Quantum Information Processing (2021) 20:343
https://doi.org/10.1007/s11128-021-03275-x

Quantum security of Grain-128/Grain-128a stream cipher against HHL algorithm

Weijie Liu[1] · **Juntao Gao**[1]

Received: 29 December 2020 / Accepted: 17 September 2021 / Published online: 15 October 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
HHL algorithm is a quantum algorithm for solving linear equation system. It can achieve an exponential improvement over the best classical algorithm. In this paper, we analyze the quantum security of Grain-128/Grain-128a stream cipher by using the HHL algorithm. Our algorithm is based on Chen and Gao's research on solving nonlinear equation system in Chen et al. (Quantum algorithm for optimization and polynomial system solving over finite field and application to cryptanalysis, 2018. arXiv:1802.03856) and Chen et al. (Quantum algorithms for Boolean equation solving and quantum algebraic attack on cryptosystems, 2017. arXiv:1712.06239). Firstly, we build a nonlinear Boolean equation system by choosing any keystream. Then, the nonlinear equation system is transformed into a special linear equation system that can be solved with the HHL algorithm. Finally, we solve the system by the HHL quantum algorithm. Our attack requires $N > 2^8$-bit keystream, and the complexity is $O(2^{21}N^{3.5}\kappa^2 e^{\epsilon}/\epsilon^{0.5})$ for Grain-128, and $O(2^{21.5}N^{3.5}\kappa^2 e^{\epsilon}/\epsilon^{0.5})$ for Grain-128a where κ is the condition number of the matrix of the corresponding linear systems and ϵ is a given error bound. Then we give a toy example of Grain family to estimate κ and briefly analyze the security of Grain-128/Grain-128a against HHL algorithm.

Keywords Grain-128 algorithm · Grain-128a algorithm · HHL quantum algorithm · Chosen-plaintext attack · Security analysis

1 Introduction

After the concept of quantum computer was proposed by Richard Feynman [1], the introduction of quantum algorithm based on quantum computer solved many problems which are complicated on classic computer. The advent of quantum computers will change the world. Compared with classical computers, quantum computers have

✉ Juntao Gao
jtgao@mail.xidian.edu.cn

[1] School of Telecommunications Engineering, Xidian University, P.O. 106, No. 2 South Taibai Road, Xi'an Shaanxi 710071, China

significant advantages in terms of computational speed. In a quantum computer, an n-bit register can store a superposition of 2^n values. Another advantage is that quantum computers can perform operations on 2^n input numbers simultaneously in one operation, which needs 2^n operations or parallel operation with 2^n different processors in a classical computer. It is easy to see that the rational use of quantum computers can greatly save computing resources and speed up computation.

In 1994, Shor proposed a quantum algorithm, which can be used to solve the factorization problem, which is difficult to solve on classical computers [2]. In 1996, Grover proposed the Grover algorithm for searching in an unstructured database [3]. Grover algorithm can search for a specific item from an unstructured database with a size of $N = 2^n$ in $O(\sqrt{n})$ steps [4]. It achieves polynomial acceleration for unstructured database search problems. With the emergence of efficient quantum algorithms [5,6], quantum computing is used in various fields such as machine learning [7], deep learning [8] and cryptography [9,10].

Quantum computing can be applied to almost all aspects of cryptography.. It can be used in encryption [11,12], key distribution [13] and security analysis [14,15]. For example, BV quantum algorithm can be used to attack block ciphers [15]. With the development of quantum computers, many cryptographic algorithms, which were originally secure in the classical computer environment, are not necessarily secure under the attack of quantum algorithms any more [16]. Therefore, it is necessary to estimate the security of typical cryptographic algorithms under quantum computing environments.

HHL algorithm [17] is a quantum algorithm for solving linear systems of equations over complex number field $\mathbb{C}$. This algorithm runs in $O(\log N)$ time under certain conditions, an exponential improvement over the best classical algorithm. This algorithm has been used in many fields such as [7,18].

In [19], Chen and Gao introduced a method of solving nonlinear equation system by HHL algorithm, and in [20], Chen and Gao explain how to rewrite the Boolean equation system as equation system over $\mathbb{C}$.

Grain is one of the three hardware-oriented algorithms in the eSTREAM project [21]. Grain [22] originally only employed an 80-bit key and 64-bit initial vector. In the last stage of the eSTREAM project, the designers proposed a new version of Grain called Grain-128 [23]. This version of the algorithm employs 128-bit key and 96-bit initial vector. Its security is higher than the previous version. In 2011, the designers of Grain proposed a new version of Grain-128 called Grain-128a [24]. This version of Grain-128 enhances security while retaining the basic structure of Grain-128; it can effectively resist all known attacks against Grain-128 at that time such as [25,26] and has built-in support for optional authentication. For a long time, apart from the dynamic cube attack in [27], no other attacks can pose a threat to Grain-128a.

Grain-128/Grain-128a consists of a linear feedback shift register (LFSR), a nonlinear feedback shift register (NFSR) and a Boolean function, all of which are typical components in stream ciphers. The structure of Grain-128/Grain-128a can be viewed as a generalized filter function generator, such that it is easy to write down the equations and determine the equation system's complexity. Many classical algorithms have been developed to attack Grain-128 algorithm. Yuseop Lee et al. in [25] proposed a related-key chosen IV attack. This attack requires $2^{26.59}$ chosen IVs, $2^{31.39}$-bit

keystream sequences to attack Grain-128 algorithm, and the computational complexity is $O(2^{27.01})$. Banik S et al. proposed a related-key chosen IV attack in [28] to attack Grain-128a. This attack requires $\gamma \cdot 2^{64}$ chosen IVs, $\gamma \cdot 2^{32}$-bit keystream sequences to attack Grain-128 algorithm, and the computational complexity is $\gamma \cdot 2^{64}$ where γ is a experimentally determined constant, and it is sufficient to estimate it as 2^8. Itai Dinur and Adi Shamir in [27] proposed a dynamic cube attack on the full version of Grain-128, which can recover the full key but only when it belongs to a large subset of 2^{-10} of the possible keys. This attack is faster than an exhaustive search over the 2^{128} possible keys by a factor of about 2^{15}. Then, Itai Dinur, Adi Shamir et al. in [26] proposed an experimentally verified attack. They use a new type of dedicated hardware to verify their attack, and the experimental results show that for about 7.5% of the keys we get a huge improvement factor of 2^{38} over exhaustive search. In [29], Wang and Fu improved the work in [26]; the computational complexity of their attack is $O(2^{74})$, which is applicable to all keys of Grain-128. Of course, there are also some quantum algorithms that can effectively attack Grain-128, such as the Grover algorithm [3]. By using this algorithm, the key of Grain-128 can be searched within $O(2^{64})$.

In this paper, we use the HHL algorithm to attack Grain-128/Grain-128a stream cipher. Firstly, we write the Grain-128/Grain-128a algorithm in the form of nonlinear Boolean equation system by algebraic methods. Then, we reduce this equation system into a nonlinear equation system over $\mathbb{C}$. Finally, we reduce this nonlinear equation system into a linear equation system and solve it by the HHL algorithm. Our attack requires only N-bit keystream, and the complexity is $O(2^{21}N^{3.5}\kappa^2 e^{\epsilon}/\epsilon^{0.5})$ for Grain-128, and $O(2^{21.5}N^{3.5}\kappa^2 e^{\epsilon}/\epsilon^{0.5})$ for Grain-128a, where κ is the condition number of the matrix of the corresponding linear systems and ϵ is a given error bound. More specifically, for $N = 512$ and $\epsilon = 0.5$, the complexity of our algorithm is $O(2^{53.7}\kappa^2)$ for Grain-128, and $O(2^{54.2}\kappa^2)$ for Grain-128a. Since the value of κ is related to the corresponding keystream segment, κ may be very large according to the keystream segment we choose, which leads to a very large complexity of our attack, even exceeding the complexity of the exhaustive attack. Because the number of rows and columns of the coefficient matrix of the linear equation system corresponding to the Grain algorithm is too large, we cannot effectively calculate the value of κ of this matrix. So at the end of the paper, we give a toy example of Grain-128 to estimate κ, and we take 50 keystream segments and the minimum $\kappa = 2^{5.9932}$. This result shows that the matrixes generated by a part of keystream segments do have small κ, which means that Grain-like algorithms have a security risk under the attack of the HHL algorithm. Specifically, compared to attacks like [25–28], when κ is small, our attack could have a less complexity. Compared to [29], our attack requires less keystream. And when $\kappa \leq 2^5$, our attack has a less complexity than using the Grover algorithm.

