electronics

MDPI

Article

High-Speed Privacy Amplification Algorithm Using Cellular Automate in Quantum Key Distribution

Yekai Lu, Enjian Bai *, Xue-qin Jiang and Yun Wu

School of Information Science & Technology, Donghua University, Shanghai 201620, China
* Correspondence: baiej@dhu.edu.cn

Abstract: Privacy amplification is an important step in the post-processing of quantum communication, which plays an indispensable role in the security of quantum key distribution systems. In this paper, we propose a Cellular Automata-based privacy amplification algorithm, which improves the speed of key distribution. The proposed algorithm is characterized by block iteration to generate secure key of arbitrary length. The core of the algorithm in this paper is to use the property that Cellular Automata can generate multiple new associated random sequences at the same time to carry out bit operations for multiple negotiation keys in the meantime and calculate in turn, so as to quickly realize the compression of negotiation keys. By analyzing the final key, the proposed algorithm has the advantages of fast key generation speed and high real-time performance. At the same time, the results of the NIST randomness test and avalanche test show that the algorithm has good randomness performance.

Keywords: quantum key distribution (QKD); privacy amplification (PA); cellular automata (CA); randomness test; avalanche test

Citation: Lu, Y.; Bai, E.; Jiang, X.-q.; Wu, Y. High-Speed Privacy Amplification Algorithm Using Cellular Automate in Quantum Key Distribution. *Electronics* **2022**, *11*, 2426. https://doi.org/10.3390/electronics11152426

Academic Editor: Wiesław Leonski

Received: 4 July 2022
Accepted: 1 August 2022
Published: 4 August 2022

1. Introduction

Quantum key distribution (QKD), based on the uncertainty principle and the No-Cloning theorem, theoretically has higher security than the existing information security schemes [1]. However, the generated key itself has no substantive information. Only when it is encrypted by the encryption algorithm as a key can the information required by both sides of the communication be transmitted [2–5]. The encrypted key needs to be transmitted in the public channel, which inevitably leads to the risk of information disclosure. In order to delete the leaked information from the negotiation key containing the leaked information, Bennet et al. proposed an important privacy amplification step in the post-processing of quantum communication [6,7], which realizes the unconditional security of the quantum key distribution system by compressing the negotiation key into an absolutely secure final key [8–10].

A common PA is to compress a string of keys through a universal hash function, and then eliminate the information leaked to attacker Eve. In this way, the security key can be obtained. The hash function is usually selected as the Toeplitz matrix, whose element is 0 or 1 [11]. Some researchers use a variety of acceleration software methods provided by fast Fourier transform (FFT) [12] to realize the privacy amplification algorithm through CPU and GPU software [13]. Its experimental efficiency is relatively good and can achieve a considerable processing rate. In [12], the researchers propose a FFT PA scheme on commercial CPU platform. The long input weak secure key is divided into many blocks, then PA procedures are parallel implemented for all sub-key blocks, and afterwards the outcomes are merged as the final secure key, but FFT also needs to consume a lot of computing resources. Moreover, for the practical quantum key distribution system, these methods of using CPU software to realize the PA algorithm have hidden dangers in security. There may be various unknown backdoors and vulnerabilities in this system, which greatly

affect the work of quantum key distribution system. Therefore, researchers propose to use a field programmable gate array (FPGA) platform to implement the privacy amplification algorithm. The algorithm implemented by FPGA is a pure hardware logic circuit with low security risks. In [14], Lu et al. proposed a PA algorithm implemented on FPGA platform. By constructing the required Toeplitz matrix on FPGA and using the characteristics of FPGA to calculate the Toeplitz matrix in parallel, they succeed in improving the running speed of the algorithm and the maximum safe coding rate of the system. In addition, the algorithm can also achieve any number of input key bits in a certain length, which is helpful for the implementation of future PA algorithms [15]. In [16], Toeplitz matrix is divided into several sub blocks, and FPGA is used to process the sub blocks in parallel to improve the operation speed. However, it only considers the reconstruction of the Toeplitz matrix, and does not involve the adequate processing of the negotiated key with the Toeplitz matrix. In view of the high requirements of hardware resources and low computing speed of Toeplitz matrix, researchers put forward some effective schemes to improve it. In [17], they propose a privacy amplification algorithm based on LFSR to save storage space and speed up operation process. For the storage of elements in the Toeplitz matrix, only one register is needed, which greatly saves hardware storage resources. In [18], Bai et al. proposed a PA algorithm based on Toeplitz matrix, which uses LFSR to save storage space and speed up the privacy amplification process. The continuous state transformation of LFSR is constructed. The results of each LFSR state are accumulated at the same time of LFSR state transition. Repeat the above steps through block iteration to obtain the final key. Because the operations of different accumulators are independent, the calculation of the final key is parallel, and the speed of the algorithm can be improved. However, due to the characteristics of its sequential transformation to produce the whole Toeplitz matrix, the rate of generating the final key is still inevitably affected.

In this paper, we propose a PA algorithm based on Cellular Automata (CA) and block structure. CA is used to generate a pseudorandom sequence with good random characteristics. The sequence performs a bit operation with the negotiation key and accumulates in blocks, so as to realize the function of compressing the longer key into the final key. Unlike the algorithm of dynamically generating Toeplitz matrix using LFSR, CA does not need to generate random sequences bit by bit like LFSR. Due to the characteristics of CA, it can generate many new random sequences in parallel, which improves the operation speed and can generate keys of any length. The National Institute of Standards and Technology (NIST) randomness test [19] and avalanche test show that the final key generated by the algorithm also has good randomness performance and a good avalanche effect [20,21].

The rest of this paper is organized as follows. Section 2 introduces some relevant principles used in this paper, including privacy amplification and Cellular Automata. Section 3 introduces the algorithm and its implementation proposed in this paper. Section 4 is the analysis of the experimental results. Finally, Section 5 is presented.

