

Effect of pseudo-random number on the security of quantum key distribution protocol

Xiao-Liang Yang(杨晓亮), Yu-Qing Li(李毓擎), and Hong-Wei Li(李宏伟)

Citation: Chin. Phys. B, 2025, 34 (**2**): 020301. DOI: 10.1088/1674-1056/ad99c9
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Multi-protocol relay chaining for large-scale quantum key distribution networks

Yuan Cao(曹原), Xiaosong Yu(郁小松), Yongli Zhao(赵永利), Chunhui Zhang(张春辉), Xingyu Zhou(周星宇), Jie Zhang(张杰), and Qin Wang(王琴)
Chin. Phys. B, 2025, 34 (**1**): 010310. DOI: 10.1088/1674-1056/ad9018

Security analysis of satellite-to-ground reference-frame-independent quantum key distribution with beam wandering

Chun Zhou(周淳), Yan-Mei Zhao(赵燕美), Xiao-Liang Yang(杨晓亮), Yi-Fei Lu(陆宜飞), Yu Zhou(周雨), Xiao-Lei Jiang(姜晓磊), Hai-Tao Wang(汪海涛), Yang Wang(汪洋), Jia-Ji Li(李家骥), Mu-Sheng Jiang(江木生), Xiang Wang(汪翔), Hai-Long Zhang(张海龙), Hong-Wei Li(李宏伟), and Wan-Su Bao(鲍皖苏)
Chin. Phys. B, 2024, 33 (**8**): 080306. DOI: 10.1088/1674-1056/ad51f6

Improvement and security analysis of multi-ring discrete modulation continuous variable quantum secret sharing scheme

Huan-Yao Jiang(姜欢窈), Min Nie(聂敏), Guang Yang(杨光), Ai-Jing Sun(孙爱晶), Mei-Ling Zhang(张美玲), and Chang-Xing Pei(裴昌幸)
Chin. Phys. B, 2024, 33 (**7**): 070303. DOI: 10.1088/1674-1056/ad3b81

General multi-attack detection for continuous-variable quantum key distribution with local local oscillator

Zhuo Kang(康茁), Wei-Qi Liu(刘维琪), Jin Qi(齐锦), and Chen He(贺晨)
Chin. Phys. B, 2024, 33 (**5**): 050308. DOI: 10.1088/1674-1056/ad2bf2

A new quantum key distribution resource allocation and routing optimization scheme

Lin Bi(毕琳), Xiaotong Yuan(袁晓同), Weijie Wu(吴炜杰), and Shengxi Lin(林升熙)
Chin. Phys. B, 2024, 33 (**3**): 030309. DOI: 10.1088/1674-1056/ad2503

SPECIAL TOPIC — Quantum communication and quantum network

Effect of pseudo-random number on the security of quantum key distribution protocol

Xiao-Liang Yang(杨晓亮), Yu-Qing Li(李毓擎), and Hong-Wei Li(李宏伟)†

Henan Key Laboratory of Quantum Information and Cryptography, SSF IEU, Zhengzhou 450000, *China*

(Received 31 July 2024; revised manuscript received 26 November 2024; accepted manuscript online 3 December 2024)

In the process of quantum key distribution (QKD), the communicating parties need to randomly determine quantum states and measurement bases. To ensure the security of key distribution, we aim to use true random sequences generated by true random number generators as the source of randomness. In practical systems, due to the difficulty of obtaining true random numbers, pseudo-random number generators are used instead. Although the random numbers generated by pseudo-random number generators are statistically random, meeting the requirements of uniform distribution and independence, they rely on an initial seed to generate corresponding pseudo-random sequences. Attackers may predict future elements from the initial elements of the random sequence, posing a security risk to quantum key distribution. This paper analyzes the problems existing in current pseudo-random number generators and proposes corresponding attack methods and applicable scenarios based on the vulnerabilities in the pseudo-random sequence generation process. Under certain conditions, it is possible to obtain the keys of the communicating parties with very low error rates, thus effectively attacking the quantum key system. This paper presents new requirements for the use of random numbers in quantum key systems, which can effectively guide the security evaluation of quantum key distribution protocols.