2 HHL algorithm and its modified version

2.1 HHL algorithm

HHL algorithm [17] is a quantum algorithm to solve linear systems of equations over $\mathbb{C}$ such as

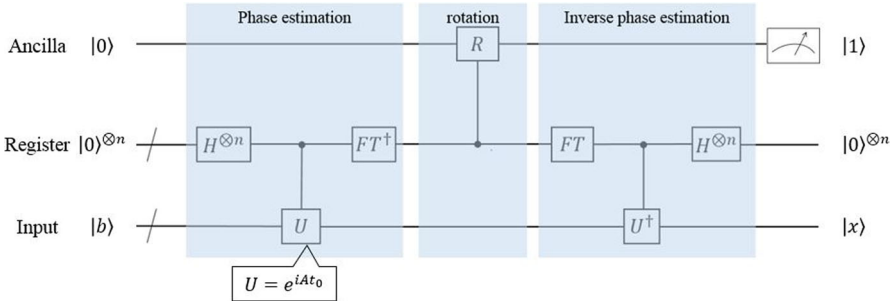

Fig. 1 HHL algorithm quantum circuit

$$A\vec{x} = \vec{b}$$

effectively, where A is the coefficient matrix and $\vec{x}$ and $\vec{b}$ are two vectors. This algorithm can achieve exponential acceleration compared to classical algorithms under certain conditions.

An s-sparse matrix is defined as a matrix in which the number of nonzero entries of each row and column is less than s. The condition number κ of a matrix is defined as the reciprocal of the absolute value of the minimum eigenvalue of the matrix when the maximum eigenvalue of the matrix is rescaled to an absolute value of 1. When a matrix is ill-conditioned, its condition number is large and vice versa.

Let $A \in \mathbb{C}^{M\times N}$ be an $M \times N$ s-sparse matrix and condition number κ. The singular value decomposition of A is defined as:

$$A = \sum_{j=1}^{n} \lambda_j |u_j\rangle\langle v_j|,$$

where $|u_j\rangle$ and $\langle v_j|$ are eigenvectors of $A^\dagger A$, $\lambda_1, \ldots, \lambda_n$ are singular values of A. For the linear equation system $A|x\rangle = |b\rangle$, HHL algorithm will give an approximation to the solution state $|x\rangle$ for the following vector:

$$\widetilde{x} = \sum_{j=1}^{n} \lambda_j^{-1} |v_j\rangle\langle v_j|b\rangle.$$

The computational complexity of the HHL algorithm is $O(\log(N+M)s\kappa^2/\epsilon)$ where ϵ denotes the given error bound and $\epsilon \in (0, 1)$. In [30], a new version of the HHL algorithm was given with complexity $O(log(N+M)s\kappa/\epsilon^3)$ (Fig. 1).

Notice that for an arbitrary vector $\vec{b}$ in linear equation system $A\vec{x} = \vec{b}$, there is no effective way to generate the quantum state $|b\rangle$. So we introduce a modified version of the HHL algorithm that all quantum states in the input step can be generated.

2.2 A modified version of HHL algorithm

This modified version of the HHL algorithm is given in [20]. It has a similar complexity, and its procedures are as follows: for an equation system $A\vec{x} = \vec{b}$, we assume $b = (1, \ldots, 1, 0, \ldots, 0)$, where the first ρ entries are 1.

Step 1 Let $\sigma = 2^{\lceil \log_2 \rho \rceil}$. Add $\sigma - \rho$ ones to b, and add $\sigma - \rho$ zero rows to A after the ρ-th row. We obtain the equation system $B\vec{x} = \vec{c}$. Because $B^\dagger B = A^\dagger A$, this step does not change the nonzero eigenvalues and the eigenvectors of $B^\dagger B$ are the same as $A^\dagger A$, so the solutions of this system $B\vec{x} = \vec{c}$ we get are same as the solutions of $A\vec{x} = \vec{b}$ we get by HHL algorithm.

Step 2 Add more zero rows to B and C such that $B \in \mathbb{C}^{2^\eta \times N}$ and $c \in \mathbb{C}^{2^\eta}$ where $\eta = \lceil \log_2 (M + \sigma + \rho) \rceil$.

Step 3 We adjust the size of c to make it satisfy the normalization condition that $|c| = 1$ and represent it in a quantum state way $|c\rangle$. Then we get the following equation system:

$$\frac{B}{\sqrt{\sigma}} |x\rangle = |c\rangle , \tag{1}$$

which has the same solution as $A\vec{x} = \vec{b}$.

Step 4 Solve the equation system (1) with HHL quantum algorithm.

It is easy to see that, unlike the original HHL algorithm, in this algorithm the quantum state $|c\rangle$ can be generated as follows:

$$|c\rangle = \otimes_{i=1}^{\eta - \lceil \log_2 \rho \rceil} |0\rangle \otimes_{i=1}^{\lceil \log_2 \rho \rceil} (H |0\rangle),$$

where H is the Hadamard operator.

In step 3, dividing B by $\sqrt{\sigma}$ takes $O(T_\rho)$ complexity. In step 4, the complexity of using the HHL algorithm is: $O(\log (N + 2^\eta) \frac{s\kappa^2}{\epsilon}) = O(\log (N + M) \frac{s\kappa^2}{\epsilon})$, and the complexity of other steps can be ignored.

Theorem 1 ([17]) *the complexity of the modified HHL algorithm is*

$$O\left(\log (N + M) \frac{s\kappa^2}{\epsilon} + T_\rho\right), \tag{2}$$

where T_ρ is the number of nonzero elements in the first ρ rows of A.

When ρ is small, the complexity is approximated as

$$O\left(\log (N + M) \frac{s\kappa^2}{\epsilon}\right).$$

3 Grain-128 algorithm

Grain-128 is a version of the Grain algorithm. It employs a 128-bit key and a 96-bit initial vector. It contains a 128-bit linear feedback shift register (LFSR), a 128-bit nonlinear feedback shift register (NFSR) and a Boolean function $h(x)$.

The feedback polynomial of the LFSR and the NFSR is denoted by $f(x)$ and $g(x)$, where

$$f(x) = 1 \oplus x^{32} \oplus x^{47} \oplus x^{58} \oplus x^{90} \oplus x^{121} \oplus x^{128}, \tag{3}$$

and

$$\begin{aligned} g(x) =& 1 \oplus x^{32} \oplus x^{37} \oplus x^{72} \oplus x^{102} \oplus x^{128} \oplus x^{44}x^{60} \oplus x^{61}x^{125} \oplus \\ & \oplus x^{63}x^{67} \oplus x^{69}x^{101} \oplus x^{80}x^{88} \oplus x^{110}x^{111} \oplus x^{115}x^{117}, \end{aligned} \tag{4}$$

where $\oplus$ denotes the addition over GF(2) (the finite field consisting of 0 and 1) in the above formula.

At the ith instance, $i \geq 0$, the bits in LFSR are denoted by $s_i, s_{i+1}, \ldots, s_{i+127}$, and the bits in NFSR are denoted by $b_i, b_{i+1}, \ldots, b_{i+127}$. According to (3) and (4), the state update functions of LFSR and NFSR are:

$$s_{i+128} = s_i \oplus s_{i+7} \oplus s_{i+38} \oplus s_{i+70} \oplus s_{i+81} \oplus s_{i+96}, \tag{5}$$

and

$$\begin{aligned} b_{i+128} =& s_i \oplus b_i \oplus b_{i+26} \oplus b_{i+56} \oplus b_{i+91} \oplus b_{i+96} \oplus b_{i+3}b_{i+67} \oplus b_{i+11}b_{i+13} \oplus \\ & \oplus b_{i+17}b_{i+18} \oplus b_{i+27}b_{i+59} \oplus b_{i+40}b_{i+48} \oplus b_{i+61}b_{i+65} \oplus b_{i+68}b_{i+84}. \end{aligned} \tag{6}$$

The Boolean function $h(x)$ has 9 variables, two of which are from the NFSR, and the other 7 are from the LFSR. Define $h(x)$ as follows:

$$h(x) = x_0x_1 \oplus x_2x_3 \oplus x_4x_5 \oplus x_6x_7 \oplus x_0x_4x_8, \tag{7}$$

where $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$ correspond to $b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}$ and s_{i+95}, respectively.