2. Preliminary Work

2.1. Principles of PA

The privacy amplification process is that the communication parties Alice and Bob use the negotiation key obtained in the quantum key distribution process to carry out security compression to obtain a relatively short final key, so as to eliminate the information that may be leaked in the traditional channel and achieve unconditional security. The eavesdropper Eve can hardly obtain any information about the key after the privacy amplification step, and the secure key used by Alice and Bob has sufficient security [22].

From the perspective of information theory, the process of PA can be regarded as the technology of extracting highly confidential shared information, security key, from a large number of shared information that is only partially secure but at risk of disclosure. Let Alice and Bob share a random variable W, such as a random bit string with a length of n, from which the eavesdropper Eve obtains the relevant random variable V. Due to the influence of interference factors, it can obtain at most t ($t \leq n$) bits of information

about W, that is, $H(W|V) \geq n - t$. In addition, to satisfying this constraint and possibly some further constraints, Alice and Bob usually do not know the specific details of the distribution of random variables. Alice and Bob hope to publicly select a compression function $g : S_n[0,1] \rightarrow S_r[0,1](n > r)$, so that Eve's partial information about W and its complete information about the compression function g can obtain the information $k = g(V)$. Because the k is almost evenly distributed, the eavesdropper cannot obtain the information about W from k [23]

$$I(k : g, V) \approx 0 \tag{1}$$

Therefore, the key obtained after the PA algorithm can be safely used as the encryption key.

2.2. Cellular Automata

Cellular Automata is a special grid dynamic model. Its characteristics are discrete in time, space and state, and the rules that change the state have local characteristics in time or space. CA is the general name of a kind of model or framework [24,25]. CA is defined as a dynamic system that changes in the discontinuous time dimension in the unit space composed of discontinuous and finite elements under certain rules. Specifically, CA consists of four main parts: cell space, state, neighborhood and rule, which are recorded as $A = (L_d, S, N, f)$ [26]. Where A denotes CA, L_d denotes cell space and d is the dimension of space, S denotes the finite discrete state set of CA, N denotes the neighborhood vector and f denotes the local conversion function.

2.2.1. Elementary Cellular Automata

Elementary Cellular Automata (ECA) is the simplest form of CA [27]. Its state number $k = 2$, neighborhood radius $r = 1$, and local transformation function f is expressed as

$$s_i{}^{t+1} = f(s_{i-1}^t s_i^t s_{i+1}^t) \tag{2}$$

The input quantity of the local conversion function is three state quantities, and each state quantity can have two choices, i.e., 0 or 1, so there are eight possible state combination modes, namely 000, 001, 010, 011, 100, 101, 110 and 111. The combination of each input state must correspond to two output states of 0 or 1. Confirm the corresponding output of each input state combination to obtain the truth table of CA, which corresponds to the rules of an ECA. Since there are 8 state combinations of ECA, each of which can correspond to two outputs, there are totally $2^8 = 256$ truth tables and 256 rules. Rule space is a collection of these 256 rules. Assign corresponding serial numbers to these 256 combinations, and record the 8-bit binary number in the right column of each combination table as decimal number to obtain the rule number, which is any integer between 0 and 255. Arrange the above eight possible combination modes in binary increasing order, and calculate the corresponding output state at the same time to obtain the truth table of the local conversion function. Table 1 shows the truth table of rule No. 150 (the binary representation of 150 is 10010110).

Table 1. Truth table of ECA (No. 150).

$s_{i-1}^t s_i^t s_{i+1}^t$	s_i^{t+1}
0 0 0	1
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	0

If we take an ECA with a length of 8 bits as an example, we set the initial value of CA to 10101010, and the rules adopted are like rule No. 150 shown in Table 1. In the first clock cycle, we give the initial value to the CA. In the second clock cycle, the 8-bit CA is updated according to the set rules. The neighborhood objects of the first bit of the CA are bit 8 and bit 2, that is, $s_1^2 = f(s_8^1 s_1^1 s_2^1)$, get the status 010 and update it to 0 according to the truth table. The second bit update is also done the same way to get the status 101, which is updated to 1 according to the truth table. The third bit gets the status 010 and updates it to 0. Four to eight bits get 101, 010, 101, 010 and 101 respectively and update it to 1, 0, 1, 0 and 1. Finally, after a round of updating, the CA gets 8 bits, and the new state is 01010101.

2.2.2. Pseudorandom Sequence

True randomness is a phenomenon that has no definite cause and effect, and only feels the result but cannot see (that is, the existing human cognitive system cannot perceive and measure) the cause, which is completely incomprehensible. At present, the randomness widely used in various fields is usually generated by chaotic systems, such as Duffing oscillator [28]. Chaos is a phenomenon that determines cause and effect, but cannot be accurately calculated and predicted by mathematics. The reason is that the initial value is sensitive and the model is complex, which belongs to certain, understandable, but imprecise predictive control.

The true random sequence can only come from natural phenomena, which are very difficult to generate in practical application, so the pseudorandom sequence generated by an artificial method is widely used [29,30] in the field of sequence cipher. The key problem of sequence cipher is to produce a long unpredictable key sequence.

Pseudorandom number generation by CA has been an active field of research in cryptography, one of the underlying motivations stemming from the advantages offered by CA when considered from a VLSI viewpoint: CA are simple, regular, locally interconnected, and modular [31]. These characteristics make them easier to implement in hardware than other models. The pseudorandom sequences generated by CA can be divided into the following three categories.

- Stationary type. No matter what the initial value of CA is, after a certain period of evolution, it will eventually enter a stationary state, that is, the state values of all cells are the same, and the evolution of this type of CA has no randomness.
- Periodic type. The CA will enter the periodic structure after a certain period of time. The evolution of this type of CA will remove some randomness, but also retain some, which can be applied to image processing.
- Chaotic type. CA will enter a random or chaotic aperiodic state after a certain period of evolution. The evolution of this type of CA has good randomness.

In this paper, the type of pseudorandom sequence generated by CA is chaotic. For ECA, a chaotic pseudorandom sequence can be generated according to the truth table rule No. 150. When we choose chaotic CA, we can't see any regular pattern in the space-time pattern, and the pattern is much richer than that of a single CA. From the Figure 1, we can see that its CA sequence has strong randomness. Using this characteristic, we can produce a pseudorandom sequence with better performance.

