. In other words, a quantum computer of a moderate (e.g., 2^{20}) size has a register of a very large (e.g., 2^{60}) size to receive outputs from the oracle, which is quite unbalanced. A potential solution to this problem is to limit the output length of oracles, but this overlooks one of the primary features of stream ciphers, which is that a long keystream can be generated from a single IV. An alternative solution could be to assume that the oracle's outputs are written into qRAM, but this approach would require a substantial amount of operations for adversaries just to read the oracle's outputs. It undermines the meaning of studying Q2 attacks because unexpectedly efficient and intriguing Q2 attacks are usually achieved by efficiently processing a superposition of many outputs from an oracle.

qBPRF Security. To remedy this, we introduce a new security definition, which we call the *quantum Booleanized PRF security*, or qBPRF security for short.

First, let us define the *Booleanization* of a function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ as the Boolean function $BF : \mathbb{F}_2^n \times \mathbb{F}^{\log m} \to \mathbb{F}_2$ such that $BF(x,i) = (F(x))_i$, and the quantum Boolianized PRF (qBPRF) security as the qPRF security of BF.

We define the qBPRF advantage of an algorithm $\mathcal{A}$ for a stream cipher $\mathsf{SC} : F_2^\kappa \times \mathbb{F}_2^{iv} \to \mathbb{F}_2^D$ as the qPRF advantage of its Booleanization, namely,

$$\mathsf{Adv}^{\mathrm{qBPRF}}_{\mathsf{SC}}(\mathcal{A}) := \left(\mathsf{Adv}^{\mathrm{qPRF}}_{B\mathsf{SC}}(\mathcal{A}) =\right) \left|\Pr\left[\mathcal{A}^{B\mathsf{SC}_K} \text{ outputs } 1\right] - \Pr\left[\mathcal{A}^{\mathsf{RF}} \text{ outputs } 1\right]\right|,$$

where $\mathcal{A}$ is allowed to make quantum queries to the oracles. We say that the stream cipher SC is qBPRF-secure if no adversary faster than the generic attack can have a non-negligible qBPRF advantage.

Put differently, we regard an attack on the cipher breaks its qBPRF security if its computational cost is less than the generic attack with Grover's algorithm while the qBPRF advantage is close to 1, and we aim to find such (fast correlation) attacks in the next section. In particular, we assume that the quantum oracle of the Booleanized version of the target cipher is given to an adversary. By considering the Booleanized versions, we can keep the output length of the oracle short while taking long keystreams into account, addressing the aforementioned issues. To prevent trivial attacks, we set an appropriate limit on D, of which the details will be discussed later.

Feasibility. The attack model in the definition of qBPRF security is quite strong because it essentially assumes an adversary can query not only IVs but also indices for keystream bits in quantum superposition. Yet, we argue that qBPRF security is worth studying and feasible in that some stream ciphers based on the CTR mode, e.g., (some members of) Salsa20 [10] and ChaCha [9] families, seem to achieve it. Below, we explain this by showing a security reduction.

Let $F : \mathbb{F}_2^\kappa \times \mathbb{F}_2^{\mathrm{iv}} \times \mathbb{F}_2^{\mathrm{ctr}} \to \mathbb{F}_2^m$ be a keyed function where $\mathbb{F}_2^\kappa$ is the key space, and $D' \gg 1$ be a parameter. Let $\mathsf{CTR}^F : \mathbb{F}_2^\kappa \times \mathbb{F}_2^{\mathrm{iv}} \to \mathbb{F}_2^{D' \cdot m}$ be the stream cipher generating a keystream segment as

$$\mathsf{CTR}^F_K(IV) := F_K(IV, 0)||F_K(IV, 1)|| \cdots ||F_K(IV, D')$$

Then, the following proposition holds.

Proposition 4. *Suppose $D' < 2^{\mathrm{iv}}$. For any quantum algorithm $\mathcal{A}$ making q queries, there is another quantum algorithm $\mathcal{B}$ making q quantum queries such that*

$$\mathsf{Adv}^{\mathrm{qBPRF}}_{\mathsf{CTR}^F}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{qPRF}}_{F}(\mathcal{B}),$$

where the time, memory complexity, and qubits required to run $\mathcal{B}$ are at most $O(1)$ times larger than those for $\mathcal{A}$.

Proof. We construct $\mathcal{B}$ so that it simply emulates the oracle for $\mathcal{A}$ by using the one given to itself. That is, when $\mathcal{A}$ queries a pair (IV, i), $\mathcal{B}$ queries $(IV, \lfloor i/m \rfloor)$ to its own oracle, truncating the response y from the oracle and sending the $(i - m \cdot \lfloor i/m \rfloor)$-th bit of y to $\mathcal{A}$. (Note that arbitrary bit of y can be computed in quantum superposition by making only a single query to $\mathcal{B}$'s oracle [46].) Using this $\mathcal{B}$, the claim of the proposition obviously holds. □

Roughly speaking, the above proposition states that CTR^F is qBPRF-secure as long as there is no attack breaking the qPRF security of F (as long as $D' < 2^{\mathrm{iv}}$). ChaCha and Salsa20 families adapt the structure of the above CTR for some F, and there have been reported no Q2 attacks distinguishing their underlying function F faster than the Grover search. Therefore, some of these ciphers, including Salsa20/12, Salsa20/20, and ChaCha20, will likely achieve the qBPRF security[5].

Upper Limit of Keystream Bit Index. When studying attacks to break the qBPRF security of a stream cipher $\mathsf{SC} : F_2^{\kappa} \times \mathbb{F}_2^{iv} \to \mathbb{F}_2^{D}$, we must set an appropriate upper limit for the keystream bit index, i.e., the parameter D, to prevent trivial attacks. For example, if the $(i + j)$-th bit of each keystream is always equal to the i-th bit for some exponentially large j, the parameter D should be less than j. Otherwise, an adversary can efficiently break the qBPRF security by, e.g., getting the first κ bits and $(j+1)$-th, …, $(j+\kappa)$-th bits of a keystream and check whether they are equal.

When studying LFSR-based stream ciphers, we set D to be the period of the underlying LFSR, which is $2^{\ell} - 1$ if LFSR's bit length is ℓ. This may exceed data limits specified by the designers of a target cipher. Still, considering that even in the classical setting, the first step is to show an attack exceeding the designers' limit (e.g., [78]), we set the limit D as large as possible in the quantum setting.

[5] Some members of the families, e.g., Salsa20/8, have already been broken in the classical setting [4], but we are unsure whether they can be converted into a Q2 attack faster than the Grover search.

Remarks. The oracle of the Booleanized version of a stream cipher enables an adversary to efficiently get the i-th bit of a keystream for arbitrarily large i (as long as i is smaller than an appropriately set upper limit). Some readers may be concerned that such oracles may significantly speed up some attacks even in the *classical* setting. Indeed, if such a classical oracle is available, the *data* complexity of some classical fast correlation attacks will be reduced to some extent because only specific bits of keystream segments are used [84]. However, we expect that the *time* complexity of fast correlation attacks will not be significantly affected because the decoding algorithms do not care much whether the size of the indices i involved in decoding procedures is large or not.

Our primary objective of studying attacks on qBPRF security is to uncover interesting properties and deepen our understanding of the power of Q2 attacks. We do not claim that the practical security of a scheme is affected, even if we discover an efficient attack that only compromises the qBPRF security. We argue that it is worthwhile to study attacks on qBPRF security because we can obtain an interesting new type of large quantum speed-up for fast correlation attacks on some LFSR-based stream ciphers, while some other stream ciphers including Salsa20/12, Salsa20/20, and ChaCha20 are almost completely intact, as we showed around Proposition 4.

Often, quantum attacks breaking a rather theoretical security notion do not immediately imply attacks that compromise more practical security notions. Still, some of such attacks later have become the indispensable basis of other attacks with much more practical implications. For example, the Q2 attacks on the Even-Mansour and FX constructions [55, 56] paved the way for the technique to exponentially reduce memory complexity in some Q1 attacks by using Simon's algorithm [14] and achieving a more-than-quadratic speed-up in the Q1 model [16]. The subsequent sections present attacks on the qBPRF security, hoping they will serve as the foundation for even more impactful attacks.

6 Quantum Fast Correlation Attack in the Q2 Model

6.1 Overview and Rough Idea

When mounting fast correlation attacks in the Q2 model, we aim to break the qBPRF security of a target stream cipher, assuming that the Booleanized version of the cipher is given as a quantum oracle.

The oracle allows an adversary to query IVs in quantum superposition as well as indices of keystreams, but we fix an arbitrarily chosen IV through an attack like in the classical and Q1 settings. Namely, the primary goal of the attack is to recover the initial state of an LFSR for a single pair of the key and an IV, and we make superposition queries only for keystream indices. The basic idea of the attack is the same as in Section 4.1. That is, we (i) prepare the quantum state $|\psi\rangle$ of Eq. (10), (ii) apply the Hadamard transform on $|\psi\rangle$, and then (iii) apply QAA to amplify the quantum amplitude of the correct decoding result $|\boldsymbol{\sigma}^{(0)}\rangle$.

The difference is as follows.

- We focus on decoding problems derived from a single linear approximation as explained below Eq. (7). In particular, the parameter ℓ is equal to the bit length of the LFSR, G is an matrix of which the i-th column is $M^{i-1} \cdot \boldsymbol{\Gamma}$ for some $\boldsymbol{\Gamma}$ (recall that M is the LFSR's state update matrix) and a decoding algorithm returns $\boldsymbol{\sigma}^{(0)} = \boldsymbol{s}^{(0)}$, the initial state of the LFSR.
- We set the parameter N (the number of columns of G) to be $2^{\ell} - 1$, the period of the LFSR. This implies that G is an $\ell \times (2^{\ell} - 1)$ matrix over $\mathbb{F}_2$. At first glance, it might seem that this would make the preparation of $|\psi\rangle$ prohibitively costly. However, our core observation is that $|\psi\rangle$ can be prepared quite efficiently by regarding the state update of LFSR as multiplication in a finite field and applying Shor's algorithm for the discrete logarithm problem to find the index i satisfying $\boldsymbol{g}_i = \boldsymbol{x}$ for a given $\boldsymbol{x}$. (The main reason for only focusing on G derived from a single linear approximation is to utilize this technique.)
- The Boolean function $f(\boldsymbol{x})$ is computed by checking whether the value $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2$ is above a certain threshold by using the quantum counting algorithm, like Kaplan et al. [52] did for quantum linear distinguishers[6]. We do not use methods depending on the structure of target ciphers because the method with the quantum counting algorithm is the most efficient for all the applications we have found so far.

As the structure of G is restricted and we compute f independently from the structure of target ciphers, our Q2 attack can be formulated as an algorithm to solve the following general problem.

Problem 1. Let M be the state update matrix (Eq. (2)) of an LFSR of length L over $\mathbb{F}_{2^n}$ and G be an $\ell \times (2^{\ell} - 1)$ matrix over $\mathbb{F}_2$ of which the i-th column vector is $M^{i-1} \cdot \boldsymbol{\Gamma}^{\top}$ for some $\boldsymbol{\Gamma} \in \mathbb{F}_2^{\ell} (= \mathbb{F}_{2^n}^{L})$, where $\ell := n \cdot L$. Let $\boldsymbol{s}^{(0)}$ be a vector in $\mathbb{F}_2^{\ell}$ and $\boldsymbol{\zeta}$ be a binary sequence of length $2^{\ell} - 1$ defined as $\boldsymbol{\zeta} := (\boldsymbol{s}^{(0)}G) \oplus \boldsymbol{e}$, where the bits of $\boldsymbol{e} = (e_0, \ldots, e_{2^{\ell}-2})$ are independently and randomly chosen in such a way that $\Pr[e_i = 1] = p$ for all i, with $p \approx (1 + c)/2$ or $(1 - c)/2$ for some $c > 0$. Given the quantum oracle of the Boolean function that returns ζ_i on an input $i \in \{0, \ldots, 2^{\ell} - 2\}$, compute $\boldsymbol{s}^{(0)}$.

The next subsection presents a quantum algorithm to solve this problem.

6.2 Formal Details

We first explain how to compute f, and then show an algorithm to prepare $|\psi\rangle$. After that, we describe the entire algorithm and provide analysis.