Take 7 variables: $b_{i+1}, b_{i+15}, b_{i+36}, b_{i+45}, b_{i+64}, b_{i+73}, b_{i+89}$ from the NFSR, one variable s_{i+93} from the LFSR, and the 1-bit output of the Boolean function $h(x)$. 1-bit keystream can be obtained by doing modulo 2 addition on the above 9 variables. We denote the keystream as k_i, where

$$\begin{aligned} k_i =& b_{i+2} \oplus b_{i+15} \oplus b_{i+36} \oplus b_{i+45} \oplus b_{i+64} \oplus b_{i+73} \oplus b_{i+89} \oplus s_{i+93} \oplus \\ & \oplus h(b_{i+12}, s_{i+8}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}, s_{i+95}). \end{aligned} \tag{8}$$

According to (5), (6), (7), (8), we can see that the structure of the Grain-128 algorithm is very regular, and we can easily express it as a nonlinear equation system and calculate its complexity. We focus on the quantum security of the Grain-128 keystream generation algorithm against the HHL algorithm and its modification.

Below we use the modified HHL algorithm in [19,20] to attack Grain-128. For convenience, in the rest of this paper, for a function f, we use $deg(f)$ and $\#f$ to

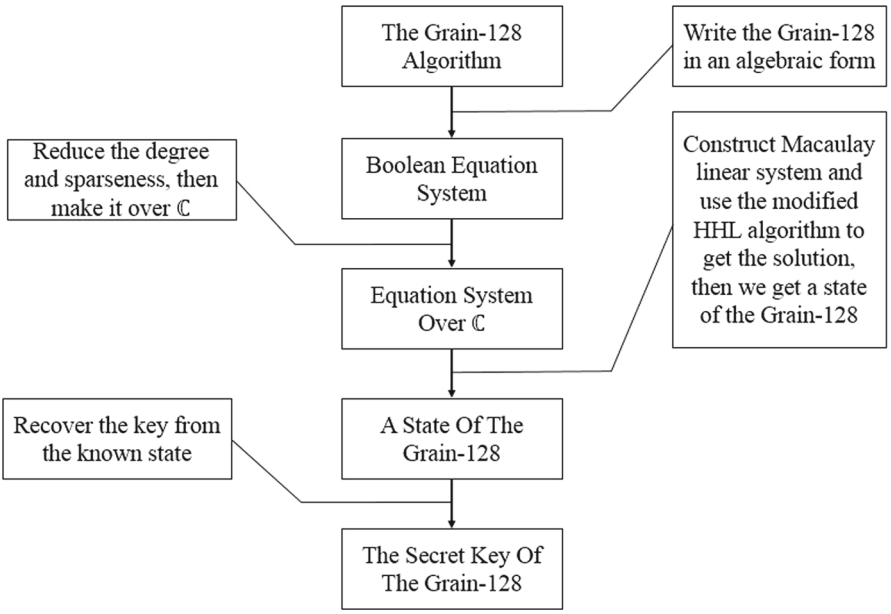

Fig. 2 Steps in this article

denote its degree and the sparseness (number of terms). In order to achieve this goal, we have to transform the Grain-128 algorithm into a linear equation system over $\mathbb{C}$, which can be solved by the modified HHL algorithm, and then, we use the modified HHL algorithm to get a state of the Grain-128 algorithm and furthermore get the secret key of the Grain-128 algorithm. The specific steps are as follows (Fig. 2).

4 Algebraic description of Grain-128

Assume that the attacker has a string of plaintext and corresponding ciphertext generated from a state $\{b_i, b_{i+1}, \ldots, b_{i+127}, s_i, s_{i+1}, \ldots, s_{i+127}\}$ where b_{i+j}, for $0 \leq j \leq 127$, denote the state in the NFSR and s_{i+j}, for $0 \leq j \leq 127$, denote the state in LFSR. Then, we can get the keystream section $K = \{k_i, k_{i+1}, \ldots, k_{i+N-1}\}$ through a simple bitwise XOR operation between the plaintext and ciphertext. Notice that the keystream section K can be completely determined by the state $\{b_i, b_{i+1}, \ldots, b_{i+127}, s_i, s_{i+1}, \ldots, s_{i+127}\}$. For every $k_j \in K$, $i \leq j \leq i + N - 1$, we can construct an equation about $\boldsymbol{b} = \{b_i, b_{i+1}, \ldots, b_{i+127}\}$ and $\boldsymbol{s} = \{s_i, s_{i+1}, \ldots, s_{i+127}\}$, where the entries in the state $\{b_i, b_{i+1}, \ldots, b_{i+127}, s_i, s_{i+1}, \ldots, s_{i+127}\}$ are viewed as the variables. For any N-bit keystream segment K, we can obtain an equation system $\mathscr{F}_1 = \mathbf{0}$ consisting of

$$f_1(\boldsymbol{b}, \boldsymbol{s}) \oplus k_i = 0,\ f_2(\boldsymbol{b}, \boldsymbol{s}) \oplus k_{i+1} = 0, \ldots,\ f_N(\boldsymbol{b}, \boldsymbol{s}) \oplus k_{i+N-1} = 0.$$

For simplicity, we denote $f_j(\boldsymbol{b}, \boldsymbol{s}) \oplus k_{i+j-1}$ as f_j, that is, the $\mathscr{F}_1$ is simplified as $\{f_1, f_2, \ldots, f_N\}$. Obviously, the equation system $\mathscr{F}_1 = \mathbf{0}$ has at least one solution. As N increases, the number of solution decreases, and when N is large enough, there is only one solution for the system $\mathscr{F}_1 = \mathbf{0}$. Here we assume that N is large enough such that $\mathscr{F}_1 = \mathbf{0}$ has an unique solution.

We can see that $\mathscr{F}_1 = \mathbf{0}$ is an algebraic form of the Grain-128 algorithm, but if we write down $\mathscr{F}_1$, we will easily find that both $\#f_j$ and $deg(f_j)$ are large when j is a large number. For example, we can roughly estimate the degree and the sparseness for f_{255}: $deg(f_{255}) \geq 200$ and $\#f_{255} \geq 7^{12}$. In the following, to make it easy to be represented and operated, we transform the $\mathscr{F}_1$ into a new equation system with small sparseness and degree.

For every $k_j \in K, i \leq j \leq i+N-1$, from (7) and (8), we get $f_{j,1} = 0$ as follows:

$$\begin{aligned} f_{j,1} \; &= b_{j+2} \oplus b_{j+15} \oplus b_{j+36} \oplus b_{j+45} \oplus b_{j+64} \oplus b_{j+73} \oplus b_{j+89} \oplus s_{j+93} \oplus \\ &\quad \oplus h(b_{j+12}, s_{j+8}, s_{j+20}, b_{j+95}, s_{j+42}, s_{j+60}, s_{j+79}, s_{j+95}) \oplus k_j = 0. \end{aligned}$$

For every $k_j \in K, i \leq j \leq i+N-2$, from (5) and (6), we get $f_{j,2} = 0$ and $f_{j,3} = 0$, respectively,

$$\begin{aligned} f_{j,2} &= b_{j+128} \oplus s_j \oplus b_j \oplus b_{j+26} \oplus b_{j+56} \oplus b_{j+91} \oplus b_{j+96} \oplus \\ &\quad \oplus b_{j+3}b_{j+67} \oplus b_{j+11}b_{j+13} \oplus b_{j+17}b_{j+18} \oplus b_{j+27}b_{j+59} \oplus \\ &\quad \oplus b_{j+40}b_{j+48} \oplus b_{j+61}b_{j+65} \oplus b_{j+68}b_{j+84} = 0. \\ f_{j,3} &= s_{j+128} \oplus s_j \oplus s_{j+7} \oplus s_{j+38} \oplus s_{j+70} \oplus s_{j+81} \oplus s_{j+96} = 0. \end{aligned}$$

where all s_j's and b_j's appearing in the above equations are viewed as variables. All the above equations form a new equation system denoted by $\mathscr{F}_2 = \mathbf{0}$.