The pseudorandom sequence generator using CA needs to assign an initial value to the CA before it starts running. The initial value of the first CA of length N can be generated randomly or fixedly according to the specific application, as hash function application, selecting the first N bits of irrational numbers e and π or the first N bits of $\sqrt{2}$ and $\sqrt{3}$.

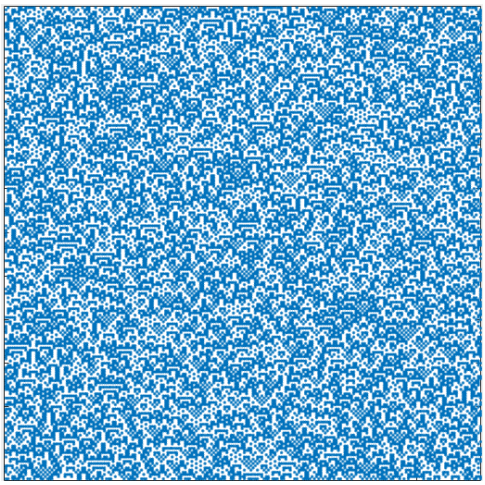

Figure 1. Pseudo chaotic spatiotemporal map of ECA (No. 150).

2.3. NIST Randomness Test

Randomness detection usually uses the method of probability and statistics to detect whether the detection sequence conforms to some characteristics of random sequence, so as to judge whether it is random [30]. Theoretically, if the detected sequence fails to pass the randomness test, it can be determined that the sequence is not random. On the contrary, if the detected sequence can pass a certain randomness test, it is uncertain whether the sequence is random, that is, passing the randomness test is a necessary and insufficient condition for the randomness of the sequence. Because the detection items in each detection method are usually designed according to the characteristics of random sequence. In fact, any set consisting of a limited number of test items cannot cover all aspects of randomness. However, in practical application, if the design of the test is sufficient to meet the specific requirements of the random sequence, and the tested sequence can pass the test, the randomness of the sequence is regarded as qualified.

Randomness detection uses the method of probability and statistics to describe the randomness of the sequence generated by random number generator or cryptographic algorithm. Different detection items describe the gap between the detected sequence and the real random sequence from different angles. Hypothesis testing is usually used for randomness testing. Hypothesis test is to put forward some hypotheses on the population when the population distribution is unknown or only know its form, but not its nature, so as to infer some properties of the population, and then judge the hypotheses according to the samples. The randomness hypothesis test is that if one aspect of the real random sequence is known to conform to the specific distribution, it is assumed that the sequence to be tested is random, and the sequence to be tested is on this side. In practical application, the surface should also conform to the specific distribution. The common method to measure randomness is the p-value method. The NIST suite includes multiple randomness tests, each of which is a returning p-value. When $p \leq 0.01$, it indicates that the sequence has not passed the corresponding test, and when $0.01 < p \leq 1$, the sequence has passed the corresponding test. The higher the p-value, the better the randomness of this sequence [19].

3. Proposed Algorithm

No matter how the designer optimizes it, the PA based on Toeplitz matrix needs to find a balance between resource consumption and time consumption, which will inevitably be greatly affected by the Toeplitz matrix itself required by the compression function. Therefore, we use CA, a tool that can generate pseudorandom sequences with good randomness, to replace Toeplitz matrix and realize the compression process from negotiation key to final key, so as to improve the speed of PA algorithm. Based on this idea, we propose a high-speed PA algorithm with memory-saving using CA.

Table 2 gives some notations involved in the algorithm. Figure 2 depicts the process of the proposed PA algorithm.

Table 2. Notations.

Notation	Definition
T	Negotiation key
n	Length of negotiation key
T_{Mi}	The i-th group negotiation key
M	Group length of negotiation key
K	Number of groups negotiating key
N_j	The j-th block negotiation key
N	Length of Cellular Automata
C	Reciprocal of key compression rate and number of blocks
H_i	Final key obtained by group i
H	Final key
m	Length of zero complement in negotiation key T