Keywords: quantum key distribution, pseudo-random, security

PACS: 03.67.Dd, 03.65.Ud, 03.67.Ac, 03.67.Mn

DOI: 10.1088/1674-1056/ad99c9 **CSTR:** 32038.14.CPB.ad99c9

1. Introduction

The security of QKD depends on the basic principle of quantum mechanics and the true random numbers to prepare and measure the quantum states, its security is independent of the eavesdrop's calculation and storage capacity. The communication parties Alice and Bob select the basis vector at random, thus the eavesdropper Eve cannot guarantee that the correct basis vector must be chosen for measurement. Once the wrong basis vector is selected, Eve can not get the right quantum states. Therefore, in a QKD protocol, both sides of the communication parties often judge the existence of the eavesdropper by the bit error rate. In the previous BB84 protocol security analysis result with perfect devices, the secure key can be generated if the bit error rate is less than 11%.

The communication parties Alice and Bob apply true random numbers to encode the quantum states, thus the eavesdropper is unable to select the correct basis vector for measurement, which ensures the security of the QKD protocol. On the contrary, if the quantum state in communication is not encoded with true random numbers, the eavesdropper will get more information at lower bit error rate. In most prepare-and-measure protocols, both the sender Alice and the receiver Bob need to generate basis vectors at random.[2] However, in actual systems, it is possible to replace true random sequences with pseudo-random sequences. As for the randomness of the sequence, the quantitative description cannot be given mathematically. In general, the randomness of the sequence is described qualitatively by comparing the statistical value of the pseudo-random sequence and that of the true random sequence. For example, the ratio of 0 to 1 in a true random sequence should be 1:1, the period of the sequence should be long enough, and so on. Currently, hypothesis testing is generally used to describe the randomness of a sequence, such as NIST's randomness test. Once passing the verification, the sequence is considered to be random. In this paper, it is proved that the security of the system cannot be guaranteed by using a pseudo-random sequence for coding quantum state. Thus, the security requirements of the random number of the quantum key distribution protocol are given.

2. Pseudo-random generation model

A pseudo-random number generator (PRNG), also known as a deterministic random bit generator (DRBG), is an algorithm for generating sequences of numbers that approximate the properties of random numbers. The sequences generated by PRNGs are not truly random, as each number is completely determined by an initial value known as the random seed (the seed is sometimes generated using a hardware random number generator that approximates true randomness). Although sequences close to true randomness can be generated using

†Corresponding author. E-mail: lihow@ustc.edu.cn

© 2025 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

hardware random number generators, PRNGs are particularly important in practice due to their generation speed and reproducibility. There are many types of PRNGs, commonly including:

Linear congruential generator (LCG): An algorithm that produces a pseudo-random sequence using a piecewise linear equation. It represents one of the oldest and most well-known PRNG algorithms, theoretically easy to understand, simple to implement, and fast, especially on computer hardware that provides modulus operations via bit truncation.

Linear feedback shift register (LFSR): A shift register whose input bit is a linear function of its previous state's output. XOR is the most common single-bit linear function: after performing XOR operations on certain bits of the register, the result is used as input, and the whole register is then shifted.

Cryptographically secure pseudo-random number generator (CSPRNG): A type of PRNG that is considered suitable for cryptographic purposes. Compared to statistical PRNGs and weaker PRNGs, the numbers generated by CSPRNGs possess additional pseudo-random properties.

Although PRNGs generate numbers that are statistically random, meeting the requirements of uniform distribution and independence, they depend on an initial value, or seed, to produce the corresponding pseudo-random sequence. An attacker might predict future elements from the initial sequence elements.

The German Federal Office for Information Security (BSI) has established four criteria for evaluating deterministic random bit generators:[3]

K1: The probabilities that the generated random number sequences are different from each other should be very high.

K2: According to certain statistical tests, it should be impossible to distinguish the generated sequences from true random sequences.

K3: Given any subsequence, it should be computationally infeasible for any attacker to predict subsequent sequences or determine the internal state of the generator.

K4: Given the internal state of the generator, it should be computationally infeasible for any attacker to determine previous sequences or the previous state of the generator.

Generators that only meet the K1 and K2 criteria can be used to compute subsequent sequences or the internal state from a given subsequence.