Note that in this system, we can substitute $\{b_j, b_{j+1}, \ldots, b_{j+127}, s_j, s_{j+1}, \ldots, s_{j+127}\}$ into $\{b_{j+128}, b_{j+129}, \ldots, b_{j+N+126}, s_{j+128}, s_{j+129}, \ldots, s_{j+N+126}\}$ by using the specification of the clocking functions in the NFSR and LFSR; then, we can get $\mathscr{F}_1 = \mathbf{0}$. In summary, we can get the following lemma.

Lemma 1 *$\mathscr{F}_2 = \mathbf{0}$ is equivalent to $\mathscr{F}_1 = \mathbf{0}$ and it contains $3N - 2$ equations; the maximum degree of $\mathscr{F}_2$ is 3 and the maximum sparseness is 14.*

For convenience, we rearrange the above polynomials f in the following order: $f_1 = f_{1,1}, \ldots, f_N = f_{N,1}, f_{N+1} = f_{1,2}, \ldots, f_{2N-1} = f_{N-1,2}, f_{2N+1} = f_{1,3}, \ldots, f_{3N-1} = f_{N-1,3}$, and use $\{x_1, x_2, \ldots, x_{2N+254}\}$ to denote variables in $\mathscr{F}_2$.

5 Reduction of Boolean polynomial system

Above we get an equation system $\mathscr{F}_2 = \mathbf{0}$, this system is a Boolean equation system over GF(2). In order to use the HHL algorithm to solve this system, we must rewrite it in a suitable form. In this section, we use the method in [20] to reduce the system $\mathscr{F}_2 = \mathbf{0}$ to a quadratic equation system over $\mathbb{C}$.

5.1 Reduce degree

We reduce every f in $\mathscr{F}_2$ into a quadratic polynomial. The process is as follows:

For a polynomial f in $\mathscr{F}_2$, we denote the kth monomial in f as

$$X_k^r = \prod_{j=1}^{r} x_{kj} = x_{k1}x_{k2}\ldots x_{kr}. \tag{9}$$

For each monomial $X_k^r (r \geq 3)$, by introducing some new variables $v_i \in \mathrm{GF}(2)$, we rewrite X_k^r as follows:

$$\begin{aligned} &v_1 \oplus x_{k1}x_{k2};\\ &v_2 \oplus v_1 x_{k3};\\ &v_3 \oplus v_2 x_{k4};\\ &\ldots\ldots;\\ &v_{r-2} \oplus v_{r-3}x_{k(r-1)};\\ &X_k^r = v_{r-2}x_{kr}. \end{aligned}$$

The monomials in $\mathscr{F}_2$ with degree 3 only appear in equations containing $h(x)$, which is the first N polynomials in $\mathscr{F}_2$. Each of them contains a monomial with degree 3; therefore, we introduce N variables and N polynomials with 2 degrees and 2 sparseness into $\mathscr{F}_2$. Then, we get a new polynomial system $\mathscr{F}_3$.

Lemma 2 *The degree of elements in $\mathscr{F}_3$ is ≤ 2, and $\mathscr{F}_3 = \mathbf{0}$ is equivalent to $\mathscr{F}_2 = \mathbf{0}$.*

For convenience, we express the introduced equations as: $f_{3N-1} = 0, \ldots,$ $f_{4N-2} = 0$ and we express all invariables in $\mathscr{F}_3$ as $\{x_1, x_2, \ldots, x_{3N+254}\}$.

5.2 Reduce sparseness

In this step, we reduce $\mathscr{F}_3$ to polynomials with 3 or less sparseness.

Let

$$f = \prod_{k_1} x_{k_1} \oplus \prod_{k_2} x_{k_2} \oplus \cdots \oplus \prod_{k_r} x_{k_r}, \tag{10}$$

f is a Boolean polynomial and $\#f = r, r \geq 4$.

In order to reduce the sparseness of (10), we find a set of equations which is equivalent to (10). We introduce some new variables u_1 and rewrite f as follows:

$$\begin{aligned}
&\prod_{k_1} x_{k_1} \oplus u_1;\\
&\prod_{k_2} x_{k_2} \oplus u_1 \oplus u_2;\\
&\cdots\cdots\\
&\prod_{k_{r-1}} x_{k_{r-1}} \oplus u_{r-2} \oplus u_{r-1};\\
&\prod_{k_r} x_{k_r} \oplus u_{r-1}.
\end{aligned} \tag{11}$$

Further simplifying polynomials in (11), we get:

$$\begin{aligned}
&\prod_{k_2} x_{k_2} \oplus \prod_{k_1} x_{k_1} \oplus u_2;\\
&\prod_{k_3} x_{k_3} \oplus u_2 \oplus u_3;\\
&\cdots\cdots\\
&\prod_{k_{r-2}} x_{k_{r-2}} \oplus u_{r-3} \oplus u_{r-2};\\
&\prod_{k_{r-1}} x_{k_{r-1}} \oplus u_{r-2} \oplus \prod_{k_r} x_{k_r}.
\end{aligned} \tag{12}$$

We use (12) to replace (10) in $\mathscr{F}_3$. Through the above operations, f is replaced by $r-2$ polynomials with sparseness 3 and $r-3$ new variables.

For each polynomial $f_i \in \mathscr{F}_3$, we have $\#f_i = 14(1 \le i \le 2N-1)$, $\#f_i = 7(2N \le i \le 3N-2)$ and $\#f_i = 2(3N-1 \le i \le 4N-2)$. By performing the above operation for all polynomials in $\mathscr{F}_3$ with sparseness ≥ 3, we get a new polynomial system consisting of

$$12 \times N + 12 \times (N-1) + 5 \times (N-1) + N = 30N - 17$$

polynomials with maximum sparseness 3 and $29N+254$ variables. We denote this new polynomial system as $\mathscr{F}_4$. Combining Lemma 2, we have the following conclusion.

Lemma 3 *$30N-17$ polynomials and $29N+254$ variables are in $\mathscr{F}_4$, the degree of polynomials in $\mathscr{F}_4$ are ≤ 2 and the sparseness of polynomials in $\mathscr{F}_4$ are ≤ 3. $\mathscr{F}_4 = \mathbf{0}$ is equivalent to $\mathscr{F}_3 = \mathbf{0}$.*

Proof We add all the polynomials in (11); then, we get f, so $\mathscr{F}_4 = \mathbf{0}$ is equivalent to $\mathscr{F}_3 = \mathbf{0}$. □

For convenience, we use $\{x_1, \ldots, x_{29N+254}\}$ to denote the variables u and x together.

5.3 Reduce Boolean equation system to equation system over $\mathbb{C}$

For an equation $f = 0$, where $f \in \mathscr{F}_4$, according to Lemma 3, we know that $\#f \leq 3$. It means that when $f = 0$ has solutions over GF(2), the value of f is 0 or 2 over $\mathbb{C}$. So we construct the following polynomial over $\mathbb{C}$:

$$g = f(f-2). \tag{13}$$

It is not difficult to see that the solutions of $g = 0$ contain the solutions of $f = 0$. We replace each polynomial $f \in \mathscr{F}_4$ by g in (13); then, we get a new polynomial system $\mathscr{F}_5$ over $\mathbb{C}$, where all solutions of $\mathscr{F}_4 = \mathbf{0}$ are in the solutions of $\mathscr{F}_5 = \mathbf{0}$.

Because $\mathscr{F}_4 = \mathbf{0}$ is a Boolean equation system, all variables x_i in $\mathscr{F}_4$ take values over GF(2). In other words, all 0, 1 solutions in $\mathscr{F}_5 = \mathbf{0}$ are the solutions of $\mathscr{F}_4 = \mathbf{0}$. So for each variable in $\mathscr{F}_5$, we first use x to replace x^m, and then, we construct the following polynomial:

$$x^2 - x. \tag{14}$$

According to Lemma 3, there are $29N + 254$ variables in $\mathscr{F}_4$, so we have to construct $29N + 254$ polynomials like (14). Adding all the polynomials, we construct into $\mathscr{F}_5$, and we obtain a new polynomial system $\mathscr{F}_6$. Through simple calculations, we get the following conclusion.

Theorem 2 *$59N + 237$ polynomials and $29N + 254$ variables are in $\mathscr{F}_6$, $\mathscr{F}_6$ is a polynomial system over $\mathbb{C}$ and $\mathscr{F}_6 = \mathbf{0}$ is equivalent to $\mathscr{F}_4 = \mathbf{0}$.*

Combining Lemmas 1, 2, 3 and Theorem 2, we get the following conclusion.