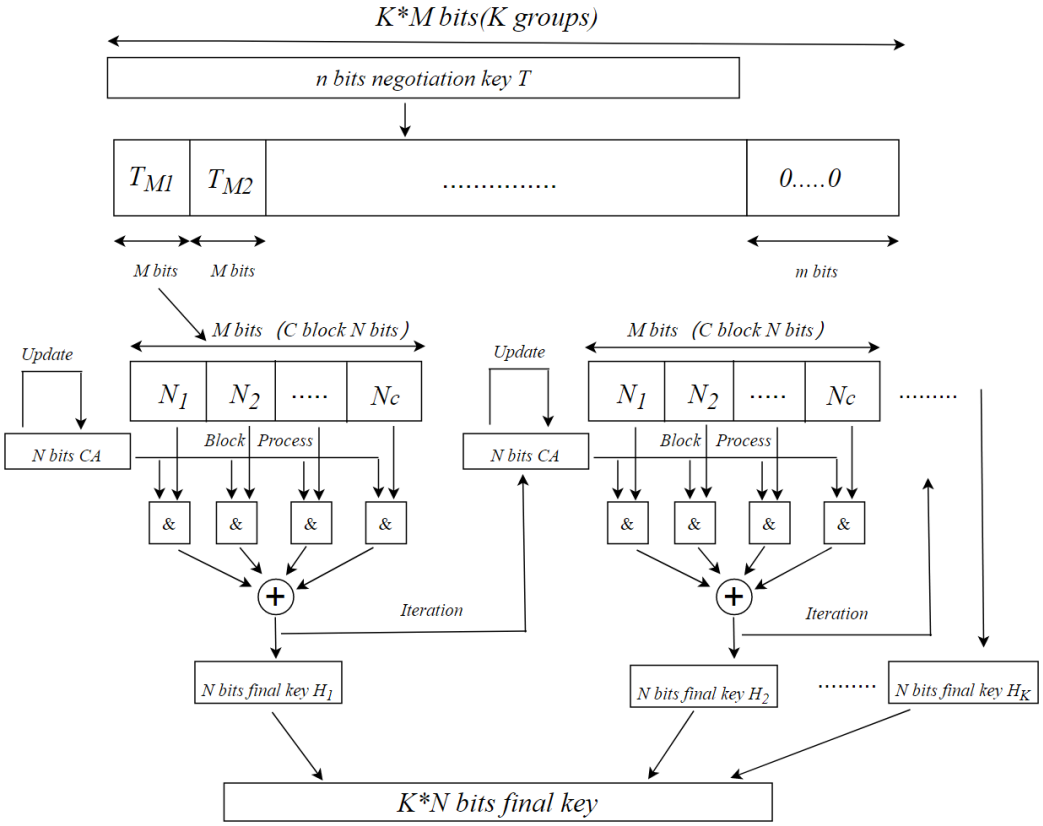

Figure 2. Schematic diagram of PA algorithm.

As can be seen from Figure 2, n bits negotiation key T is firstly divided into K groups $[T_{M1}, T_{M2}, \cdots, T_{MK}]$ with length M. After dividing the negotiation key, process the group negotiation key in turn. For example, for the group negotiation key T_{M1} with length M, we further divide it into C blocks and each block with length N. To make M satisfy $M = C \times N$, it is necessary to add a zero sequence of $m = M \times K - n$ bits after the original negotiation key T. The core of the algorithm is to use CA to generate multiple pseudorandom sequences with good randomness at the same time, and successfully compress the group negotiation key of length M into the final key H_1 of length N through operation. After obtaining the result of the final key, we use the idea of iteration to take the current N bits final key H_1

as the initial value of CA used in the next group, the same algorithm is used to obtain the next N bits final key. The above steps are repeated until all the group negotiation keys are processed and the final required $K \times N$ bits final key H is generated. Figure 3 shows the processing flow of the algorithm. The specific process of the algorithm will be described in detail below.

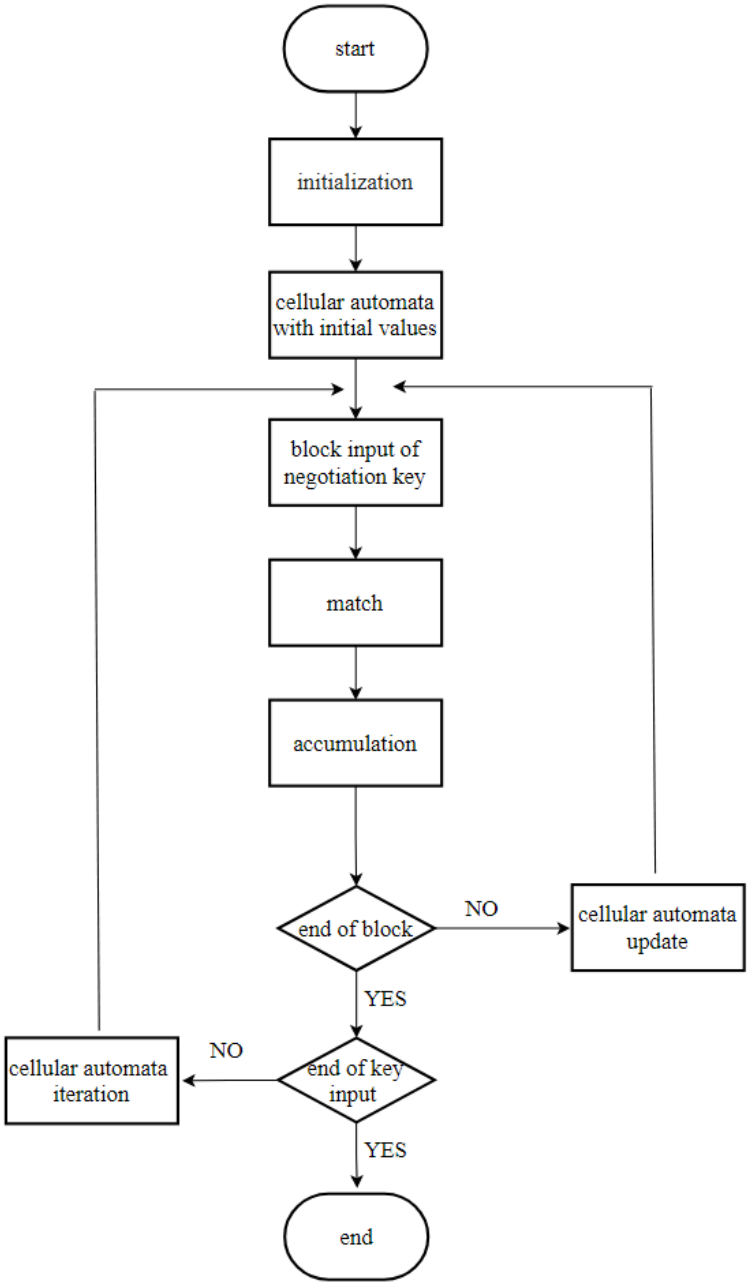

Figure 3. Flow chart of PA algorithm.

Step 1: Set the parameters according to the requirements, such as the length N of CA and the reciprocal C of the compression rate of the final key. After setting the parameters, divide the received n bits negotiation key T into small groups with length M. The last T_{Mk} may not meet the requirement of length M. If the length does not meet the requirement, we will add a sufficient number of zeros m in the last group to make it meet the requirement of length.

Step 2: Initialize CA and set its running rules. In order to make the pseudorandom sequence generated by CA have good randomness, we need to adopt appropriate rules.

Our algorithm selects rule 150 shown in Table 1 among the 256 rules of ECA. Under this rule, the sequence space-time map generated by CA has obvious chaotic characteristics. As for the choice of the initial value of CA, we select the fixed first N bits of e, π, $\sqrt{2}$ or $\sqrt{3}$.