We denote the i-th column vector of G by $\boldsymbol{g}_i$ as before and put $N := 2^{\ell} - 1 (= 2^{nL} - 1)$. Note that $\boldsymbol{g}_i \neq \boldsymbol{g}_j$ for $i \neq j$ and that $\{\boldsymbol{g}_i\}_{1 \leq i \leq N} = \mathbb{F}_2^{\ell} \backslash \{\mathbf{0}\} \left(= \mathbb{F}_2^{nL} \setminus \{\mathbf{0}\}\right)$

[6] Saying it differently, we prepare a superposition of $|\boldsymbol{x}\rangle$ with the amplitude being proportional to the correlation $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})$, and then amplify the "good" $\boldsymbol{x}$ by computing the correlation again and checking whether it is large enough. The idea of using a single value for both preparation and amplification is not new and has already appeared in, e.g., [5].

follows from the definition of G and Eq. (5). In particular, we have that $\Psi(\boldsymbol{g}_i) = (-1)^{\zeta_i}$ for all i and $\sum_{\boldsymbol{w}\neq\boldsymbol{0}} |\Psi(\boldsymbol{w})|^2 = 2^\ell - 1 (= N)$, which implies

$$|\psi\rangle = \sum_{1\leq i\leq N} \frac{(-1)^{\zeta_{i-1}}}{\sqrt{N}} |\boldsymbol{g}_i\rangle .$$

In what follows, we use these properties without any mention.

Computing a Boolean Function for QAA by Quantum Counting. Here we explain how to compute $f(\boldsymbol{x})$ such that $f(\boldsymbol{x}) = 1$ iff $\boldsymbol{x} = \boldsymbol{s}^{(0)}$.

Define a Boolean function f' by $f'(\boldsymbol{x}) = 1$ iff $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \geq 3c^2/8$. Then, the claim at the end of Section 3.3 ensures $\Pr_{K,IV}[f(\boldsymbol{x}) = f'(\boldsymbol{x})$ for all $\boldsymbol{x}] \geq 0.9$.

With this in mind, we implement the unitary operator $\mathcal{S}_{f'}$ satisfying $\mathcal{S}_{f'} |\boldsymbol{x}\rangle = (-1)^{f'(x)} |\boldsymbol{x}\rangle$. To implement this, we count the number of i satisfying $(\boldsymbol{x}G)_i = \zeta_i$ for each $i = 0, 1, \ldots, N-1$ by using the quantum counting algorithm with a sufficiently high precision. Proposition 3 ensures that the error probability of the quantum counting algorithm is as small as 0.2, but this is still large if the algorithm is used in QAA as a subroutine. To make the error probability small enough, we run multiple instances and perform a majority vote.

Concretely, to implement $\mathcal{S}_{f'}$, we run the following algorithm. Here, r is a parameter fixed later, and $h_{\boldsymbol{x}} : \{0, \ldots, N\} \to \mathbb{F}_2$ is the Boolean function defined[7] by $h_{\boldsymbol{x}}(i) = (\boldsymbol{x}G)_i \oplus \zeta_i \oplus 1$ for $0 \leq i \leq N-1$ and $h_{\boldsymbol{x}}(N) = 0$.

Algorithm JDG.

```
0. (Assume a basis state $|\boldsymbol{x}\rangle$ is given as an input.)
1. For $j = 1, \ldots, r$, perform the following procedure.
   (a) Run the quantum counting algorithm (QC on page 9) with $q = 2^7/c$ to
       compute an estimation of $Z := |h_{\boldsymbol{x}}^{-1}(1)|$. Let $\tilde{Z}_j$ be the resulting output.
   (b) Let $\tilde{C}_j := \frac{2\tilde{Z}_j - N}{N}$. Compute $(\tilde{C}_j)^2$ and write it into a new auxiliary
       register.
   (c) Uncompute Step (a).
2. Check whether at least $r/2$ values among $(\tilde{C}_1)^2, \ldots, (\tilde{C}_r)^2$ are greater than or
   equal to $3c^2/8$. If so, multiply the entire state by the phase $(-1)$. Otherwise,
   do nothing.
3. Uncompute Step 1.
```

Proposition 5. *Assume $\ell \geq 10$, $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \leq c^2/4$ holds for all $\boldsymbol{x} \neq \boldsymbol{s}^{(0)}$, and $\mathrm{Cor}(\boldsymbol{s}^{(0)}G, \boldsymbol{\zeta})^2 \geq c^2/2$. By abuse of notation, let* JDG *also denote the unitary operator corresponding to the above algorithm. Then, $f = f'$ holds, and the operator norm of* (JDG$-\mathcal{S}_f$) *is upper bounded as $\|\mathsf{JDG} - \mathcal{S}_f\|_{\mathrm{op}} \leq 2^{(\ell/2)-(0.1r)+1}$. The depth required to implement* JDG *on a quantum circuit is at most about $2^{11} r\ell^3/c$, and* JDG *makes $2^8 r/c$ queries to the oracle.*

[7] $h_{\boldsymbol{x}}$ is defined so that $h_{\boldsymbol{x}}(i) = 1$ iff $(\boldsymbol{x}G)_i = \zeta_i$ for $i < N$. The domain size is set as $(N+1)$ instead of N to make it a power of two.

See Section D in the appendix for a proof. We will set $r = 25\ell$ such that the difference $\|\mathsf{JDG} - \mathcal{S}_f\|_{\mathrm{op}}$ is extremely small ($\leq O(2^{-2\ell})$) and we can use JDG instead of $\mathcal{S}_f$ in QAA while keeping the success probability almost unchanged.

The amount of auxiliary qubits required for JDG is at most the maximum of

- the qubits needed to compute $h_{\boldsymbol{x}}$,
- the qubits needed to perform the classical computation in Step 2,
- the qubits needed for Calc on page 8,

which is in $O(\ell^2)$. (See Section E in the appendix for details on the qubits required for $h_{\boldsymbol{x}}$. For Calc, we assume that the sin function is approximately computed using a constant number of terms in the Taylor expansion.)

How to Prepare $|\psi\rangle$. Roughly speaking, we prepare $|\psi\rangle$ by (i) making a superposition of all $\boldsymbol{x} \in \mathbb{F}_2^\ell \setminus \{\boldsymbol{0}\}$, (ii) compute i such that the i-th column of G (denoted by $\boldsymbol{g}_i$) is equal to $\boldsymbol{x}$, and (iii) multiply the phase $(-1)^{\zeta_{i-1}}$ by querying to the quantum oracle.

To perform (ii), we utilize Shor's algorithm and the relationship between the state update of LFSR and multiplication in the finite field. Let ξ be the isomorphism defined in Eq. (3). The most important observation is that we have

$$i = \log_\alpha\left(\xi(\boldsymbol{g}_i)/\xi(\boldsymbol{\Gamma})\right) + 1 = \log_\alpha\left(\xi(\boldsymbol{g}_i)\right) - \log_\alpha\left(\xi(\boldsymbol{\Gamma})\right) + 1 \tag{18}$$

for each i because

$$\xi\left(\boldsymbol{g}_i\right) = \xi\left(\boldsymbol{\Gamma}(M^\top)^{i-1}\right) \underset{Eq.\ (4)}{=} \xi(\boldsymbol{\Gamma}) \cdot \alpha^{i-1},$$

holds. Therefore, we can compute i such that $\boldsymbol{g}_i = \boldsymbol{x}$ for a given $\boldsymbol{x} \in \mathbb{F}_2^\ell \setminus \{\boldsymbol{0}\}$ as $i = \log_\alpha\left(\xi(\boldsymbol{g}_i)\right) - \log_\alpha\left(\xi(\boldsymbol{\Gamma})\right) + 1$, applying Shor's discrete logarithm in the multiplicative group $\mathfrak{F}^\times = (\mathbb{F}_{2^n}[x]/(f^*(x)))^\times \cong \mathbb{Z}/N\mathbb{Z}$.

First, we describe our algorithm when a unitary operator DLOG satisfying

$$\mathsf{DLOG}\,|\boldsymbol{x}\rangle\,|0\rangle = |\boldsymbol{x}\rangle\,|\log_\alpha(\boldsymbol{x})\rangle$$

is available as a quantum circuit. After that, we discuss the complexity to approximate it with sufficeintly high precision by using Shor's algorithm.

Algorithm $\mathsf{PREP2}$.

1. Prepare the superposition
$$\sum_{\boldsymbol{x} \in \mathbb{F}_2^\ell \setminus \{\boldsymbol{0}\}} \frac{1}{\sqrt{N}}\,|\boldsymbol{x}\rangle\,.$$
2. For each basis state $\boldsymbol{x}$, run DLOG twice to compute $\log_\alpha(\xi(\boldsymbol{x}))$ and $\log_\alpha(\xi(\boldsymbol{\Gamma}))$. Then compute $i := \log_\alpha(\xi(\boldsymbol{x})) - \log_\alpha(\xi(\boldsymbol{\Gamma})) + 1$. Now the state is
$$\sum_{\boldsymbol{x} \in \mathbb{F}_2^\ell \setminus \{\boldsymbol{0}\}} \frac{1}{\sqrt{N}}\,|\boldsymbol{x}\rangle\,|\log_\alpha(\xi(\boldsymbol{x})))\rangle\,|\log_\alpha(\xi(\boldsymbol{\Gamma}))\rangle\,|i\rangle\,.$$

3. By querying $(i-1)$ to the oracle, multiply the entire state by the phase $(-1)^{\zeta_{i-1}}$ (Recall that now we are assuming the oracle that returns ζ_i for the input i in quantum superposition.)
4. Uncompute Step 2. Now, the state is $|\psi\rangle$.

Both the T-depth and the number of ancillary qubits are dominated by Step 2 (and its uncomputation). By running 8ℓ instances of Shor's algorithm DLOG can be approximated with error (w.r.t. operator norm) in $O(2^{-2\ell})$, T-depth at most $2^8\ell^3+O(\ell^2)$, and $O(\ell^2)$ ancillary qubits (see Section F in the appendix for details). Therefore, $\mathsf{PREP2}$ can be implemented with error in $O(2^{-2\ell})$, T-depth $2^9\ell^3+O(\ell^2)$, and $O(\ell^2)$ ancillary qubits.

The Entire Algorithm and Analysis. Our algorithm solving Problem 1 runs as follows.

Algorithm $\mathsf{QFCA2}$. Run $\mathsf{QAAw/oKp}$ on p.8 with $U := H^{\otimes \ell}\cdot \mathsf{PREP2}$ on the Boolean function $f:\mathbb{F}_2^\ell \to \mathbb{F}_2$ such that $f(\boldsymbol{x})=1$ iff $\boldsymbol{x}=\boldsymbol{s}^{(0)}$. Here, f is computed by using the algorithm JDG. The parameter r for JDG is chosen as $r:=25\ell$.

Below we analyze the complexity of $\mathsf{QFCA2}$ assuming $\ell \geq 10$ (so that the assumption of Proposition 5 will be satisfied) and $c \leq \ell^{-1}$, which is the case for the applications we will see later.

Let p be the probability that we obtain an $\boldsymbol{x}$ satisfying $f(\boldsymbol{x})=1$ when we measure the state $H^{\otimes\ell}\cdot\mathsf{PREP2}\,|0^\ell\rangle$ $(=H^{\otimes\ell}\,|\psi\rangle)$. That is,

$$p=\frac{N^2\cdot \mathrm{Cor}(\boldsymbol{s}^{(0)}G,\boldsymbol{\zeta})^2}{2^\ell\cdot\sum_{\boldsymbol{w}}|\Psi(\boldsymbol{w})|^2}=\frac{2^\ell}{2^\ell-1}\cdot\mathrm{Cor}(\boldsymbol{s}^{(0)}G,\boldsymbol{\zeta})^2. \tag{19}$$

By the claim at the end of Section 3.3, $\mathrm{Cor}(\boldsymbol{x}G,\boldsymbol{\zeta})^2\leq c^2/4$ holds for all $\boldsymbol{x}\neq\boldsymbol{s}^{(0)}$, with probability at least 0.9 (when K and IV are randomly chosen). Provided this condition hold,

$$p\geq\frac{2^\ell}{2^\ell-1}\cdot c^2\approx c^2 \tag{20}$$

follows from Eq. (19), and the attack finds the correct decoding result $\boldsymbol{s}^{(0)}$ in expected time complexity at most about

$$T_{\mathrm{total}}=(9/2)p^{-1/2}\left(2\cdot T_{\mathrm{prepare}}+T_f\right)\lessapprox(9/2)\frac{1}{c}\left(2\cdot T_{\mathrm{prepare}}+T_f\right) \tag{21}$$

where T_{prepare} (resp., T_f) is the running time of the algorithm $\mathsf{PREP2}$ (resp., JDG). Since we set $r=25\ell$ for JDG,

$$T_f\lessapprox 25\cdot 2^{11}\ell^4/c$$

follows from Proposition 5. In addition, $c\leq\ell^{-1}$ is assumed. Meanwhile, as discussed before, $T_{\mathrm{prepare}}=2^9\ell^3+O(\ell^2)$ holds. Hence $T_{\mathrm{prepare}}\ll T_f$ holds and the total time complexity can be estimated as

$$T_{\text{total}} \lessapprox (9/2) \cdot \frac{1}{c} \cdot \left(25 \cdot 2^{11} \ell^4/c\right) \leq 2^{18} \cdot \ell^4/c^2. \tag{22}$$

In addition, since PREP2 makes only $O(1)$ queries, the number of queries made by QFCA2 can be approximated by the number of queries made by JDG multiplied by the number of applications of JDG. Therefore, the number of quantum queries made by QFCA2 is at most about $(9/2)p^{-1/2} \cdot (2^8(25\ell)/c) \lessapprox 2^{15}\ell/c^2$.