Theorem 3 *$\mathscr{F}_6 = \mathbf{0}$ is equivalent to $\mathscr{F}_1 = \mathbf{0}$.*

In other words, if we solve the equation system $\mathscr{F}_6 = \mathbf{0}$, then we solve the equation system $\mathscr{F}_1 = \mathbf{0}$ and then break the Grain-128 algorithm.

6 Solve nonlinear polynomial system over $\mathbb{C}$

In this section, we first construct the Macaulay linear system of $\mathscr{F}_6$; then, we solve the Macaulay linear system by using the modified HHL quantum algorithm. The method comes from [20].

6.1 Construct Macaulay matrix

For a nonlinear polynomial system $\mathscr{F}$, let $\mathbb{X}$ be the set of variables in $\mathscr{F}$, i.e., $\mathbb{X} = \{x_1, x_2, \ldots, x_n\}$, and m be the set of all the monomials of variables in $\mathbb{X}$. We use the ascending degree reverse lexicographic (DRL) monomial order to arrange m, that is: $m_0 = 1, m_1 = x_1, \ldots, m_n = x_n, m_{n+1} = x_1^2, \ldots$, where $m_i \in m$. Let $m_{\leq d} = \{m_0, m_1, \ldots, m_{Q_d - 1}\}$ be the set of monomials in m whose degree $\leq d$, where

$$Q_d = \binom{d+n}{n} = O((d+n)^{\min\{n,d\}}). \tag{15}$$

Let $\{f_1, f_2, \ldots, f_r\}$ be polynomials in $\mathscr{F}$ with $d_i = deg(f_i)(i = 1, \ldots, r)$, and $D \geq max_{i=1}^{r} d_i$, we can construct a Macaulay linear system for $\mathscr{F}$:

$$\begin{pmatrix} m_0 f_1 \\ \ldots \\ m_0 f_r \\ m_1 f_1 \\ \ldots \\ m_{Q_D - d_r - 1} f_r \end{pmatrix} \begin{pmatrix} \ldots \\ \ldots \\ \ldots \\ \ldots \\ \ldots \\ \ldots \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \ldots \\ m_{Q_D - 1} \end{pmatrix} = \begin{pmatrix} -f_1(0) \\ \ldots \\ -f_r(0) \\ 0 \\ \ldots \\ 0 \end{pmatrix},$$

denoted as

$$\mathscr{M}_{\mathscr{F},D} \boldsymbol{m}_D = \boldsymbol{b}_{\mathscr{F},D}, \tag{16}$$

where $\boldsymbol{m}_D = (m_1, m_2, \ldots, m_{Q_D - 1})^T$ is the vector that contains the variables $m_1, m_2, \ldots, m_{Q_D - 1}$ and $\mathscr{M}_{\mathscr{F},D}$ is the coefficient matrix of $\boldsymbol{m}_D$ composed by the coefficients of m_i in $m_0 f_1, \ldots, m_0 f_r, m_1 f_0, \ldots, m_{Q_D - d_r - 1} f_r$.

$\mathscr{M}_{\mathscr{F},D}$ is called the Macaulay matrix of $\mathscr{F}$, and (16) is called the Macaulay linear system of $\mathscr{F}$.

Then, we introduce the solving degree:

Definition 1 ([20]) Let $\mathscr{F} = \{f_1, \ldots, f_r\}$. The solving degree of $\mathscr{F}$ means the maximum degree that occurs during the Gröbner basis computing process for $\mathscr{F}$ by Buchberger's algorithm.

We use $Sdeg(\mathscr{F})$ to denote the solving degree of $\mathscr{F}$.

Definition 2 ([20]) For a polynomial system $\mathscr{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$, $\mathscr{F}$ satisfies Lazard's condition meaning that $F^h = \{f_1^h, \ldots, f_r^h\}$ has a finite number of solutions in the projective space where f^h denotes the homogenization of f.

Relying on Lazard's condition, [31] gives the upper bound of the solving degree:

Theorem 4 ([31]) *Denote $d = max_i d_i$, $Sdeg(\mathscr{F}) \leq (n+1)(d-1)+2$ if $\mathscr{F}$ satisfies Lazard's condition.*

Theorem 5 ([32,33]) *In the F4 algorithm and the XL algorithm, D is the solving degree of $\mathscr{F}$ when the Gröbner basis of $(\mathscr{F})$ can be obtained from $\mathscr{M}_{\mathscr{F},D}\boldsymbol{m}_D = \boldsymbol{b}_{\mathscr{F},D}$ by using Gaussian elimination over $\mathbb{C}$.*

[20] gives the form of the solution of the Macaulay linear system, which has a unique solution.

Theorem 6 ([20]) *Let $\mathscr{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ and $\mathscr{F} = \mathbf{0}$ has a unique solution $\boldsymbol{a}$, let $D = Sdeg(\mathscr{F})$, the form of the solution of the Macaulay linear system $\mathscr{M}_{\mathscr{F},D}\boldsymbol{m}_D = \boldsymbol{b}_{\mathscr{F},D}$ is*

$$\hat{\boldsymbol{m}}_D = \boldsymbol{m}_D(\boldsymbol{a}) + \sum_{m_k \in \mathscr{N}_{\mathscr{F},D}} \mu_k \boldsymbol{e}_k,$$

where $\mathscr{N}_{\mathscr{F},D}$ denotes the set of monomials m_k in $\boldsymbol{m}_D$ such that the k-th column of $\mathscr{M}_{\mathscr{F},D}$ is $\mathbf{0}$, μ_k is an arbitrary complex number and $\boldsymbol{e}_k$ is the k-th unit vector in $\mathbb{C}^{Q_D - 1}$.

It can be seen that the solutions of $\mathscr{F} = 0$ are equal to the solution of the first r variables in $\mathscr{M}_{\mathscr{F},D}\boldsymbol{m}_D = \boldsymbol{b}_{\mathscr{F},D}$, so we can get a solution of $\mathscr{F} = 0$ using the modified HHL algorithm. We can get the complexity according to (2) and (16):

$$O\left(\log\left(\sum_{i=1}^{r} Q_{D-d_i} + (Q_D - 1)\right)\frac{(\sum_{i=1}^{r} \#f_i)\kappa^2}{\epsilon} + T_\rho\right)$$
$$= O(\log(n+D)min\{n, D\}T\left(\frac{\kappa^2}{\epsilon}\right) + T_\rho),$$

where $T = \sum_{i=1}^{r} \#f_i \geq T_\rho$. According to Theorem 4, let D take its upper bound, i.e., $D = (n+1)(d-1)+2$ and ignore T_ρ, the final complexity is

$$O(\log(nd)nT\kappa^2/\epsilon). \tag{17}$$

Therefore, we can easily transform $\mathscr{F}_6$ into a linear equation system and use HHL algorithm to get the solutions. By the way, according to [20], the scale of Macaulay matrix is

$$\left(\sum_{i=1}^{r} Q_{D-d_i}\right) \times (Q_D - 1). \tag{18}$$

Until here, we successfully use HHL algorithm in Grain-128 algorithm, but there is still an unresolved problem: the solution of $\mathscr{F}_6$ is stored in register in a quantum superposition state $|x\rangle = \sum a\,|e\rangle$. In the next step, we give an algorithm to find the solutions of $\mathscr{F}_6$.