Step 3: We further divide the group negotiation key T_{M1} with length M into blocks with length N, and then we combine these blocks with the N bits sequence generated by CA to perform bit and operation. The specific operation is that the first N length block T_{M11} is combined with the initial value of the CA IV, denoted as $s_i^1, i = 1, 2, \cdots, N$ and the result is put into the N bits accumulator. After the first block operation, using the characteristics of the CA, we can update the N bits data of the CA at the same time, and the updated result $s_i^2, i = 1, 2, \cdots, N$ performs bit and operation with the negotiation key of the next N length block T_{M12}. The result is also put into the N bits accumulator and performs modulo-2 addition with the previous result. Repeat the above steps until the C blocks M length negotiation key is calculated, we can take the value of the accumulator as the final key H_1. Using the idea of iteration, we take the result of the final key H_1 as the N bits initial value of the next CA, repeat the above process again, and finally complete the calculation of the last group to obtain the final key H_K.

Step 4: After all the negotiation keys are calculated, we can get the final key with K blocks length of N, and we combine it into a final security key H. Because the final key of the algorithm is obtained in blocks, using this feature, we can output the final result of the privacy amplification process in real time, which improves the data throughput of the hardware implementation.

We summarize the proposed PA algorithm as follow:

$$n + m = K \times M \tag{3}$$

$$H_0 \leftarrow IV \tag{4}$$

$$H_j = g(T_{Mj}, H_{j-1}) = \bigoplus_{l=1}^{C} T_{Mjl} \& a_l, j = 1, 2, \cdots, K \tag{5}$$

$$a_l = (s_1^l s_2^l \cdots s_N^l), l = 1, 2, 3, \cdots, C \tag{6}$$

$$a_1 = (s_1^1 s_2^1 \cdots s_N^1) \leftarrow H_{j-1}, l = 1, j = 1, 2, \cdots, K \tag{7}$$

$$s_i^l = f(s_{i-1}^{l-1} s_i^{l-1} s_{i+1}^{l-1}), s_0^{l-1} = s_N^{l-1}, s_{N+1}^{l-1} = s_1^{l-1}, l \neq 1, i = 1, 2, \cdots, N \tag{8}$$

$$H = \{H_1, H_2, \cdots, H_K\} \tag{9}$$

The proposed PA algorithm can be regarded as an (M, N)-family of hash functions as follows. We associate a hash function h such that for any message T_{Mj} of binary length M, $h(T_{Mj})$ is defined as $\bigoplus_{l=1}^{C} T_{Mjl} \& a_l$.

The $\epsilon-$balanced hash function is defined [11] as follows.

Definition 1. *A family of hash functions is called $\epsilon-$balanced if*

$$\forall T \neq 0, c, Pr_h(h(T) = c) \leq \epsilon.$$

Theorem 1. *For any values of N and M the above defined family of hash functions is $\epsilon-$balanced for $\epsilon \leq \frac{1}{2^N}$.*

Proof. To show that the family is $\epsilon-$balanced, notice that any non-zero message T_{Mj} of length M and any string c of length N, $h(T_{Mj}) = c$ iff $\bigoplus_{l=1}^{C} T_{Mjl} \& a_l = c$. Since a_l generated by ECA under rule 150 has random characteristics, the vector T_{Mj} assumes this value c with probability of $\frac{1}{2^N}$, and therefore $Pr(h(T_{Mj}) = c)$ happens with at most this probability. □

4. Experimental Results

The amount of computing resources consumed, the rate of generating the final key, the randomness of generating the final key, and the sensitivity to the change of the input key are four important indicators to evaluate the performance of a PA algorithm. In addition, we will also test the influence of the initial value of CA on the experimental results. In order to evaluate the algorithm we proposed in this paper, we carried out simulation experiments on Matlab R2020a platform, and analyzed the experimental results with [12,18] in these four aspects.

The computer used in this simulation experiment is configured as AMD Ryzen 5 5600 h with Radeon graphics 3.30 GHz, 16.0 GB memory and win11 operating system.

4.1. Analysis of Memory

The length of the negotiation key is usually long enough, due to the finite scale effect [32,33]. However, with the increase of the length of the negotiation key, the memory of Toeplitz matrix based PA will increase, and the storage of Toeplitz matrix elements is an obstacle.

If the algorithm in [12] is used, the privacy amplification process is carried out by dividing the Toeplitz matrix into several sub matrices and using FFT fast operations on these sub matrices. Using this method, we need to store the information with the size of $2(K \times N + M)$ bits.

If the method in [18] is used and the Toeplitz matrix is generated by LFSR, we need to store the N bits LFSR state and N bits accumulator, and need $2 \times K \times N$ bits to complete the whole PA process, $2(K \times N + N)$ bits in all. However, when N is large, finding the irreducible polynomial of length N is a troublesome problem.

In our algorithm, the state of CA changes iteratively by itself. When the size of CA is N, only N bits need to be stored, and we need $K \times N$ bits to store the final key. With only $(2 + K) \times N$ bits, the whole PA process can be completed without looking for the irreducible polynomial required by LFSR.

4.2. Analysis of Final Key Generation Rate

In order to verify the advantages of this scheme, we will compare the key generation rates of different schemes under the same conditions. Firstly, we set several different negotiation key lengths from 0.64 million bits to 5.12 million bits, which is convenient for calculation. The compression ratio N/M is set to 0.1, the length of CA is set to 128 bits, the length of LFSR is the same as that of CA [18], and the size of sub block meets the requirements of each algorithm.

The experimental results are shown in Figure 4. We can see that under the experimental conditions we set, the time consumption of the PA algorithm using CA is almost half that of the block LFSR algorithm [18] and a quarter of that of the FFT algorithm [12]. It can be seen that under certain conditions, the proposed PA algorithm has advantages in key generation rate and algorithm execution speed.

However, other factors will affect the execution speed of the algorithm and the generation rate of the key. Therefore, we then changed the conditions of the compression rate to 0.1, 0.2 and 0.5, respectively, and observed the impact of different compression rates on several algorithms (the length of the experimental negotiation key is 1.28 million bits). The results are shown in the Table 3.