Next, we analyze the success probability. Failure of QFCA2 is attributed to the following four factors:

(1) Whether the assumption of Proposition 5 about the correlations (i.e., the assumption of the claim at the end of Section 3.3) is satisfied or not.
(2) The error in JDG approximating $\mathcal{S}_f$ (provided the assumption of Proposition 5 is satisfied).
(3) The error in approximating DLOG.
(4) Failure of QAA to find the correct value $\boldsymbol{s}^{(0)}$.

The error probability coming from (1) is at most 0.1 (because of the claim at the end of Section 3.3 and the assumption $\ell \geq 10$), and (4) is already taken into account in the expected complexity of QAA shown in Proposition 2. Hence, if (2) and (3) could be ignored, the algorithm would successfully recover $\boldsymbol{s}^{(0)}$ in the expected time complexity shown in Eq. (22) with a probability of at least 0.9 (with respect to the randomness of K and IV).

Regarding (2), the distance between JDG and $\mathcal{S}_f$ is at most $O(2^{-2\ell})$. This means that the failure probability of QAA with t applications of $\mathcal{S}_f$ increases by $O(t \cdot 2^{-2\ell})$ if $\mathcal{S}_f$ is replaced with JDG. The same holds for the approximation of DLOG. As the overall complexity of QFCA2 is $O(\ell^4 c^{-2})$, the success probability decreases by $O(\ell^4 c^{-2} 2^{-2\ell})$ in total when (2) and (3) are taken into account. Therefore, the success probability of the algorithm can be estimated as at least $0.9 - O(\ell^4 c^{-2} 2^{-2\ell})$.

Summary. Assuming $\ell \geq 10$ and $c \leq \ell^{-1}$, QFCA2 solves Problem 1 with time and query complexity (approximately) at most $2^{18}\ell^4/c^2$ and $2^{15}\ell/c^2$. The probability of success is estimated as at least $0.9 - O(\ell^4 c^{-2} 2^{-2\ell})$. The number of ancillary qubits required is $O(\ell^2)$ since both JDG and (the approximation of) DLOG are implemented with $O(\ell^2)$ qubits.

In the applications below, we will only discuss the cases where the term $O(\ell^4 c^{-2} 2^{-2\ell})$ is negligibly small.

Remark 4. If QFCA2 is applied to a stream cipher with an ℓ-bit LFSR, it requires $O(\ell^2)$ qubits. Meanwhile, the exhaustive key search with Grover's algorithm would require only $O(\ell)$ qubits. Strictly speaking, the validity of a dedicated quantum attack such as QFCA2 should be compared to the parallelized Grover search using the same amount of qubits. However, $O(\ell^2)$ qubits would allow to run only $O(\ell)$ parallel instances, which yields a speed-up by a factor of at most $O(\sqrt{\ell})$. This factor is not so large as to affect the validity of the attacks considered in this paper, and the cost of the Grover search also varies depending on implementations of a target cipher. Hence, in what follows, we do not take parallelization into account.

6.3 Applications

We show applications of QFCA2 on SNOW 2.0 [31], SNOW 3G [34], and Sosemanuk [6]. Our goal is to break the qBPRF security of the ciphers when the quantum oracle of the Booleanized version of the cipher is given.

SNOW 2.0 SNOW 2.0 is a stream cipher designed by Ekdahl and Johansson [31], which is standardized by ISO/IEC [48]. It consists of an LFSR of length $L = 16$ over $\mathbb{F}_{2^{32}}$ (512 bits long in total) and a finite state machine that keeps 64-bit states. The state update and keystream generation are carried out in 32-bit words. The cipher outputs a 32-bit keystream segment at each clock, updating the internal state registers. (see Figure 2). The key length is either 128 or 256 bit, and IVs are 128 bits. In the initialization phase, a key and an IV are linearly expanded and loaded to the registers and then mixed by updating the states 32 times, with the output bits fed back to the LFSR.

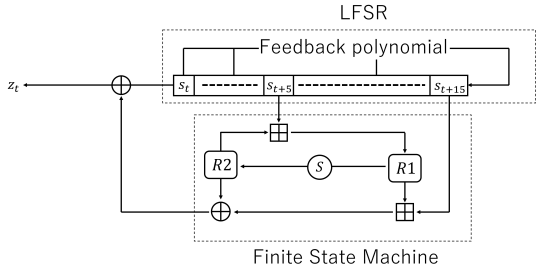

Fig. 2. SNOW 2.0. Each line corresponds to a 32-bit word. $R1$ and $R2$ are additional 32-bit registers. Modular additions are denoted by $\boxplus$. The circled “S” at the center is a non-linear permutation.

Linear Approximations and Classical Attacks. In the classical setting, many (linear attacks and) fast correlation attacks have been proposed on SNOW 2.0 [72, 87, 57, 92, 36, 40, 41]. Among others, [36, 40, 39] found linear approximation

$$\langle \boldsymbol{s}^{(t)}, \boldsymbol{\Gamma} \rangle_{\mathbb{F}_2} \approx \langle z_t, \boldsymbol{\Lambda}_1 \rangle_{\mathbb{F}_2} \oplus \langle z_{t+1}, \boldsymbol{\Lambda}_2 \rangle_{\mathbb{F}_2} \tag{23}$$

for some $\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2 \in \mathbb{F}_{2^{32}}$ and $\boldsymbol{\Gamma} \in \mathbb{F}_{2^{512}}$ which holds with absolute linear correlation $2^{-14.411}$. Here, $\boldsymbol{s}^{(t)}$ is the state of the LFSR at clock t, and $z_t \in \mathbb{F}_{2^{32}}$ is the 32-bit word (keystream segment) output by the cipher at clock t. A recent work by Gong et al. [39] also found multiple approximations with the same absolute correlation. As far as we know, $2^{-14.411}$ is the highest (absolute) linear correlation of SNOW 2.0 that has been found so far.

The current fastest classical attack on SNOW 2.0 is the fast correlation attack in [41] that uses a few linear correlations, including the above, which recovers not only the initial state of the LFSR but also the key with about 2^{159} data and 2^{162} time complexity.

Application of QFCA2. We apply QFCA2 on the decoding problem (Problem 1) derived from the linear approximation of Eq. (23). The parameters are set as $c := 2^{-14.411}$ and $\ell := 512$, and the sequence $\boldsymbol{\zeta} = (\zeta_0, \zeta_2, \ldots, \zeta_{N-1})$ is defined as $\zeta_t := \langle z_t, \boldsymbol{\Lambda}_1 \rangle_{\mathbb{F}_2} \oplus \langle z_{t+1}, \boldsymbol{\Lambda}_2 \rangle_{\mathbb{F}_2}$.

Recall that we aim to break the qBPRF security. Here, we briefly review the attack model. We assume the quantum oracle of the Booleanized oracle of SNOW 2.0, which we denote by $O_{B\mathrm{SNOW2.0}}$. Given a superposition of indices i as an input, the oracle returns the i-th bit of the keystream in quantum superposition. Since the period of LFSR is $2^\ell - 1$, the upper limit of i is set as $i < 2^\ell - 1$ to prevent trivial attacks. (The oracle also allows an adversary to query IV. However, when mounting fast correlation attacks, we choose an IV arbitrarily and fix it during the attack, as in classical attacks.)

Problem 1 (and QFCA2) assumes the quantum oracle that returns ζ_i for each i (in superposition), whereas the oracle $O_{B\mathrm{SNOW2.0}}$ returns a keystream bit of SNOW 2.0. Thus, to apply QFCA2, we simulate the oracle of ζ_i by using $O_{B\mathrm{SNOW2.0}}$ as follows[8].

0. (Assume a basis state $|t\rangle$ is given as an input)
1. Query $t, t+1, \ldots, t+63$ to $O_{B\mathrm{SNOW2.0}}$ to obtain $z_t, z_{t+1} \in \mathbb{F}_{32}$.
2. Compute $\zeta_t := \langle z_t, \boldsymbol{\Lambda}_1 \rangle_{\mathbb{F}_2} \oplus \langle z_{t+1}, \boldsymbol{\Lambda}_2 \rangle_{\mathbb{F}_2}$.
3. Copy the value ζ_t into the output register.
4. Uncompute Step 1-2.

Since $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$ are predetermined constants, Step 2 can be executed by only applying CNOT gates. Hence, the T-depth of Step 2 is zero, and the above simulation requires $2 \times 64 = 2^7$ depth and the same amount of queries to $O_{B\mathrm{SNOW2.0}}$.

Using this simulation, QFCA2 recovers the initial state of the LFSR of SNOW 2.0 (and breaks its qBPRF security) with time complexity at most $2^7 \cdot \big(2^{18} \cdot (512)^4 \cdot (2^{14.411})^2\big) \leq 2^{89.3}$, making quantum queries at most $2^7 \cdot \big(2^{15} \cdot (512) \cdot (2^{14.411})^2\big) \leq 2^{59.3}$ times.

On the other hand, the running time of the exhaustive key search with the Grover search is at least $2^{10} \cdot 2^{\kappa/2}$ for κ-bit keys, because of the following reasons: The Grover search performs $2^{\kappa/2}$ iterations to search for a κ-bit secret key. It evaluates a Boolean function f such that $f(\boldsymbol{x}) = 1$ iff $\boldsymbol{x}$ matches the secret key

[8] Strictly speaking, the last bit ζ_N cannot be computed due to the upper limit of the index i that can be queried to $O_{B\mathrm{SNOW2.0}}$. So, we just set $\zeta_N := 0$. Since N is quite large ($N = 2^{512} - 1$), this modification has little effect on the attack complexity and the success probability.

in each iteration. The only way (that we are aware of) to implement such f is to compute a keystream segment for each input $\boldsymbol{x}$ and check whether it matches the real keystream segment. As the initialization phase requires 32 state updates and each update involves at least one 32-bit modular addition (in the finite state machine), the T-depth of f should be at least $32 \cdot 32 = 2^{10}$.

In particular, when the key length is 256, our attack (time complexity $2^{89.3}$) is significantly faster than the generic attack by the Grover search (time complexity at least 2^{138}).

A Note on Key Recovery. Our primary aim here is to break the qBPRF security of SNOW 2.0. Still, once the LFSR's initial state is recovered, the remaining 64-bit state of the finite state machine can also be recovered with at most about 2^{64} classical operations. In particular, since the initialization phase of SNOW 2.0 is reversible, we can recover the secret key with almost the same complexity.

Remark 5. A previous work [27] shows that a quantum guess-and-determine attack on SNOW 2.0 with 256-bit keys breaks the cipher in time around 2^{88}. However, the attack uses a large quantum computer of size around 2^{88} to run a parallelized Grover search, whereas our paper does not consider parallel computation. In addition, [27] defines the unit of time (resp., size) as the time to execute the target cipher once (resp., the size to implement the target cipher). Under this cost metric, if a quantum computer of size 2^{88} is available, the generic attack (simple parallelized Grover search) recovers a 256-bit key with time complexity about $\sqrt{2^{256}/2^{88}} = 2^{84}$. Thus, the attack on SNOW 2.0 with 256-bit keys is slower than the generic attack.

We also applied QFCA2 on SNOW 3G [34] and Sosemanuk [6], of which the structures are quite close to SNOW 2.0. For SNOW 3G, the time and query complexity are $2^{102.9}$ and $2^{72.9}$, which is slower than the Grover search but significantly faster than the classical attacks [72, 88, 40, 41, 39]. On Sosemanuk, the time and query complexity are $2^{101.11}$ and $2^{73.15}$. This is slower than the quantum guess-and-determine attack [27], but faster than the Grover search for long keys (e.g., 256-bit keys). See Section G in the appendix for details.