6.2 Find a solution

For an equation system $\mathscr{F} = \mathbf{0}$ which is known has solutions, we give the following algorithm to find one of its solutions:

Input $\mathscr{F} \in \mathbb{C}(\mathbb{X})$ where $\mathbb{X} = \{x_1, \ldots, x_n\}$, an error bound $\epsilon \in (0, 1)$.
Output A solution of $\mathscr{F} = \mathbf{0}$.
Step 1 If $\mathscr{F}(\mathbf{0}) = \mathbf{0}$, return $\mathbf{0}$. If $\mathscr{F}(\mathbf{1}) = \mathbf{0}$, return $\mathbf{1}$.
Step 2 Let $\mathscr{F}' = \mathscr{F}$.
Step 3 Use the modified HHL algorithm to the Macaulay linear system of $\mathscr{F}'$, we obtain a state $|\hat{m}\rangle$ with the error bound $\sqrt{\epsilon/n}$, where ϵ is an arbitrary number chosen in (0, 1).
Step 4 Measuring $|\hat{m}\rangle$, we obtain a state $|e_k\rangle$ or equivalently, the number k.
Step 5 Let $m_k = \prod_{i=1}^{u_k} x_{n_1}$. Set $x_{n_1} = 1$ in $\mathscr{F}' \in \mathbb{C}(\mathbb{X})$ for $i = 1, \ldots, u_k$.
Step 6 Remove 0 from $\mathscr{F}'$. Set $\mathbb{X} = \mathbb{X} \setminus \{x_{n_i} | i = 1, \ldots, u_k\}$.
Step 7 If $1 \in \mathscr{F}'$ or $\mathbb{X} = \emptyset$, then roll back to Step 2.
Step 8 If $\mathscr{F}' \neq \emptyset$ and $\mathscr{F}'(\mathbf{0}) = \mathbf{0}$, then go to Step 3.
Step 9 Return $(a_1, \ldots, a_n)$ where $a_i = 0$ if $x_i \in \mathbb{X}$ else $a_i = 1$.

We explain the principle of the algorithm. In Step 1 to Step 3, we construct Macaulay matrix for $\mathscr{F}' = \mathbf{0}$ and use the modified HHL algorithm on it; then, we can get a

solution of $\mathscr{F}' = \mathbf{0}$ as a quantum state. In Step 4, we measure this quantum state to collapse it to a basis state $|e_k\rangle$ with probability and get this basis state; equivalently, we find the m which is corresponding to this basis state. In Step 5, we set all the x that make up m to 1. In Step 6, we bring the determined value of x into $\mathscr{F}'$ and sort it. In Step 7 and Step 8, we test the simplified $\mathscr{F}'$. If $1 \in \mathscr{F}'$ or $\mathbb{X} = \emptyset$, this indicates that the calculation in Step 3 has gone wrong, so we go to Step 2, and if $\mathscr{F}' \neq \emptyset$ and $\mathscr{F}'(\mathbf{0}) \neq \mathbf{0}$, there exist other x equal to 1, so we go back to Step 3 to find them. If the conditions in Step 7 and Step 8 are not met, then we have found all x which are equal to 1. Then in Step 9, we output the solution we get.

Next, we analyze the complexity of this algorithm. Firstly, we give the result of complexity:

Theorem 7 *The complexity of this algorithm is*

$$O((n+T)n^{2.5}e^{\epsilon}\kappa^2/\epsilon^{0.5}), \tag{19}$$

where κ is the maximal condition number for all Macaulay matrixes in Step 3.

To prove Theorem 7, we perform the following analysis.

We first analyze the composition of the number of algorithm rounds.

Lemma 4 *The time the loop from Step 3 to Step 8 occurs $\leq n$ and the time the loop from Step 2 to Step 7 occurs $= e^{\epsilon}$ by averaging.*

Proof Because if the algorithm proceeds to Step 8, at least one value of x can be obtained, the situation that the loop from Step 3 to Step 8 occurs can only be performed at most n times.

Let $|m_D\rangle = \sum_{j=1}^{Q_D-1} \alpha_j |e_j\rangle$ be the solution state and $|\hat{m}_D\rangle = \sum_{j=1}^{Q_D-1} \beta_j |e_j\rangle$ be the approximate state obtained with the modified HHL algorithm. When we measure $|\hat{m}_D\rangle$, we can get a state $|e_k\rangle$, and the probability of $m_k(a) = 0$ is $\left\| \sum_{k,\alpha_k=0} \beta_k |e_k\rangle \right\|^2 < \left\| \sum_{k=1}^{Q_D-1} (\beta_k - \alpha_k) |e_k\rangle \right\|^2 = \left\| \hat{m}_D - m_D \right\|^2 < \epsilon/n$. So the probability of the situation that the loop from Step 2 to Step 7 occurs is ϵ/n for each round. Here we assume that the loop from Step 3 to Step 8 runs n times so the probability of not entering Step 7 is $(1-\epsilon/n)^n$. For $\epsilon \in (0,1)$, n is a large number; this probability is approximately equal to

$$\lim_{n\to\infty} (1-\epsilon/n)^n = \lim_{n\to\infty} \left(\frac{\frac{n}{\epsilon}-1}{\frac{n}{\epsilon}} \right)^n = \lim_{n\to\infty} \frac{1}{\left(1+\frac{1}{\frac{n}{\epsilon}-1}\right)^{(\frac{n}{\epsilon}-1)\frac{n}{\frac{n}{\epsilon}-1}}}$$

$$= \lim_{n\to\infty} \frac{1}{e^{\frac{n}{\frac{n}{\epsilon}-1}}} = \frac{1}{e^{\epsilon}}.$$

In this algorithm, n is usually a large number, so we use $\frac{1}{e^{\epsilon}}$ to approximate the probability. By averaging, we assume that the time the loop from Step 2 to Step 7 occurs is e^{ϵ}. □

Then, we calculate the complexity of a loop.

Lemma 5 *The complexity of each loop is:*

$$O\left(\log\left(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1\right)(2n+T)\kappa^2\sqrt{n/\epsilon}\right). \tag{20}$$

Proof Since the complexity of the other steps is negligible, the complexity of this loop is equal to the complexity of Step 3. For $D \leq (n+1)(d-1)+2$ and the $\mathscr{M}_{\mathscr{F},D}$ is of dimension $(\sum_{i=1}^{r} Q_{D-d_i}) \times (Q_D - 1)$ and $(2n+T)$-sparseness, by Eqs. (2) and (17), the complexity of Step 3 is

$$O\left(c\log\left(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1\right)(2n+T)\kappa^2\sqrt{n/\epsilon}\right)$$

This is also the complexity of each loop. □

By the above discussion and (20), the total complexity of this algorithm is:

$$\begin{aligned} &e^{\epsilon}\sum_{j=0}^{n-1}\left(c\log\left(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1\right)(2(n-j)+T)\kappa^2\sqrt{\frac{n}{\epsilon}}\right) \\ &= ce^{\epsilon}\log\left(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1\right)(n(n+1)+nT)\kappa^2\sqrt{\frac{n}{\epsilon}}. \end{aligned} \tag{21}$$

Gao et al. gives the upper bound of $\log\left(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1\right)$ in [20].

Lemma 6 ([20])

$$\log\left(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1\right) \leq \sqrt{2}(n\log_2 e + \log_2 r). \tag{22}$$

According to (21) and (22), the complexity of this algorithm is:

$$\begin{aligned} &\sqrt{2}ce^{\epsilon}(n\log_2 e + \log_2 r)(n(n+1)+nT)\kappa^2\sqrt{\frac{n}{\epsilon}} \\ &= \sqrt{2}cn^{1.5}e^{\epsilon}\kappa^2\frac{1}{\epsilon^{0.5}}(n\log_2 e + \log_2 r)(n+1+T), \end{aligned}$$

and because $r \leq T$, the complexity is equal to

$$O((n+T)n^{2.5}e^{\epsilon}\kappa^2/\epsilon^{0.5}),$$

Table 1 The complexities of our algorithm

The value of N	Complexity
256	$O(2^{50.2}\kappa^2)$
320	$O(2^{51.2}\kappa^2)$
384	$O(2^{52.3}\kappa^2)$
448	$O(2^{53}\kappa^2)$
512	$O(2^{53.7}\kappa^2)$

where κ is the maximal condition number for all Macaulay matrixes in Step 3.

We use this algorithm to $\mathscr{F}_6$; then, we can obtain a solution of $\mathscr{F}_6 = 0$; this means that we get the state of the register at a certain moment. Then, we use a key recover attack in [34] so that we can find the secret key.

7 Complexity analysis

In this section, we analyze the complexity of this attack (Table 1).

The construction and the reduction steps can be completed in $O(N)$ on a classical computer, and the key recovery attack in [34] can be completed in $O(2^8)$. The complexity of these parts can be ignored.