Table 3. Effect of compression rate on algorithm running time.

Compression Ratio	FFT(s) [12]	LFSR(s) [18]	CA(s)
0.1	9.41	5.23	2.25
0.2	9.32	5.21	2.21
0.5	9.18	5.05	2.10

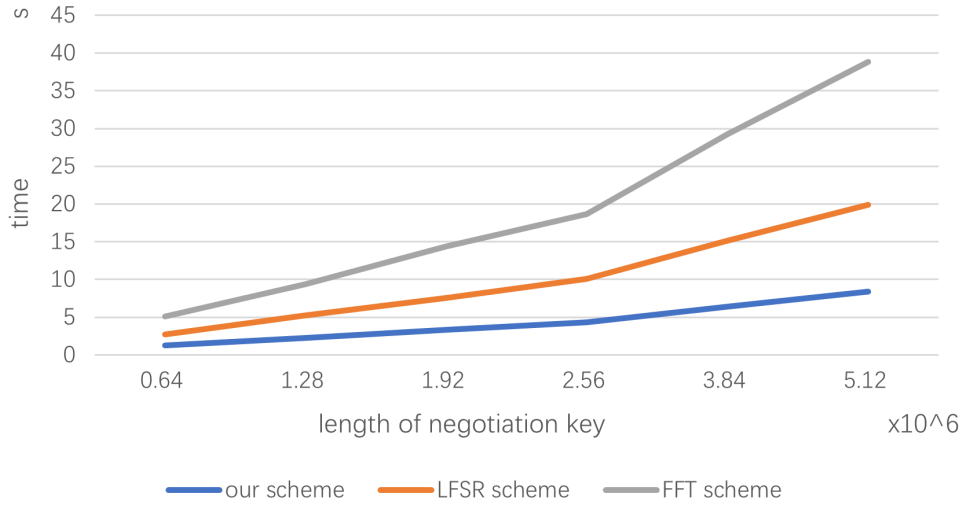

Figure 4. Final key generation rate comparison of LFSR scheme [18], FFT scheme [12] and proposed scheme when $N/M = 0.1$.

It can be seen from the table that the change of compression rate has little impact on several algorithms, because the change of compression rate will affect the size of the data to be processed, but the change of data size does not account for the overall negotiation key size, so the compression rate only has a slight impact on the execution time of the algorithm.

Finally, we investigate the effect of block length on the running time of the algorithm. Due to the influence of algorithm design, we only compare the block LFSR and CA algorithms. The block length of negotiation key in the two scheme is that of the length of CA and LFSR. Observe the effects of different lengths of CA and LFSR on the operation speed of the two algorithms (the length of the experimental negotiation key is 1.28 million bits and the compression rate is 0.1). The results are shown in the Table 4.

Table 4. Influence of block length on algorithm running time.

Block Length	LFSR(s) [18]	CA(s)
64	4.83	2.46
128	5.11	2.21
256	6.18	2.13
512	8.45	2.29

It can be seen that with the increase of the length of pseudorandom sequence generators such as CA and LFSR, the execution time of the PA algorithm using CA changes slightly, while the time consumption of the block LFSR PA algorithm [18] increases with the increase of the block length, which shows that the PA algorithm using CA is less sensitive to the block length. The CA with appropriate length can be selected according to the actual needs, while the LFSR can only select the LFSR with a fixed range to meet the appropriate length due to the difficulty of finding the higher-order primitive polynomial and the high sensitivity to the change of block length. We can find that the PA algorithm using CA has advantages in this aspect.

4.3. Randomness Analysis of Final Key

For the randomness analysis of the final key, we choose to use NIST test tools [19] to analyze the final binary key.

We set the length of the negotiation key to 1.28 million bits and the compression rate to 0.1. We test the final key generated by CA with the length of 64 bits, 128 bits and 256 bits, respectively, and analyze the influence of CA with different length on the randomness of the key. The results are shown in the Table 5 below.

Table 5. NIST randomness test.

Test Items	64 Bits	128 Bits	256 Bits
Frequency	0.350485	0.534146	0.739918
Block Frequency	0.739918	0.534146	0.522325
Cumulative Sums	0.422325	0.350485	0.911413
Runs	0.122325	0.350485	0.122325
Longest Run	0.911413	0.350485	0.535446
Rank	0.422325	0.534146	0.594134
FFT	0.739918	0.350485	0.613309
NonOverlapping Template	0.750485	0.739918	0.991468
Overlapping Template	0.534146	0.754686	0.834526
Approximate Entropy	0.035174	0.122325	0.066882
Serial	0.522325	0.739918	0.650485
Linear Complexity	0.911413	0.350485	0.534146

From the data analysis shown in the Table 5, the longer the length of CA, the more binary digital sequences can be represented by it, so the randomness of the pseudorandom sequences generated by it is stronger. When it is applied to the PA algorithm, the randomness of the final key generated by it with the length of CA is also similar to the pseudorandom sequence of CA. With the increase of the length of CA, the randomness of the final key increases slightly. It can be seen that although the three lengths of CA based PA algorithms have different sizes in each NIST test project, on the whole, the p value of CA with large length is greater than that of CA with shorter length.

Moreover, because our algorithm uses the iterative structure, the initial value of the next CA is determined by the accumulation of the previous register, rather than the continuous transformation of the first initial value of the CA, which strengthens the correlation between the iterative block structures.

4.4. Avalanche Analysis of Final Key

Avalanche effect is an ideal characteristic of hash algorithm. Avalanche effect means that even the smallest change in the input (for example, reversing one binary bit) will lead to drastic changes in the results generated by the algorithm. Strict avalanche criterion is the formalization of avalanche effect. The criterion points out that when any input at the input is reversed, each bit in the output has a 0.5 probability of changing [30]. In order to amplify the influence of interference on eavesdropper Eve, we need the final key to have good avalanche characteristics. If the key does not show a certain degree of avalanche characteristics, we can think that its randomness is poor.