6.4 Disuccsions

We also tried speeding up other classical fast correlation attacks [83, 78, 86, 84, 77, 93, 59, 39], which recover the initial state (or even the secret key) of Grain v1 [45], Grain-128 [44], Grain-128a [1], Fruit-v2 [38], Fruit-80 [37], Plantlet [65], SNOW-V [32], and SNOW-Vi [33] faster than the classical exhaustive key search. However, we have not found Q2 attacks faster than the exhaustive key search with Grover's algorithm.

Except for SNOW-V/Vi, the problem is that the absolute correlations are too small. For SNOW-V/Vi, which uses 512-bit LFSRs and 256-bit keys, the absolute correlations are in a moderate order ($> 2^{-50}$), but still the time complexity of QFCA2 becomes at least 2^{150} due to the factor $2^{18}\ell^4$ in Eq. (22) with $\ell = 512$.

Acknowledgements

We thank anonymous reviewers for their insightful comments.

References

1. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of grain-128 with optional authentication. Int. J. Wirel. Mob. Comput. **5**(1), 48–59 (2011)
2. Ågren, M., Löndahl, C., Hell, M., Johansson, T.: A survey on fast correlation attacks. Cryptogr. Commun. **4**(3-4), 173–202 (2012)
3. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(6), 818–830 (2013)
4. Aumasson, J., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New features of latin dances: Analysis of salsa, chacha, and rumba. In: FSE 2008, Revised Selected Papers. LNCS, vol. 5086, pp. 470–488. Springer (2008)
5. Bera, D., Tharrmashastha, S.: Quantum and randomised algorithms for non-linearity estimation. ACM Transactions on Quantum Computing **2**(2) (June 2021)
6. Berbain, C., Billet, O., Canteaut, A., Courtois, N.T., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: Sosemanuk, a fast software-oriented stream cipher. In: New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 98–118. Springer (2008)
7. Berbain, C., Gilbert, H.: On the security of IV dependent stream ciphers. In: Biryukov, A. (ed.) FSE 2007, Revised Selected Papers. LNCS, vol. 4593, pp. 254–273. Springer (2007)
8. Berbain, C., Gilbert, H., Maximov, A.: Cryptanalysis of grain. In: Robshaw, M.J.B. (ed.) FSE 2006, Revised Selected Papers. LNCS, vol. 4047, pp. 15–29. Springer (2006)
9. Bernstein, D.J.: ChaCha, a variant of Salsa20. In: Workshop Record of SASC. vol. 8 (2008)
10. Bernstein, D.J.: The Salsa20 family of stream ciphers. In: New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 84–97. Springer (2008)
11. Biham, E., Anderson, R.J., Knudsen, L.R.: Serpent: A new block cipher proposal. In: Vaudenay, S. (ed.) FSE '98, Proceedings. LNCS, vol. 1372, pp. 222–238. Springer (1998)
12. Boneh, D., Zhandry, M.: Quantum-secure message authentication codes. In: EUROCRYPT 2013, Proceedings. LNCS, vol. 7881, pp. 592–608. Springer (2013)
13. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline simon's algorithm. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Proceedings, Part I. LNCS, vol. 11921, pp. 552–583. Springer (2019)
14. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline simon's algorithm. In: ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 552–583. Springer (2019)
15. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: On quantum slide attacks. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019, Revised Selected Papers. LNCS, vol. 11959, pp. 492–519. Springer (2019)

16. Bonnetain, X., Schrottenloher, A., Sibleyras, F.: Beyond quadratic speedups in quantum attacks on symmetric schemes. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Proceedings, Part III. LNCS, vol. 13277, pp. 315–344. Springer (2022)
17. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik: Progress of Physics **46**(4-5), 493–505 (1998)
18. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics **305**, 53–74 (2002)
19. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: LATIN 1998. LNCS, vol. 1380, pp. 163–169. Springer (1998)
20. Canteaut, A., Trabbia, M.: Improved fast correlation attacks using parity-check equations of weight 4 and 5. In: EUROCRYPT 2000, Proceeding. LNCS, vol. 1807, pp. 573–588. Springer (2000)
21. Canteut, A.: LFSR-based stream ciphers, https://www.rocq.inria.fr/secret/Anne.Canteaut/MPRI/chapter3.pdf (Accessed on September 19, 2024)
22. Chepyzhov, V.V., Johansson, T., Smeets, B.J.M.: A simple algorithm for fast correlation attacks on stream ciphers. In: FSE 2000, Proceedings. LNCS, vol. 1978, pp. 181–195. Springer (2000)
23. Chepyzhov, V.V., Smeets, B.J.M.: On A fast correlation attack on certain stream ciphers. In: EUROCRYPT '91, Proceedings. LNCS, vol. 547, pp. 176–185. Springer (1991)
24. Cho, J.Y., Hermelin, M.: Improved linear cryptanalysis of SOSEMANUK. In: Lee, D.H., Hong, S. (eds.) ICISC 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5984, pp. 101–117. Springer (2009)
25. Chose, P., Joux, A., Mitton, M.: Fast correlation attacks: An algorithmic point of view. In: EUROCRYPT 2002, Proceedings. LNCS, vol. 2332, pp. 209–221. Springer (2002)
26. Collard, B., Standaert, F., Quisquater, J.: Improving the time complexity of matsui's linear cryptanalysis. In: Nam, K., Rhee, G. (eds.) ICISC 2007, Proceedings. LNCS, vol. 4817, pp. 77–88. Springer (2007)
27. Ding, L., Wu, Z., Zhang, G., Shi, T.: Quantum guess and determine attack on stream ciphers. Comput. J. **67**(1), 292–303 (2024)
28. Dong, X., Sun, S., Shi, D., Gao, F., Wang, X., Hu, L.: Quantum collision attacks on aes-like hashing with low quantum random access memories. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 727–757. Springer (2020)
29. ECRYPT: eSTREAM: ECRYPT stream cipher project, https://www.ecrypt.eu.org/stream/
30. Einsele, S., Wunder, G.: Quantum speed-up of fast correlation attacks against stream ciphers. Crypto day matters 36
31. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: SAC 2002, Revised Papers. LNCS, vol. 2595, pp. 47–61. Springer (2002)
32. Ekdahl, P., Johansson, T., Maximov, A., Yang, J.: A new SNOW stream cipher called SNOW-V. IACR Trans. Symmetric Cryptol. **2019**(3), 1–42 (2019)
33. Ekdahl, P., Maximov, A., Johansson, T., Yang, J.: Snow-vi: an extreme performance variant of SNOW-V for lower grade cpus. In: WiSec 2021. pp. 261–272. ACM (2021)
34. ETSI/SAGE: Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification. Version 1.1 (2006)

35. Feng, X., Liu, J., Zhou, Z., Wu, C., Feng, D.: A byte-based guess and determine attack on SOSEMANUK. In: ASIACRYPT 2010, Proceedings. LNCS, vol. 6477, pp. 146–157. Springer (2010)
36. Funabiki, Y., Todo, Y., Isobe, T., Morii, M.: Several milp-aided attacks against SNOW 2.0. In: CANS 2018, Proceedings. LNCS, vol. 11124, pp. 394–413. Springer (2018)
37. Ghafari, V.A., Hu, H.: Fruit-80: A secure ultra-lightweight stream cipher for constrained environments. Entropy **20**(3), 180 (2018)
38. Ghafari, V.A., Hu, H., Chen, Y.: Fruit-v2: Ultra-lightweight stream cipher with shorter internal state. IACR Cryptology ePrint Archive 2016/355 (2016)
39. Gong, X., Hao, Y., Wang, Q.: Combining milp modeling with algebraic bias evaluation for linear mask search: improved fast correlation attacks on snow. Des. Codes Cryptogr. **92**, 1663–1728 (2024)
40. Gong, X., Zhang, B.: Fast computation of linear approximation over certain composition functions and applications to SNOW 2.0 and SNOW 3g. Des. Codes Cryptogr. **88**(11), 2407–2431 (2020)
41. Gong, X., Zhang, B.: Comparing large-unit and bitwise linear approximations of SNOW 2.0 and SNOW 3g and related attacks. IACR Trans. Symmetric Cryptol. **2021**(2), 71–103 (2021)
42. Grassi, L., Naya-Plasencia, M., Schrottenloher, A.: Quantum algorithms for the k -xor problem. In: ASIACRYPT 2018, Proceedings, Part I. LNCS, vol. 11272, pp. 527–559. Springer (2018)
43. Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. In: ACM STOC 1996. pp. 212–219. ACM (1996)
44. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: Proceedings 2006 IEEE International Symposium on Information Theory, ISIT 2006, The Westin Seattle, Seattle, Washington, USA, July 9-14, 2006. pp. 1614–1618. IEEE (2006)
45. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain family of stream ciphers. In: New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 179–190. Springer (2008)
46. Hosoyamada, A., Sasaki, Y.: Quantum Demirci-Selçuk Meet-in-the-Middle Attacks: Applications to 6-Round Generic Feistel Constructions. In: SCN 2018. LNCS, vol. 11035, pp. 386–403. Springer (2018)
47. Hosoyamada, A., Sasaki, Y.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 249–279. Springer (2020)
48. ISO/IEC: 18033-4:2011 Information technology — Security techniques — Encryption algorithms. Part 4 Stream Ciphers (2011)
49. Johansson, T., Jönsson, F.: Fast correlation attacks based on turbo code techniques. In: CRYPTO '99, Proceedings. LNCS, vol. 1666, pp. 181–197. Springer (1999)
50. Johansson, T., Jönsson, F.: Improved fast correlation attacks on stream ciphers via convolutional codes. In: EUROCRYPT '99, Proceeding. LNCS, vol. 1592, pp. 347–362. Springer (1999)
51. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: CRYPTO 2016, Part II. LNCS, vol. 11693, pp. 207–237. Springer (2016)
52. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2016**(1), 71–94 (2016)

53. Kitaev, A.Y.: Quantum measurements and the abelian stabilizer problem. arXiv preprint quant-ph/9511026 (1995)
54. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In: ISIT 2010. pp. 2682–2685. IEEE (2010)
55. Kuwakado, H., Morii, M.: Security on the quantum-type Even-Mansour cipher. In: ISITA 2012. pp. 312–316. IEEE (2012)
56. Leander, G., May, A.: Grover Meets Simon - Quantumly Attacking the FX-construction. In: ASIACRYPT 2017. LNCS, vol. 10625, pp. 161–178. Springer (2017)
57. Lee, J., Lee, D.H., Park, S.: Cryptanalysis of sosemanuk and SNOW 2.0 using linear masks. In: Pieprzyk, J. (ed.) ASIACRYPT 2008, Proceedings. LNCS, vol. 5350, pp. 524–538. Springer (2008)
58. Ma, S., Jin, C., Guan, J.: Improved fast correlation attack on snow 3g stream cipher (2023), available at SSRN: https://ssrn.com/abstract=4501579
59. Ma, S., Jin, C., Shi, Z., Cui, T., Guan, J.: Correlation attacks on snow-v-like stream ciphers based on a heuristic milp model. IEEE Transactions on Information Theory, Early Access (2023)
60. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993, Proceedings. LNCS, vol. 765, pp. 386–397. Springer (1993)
61. Meier, W.: Fast correlation attacks: Methods and countermeasures. In: FSE 2011, Revised Selected Papers. LNCS, vol. 6733, pp. 55–67. Springer (2011)
62. Meier, W., Staffelbach, O.: Fast correlation attacks on stream ciphers (extended abstract). In: EUROCRYPT '88, Proceedings. LNCS, vol. 330, pp. 301–314. Springer (1988)
63. Mihaljevic, M.J., Fossorier, M.P.C., Imai, H.: Fast correlation attack algorithm with list decoding and an application. In: FSE 2001, Revised Papers. LNCS, vol. 2355, pp. 196–210. Springer (2001)
64. Mihaljevic, M.J., Golic, J.D.: A fast iterative algorithm for A shift register initial state reconstruction given the nosiy output sequence. In: AUSCRYPT '90, Proceedings. LNCS, vol. 453, pp. 165–175. Springer (1990)
65. Mikhalev, V., Armknecht, F., Müller, C.: On ciphers that continuously access the non-volatile key. IACR Trans. Symmetric Cryptol. **2016**(2), 52–79 (2016)
66. Mitzenmacher, M., Upfal, E.: Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis (2nd edition). Cambridge university press (2017)
67. Nam, Y., Su, Y., Maslov, D.: Approximate quantum fourier transform with $O(n \log(n))$ T gates. npj Quantum Information **6** (2020), Article number: 26
68. National Institute of Standards and Technlology: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016), https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf
69. Naya-Plasencia, M., Schrottenloher, A.: Optimal merging in quantum k-xor and k-xor-sum algorithms. In: EUROCRYPT 2020, Proceedings, Part II. LNCS, vol. 12106, pp. 311–340. Springer (2020)
70. Nie, J., Zhu, Q., Li, M., Sun, X.: Quantum circuit design for integer multiplication based on schönhage–strassen algorithm. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **42**(12), 4791–4802 (2023). https://doi.org/10.1109/TCAD.2023.3279300
71. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press (2010)