In $\mathscr{F}_6$, according to Theorem 2, we know that $n = 29N + 254$ and $T = \sum_{i=1}^{59N+237} \#f_i = 12 \times (29N - 17) + 6 \times N + 2 \times (29N + 254) = 412N + 304$. According to (19), we can compute the complexity of this algorithm:

Theorem 8 *The total complexity of attacking Grain-128 with our algorithm is*

$$O(2^{21}N^{3.5}\kappa^2 e^{\epsilon}/\epsilon^{0.5}).$$

Proof We bring all known values into (20); then, we get the complexity of this algorithm C:

$$\begin{aligned} C &= O((29N + 254)^{2.5}(29N + 254 + 412N + 304)\kappa^2 e^{\epsilon}/\epsilon^{0.5}) \\ &= O(2^{21}N^{3.5}\kappa^2 e^{\epsilon}/\epsilon^{0.5}). \end{aligned}$$

□

Next, if we determine the values of N and ϵ, we can get a more specific value. For $\epsilon = 0.5$, the complexities of our algorithm for different N are:

We can see that this result is only related to κ, the condition number of A. If κ is small enough, the complexity of our algorithm is far less than the complexity of other algorithms such as [4,25–28]; otherwise, the complexity is very large.

8 Security analysis of Grain-128a against HHL algorithm

8.1 Grain-128a

Grain-128a is a new version of the stream cipher Grain-128; it can effectively resist all known attacks on Grain-128 before [24].

The structure of Grain-128a is similar to Grain-128. It employs a 128-bit key and a 96-bit initial vector and contains a 128-bit linear feedback shift register (LFSR), a 128-bit nonlinear feedback shift register (NFSR) and a Boolean function $h(x)$ as same as Grain-128.

We use $s_i, s_{i+1}, \ldots, s_{i+127}$ and $b_i, b_{i+1}, \ldots, b_{i+127}$ to denote the bits in LFSR and NFSR. At the ith instance, we can express Grain-128a as follows:

$$\begin{aligned}
s_{i+128} &= s_i \oplus s_{i+7} \oplus s_{i+38} \oplus s_{i+70} \oplus s_{i+81} \oplus s_{i+96},\\
b_{i+128} &= s_i \oplus b_i \oplus b_{i+26} \oplus b_{i+56} \oplus b_{i+91} \oplus b_{i+96} \oplus b_{i+3}b_{i+67} \oplus b_{i+11}b_{i+13} \oplus\\
&\quad \oplus b_{i+17}b_{i+18} \oplus b_{i+27}b_{i+59} \oplus b_{i+40}b_{i+48} \oplus b_{i+61}b_{i+65} \oplus b_{i+68}b_{i+84}\\
&\quad \oplus b_{i+88}b_{i+92}b_{i+93}b_{i+95} \oplus b_{i+22}b_{i+24}b_{i+25} \oplus b_{i+70}b_{i+78}b_{i+82},\\
h(x) &= x_0x_1 \oplus x_2x_3 \oplus x_4x_5 \oplus x_6x_7 \oplus x_0x_4x_8,
\end{aligned}$$

and

$$\begin{aligned}
k_i =& b_{i+2} \oplus b_{i+15} \oplus b_{i+36} \oplus b_{i+45} \oplus b_{i+64} \oplus b_{i+73} \oplus b_{i+89} \oplus s_{i+93} \oplus\\
&\oplus h(b_{i+12}, s_{i+8}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}, s_{i+94}).
\end{aligned}$$

where $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$ correspond to $b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}$ and s_{i+94}, respectively.

Next, we will attack Grain-128a with the method in this paper and analyze its complexity.

8.2 Security analysis of Grain-128a

We attack the Grain-128a algorithm in the same way as Sects. 4, 5, 6 and 7 in this paper. By rewriting the Grain-128a algorithm, we get a polynomial system over $\mathbb{C}$ with $73N + 223$ polynomials and $36N + 247$ variables. Further, we get $n = 30N + 247$ and $T = 486N + 230$.

According to (19), we can compute the complexity of this algorithm:

Theorem 9 *The total complexity of attacking Grain-128a with our algorithm is:*

$$O(2^{21.5}N^{3.5}\kappa^2 e^{\epsilon}/\epsilon^{0.5}).$$

It can be seen that the complexity of attacking Grain-128a with our algorithm is close to that of attacking Grain-128. This is undoubtedly a good result.

9 A toy example about κ

Based on the complexity we obtained above, it is not difficult to see that the value of κ is very important for the complexity of our algorithm. However, the row and column of the Macaulay matrix we get in this paper are too large. According to (18), if we let $N = 256$, in the worst case, the scale of the Macaulay matrix we generate is up to $2^{18388} \times 2^{18373}$. Obviously, such a large matrix cannot be effectively processed, so we build a toy example of Grain family with 8-bit secret key K and 6-bit IV and calculate its κ.

Our toy example is as follows:

Initialization phase:

$$\begin{aligned} h_i &= b_{i+1}s_{i+1} \oplus s_{i+4}s_{i+5} \oplus b_{i+1}b_{i+7}s_{i+7}, \\ k_i &= b_{i+1} \oplus b_{i+5} \oplus s_{i+5} \oplus h_i, \\ s_{i+8} &= s_i \oplus s_{i+3} \oplus s_{i+7} \oplus k_i, \\ b_{i+8} &= s_i \oplus b_i \oplus b_{i+7} \oplus b_{i+3}b_{i+5} \oplus k_i. \end{aligned}$$

Keystream generation phase:

$$\begin{aligned} h_i &= b_{i+1}s_{i+1} \oplus s_{i+4}s_{i+5} \oplus b_{i+1}b_{i+7}s_{i+7}, \\ k_i &= b_{i+1} \oplus b_{i+5} \oplus s_{i+5} \oplus h_i, \\ s_{i+8} &= s_i \oplus s_{i+3} \oplus s_{i+7}, \\ b_{i+8} &= s_i \oplus b_i \oplus b_{i+7} \oplus b_{i+3}b_{i+5}. \end{aligned}$$

where $s_i, \ldots, s_{i+7}$ and $b_i, \ldots, b_{i+7}$ are the bits in LFSR and NFSR at the ith instance and k_i is the keystream at the ith instance.

Let $N = 8$ and $D = 3$, we generate a keystream with $K = 01011011$ and IV$= 010101$ and randomly choose 50 keystream segments from it. We generate the Macaulay matrix for each keystream segments, the scale of the matrices we generate is about 4700×14500, and then, we calculate the value of κ of these matrices. The results are as follows:

From Table 2, we can see that among the 50 keystream segments, there are 3 keystream segments: 01010100, 01010101 and 10101010 that can generate small κ, which is about 2^6. This toy example shows that small κ must exist. Therefore, when we choose a keystream segment corresponding to a small κ; our attack is effective for this toy example.

For our attack, this result illustrates:

- Different keystream segments have a great influence on the condition number κ of the generated matrix.
- Keystream segments corresponding to small κ are existed.
- When we choose the keystream segment corresponding to a small κ, our attack is effective.

Table 2 The value of κ of our toy example corresponds to different keystream segments

Keystream segment	The value of κ	Keystream segment	The value of κ
00001010	$2^{50.7855}$	00010101	$2^{50.7794}$
00011000	$2^{50.8680}$	00011010	$2^{50.7511}$
00100101	$2^{50.7382}$	00101101	$2^{50.6557}$
00110001	$2^{50.8711}$	00111010	$2^{50.7985}$
01000110	$2^{50.8802}$	01001011	$2^{50.6530}$
01010100	$2^{5.9932}$	01010101	$2^{5.9932}$
01011000	$2^{50.7327}$	01011010	$2^{50.7129}$
01011011	$2^{50.7127}$	01011100	$2^{50.7172}$
01101001	$2^{50.8144}$	01101010	$2^{50.6601}$
01110101	$2^{50.6013}$	01111010	$2^{50.7896}$
10001010	$2^{50.7540}$	10010110	$2^{50.8289}$
10011101	$2^{50.7918}$	10100001	$2^{50.8169}$
10100011	$2^{50.8370}$	10100100	$2^{50.7447}$
10100111	$2^{50.6782}$	10101000	$2^{50.7739}$
10101010	$2^{6.0305}$	10101110	$2^{50.7195}$
10110100	$2^{50.7338}$	10110110	$2^{50.7991}$
10111001	$2^{50.7988}$	11000111	$2^{50.7865}$
11001010	$2^{50.6978}$	11010010	$2^{50.7914}$
11100011	$2^{50.8643}$	11100101	$2^{50.8024}$
11101010	$2^{50.7890}$	11111000	$2^{50.7851}$
00000011	$2^{50.8861}$	00011111	$2^{50.7899}$
00111000	$2^{50.8533}$	01100011	$2^{50.7881}$
01110001	$2^{50.7941}$	10000001	$2^{50.8280}$
10001110	$2^{50.7872}$	10101100	$2^{50.5908}$
11000000	$2^{50.8360}$	11100000	$2^{50.8174}$

- The keystream segments corresponding to small κ have no obvious characteristics; we only calculate them by computer through simulation. This is an open problem and needs further study.