Therefore, we conduct an avalanche test on the final key obtained by our algorithm. The length of the negotiation key we use is 1.28 million bits, the compression rate is 0.1, and the length of CA is 128 bits. Change the input 1, 2, 3, 4 and 5 binary bits, respectively, and observe the percentage of the number of bits of the final key flipped in the total length. The corresponding results are given in the Table 6.

Table 6. Avalanche effect test.

Input Flip (Bit)	Output Flip Scale
1	49.9%
2	49.6%
3	49.6%
4	49.8%
5	49.6%

As can be seen from Table 6, due to the iterative structure used in the algorithm, the overall connection of the final key is strong. The reversal of one input can cause the reversal of about 50% of the digit value of the whole final key. From the data in the table, we can get that the final key generated by the PA algorithm proposed in this paper has a good avalanche effect.

4.5. Influence of CA Initial Value

In this paper, CA is used to replace LFSR in the process of privacy amplification. For the chaotic CA used in this paper, its initial value sensitivity and complex system model are the sources of its randomness. Therefore, we need to test the influence of the initial value changed in a wide range on the experimental results to ensure that the proposed algorithm is universal for most of the initial value selection.

Avalanche effect reflects randomness to a certain extent and is more intuitive. We conduct avalanche test on the final key obtained from different CA initial values. We can change the initial value by changing the proportion R of 1 s in the CA initial values. In this experiment, the selected R is 0.1, 0.5 and 0.9, respectively. The length of the negotiation key we use is 1.28 million bits, the compression rate is 0.1, and the length of CA is 128 bits. Change the input 1, 2, and 3 binary bits, and observe the percentage of the number of bits of the final key flipped in the total length. The corresponding results are given in Table 7.

Table 7. Influence of CA initial value.

Proportion of 1 s (R)	Input Flip (Bit)	Output Flip Scale
0.1	1	49.8%
	2	49.7%
	3	49.9%
0.5	1	49.9%
	2	49.6%
	3	49.6%
0.9	1	50.0%
	2	49.8%
	3	49.7%

From Table 7, we can see that the avalanche effect is still good and still tends to be close to the strict avalanche criterion when the final key generated by the proposed PA algorithm changes the initial value of CA in a large range. We can get from it that the initial value selection of CA has little effect on the final key generated by the proposed algorithm.

5. Conclusions

In this paper, a high-speed PA algorithm for QKD system is proposed. CA is used to to construct a hash function for the secure key distribution. Different from using LFSR-based PA to establish Toeplitz matrix by sequential shift, the algorithm proposed in this paper uses CA, which can simultaneously iterate the sequence of N bits CA length, and calculate the input of a multi bits negotiation key at the same time, the speed of the algorithm is then improved. This algorithm also uses the idea of block iteration to process the overall negotiation key block by block, and strengthen the correlation of each block generated key through iteration. Finally, the analysis results show that the algorithm saves hardware memory resources and improves the running speed. The final shared key also has good randomness performance.

The algorithm proposed in this paper can carry out the parallel processing of multi bits negotiation key input, and it improves the speed. However, when processing the block negotiation keys, we still need to wait for the update and iteration of the CA itself, and the waiting time will slow down the running speed of the algorithm. Therefore, in order to further improve the generation rate based on the algorithm in this paper, when

the hardware resources are sufficient, we can choose the improved scheme of exchanging resources for time, that is, increasing the number of CA. When the initial values are the same, at the same time, the multi block negotiation key is processed in parallel to improve the speed of the algorithm.

The security of a classical cryptosystem depends on the security of the key. The key can be distributed by the QKD system in this paper. The key of block cipher and public key cipher is usually very short, ranging from 128 bits to 4096 bits, the last corresponding bits of the final key can be used as the key. Another advantage of this algorithm is that the key generated in real time can be directly used as the encryption key of the classical stream cipher system, so as to achieve the effect of One-Time-Pad.

In addition, the CA can be implemented in FPGA to further improve the processing efficiency, which will be studied in the future.

Author Contributions: Y.L., software, writing—original draft preparation; E.B., conceptualization, methodology, software, writing—review and editing; X.-q.J., writing—review and editing, project administration, funding acquisition. Y.W., supervision, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Fundation of Shanghai (Grant no. 20ZR1400700) and the Shanghai Municipal Science and Technology Major Project (Grant no. 2019SHZDZX01), China.

Institutional Review Board Statement: This study did not involve humans or animals.