72. Nyberg, K., Wallén, J.: Improved linear distinguishers for SNOW 2.0. In: Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4047, pp. 144–162. Springer (2006)
73. Sanders, Y.R., Low, G.H., Schere, A., Berry, D.W.: Black-box quantum state preparation without arithmetic. Phys. Rev. Lett. **122**, 020502 (Jan 2019)
74. Santoli, T., Schaffner, C.: Using simon's algorithm to attack symmetric-key cryptographic primitives. Quantum Inf. Comput. **17**(1&2), 65–78 (2017)
75. Schrottenloher, A.: Improved quantum algorithms for the k-xor problem. In: SAC 2021, Revised Selected Papers. LNCS, vol. 13203, pp. 311–331. Springer (2021)
76. Schrottenloher, A.: Quantum linear key-recovery attacks using the QFT. In: CRYPTO 2023, Proceedings, Part V. LNCS, vol. 14085, pp. 258–291. Springer (2023)
77. Shi, Z., Jin, C., Jin, Y.: Improved linear approximations of SNOW-V and snow-vi. IACR Cryptology ePrint Archive 2021/1105 (2021)
78. Shi, Z., Jin, C., Zhang, J., Cui, T., Ding, L., Jin, Y.: A correlation attack on full SNOW-V and snow-vi. In: EUROCRYPT 2022, Proceedings, Part III. LNCS, vol. 13277, pp. 34–56. Springer (2022)
79. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE Computer Society (1994)
80. Siegenthaler, T.: Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans. Inf. Theory **30**(5), 776–780 (1984)
81. Simon, D.R.: On the Power of Quantum Computation. In: 35th Annual Symposium on Foundations of Computer Science. pp. 116–123 (1994). https://doi.org/10.1109/SFCS.1994.365701
82. Thapliyal, H., Varun, T.S.S., Muñoz-Coreas, E., Britt, K.A., Humble, T.S.: Quantum circuit designs of integer division optimizing t-count and t-depth. In: iNIS 2017, Proceedings. pp. 123–128. IEEE (2017)
83. Todo, Y., Isobe, T., Meier, W., Aoki, K., Zhang, B.: Fast correlation attack revisited - cryptanalysis on full grain-128a, grain-128, and grain-v1. In: Crypto 2018, Proceedings, Part II. LNCS, vol. 10992, pp. 129–159. Springer (2018)
84. Todo, Y., Meier, W., Aoki, K.: On the data limitation of small-state stream ciphers: Correlation attacks on fruit-80 and plantlet. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019, Revised Selected Papers. LNCS, vol. 11959, pp. 365–392. Springer (2019)
85. Wagner, D.A.: A generalized birthday problem. In: Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings. LNCS, vol. 2442, pp. 288–303. Springer (2002)
86. Wang, S., Liu, M., Lin, D., Ma, L.: On grain-like small state stream ciphers against fast correlation attacks: Cryptanalysis of plantlet, fruit-v2 and fruit-80. Comput. J. **66**(6), 1376–1399 (2023)
87. Watanabe, D., Biryukov, A., Cannière, C.D.: A distinguishing attack of SNOW 2.0 with linear masking method. In: SAC 2003, Revised Papers. LNCS, vol. 3006, pp. 222–233. Springer (2003)
88. Yang, J., Johansson, T., Maximov, A.: Vectorized linear approximations for attacks on SNOW 3g. IACR Trans. Symmetric Cryptol. **2019**(4), 249–271 (2019)
89. Zeng, K., Yang, C., Rao, T.R.N.: An improved linear syndrome algorithm in cryptanalysis with applications. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO '90, Proceedings. LNCS, vol. 537, pp. 34–47. Springer (1990)

90. Zhandry, M.: How to construct quantum random functions. In: FOCS. pp. 679–687. IEEE Computer Society (2012)
91. Zhang, B., Liu, R., Gong, X., Jiao, L.: Improved fast correlation attacks on the Sosemanuk stream cipher. IACR Trans. Symmetric Cryptol. **2023**(4), 83–111 (2023)
92. Zhang, B., Xu, C., Meier, W.: Fast correlation attacks over extension fields, large-unit linear approximation and cryptanalysis of SNOW 2.0. In: CRYPTO 2015, Proceedings, Part I. LNCS, vol. 9215, pp. 643–662. Springer (2015)
93. Zhou, Z., Feng, D., Zhang, B.: Efficient and extensive search for precise linear approximations with high correlations of full SNOW-V. Des. Codes Cryptogr. **90**(10), 2449–2479 (2022)

A Implementations of Multiplication in $\mathbb{F}_{2^n}$ on Quantum Circuits

For completeness, this section discusses the implementation cost (controlled) multiplication in $\mathbb{F}_{2^n}$ on quantum circuits. The implementations are basic and straightforward, but we will give them here for a precise complexity analysis.

Let $f(x) \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n over $\mathbb{F}_2$. This section identifies $\mathbb{F}_{2^n}$ with $\mathbb{F}_2[x]/(f(x))$. Each element of $\mathbb{F}_2[x]/(f(x))$ is represented as a polynomial $\beta(x) = \sum_{0 \le i < n} b_i x^i$, where $b_0, \ldots, b_{n-1} \in \mathbb{F}_2$, which is further identified with the vector $\boldsymbol{b} = (b_0, \ldots, b_n) \in \mathbb{F}_2^n$.

Proposition 6. *There is a quantum circuit realizing the unitary operator of the field multiplication, i.e., the unitary operator* Mult *such that*

$$\mathsf{Mult}\,|\beta\rangle\,|\gamma\rangle\,|\delta\rangle = |\beta\rangle\,|\gamma\rangle\,|\delta \oplus ((\beta \cdot \gamma) \bmod f)\rangle$$

for $\beta, \gamma, \delta \in \mathbb{F}_2[x]/(f(x))$ *with depth at most* $3n^2$ *and at most* $2n$ *auxiliary qubits.*

Proof. For arbitrary i, the unitary operator

$$\mathsf{CAdd}_{x^i} : |b\rangle\,|\delta\rangle \mapsto |b\rangle\,|\delta \oplus (x^i \bmod f)\rangle \quad (b \in \mathbb{F}_2, \delta \in \mathbb{F}_2[x]/(f(x)))$$

can be implemented by using at most n CNOT gates. In addition, the unitary operator that computes the multiplication of two polynomials of degree $< n$ in $\mathbb{F}_2[x]$ (not in $\mathbb{F}_2[x]/(f(x))$), i.e., the operator

$$\widetilde{\mathsf{Mult}}\,|\beta\rangle\,|\gamma\rangle\,|\delta\rangle = |\beta\rangle\,|\gamma\rangle\,|\delta \oplus (\beta \cdot \gamma)\rangle$$

can be computed by using at most n^2 Toffoli gates in a straightforward manner (to compute all $b_i \cdot c_j$ for all i and j). We implement Mult as follows.

1. Apply $\widetilde{\mathsf{Mult}}$ to compute $\beta \cdot \gamma$ (in $\mathbb{F}_2[x]$, not in $\mathbb{F}_2[x]/(f(x))$). Suppose $\beta \cdot \gamma$ is represented as $\beta \cdot \gamma = \sum_{0 \le i \le 2n-2} u_i x^i$.
2. Apply CAdd_{x^i} with $b = u_i$ for $i = 0, 1, \ldots, 2n-2$ in sequential order.
3. Uncompute Step 1.

This procedure requires at most $2n$ additional auxiliary quibts and T-depth at most $3n^2$ (here, we used the fact that the T-depth for the Toffoli gate is at most 3 [3]). □

Proposition 7. *There is a quantum circuit realizing the unitary operator* ExMult *satisfying*

$$\mathsf{ExMult}\,|i\rangle\,|\beta\rangle\,|\delta\rangle = |i\rangle\,|\beta\rangle\,|\delta \oplus (\beta^i \bmod f)\rangle$$

for $i \in \{0, \ldots, 2^n - 1\}$ *and* $\beta, \delta \in \mathbb{F}_2[x]/(f(x))$ *with depth at most* $3n^3 + 3n^2$ *and at most* $(2n^2 + 2n)$ *auxiliary qubits.*

Proof. We implement ExMult as follows.

1. Copy β into a new register to obtain
$$|i\rangle\,|\beta\rangle\,|\delta\rangle \otimes |\beta\rangle\,.$$

2. Apply Mult to compute $(\beta^2 \bmod f)$ and obtain
$$|i\rangle\,|\beta\rangle\,|\delta\rangle \otimes |\beta\rangle\,|\beta^2 \bmod f\rangle\,.$$

3. Copy $(\beta^2 \bmod f)$ into a new register, apply Mult to compute $(\beta^4 \bmod f)$, and obtain
$$|i\rangle\,|\beta\rangle\,|\delta\rangle \otimes |\beta\rangle\,|\beta^2 \bmod f\rangle\,|\beta^2 \bmod f\rangle\,|\beta^4 \bmod f\rangle\,.$$

4. Compute $(\beta^4 \bmod f), (\beta^8 \bmod f), \ldots, (\beta^{2^{n-1}} \bmod f)$ similarly to obtain
$$|i\rangle\,|\beta\rangle\,|\delta\rangle \otimes |\beta\rangle \left(\bigotimes_{j=2}^{2^n-2} |\beta^{2^j} \bmod f\rangle\,|\beta^{2^j} \bmod f\rangle \right) |\beta^{2^n-1} \bmod f\rangle\,.$$

5. Suppose i is represented as a binary sequence $i_0||i_1||\cdots||i_{n-1}$. For $j = 0, \ldots, n-1$, apply Toffoli gates to add the value $i_j \cdot \left(\beta^{2^j} \bmod f\right)$ into the δ register. Now, the state is
$$|i\rangle\,|\beta\rangle\,|\delta \oplus \left(\beta^i \bmod f\right)\rangle \otimes |\beta\rangle \left(\bigotimes_{j=2}^{2^n-2} |\beta^{2^j} \bmod f\rangle\,|\beta^{2^j} \bmod f\rangle \right) |\beta^{2^n-1} \bmod f\rangle\,.$$

6. Uncompute Steps 1-4.

By Proposition 6, Steps 2-4 can be performed with at most $n \cdot (3n^2)$ T-depth and $2n$ additional auxiliary qubits. Step 5 requires T-depth at most $3n^2$ (as the T-depth of Toffoli is at most 3 [3]). Therefore, the above algorithm requires depth at most

$$3n^3 + 3n^2$$

and auxiliary qubits at most

$$2n \cdot n + 2n = 2n^2 + 2n$$

in total. □

B On the Depth and Qubits to Implement $\Lambda_q(Q(H^{\otimes n}, f))$

Recall that, for an arbitrary unitary operator W acting an n-qubit states, the operator $\Lambda_q(W)$ acts on $(\log_2 q + n)$-qubit states as

$$\Lambda_q(W) \left|i\right\rangle \left|x\right\rangle = \left|i\right\rangle (W^i \left|x\right\rangle).$$

Here, $0 \leq i \leq q-1$. Suppose that i is decomposed into a binary sequence as $i = i_{\log_2(q)-1} \cdots i_1 i_0$ ($i_j \in \mathbb{F}_2$ for each j).

For a non-negative integer j, let CW^{2^j} denote the controlled-W^{2^j} operator such that

$$CW^{2^j} \left|b\right\rangle \left|x\right\rangle = \begin{cases} \left|b\right\rangle \left|x\right\rangle & \text{if } b = 0 \\ \left|b\right\rangle (W^{2^j} \left|x\right\rangle) & \text{if } b = 1 \end{cases}$$

for $b \in \mathbb{F}_2, x \in \mathbb{F}_2^n$. We assume that $\Lambda_q(W)$ is realized by applying CW^{2^j}, where the control qubit is i_j and the target qubits are the least significant n qubits (corresponding to $\left|x\right\rangle$ in the initial state), for $j = 0, \ldots, \log_2(q) - 1$, as done in quantum phase estimation [53, 71].

The quantum counting algorithm (QC on page 9) invokes $\Lambda_q(W)$ with $W = Q(H^{\otimes n}, f) := -H^{\otimes n}\mathcal{S}_0 H^{\otimes n}\mathcal{S}_f$ for some Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$.

When $W = Q(H^{\otimes n}, f)$, we implement CW^{2^j} by just iteratively applying $C(Q(H^{\otimes n}, f))$ (i.e., the controlled-$Q(H^{\otimes n}, f)$ operator) 2^j times.