10 Multiple solutions

In rare cases, the plaintext and ciphertext we can get are limited, that is, the system of equations we construct has multiple solutions. Next, we give a method to solve this situation by finding all the solutions.

If $\mathscr{F}_6$ has more than one solution, we construct a new equation system on the basis of $\mathscr{F}_6$. We add the corresponding variables $\bar{x}_0, \ldots, \bar{x}_{255}$ to the 256 variables, which

appear in $\mathscr{F}_1$. For every $\bar{x}_i (i = 0, \ldots, 255)$, we have equation

$$\bar{x}_i + x_i - 1 = 0.$$

Then, for every $\{x_1, x_2, \ldots, x_{255}\}$ we have got, we construct an equation:

$$\prod_{x_k=0} \bar{x}_k \prod_{x_k=1} x_k = 0.$$

Add all equations above into $\mathscr{F}_6$, we get a new system denoted by $\mathscr{F}_6'$, then we use the method on $\mathscr{F}_6'$ in Sect. 6, we can get one of the solutions of $\mathscr{F}_6$. We can see that the solutions we get in $\mathscr{F}_6$ cannot be got again in $\mathscr{F}_6'$ because it will make these equations we add above invalid. When we get a solution, we verify it, if this solution is wrong, we construct the above system and get another solution until we find the right solution.

11 Conclusions

This paper proposes a quantum algorithm based on HHL algorithm to attack Grain-128 and Grain-128a stream ciphers. Our algorithm is very flexible in selecting the location and length of plaintext and ciphertext pair. The complexity of our algorithm is only related to the condition number κ, and even if there is not enough length of plaintext and ciphertext pair, our algorithm is still valid. We give a toy example to estimate κ. The result shows that the keystream segments corresponding to small κ do exist. This makes us take a positive attitude toward our algorithm. We think that when we choose a keystream segment corresponding to a small κ, our algorithm can effectively attack Grain-128 and Grain-128a.

It is worth mentioning that there is still a room for improvement. On the basis of our algorithm, the later research can start with how to find the keystream segments corresponding to small κ and how to restore the key by using the quantum state we get by HHL algorithm.

Acknowledgements This work is supported in part by the Key Research and Development Program of Shaanxi (No. 2021ZDLGY06-04), the Natural Science Foundation of China (No. 61303217, 61502372), Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS201802).

References

1. Feynman, R.P.: Simulating physics with computers. Int. J. Theor. Phys. **21**(6), 467–488 (1982)
2. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. IEEE , pp. 124–134 (1994)
3. Grover, L.K.: A fast quantum mechanical algorithm for database search (1996). preprint arXiv:quant-ph/9605043
4. Raj G., Singh D., Madaan A.: Analysis of classical and quantum computing based on Grover and Shor algorithm. In: Satapathy S., Bhateja V., Das, S. (eds) Smart Computing and Informatics. Smart

Innovation, Systems and Technologies, vol. 78. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-5547-8_43
5. Rötteler, M.: A survey of some recent results. Informatik-Forschung und Entwicklung **21**(1–2), 3–20 (2006)
6. Montanaro, A.: Quantum algorithms: an overview. NPJ Quantum Inf. **2**, 15023 (2016)
7. Biamonte, J., Wittek, P., Pancotti, N., et al.: Quantum machine learning. Nature **549**(7671), 195 (2017)
8. Wiebe, N., Kapoor, A., Svore, K.M.: Quantum deep learning (2014). arXiv preprint arXiv:1412.3489
9. Jordan, S.P., Liu, Y.K.: Quantum cryptanalysis: shor, grover, and beyond. IEEE Secur. Priv. **16**(5), 14–21 (2018)
10. Li, H..W.: Quantum Algorithms and its Applications in Cryptography. Institute of Information Engineering Chinese Academy of Sciences, Beijing (2015)
11. Alagic, G., Gagliardoni, T., Majenz, C.: Unforgeable quantum encryption. In: Annual international conference on the theory and applications of cryptographic techniques, pp. 489–519. Springer, Cham (2018)
12. Alagic, G., Russell, A.: Quantum-secure symmetric-key cryptography based on hidden shifts. In: Annual international conference on the theory and applications of cryptographic techniques, pp. 65–93. Springer, Cham (2017)
13. Broadbent, A., Schaffner, C.: Quantum cryptography beyond quantum key distribution. Des. Codes Cryptogr. **78**(1), 351–382 (2016)
14. Bonnetain, X., Plasencia, M.N., Schrottenloher, A.: Quantum security analysis of AES. IACR Trans. Symmetric Cryptol. **2019**, 55–93 (2019)
15. Xie, H.Q., Yang, L.: Using Bernstein Vazirani algorithm to attack block ciphers. Des. Codes Cryptogr. **87**(5), 1161–1182 (2019)
16. Farik, M., Ali, S.: The need for quantum-resistant cryptography in classical computers. In: 2016 3rd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE). IEEE , pp. 98–105 (2016)
17. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Phys. Rev. Lett. **103**(15), 150502 (2009)
18. Rebentrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. Phys. Rev. Lett. **113**(13), 130503 (2014)
19. Chen, Y.A., Gao, X.S., Yuan, C.M.: Quantum algorithm for optimization and polynomial system solving over finite field and application to cryptanalysis (2018). arXiv preprint arXiv:1802.03856
20. Chen, Y.A., Gao, X.S.: Quantum algorithms for Boolean equation solving and quantum algebraic attack on cryptosystems (2017). arXiv preprint arXiv:1712.06239
21. eSTREAM-ECRYPT steam cipher project. http://www.ecrypt.eu.org/stream/
22. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. IJWMC **2**(1), 86–93 (2007)
23. Hell, M., Johansson, T., Maximov, A.: A stream cipher proposal: Grain-128. In: 2006 IEEE international symposium on information theory. IEEE , pp. 1614–1618 (2006)
24. Martin, Å., et al.: Grain-128a: a new version of Grain-128 with optional authentication. Int. J. Wirel. Mob. Comput. **5**, 48–59 (2011)
25. Lee, Y., Jeong, K., et al.: Related-key chosen IV attacks on Grain-v1 and Grain-128. In: Australasian conference on information security and privacy, pp. 321–335. Springer, Berlin, Heidelberg (2008)
26. Dinur, I., Guneysu, T., Paar, C., Shamir, A., Zimmermann, R.: An experimentally verified attack on full Grain-128 using dedicated reconfigurable hardware. In: International conference on the theory and application of cryptology and information security. Springer, Berlin, Heidelberg, pp. 327–343 (2011)
27. Dinur, I., Shamir, A.: Breaking Grain-128 with dynamic cube attacks. In: International workshop on fast software encryption. Springer, Berlin, Heidelberg, pp. 167–187 (2011)
28. Banik, S., Maitra, S., Sarkar, S., Meltem, Sönmez. T.: A chosen IV related key attack on Grain-128a. (eds) Information Security and Privacy. ACISP, Lecture Notes in Computer Science, vol. 7959. Springer, Berlin, Heidelberg (2013)
29. Fu, X.M., Wang, X.Y., et al.: Determining the nonexistent terms of non-linear multivariate polynomials: how to break Grain-128 more efficiently. IACR Cryptol. ePrint Arch. **2017**, 412 (2017)
30. Ambainis, A.: Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations (2010). arXiv preprint arXiv:1010.4458
31. Caminata, A., Gorla, E.: Solving multivariate polynomial systems and an invariant from commutative algebra (2017). arXiv preprint arXiv:1706.06319

32. Faugere, J.C.: A new efficient algorithm for computing Gröbner bases (F4)[J]. J. Pure Appl. Algebra **139**(1–3), 61–88 (1999)
33. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: International conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, pp. 392–407 (2000)
34. Tang, Y.L., Han, D., Li, Z.C.: Key recover attack on stream Cipher Grain-128 and its improvement. Comput. Appl. Softw. **33**(5), 298–301 (2016)

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