Data Availability Statement: Data are available in a publicly accessible repository that does not issue DOIs. Publicly available datasets were analyzed in this study, and the data used to support the findings of this study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wolf, R. *Quantum Key Distribution: An Introduction with Exercises*; Lecture Notes in Physics; Springer: Berlin/Heidelberg, Germany, 2021.
2. Gisin, N.; Ribordy, G.; Tittel, W.; Zbinden, H. Quantum Cryptography. *Rev. Mod. Phys.* **2002**, *74*, 145–195. [CrossRef]
3. Weedbrook, C.; Pirandola, S.; García-Patrón, R.; Cerf, N.J.; Ralph, T.C.; Shapiro, J.H.; Lloyd, S. Gaussian Quantum Information. *Rev. Mod. Phys.* **2012**, *84*, 621–669. [CrossRef]
4. Grosshans, F.; Grangier, P. Continuous Variable Quantum Cryptography Using Coherent States. *Phys. Rev. Lett.* **2002**, *88*, 057902. [CrossRef] [PubMed]
5. Gilbert, G.; Hamrick, M. Secrecy, Computational Loads and Rates in Practical Quantum Cryptography. *Algorithmica* **2002**, *34*, 314–339.
6. Bennett, C.H.; Zekrifa D.M.S.; Robert, J.M. Privacy Amplification by Public Discussion. *SIAM J. Comput.* **2012**, *17*, 210–229. [CrossRef]
7. Bennett, C.H.; Brassard, G.; Crepeau, C.; Maurer, U.M. Generalized Privacy Amplification. *IEEE Trans. Inf. Theory* **1995**, *41*, 1915–1923. [CrossRef]
8. Melki, R.; Noura, H.N.; Mansour, M.M.; Chehab, A. A Survey on OFDM Physical Layer Security. *Phys. Commun.* **2019**, *32*, 1–30. [CrossRef]
9. Bottarelli, M.; Epiphaniou, G.; Ben Ismail, D.K.; Karadimas, P.; Al-Khateeb, H. Physical Characteristics of Wireless Communication Channels for Secret Key Establishment: A Survey of the Research. *Comput. Secur.* **2018**, *78*, 454–476. [CrossRef]
10. Zhang, J.Q.; Duong, T.Q.; Marshall, A.; Woods, R. Key Generation from Wireless Channels: A Review. *IEEE Access* **2017**, *4*, 614–626. [CrossRef]
11. Carter, J.L.; Wegman, M.N. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.* **1979**, *18*, 143–154. [CrossRef]
12. Tang, B.Y.; Liu, B.; Zhai, Y.P.; Wu, C.Q.; Yu, W.R. High-speed and Large-scale Privacy Amplification Scheme for Quantum Key Distribution. *Sci. Rep.* **2019**, *9*, 15733. [CrossRef]
13. Wang, X.Y.; Zhang, Y.C.; Yu, S.; Guo, H. High-speed Implementation of Length-compatible Privacy Amplification in Continuous-variable Quantum Key Distribution. *IEEE Photonics J.* **2018**, *10*, 7600309.
14. Lu, H.B.; Zhao, J.; Yin, Z.J. Implementation of Security Enhancement Algorithm for High Speed QKD System Based on FPGA. *Quantum J. Electron.* **2019**, *2*, 197–205.
15. Zidan, M.; Aldulaimi, S.; Eleuch, H. Analysis of the Quantum Algorithm based on Entanglement Measure for Classifying Boolean Multivariate Function into Novel Hidden Classes: Revisited. *Appl. Math. Inf. Sci.* **2021**, *15*, 643–647.

16. Yang, S.S.; Bai, Z.L.; Wang, X.Y.; Li, Y.M. FPGA-based Implementation of Size-adaptive Privacy Amplification in Quantum Key Distribution. *IEEE Photonics J.* **2017**, *9*, 7600308. [CrossRef]
17. Li, D.W.; Huang, P.; Zhou, Y.M.; Li, Y.; Zeng, G.H. Memory-saving Implementation of High-speed Privacy Amplification Algorithm for Continuous-variable Quantum Key Distribution. *IEEE Photonics J.* **2018**, *10*, 7600712. [CrossRef]
18. Bai, E.J.; Jiang, X.-Q.; Wu, Y. Memory-Saving and High-Speed Privacy Amplification Algorithm Using LFSR-Based Hash Function for Key Generation. *Electronics* **2022**, *11*, 377. [CrossRef]
19. Bassham, L.; Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Leigh, S.; Levenson, M.; Vangel, M.; Heckert, N.; Banks, D. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Special Publication (NIST SP); National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001. Available online: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762 (accessed on 25 January 2022).
20. Obada, A.; Abo-Kahla, D.; Metwally, N.; Abdel-Aty, M. The Quantum Computational Speed of a Single Cooper Pair Box. *Phys. E Low-Dimens. Syst. Nanostruct.* **2011**, *43*, 1792–1797. [CrossRef]
21. Zidan, M.; Abdel-Aty, A.; Khalil, A.; Abdel-Aty, M.; Eleuch, H. A Novel Efficient Quantum Random Access Memory. *IEEE Access* **2021**, *9*, 151775–151780. [CrossRef]
22. Bennett, C.H.; Brassard, G. Quantum Cryptography: Public Key Distribution and Coin Tossing. *Theor. Comput. Sci.* **2014**, *560*, 7–11. [CrossRef]
23. Diamanti, E.; Leverrier, A. Distributing Secret Keys with Quantum Continuous Variables: Principle, Security and Implementations. *Entropy* **2015**, *17*, 6072–6092. [CrossRef]
24. Wolfram, S. Statistical Mechanics of Cellular Automata. *Rev. Mod. Phys.* **1983**, *5*, 601–644. [CrossRef]
25. Wolfram, S. Universality and Complexity in Cellular Automata. *Phys. D Nonlinear Phenom.* **1984**, *10*, 1–35. [CrossRef]
26. Yacoubi, S.E. A Mathematical Method for Control Problems on Cellular Automata Models. *Int. J. Syst. Sci.* **2008**, *39*, 529–538. [CrossRef]
27. Rosenblueth, D.A.; Gershenson, C. A Model of City Traffic Based on Elementary Cellular Automata. *Complex Syst.* **2011**, *19*, 305–322. [CrossRef]
28. Teklu, B.; Ferraro, A.; Paternostro, M.; Paris, M.G.A. Nonlinearity and Nonclassicality in a Nanomechanical Resonator. *EPJ Quantum Technol.* **2015**, *2*, 16. [CrossRef]
29. Menezes, A.; Oorschot, P.V.; Vanstone, S. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1997.
30. Luby, M. *Pseudorandomness and Cryptographic Applications*; Princeton University Press: Princeton, NJ, USA, 1996.
31. Tomassini, M.; Sipper, M.; Perrenoud, M. On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata. *IEEE Trans. Comput.* **2000**, *49*, 1146–1151.
32. Lucamarini, M.; Patel, K.; Dynes, J.; FrHlich, B.; Sharpe, A.E.; Dixon, A.R.; Yuan, Z.L.; Penty, R.V.; Shields, A.J. Efficient Decoy-state Quantum Key Distribution with Quantified Security. *Opt. Exp.* **2013**, *21*, 24550–24565. [CrossRef]
33. Scarani, V.; Renner, R. Quantum Cryptography with Finite Resources: Unconditional Security Bound for Discrete Variable Protocols with One-way Post Processing. *Phys. Rev. Lett.* **2008**, *100*, 200501. [CrossRef]