The controlled-$Q(H^{\otimes n}, f)$ operator is realized as applying the controlled versions of $\mathcal{S}_f$, $H^{\otimes n}$, $\mathcal{S}_0$, and $-H^{\otimes n}$ in sequential order. The controlled operator of $\pm H^{\otimes n}$ can be implemented only with Clifford gates without auxiliary qubits. In addition, the controlled-$\mathcal{S}_f$ operator (resp., controlled-$\mathcal{S}_0$ operator) can be implemented with T-depth D_f (resp., D_0) if the T-depth to implement $\mathcal{S}_f$ is D_f (resp., D_0) with n auxiliary qubits. $D_f \gg D_0$ holds in typical cases, and thus, the controlled-$Q(H^{\otimes n}, f)$ operator can be implemented with T-depth at most D_f and n auxiliary qubits.

Summarizing the above arguments, the T-depth required to implement the operator $\Lambda_q(Q(H^{\otimes n}, f))$ is at most about

$$\sum_{0 \leq j \leq \log_2(q)-1} 2^j \cdot D_f = q \cdot D_f.$$

The number of auxiliary qubits needed is n.

C On the Claim at the End of Section 3.3

We begin with Eq. (9). Let $\boldsymbol{r}, \boldsymbol{r}'$ be two random binary sequence of length N. Then,

$$X := \#\{0 \leq i \leq N-1 : r_i = r'_i\}$$

follows the binomial distribution $B(N, 1/2)$, which is approximated by the normal distribution $\mathcal{N}(N/2, N/4)$. Thus

$$\sqrt{N} \cdot \mathrm{Cor}(\boldsymbol{r}, \boldsymbol{r}') = \frac{X - N/2}{\sqrt{N}/2} \tag{24}$$

approximately follows the standard normal distribution, and so does $\sqrt{N} \cdot \mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})$ for $\boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)}$ by the heuristic assumption. Hence we have

$$\begin{aligned}
&\Pr_{K,IV}\left[\text{There is an } \boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)} \text{ such that } \mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \geq c^2/4\right] \\
&\quad \leq \sum_{\boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)}} \Pr_{K,IV}\left[\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \geq c^2/4\right] \\
&\quad \leq \sum_{\boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)}} \Pr_{K,IV}\left[N \cdot \mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \geq Nc^2/4\right] \\
&\quad \leq \sum_{\boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)}} \Pr_{K,IV}\left[N \cdot \mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \geq 2\ell'\right] = \sum_{\boldsymbol{x} \neq \boldsymbol{\sigma}^{(0)}} 2 \Pr_{K,IV}\left[\sqrt{N} \cdot \mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta}) \geq \sqrt{2\ell'}\right] \\
&\quad \lessapprox 2^{\ell'} \cdot 2 \cdot \frac{1}{\sqrt{2\pi}} \int_{\sqrt{2\ell'}}^{\infty} e^{-t^2/2} dt \underset{(*)}{=} \frac{2^{\ell'+1}}{\sqrt{2\pi}} \int_{\ell'}^{\infty} \frac{1}{\sqrt{2t'}} e^{-t'} dt' \leq \frac{2^{\ell'+1}}{\sqrt{2\pi}} \int_{\ell'}^{\infty} \frac{1}{\sqrt{2}} e^{-t'} dt' \\
&\quad = \frac{2^{\ell'+1}}{\sqrt{2\pi}} \cdot \frac{1}{\sqrt{2}} \cdot e^{-\ell'} \leq \left(\frac{2}{e}\right)^{\ell'},
\end{aligned}$$

where we put $t' := t^2/2$ at $(*)$. Therefore, Eq. (8) follows.

Next, we focus on Eq. (9). Assume $\mathbf{Ex}_{K,IV}\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta})\right] = c > 0$. Then, the variable

$$Y := \#\{0 \leq i \leq N-1 : (\boldsymbol{\sigma}^{(0)}G)_i = \zeta_i\}$$

approximately follows the Binomial distribution $B(N, \frac{1+c}{2})$, because the equation $(\boldsymbol{\sigma}^{(0)}G)_i = \zeta_i$ holds with probability $\frac{1+c}{2}$ almost independently for each i. Since $B(N, \frac{1+c}{2})$ is approximated by the normal distribution $\mathcal{N}\left(N(\frac{1+c}{2}), N(\frac{1-c^2}{4})\right)$, the variable $\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta}) = (2Y - N)/N$ approximately follows the normal distribution $\mathcal{N}\left(c, \frac{1}{N} \cdot \frac{1-c^2}{4}\right)$. Since the standard deviation of this distribution can be upper bounded as

$$\mathsf{sd} := \sqrt{\frac{1}{N} \cdot \frac{1-c^2}{4}} \leq \sqrt{\frac{c^2}{8\ell'} \cdot \frac{1}{4}} \leq \frac{c}{4\sqrt{2}},$$

we have

$$\Pr\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta}) \geq c/2\right] \geq \Pr\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta}) \geq c - 2\mathsf{sd}\right] \gtrapprox 0.95.$$

Hence Eq. (9) holds if $\mathbf{Ex}_{K,IV}\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta})\right] > 0$. Similar arguments also work for $\mathbf{Ex}_{K,IV}\left[\mathrm{Cor}(\boldsymbol{\sigma}^{(0)}G, \boldsymbol{\zeta})\right] < 0$.

D Proof of Proposition 5

$f = f'$ immediately follows from the assumption and the definition of the algorithm. In what follows, we show the claims on the number of queries, the depth, and the operator norm.

About the Claim on the Number of Queries. JDG makes queries only at Step 1-(a) when applying QC. Since each instance of QC makes $q = 2^7/c$ queries, JDG makes $2rq = 2^8r/c$ queries in total.

About the Claim on the Depth. Next, we show the claim about depth.

First, we consider the depth to compute $h_{\boldsymbol{x}}(i) = (\boldsymbol{x}G)_i \oplus \zeta_i$ for a given $(\boldsymbol{x}, i)$, which we denote by D_h. By definition of G,

$$\begin{aligned}(\boldsymbol{x}G)_i &= \langle \boldsymbol{x}, \boldsymbol{g}_{i+1}^\top \rangle_{\mathbb{F}_2} \\ &= \langle \boldsymbol{x}, \boldsymbol{\Gamma}(M^\top)^i \rangle_{\mathbb{F}_2}\end{aligned}$$

holds. In addition, by Eq. (4),

$$\begin{aligned}\boldsymbol{\Gamma}(M^\top)^i &= \xi^{-1}\left(\xi\left(\boldsymbol{\Gamma}(M^\top)^i\right)\right) \\ &= \xi^{-1}\left(\xi\left(\boldsymbol{\Gamma}\right)\alpha^i\right)\end{aligned}$$

holds. Thus, $h_{\boldsymbol{x}}(i)$ can be computed as follows.

1. Compute α^i by using ExMult of Proposition 7 (identifying the field $\mathfrak{F} = \mathbb{F}_q^L[x]/(f(x))$ with $\mathbb{F}_2^\ell[x]/(f'(x))$ for some polynomial f').
2. Multiply $\xi(\boldsymbol{\Gamma})$ by α^i with Mult of Proposition 6 to obtain $\xi\left(\boldsymbol{\Gamma}\right)\alpha^i = \xi\left(\boldsymbol{g}_{i+1}^\top\right)$.
3. Compute the inner product $\langle \boldsymbol{x}, \boldsymbol{g}_{i+1}^\top \rangle_{\mathbb{F}_2}$. (Note that $\boldsymbol{g}_{i+1}^\top$ is immediately determined from $\xi\left(\boldsymbol{g}_{i+1}^\top\right)$ due to the simplicity of the definition of ξ.)
4. Querying i to the oracle, compute $\langle \boldsymbol{x}, \boldsymbol{g}_{i+1}^\top \rangle_{\mathbb{F}_2} \oplus \zeta_i$ $(= h_{\boldsymbol{x}}(i))$.
5. Uncpmpute Step 1-3.

Step 1 requires T-depth $3\ell^2$ by Propositon 6. Step 2 requires T-depth $3\ell^3 + 3\ell^2$ by Proposition 7. Step 3 can be performed with ℓ Toffoli gates, of which the T-depth is at most by 3ℓ by [3]. Step 4 uses a single oracle gate. Therefore, the total depth D_h to compute $h_{\boldsymbol{x}}(i)$ is

$$D_h = 2 \cdot \left(3\ell^2 + (3\ell^3 + 3\ell^2) + 3\ell\right) + 1 \leq 6(\ell+1)^3 \leq 8\ell^3, \tag{25}$$

where we used the assumption $\ell \geq 10$ for the last inequality.

From the definition of JDG and the explanation below Proposition 3, it follows that the depth required for Step 1-(a) of JDG on a quantum circuit is at most about

$$2^7 r D_h/c \leq 2^{10} r\ell^3/c.$$

Since addition and multiplication of $O(\ell)$-bit integers can be computed in depth $O(\ell^2)$ (using schoolboock multiplications), the depths required for Step 1-(b), 1-(c), and 2 are quite quite small compared to the depth required for Step 1-(a). Hence the total depth is at most about

$$2 \cdot 2^{10} r \ell^3 / c = 2^{11} r \ell^3 / c.$$

About the Claim on Operator Norm. Note that the domain size of $h_{\boldsymbol{x}}$ is $N + 1 = 2^\ell$ for all $\boldsymbol{x}$. Recall that we denote the value $|h_{\boldsymbol{x}}^{-1}(1)|$ by Z when $\boldsymbol{x}$ is fixed. Here, we show the following lemma.

Lemma 1. *Assume $\ell \geq 10$. Let $\boldsymbol{x}$ be an arbitrary element of $\mathbb{F}_2^\ell$. If $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \leq c^2/4$, then*

$$\sqrt{Z(2^\ell - Z)} \leq \left(\frac{1}{2} + \frac{c}{4}\right) 2^\ell \tag{26}$$

holds. If $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \geq c^2/2$, then

$$\sqrt{Z(2^\ell - Z)} \leq \left(\frac{1}{\sqrt{2}} + \frac{1}{4}\right) 2^\ell \tag{27}$$

holds.

Proof. First, we show Eq. (26). As $|\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})| = |(2Z - N)/N| \leq c/2$ holds by assumption, we have

$$|Z - N/2| \leq cN/4,$$

which implies

$$\frac{N}{2} - \frac{c}{4}N \leq Z \leq \frac{N}{2} + \frac{c}{4}N.$$

Hence

$$\begin{aligned}
\sqrt{Z(2^\ell - Z)} &= \sqrt{Z(N + 1 - Z)} \\
&\leq \sqrt{\left(\frac{N}{2} + \frac{c}{4}N\right)\left(\frac{N}{2} + \frac{c}{4}N + 1\right)} \\
&\leq \sqrt{\left(\frac{N}{2} + \frac{c}{4}N + \frac{1}{2}\right)\left(\frac{N}{2} + \frac{c}{4}N + \frac{1}{2}\right)} \\
&\leq \left(\frac{1}{2} + \frac{c}{4}\right)(N + 1) \\
&= \left(\frac{1}{2} + \frac{c}{4}\right) 2^\ell
\end{aligned}$$

follows.

Next, we show Eq. (27). Since $|\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})| = |(2Z - N)/N| \geq c/\sqrt{2}$ holds by assumption, we have

$$Z \leq \frac{N}{2} - \frac{cN}{2\sqrt{2}} \text{ or } \frac{N}{2} + \frac{cN}{2\sqrt{2}} \leq Z.$$

In both cases,

$$\sqrt{Z(N-Z)} \leq \sqrt{(N/2 - cN/2\sqrt{2})N} = N \cdot \sqrt{\frac{1}{2} - \frac{c}{2\sqrt{2}}} \leq \frac{N}{\sqrt{2}}$$

holds. Therefore

$$\begin{aligned}\sqrt{Z(2^\ell - Z)} &= \sqrt{Z(N-Z) + Z} \leq \sqrt{Z(N-Z)} + \sqrt{Z} \\ &\leq \frac{N}{\sqrt{2}} + \sqrt{N} \leq \frac{2^\ell}{\sqrt{2}} + \sqrt{2^\ell} \\ &\leq \left(\frac{1}{\sqrt{2}} + \frac{1}{4}\right) \cdot 2^\ell\end{aligned}$$

follows, where we used the assumption $\ell \geq 10$ at the last inequality. □

Suppose we run the algorithm JDG on a basis state $|\boldsymbol{x}\rangle$ with $\boldsymbol{x} \neq \boldsymbol{s}^{(0)}$, and measure the entire state at the end of Step 1 of the algorithm. Let X be the number defined by

$$\#\left\{1 \leq j \leq r : (\tilde{C}_j)^2 \geq 3c^2/8\right\}.$$

Since $q = 2^7/c$ and $0 \leq c \leq 1$, it follows that

$$\begin{aligned}\text{(the right hand side of Eq. (1) with } n = \ell) &\overset{(*)}{<} \frac{\pi(1+\frac{c}{2})2^\ell}{q} + \frac{\pi^2 2^\ell}{q^2} \\ &\leq \frac{\pi 2^\ell}{q} + \frac{\pi 2^\ell}{2q} + \frac{\pi^2 2^\ell}{q^2 c^2}. \\ &\leq \left(\frac{\pi}{2^7} + \frac{\pi}{2^8} + \frac{\pi^2}{2^{14}}\right) c 2^\ell \\ &\leq 2^{-4.5} c 2^\ell, \end{aligned} \tag{28}$$

where we used the assumption that $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \leq c^2/4$ holds for all $\boldsymbol{x} \neq \boldsymbol{s}^{(0)}$ and Eq. (26) at $(*)$. In addition, for each j,

$$\begin{aligned}
\Pr\left[(\tilde{C}_j)^2 \geq 3c^2/8\right] &= \Pr\left[\left|\frac{2\tilde{Z}_j - N}{N}\right| \geq \sqrt{3/8}c\right] \\
&\leq \Pr\left[\left|\frac{2Z - N}{N}\right| + \left|\frac{2(Z - \tilde{Z}_j)}{N}\right| \geq \sqrt{3/8}c\right] \\
&\overset{(*)}{\leq} \Pr\left[c/2 + \left|\frac{2(Z - \tilde{Z}_j)}{N}\right| \geq \sqrt{3/8}c\right] \\
&= \Pr\left[\left|Z - \tilde{Z}_j\right| \geq \frac{\sqrt{3} - \sqrt{2}}{4\sqrt{2}}cN\right] \\
&\leq \Pr\left[\left|Z - \tilde{Z}_j\right| \geq 2^{-4.16}cN\right] \\
&\overset{(**)}{\leq} \Pr\left[\left|Z - \tilde{Z}_j\right| \geq 2^{-4.26}c2^{\ell}\right] \\
&\overset{(***)}{\leq} 0.2
\end{aligned}$$

where we used the assumption that $\mathrm{Cor}(\boldsymbol{x}G, \boldsymbol{\zeta})^2 \leq c^2/4$ holds for all $\boldsymbol{x} \neq \boldsymbol{s}^{(0)}$ (and thus $|(2Z - N)/N| \leq c/2$) at $(*)$, the assumption $\ell \geq 10$ at $(**)$, and Proposition 3 with Eq. (28) at $(***)$. Hence, the random variable X follows a binomial distribution $B(r, \mathsf{pr})$ with $\mathsf{pr} \leq 0.2$. By Chernoff bound, we have

$$\begin{aligned}
\Pr\left[X \geq r/2\right] &= \Pr\left[X \geq (1 + 1.5) \times (0.2r)\right] \leq \Pr\left[X \geq (1 + 1.5) \times (\mathsf{pr} \cdot r)\right] \\
&\leq \left(\frac{e^{1.5}}{(1 + 1.5)^{1+1.5}}\right)^{\mathsf{pr} \cdot r} \leq \left(\frac{1}{2}\right)^{0.2 \cdot r}
\end{aligned}$$

This implies that

$$\left\|\mathsf{JDG}\left|\boldsymbol{x}\right\rangle - \mathcal{S}_{f'}\left|\boldsymbol{x}\right\rangle\right\| \leq 2\left(\frac{1}{2}\right)^{0.1r} \tag{29}$$

for $\boldsymbol{x} \neq \boldsymbol{s}^{(0)}$.

Next, suppose we run the algorithm JDG on a basis state $|\boldsymbol{s}^{(0)}\rangle$, and measure the entire state at the end of Step 1 of the algorithm. Let Y be the number defined by

$$\#\left\{1 \leq j \leq r : (\tilde{C}_j)^2 < 3c^2/8\right\}.$$

Since $q = 2^7/c$ and $0 \leq c \leq 1$, it follows that

$$\begin{aligned}
\text{(the right hand side of Eq. (1) with } n = \ell) &\overset{(*)}{<} \frac{\pi(\sqrt{2}+\frac{1}{2})2^\ell}{q} + \frac{\pi^2 2^\ell}{q^2} \qquad (30)\\
&\leq \frac{\pi\sqrt{2}\cdot 2^\ell}{q} + \frac{\pi 2^\ell}{2q} + \frac{\pi^2 2^\ell}{q^2 c^2}.\\
&\leq \left(\frac{\pi}{2^{6.5}} + \frac{\pi}{2^8} + \frac{\pi^2}{2^{14}}\right) c2^\ell\\
&\leq 2^{-4} c2^\ell, \qquad (31)
\end{aligned}$$

we used the assumption that $\mathrm{Cor}(\boldsymbol{s}^{(0)}G, \boldsymbol{\zeta})^2 \geq c^2/2$ holds and Eq. (27) at $(*)$. For each j, we have

$$\begin{aligned}
\Pr\left[(\tilde{C}_j)^2 < 3c^2/8\right] &= \Pr\left[\left|\frac{2\tilde{Z}_j - N}{N}\right| < \sqrt{3/8}c\right]\\
&\leq \Pr\left[\left|\frac{2Z - N}{N}\right| - \left|\frac{2(Z - \tilde{Z}_j)}{N}\right| < \sqrt{3/8}c\right]\\
&\overset{(*)}{\leq} \Pr\left[c/\sqrt{2} - \left|\frac{2(Z - \tilde{Z}_j)}{N}\right| < \sqrt{3/8}c\right]\\
&= \Pr\left[\left|Z - \tilde{Z}_j\right| > \frac{2-\sqrt{3}}{2\sqrt{2}} cN\right]\\
&\leq \Pr\left[\left|Z - \tilde{Z}_j\right| > 2^{-3.4} cN\right]\\
&\overset{(**)}{\leq} \Pr\left[\left|Z - \tilde{Z}_j\right| > 2^{-3.5} c2^\ell\right]\\
&\overset{(***)}{\leq} 0.2
\end{aligned}$$

where we used the assumption that $\mathrm{Cor}(\boldsymbol{s}^{(0)}G, \boldsymbol{\zeta})^2 \geq c^2/2$ holds (and thus $|(2Z - N)/N| \geq c/\sqrt{2}$) at $(*)$, the assumption $\ell \geq 10$ at $(**)$, and used Proposition 3 and Eq. (31) at $(***)$. Therefore, we can show

$$\left\|\mathsf{JDG}\,|\boldsymbol{s}^{(0)}\rangle - \mathcal{S}_{f'}\,|\boldsymbol{s}^{(0)}\rangle\right\| \leq 2\left(\frac{1}{2}\right)^{0.1r} \qquad (32)$$

in the same way we showed Eq. (29).

Let $|\phi\rangle := \sum_{\boldsymbol{x}\in\mathbb{F}_2^\ell} \phi_{\boldsymbol{x}} |\boldsymbol{x}\rangle$ be an arbitrary ℓ-qubit quantum state ($\phi_{\boldsymbol{x}} \in \mathbb{C}$ for each $\boldsymbol{x}$). Then we have

$$\begin{aligned}
\|\mathsf{JDG}\,|\phi\rangle - \mathcal{S}_{f'}\,|\phi\rangle\| &\leq \sum_{\boldsymbol{x}\in\mathbb{F}_2^\ell} |\phi_{\boldsymbol{x}}|\, \|\mathsf{JDG}\,|\boldsymbol{x}\rangle - \mathcal{S}_{f'}\,|\boldsymbol{x}\rangle\| \\
&\overset{(*)}{\leq} 2^{-(0.1r)+1} \sum_{\boldsymbol{x}\in\mathbb{F}_2^\ell} |\phi_{\boldsymbol{x}}| \\
&\overset{(**)}{\leq} 2^{-(0.1r)+1} \cdot \sqrt{2^\ell \sum_{\boldsymbol{x}\in\mathbb{F}_2^\ell} |\phi_{\boldsymbol{x}}|^2} \\
&\leq 2^{(\ell/2)-(0.1r)+1},
\end{aligned}$$

where we used Eq. (29) and Eq. (32) for $(*)$ and Jensen's inequality for $(**)$. Therefore, $\|\mathsf{JDG} - \mathcal{S}_f\|_{op} = \|\mathsf{JDG} - \mathcal{S}_{f'}\|_{op} \leq 2^{(\ell/2)-(0.1r)+1}$ follows.

E On the Qubits Required to Compute $\boldsymbol{h}$

Assume $h_{\boldsymbol{x}}(i)$ is implemented as in the five steps above Eq. (25). Then, ExMult of Step 1 and Mult of Step 2 use at most $O(\ell^2)$ qubits and $O(\ell)$ qubits, respectively (due to Proposition 6 and Proposition 7), and other steps require at most $O(\ell)$ qubits. Hence the number of qubits required is $O(\ell^2)$ in total.

F On Discrete Logarithm

Section F.1 reviews the details of Shor's algorithm for discrete logarithms with constant error probability. Section F.1 explains how much computational resource is needed to make the error probability exponentially small and approximate the unitary operator DLOG satisfying

$$\mathsf{DLOG}\,|\boldsymbol{x}\rangle\,|\boldsymbol{0}\rangle = |\boldsymbol{x}\rangle\,|\log_\alpha \boldsymbol{x}\rangle$$

so that it can be incorporated into fast correlation attacks.

F.1 Algorithm with Constant Error Probability

We basically follow the style of [71, Section 5], modifying details to fit our problem in question. Our goal is to compute $\log_\alpha \beta$ for $\alpha, \beta \in (\mathbb{F}_{2^\ell})^\times$, where α is a generator of the multiplicative group and $0 \leq \log_\alpha \beta < N := |\mathbb{F}_{2^\ell}^\times| = 2^\ell - 1$.

Let $\epsilon > 0$ be a parameter (which will be fixed later) and $t := 2\ell + 2\lceil \log(2 + 1/\epsilon) \rceil$. The algorithm runs as follows.

Quantum Algorithm for Discrete Logarithm.

1. Prepare the initial state $|0^t\rangle|0^t\rangle|0^\ell\rangle$. (The rightmost register will store elements in $\mathbb{F}_{2^\ell}$.)
2. Apply the Hadamrd operators to the left and middle registers to obtain
$$\frac{1}{2^t}\sum_{0\le x,y<2^t}|x\rangle|y\rangle|0^\ell\rangle$$
3. For each x and y, compute $\beta^x\alpha^y$ and store the result in the right register to obtain
$$\frac{1}{2^t}\sum_{0\le x,y<2^t}|x\rangle|y\rangle|\beta^x\alpha^y\rangle.$$
4. Apply the inverse of QFT over $\mathbb{Z}/2^t\mathbb{Z}$ to the left and middle registers. Then, measure the left and middle registers to obtain two t-bit strings $z = z_1||z_2||\cdots||z_t$ and $w = w_1||\cdots||w_t$ ($z_i, w_i \in \{0,1\}$).
5. Identifying z (resp., w) with the binary fraction $0.z_1\ldots z_t$ (resp., $0.w_1\ldots w_t$), compute the integer which is closest to $z\cdot N$ (resp., $w\cdot N$) and denote it by Z (resp., W).
6. Do Steps 1-5 again to obtain another pair (Z', W').
7. Check if W and W' are coprime. If so, go to the next step. If not, output $\perp$ (which means that the algorithm has failed.)
8. Compute integers (X, Y) such that $XW + YW' = 1$. Then, set $\rho := XZ + YZ' \bmod N$.
9. Compute α^ρ and check if $\alpha^\rho = \beta$. If so, output ρ. Otherwise, output $\perp$.

Note that the above algorithm is presented in a way to contain intermediate measurements, but it is straightforward to make it measurement-free.

Complexity. Steps 1 and 2 do not require T gates nor ancillary qubits.

Step 3 can be performed by computing β^x and α^y on auxiliary registers with ExMult of Proposition 7 (in parallel), then multiplying β^x and α^y with Mult of Proposition 6. This requires T-depth $3\ell^3 + O(\ell^2)$ and $O(\ell^2)$ qubits.

The QFT for Step 4 can be approximated with negligible error with $O(t\log t) = O(\ell\log\ell)$ T gates with $O(t) = O(\ell)$ qubits [67].

Step 5 requires some arithmetic computations but their complexity is much smaller than that for Steps 7 and 8 explained later.

Step 6 can be performed independently from Steps 1-5, and so the T-depth for Steps 1-6 is $3\ell^3 + O(\ell^2)$ and the number of ancillary qubits are $O(\ell^2)$ in total (ignoring the cost of Step 5).

Steps 7 and 8 are performed by running the (extended) Euclidean algorithm with $O(t) = O(\ell)$ arithmetic operations, each of which requires $O(\ell)$ T-depth and $O(\ell^2)$ ancillary qubits [82, 70]. Hence the total T-depth and ancillary qubits for Steps 7 and 8 are $O(\ell^2)$ and $O(\ell^2)$, respectively.

Step 9 uses ExMult again to compute α^ρ, and requires T-depth $3\ell^3 + O(\ell^2)$ with $O(\ell^2)$ qubits.

In summary, (a measurement-free version of) the algorithm can be implemented on a quantum circuit of T-depth at most $6\ell^3+O(\ell^2)$ with $O(\ell^2)$ ancillary qubits.

Success Probability. Some calculations show that the state after Step 3 is equal to

$$\sum_{0\leq a<N} \frac{1}{\sqrt{N}} \left(\sum_{0\leq x<2^t} \frac{1}{\sqrt{2^t}} e^{\frac{2\pi i}{N}\cdot ax \log_\alpha \beta} |x\rangle \right) \left(\sum_{0\leq y<2^t} \frac{1}{\sqrt{2^t}} e^{\frac{2\pi i}{N}\cdot ay} |y\rangle \right) |\psi_a\rangle \,,$$

where $|\psi_a\rangle = \frac{1}{\sqrt{N}} \sum_{0\leq b<N} e^{-\frac{2\pi i}{N}\cdot ab} |\alpha^b\rangle$. Since $\langle\psi_{a'}|\psi_a\rangle = 0$ for $a \neq a'$, the success probability remains unchanged even if Step 4 is modified as follows.

Step 4-(i). Measure the third register with the orthonormal basis $\{|\psi_a\rangle\}_{0\leq a<N}$, when each $|\psi_a\rangle$ is obtained with probability $1/N$ and the state of the left and middle registers collapses into

$$\left(\sum_{0\leq x<2^t} \frac{1}{\sqrt{2^t}} e^{\frac{2\pi i}{N}\cdot ax \log_\alpha \beta} |x\rangle \right) \left(\sum_{0\leq y<2^t} \frac{1}{\sqrt{2^t}} e^{\frac{2\pi i}{N}\cdot ay} |y\rangle \right).$$

Step 4-(ii). As before, apply the inverse QFT to the remaining two registers, measure them according to the computational basis, and obtain (z, w).

By the discussions in [71, Section 5], the following things hold.

- Suppose that $|\psi_a\rangle$ with $a \neq 0$ is obtained at Step 4-(i). Then, by the discussions in [71, Section 5] about phase estimation, w and z at Step 4-(ii) are 2ℓ-bit approximations of $((\log_\alpha \beta)\cdot a \bmod N)/N$ and a/N with probability at least $1-\epsilon$. The same is true for Z' and W'.
- If W and W' are chosen independently and uniformly at random, then W and W' are coprime with probability at least $1/4$.

In addition, if z and w are good approximations and W, W' are coprime, then $\rho = XZ + YZ' \equiv (\log_\alpha \beta)(XW + YW') = \log_\alpha \beta$ holds modulo N. As $|\psi_a\rangle$ is obtained with probability $1/N$ for each a at Step 4-(i), the overall success probability of the algorithm is at least

$$1/4 - \epsilon - O(1/N). \tag{33}$$

We choose ϵ so that this success probability will be at least $1/3$ for $\ell \geq 10$ (e.g., $\epsilon = 1/10$).

F.2 Algorithm with Exponentially Small Error Probability

To make the error probability exponentially small and approximate DLOG, we run the following procedure.

0. (Assume a basis state $|\boldsymbol{x}\rangle$ is given as an input.)
1. For $j = 1, \ldots, r$, perform the following procedure.
   (a) Run the quantum algorithm described in the previous subsection (without measurement) and let ρ_j denote the result. (ρ_j is either the correct value $\log_\alpha \boldsymbol{x}$ or $\perp$.)
   (b) Copy ρ_j into a new auxiliary register.
   (c) Uncompute Step (a).
2. Check at least one of $\rho_1, \ldots, \rho_r$ is not $\perp$. If so, write the value to the output register. Otherwise, do nothing.
3. Uncompute Step 1.

Running the procedure and measuring the output register, we obtain the desired value $\log_\alpha \boldsymbol{x}$ with probability at least

$$1 - (2/3)^r.$$

So, setting $r := 8\ell$, the above algorithm approximates DLOG with the error (measured with the operator norm) at most $2^{-2\ell}$, when the T depth is at most $32\ell \cdot (6\ell^3 + O(\ell^2)) \leq 2^8\ell^3 + O(\ell^2)$ and the auxiliary qubits required is at most $O(\ell^2)$.

G Applications of QFCA2 to SNOW 3G and Sosemanuk

SNOW 3G. SNOW 3G is a stream cipher designed by ETSI/SAGE and specified by 3GPP for use in UMTS and LTE [34]. The design is basically the same as SNOW 2.0, but the finite state machine is modified and has 96-bit internal states. The LFSR is unchanged (see Figure 3). Both the key and IV lengths are 128. Like SNOW 2.0, the initialization phase linearly maps a key and an IV into internal registers in a linear manner and then updates the state 32 times, with the output bits fed back to the LFSR.

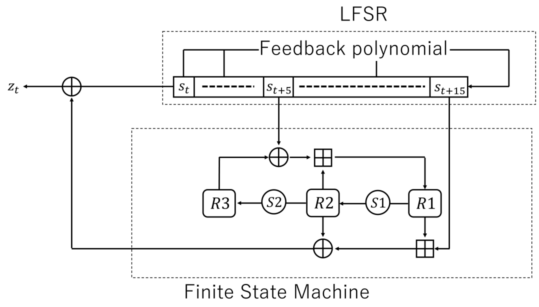

Fig. 3. SNOW 3G. Each line corresponds to a 32-bit word. $R1$, $R2$, and $R3$ are additional 32-bit registers. Modular additions are denoted by ⊞. The circled "S1" and "S2" are non-linear permutations.

Linear Approximations and Classical Attacks. So far, no work has shown a classical attack faster than the exhaustive key search, but many previous works have studied how efficient fast correlation attacks (and linear attacks) on SNOW 3G can be [72, 88, 40, 41, 39]. The (bitwise) linear approximation with the current highest absolute correlation is the one[9] found by Gong et al. [39], which has the form

$$\langle \boldsymbol{s}^{(t)}, \boldsymbol{\Gamma} \rangle_{\mathbb{F}_2} \approx \langle z_t, \boldsymbol{\Lambda}_1 \rangle_{\mathbb{F}_2} \oplus \langle z_{t+1}, \boldsymbol{\Lambda}_2 \rangle_{\mathbb{F}_2} \oplus \langle z_{t+2}, \boldsymbol{\Lambda}_3 \rangle_{\mathbb{F}_2} \tag{34}$$

for some $\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2, \boldsymbol{\Lambda}_3 \in \mathbb{F}_{2^{32}}$, and $\boldsymbol{\Gamma} \in \mathbb{F}_{2^{512}}$ and holds with absolute linear correlation $2^{-20.386}$. Using multiple linear approximations including the above one, they presented a fast correlation attack data and time complexity $2^{170.81}$ and $2^{174.95}$, respectively.

Application of QFCA2*.* The application of QFCA2 on SNOW 3G is almost identical to the one we showed on SNOW 2.0. The difference is as follows. First, since the correlation of the linear approximation is now $2^{-20.386}$ instead of $2^{-14.411}$, the term $(2^{14.411})^2$ is replaced with $(2^{20.386})^2$. Second, since there are three terms on the right-hand side of Eq. (34) while there are two in Eq. (34), the cost to simulate the oracle of ζ_i increases by a factor of 1.5. Hence, the query and time complexity of QFCA2 when applied to SNOW 3G is

$$2^{59.3} \times \frac{(2^{20.386})^2}{(2^{14.411})^2} \cdot 1.5 \leq 2^{72.9}$$

and

$$2^{89.3} \times \frac{(2^{20.386})^2}{(2^{14.411})^2} \cdot 1.5 \leq 2^{102.9},$$

respectively. (Since the initialization phase of SNOW 3G is also reversible, we can recover the secret key with some additional classical operations like the attack on SNOW 2.0.)

The time complexity of the above attack, $2^{102.9}$, will be worse than an exhaustive key search using Grover's algorithm on 128-bit keys[10], just as existing classical attacks are slower than the classical generic attack. Still, it is significantly lower compared to the current best time complexity $2^{174.95}$ of classical fast correlation attacks.

Sosemanuk. Sosemanuk [6] is a stream cipher designed by Berbain et al., which is included in the eSTREAM portfolio [29]. Following the design of the SNOW family, Sosemanuk consists of an LFSR and a finite state machine, carrying out state update and key generation in 32-bit words. The LFSR is defined over $\mathbb{F}_{2^{32}}$ and of length $L = 10$, and thus the total bit length is $\ell = 320$. The finite state machine keeps 64-bit states (see Figure 4). The key length can be any integer

[9] A recent preprint paper [58] also reports a linear approximation with the same absolute correlation.

[10] Similarly to the analysis on SNOW 2.0, the complexity of the Grover search will be at least $2^{10} \cdot 2^{128/2} = 2^{74}$.

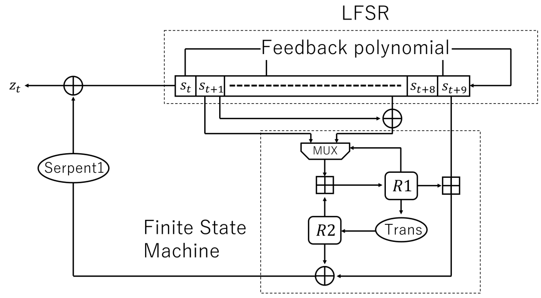

Fig. 4. Sosemanuk. $R1$ and $R2$ are additional registers. "Trans" is a permutation, and MUX is a function that outputs either s_{t+1} or $s_{t+1} \oplus s_{t+8}$ depending on a bit of $R1$. "Serpent 1" denotes a permutation derived from Serpent (in fact, it processes four consecutive output words from the finite state machine simultaneously, but we omit the details).

between 128 and 256. The initialization phase mixes a key and an IV using the key scheduling algorithm and round functions of Serpent [11], loading the result into Sosemanuk's internal registers. The designers claim 128-bit security for all the key lengths from 128 to 256.

Linear Approximations and Classical Attacks. Lee et al. [57] found the following linear approximation that holds with (absolute) correlation $c = 2^{-21.41}$ for some $\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2 \in \mathbb{F}_{2^{32}}$, and $\boldsymbol{\Gamma} \in \mathbb{F}_{2^{320}}$, yielding a fast correlation attack of time complexity around 2^{147}.

$$\langle \boldsymbol{s}^{(t)}, \boldsymbol{\Gamma} \rangle_{\mathbb{F}_2} \approx \langle z_t, \boldsymbol{\Lambda}_1 \rangle_{\mathbb{F}_2} \oplus \langle z_{t+3}, \boldsymbol{\Lambda}_2 \rangle_{\mathbb{F}_2}$$

The attack was later improved by multiple works [24, 92, 91], and the current most efficient attack, which is also a fast correlation attack, breaks the cipher with data and time complexity around 2^{135} [91] by using a linear approximation with the same (absolute) correlation and an advanced decoding technique.

Application of QFCA2. The application of QFCA2 on Sosemanuk is again almost identical to the one we showed on SNOW 2.0. The difference are that the (absolute) correlation is now $2^{-21.41}$ instead of $2^{-14.411}$, and the LFSR length is 320-bit. Similarly to the analysis on SNOW 2.0, the query and time complexity of QFCA2 when applied to Sosemanuk become $2^7 \cdot \left(2^{15} \cdot (320) \cdot (2^{21.41})^2\right) \leq 2^{73.15}$ and $2^7 \cdot \left(2^{18} \cdot (320)^4 \cdot (2^{21.41})^2\right) \leq 2^{101.11}$, respectively.

The attack is slower than the quantum version [27] of a classical guess-and-determine attack [35], which breaks the cipher in the *Q1* model with time complexity around 2^{88}. Still, it is faster than the exhaustive key search with Grover's algorithm for long (e.g., 224 or 256-bit) keys. (Sosemanuk's initialization phase is irreversible, so the attack does not extend to key recovery.)