3. Application and attack methods of pseudo-random sequences

Although pseudo-random sequences are not absolutely secure in theory, they are quick to implement, and eavesdroppers require substantial resources to crack them, especially large computational power. Since pseudo-random sequences are designed based on mathematical models, attacks on them can also be implemented using certain mathematical methods. Generally, there are two main approaches:

1. Since the pseudo-random sequence is generated by the expansion of the random number seed through a mathematical model, the eavesdropper can generate the same random number sequence again based on the same random number model when the random number seed is known. This approach can come down to taking the random number seed as the unknown quantity and solving the algebraic equation about random number seeds when we know the contents of some random number sequences.

2. In pseudo-random sequences, random numbers are not independent of each other, and there is correlation between them. Therefore, we can derive the random number of the unknown part through the known partial random number, and finally get the information of the whole sequence of random numbers.

Since pseudorandom numbers are generated through random seeds, in theory, if we obtain a sufficient amount of random numbers, we can predict the subsequent random number sequence. However, some pseudorandom numbers require an enormously large amount of known random numbers to predict the subsequent sequence, making these pseudorandom numbers practically unpredictable. We classify pseudo-random numbers as weak pseudo-random numbers and strong pseudo-random numbers based on whether they possess unpredictability.

Weak pseudo-random numbers are generated according to certain patterns to produce a random value. These numbers are characterized by possessing only a certain degree of randomness and failing to achieve unpredictability.

Strong pseudo-random numbers exhibit stronger random number characteristics, fulfilling both randomness and unpredictability.

According to the four evaluation criteria established by the BSI, weak pseudo-random number generators only meet the K1 and K2 generation standards, while strong pseudo-random number generators meet the K3 generation standard. Random number generators such as counters, linear congruential generators, and linear shift registers only satisfy the K1 and K2 generation standards, therefore they are all weak pseudo-random number generators. For random number sequences generated by weak pseudo-random number generators, we can predict subsequent random number sequences based on known sequences. Two examples are given below.

1. The rand function, frequently used in the C language. The generation of the rand function starts with a seed value, which is usually set through the srand function. If the seed is not explicitly set, the default seed value is 1. Calling srand(time(NULL)) initializes the seed using the current time. The generation of random numbers is based on the LCG algo-

rithm, where the rand function generates the next value based on the current state of the generator. To predict the next random number value of an LCG, one needs to know the specific parameters used by the generator, namely, the multiplier a, increment c, modulus m, and the current random number X_n. The formula for generating the next random number is[4]

$$X_{n+1} = (a \times X_n + c) \mod m.$$

These parameters can be solved by observing the generated pseudo-random sequence. Here's a simplified explanation of the process.

Collect data: Generate at least four consecutive pseudo-random numbers X_1, X_2, X_3, X_4.

Establish equations: Based on the LCG formula $X_{n+1} = (a \times X_n + c) \mod m$, we can establish three equations

$$\begin{cases} X_2 = (a \times X_1 + c) \mod m, \\ X_3 = (a \times X_2 + c) \mod m, \\ X_4 = (a \times X_3 + c) \mod m. \end{cases}$$

Eliminate the increment c: By eliminating the increment c, it simplifies to solving for the multiplier a and modulus m. Let $\Delta_1 = X_2 - X_1$, $\Delta_2 = X_3 - X_2$, $\Delta_3 = X_4 - X_3$, we get a new equation

$$a = \frac{\Delta_2 - \Delta_3}{\Delta_1 - \Delta_2} \mod m.$$

Solve for modulus m: By solving the system of equations and simplifying the modulus m, we can determine the value of m.

Determine the increment c: Once a and m are known, use any of the pseudo-random numbers to solve for the increment c:

$$c = (X_2 - a \times X_1) \mod m.$$

When we know these parameters and the current random number X_n, we can substitute these values to calculate the next random number X_{n+1}. In fact, in most programming environments, the parameters of the LCG are usually fixed. As long as we know these fixed parameters and the current state, we can accurately predict the next random number.

2. LFSR. In pseudo-random sequence generation algorithms, LFSRs are commonly used as the initial source of randomness. The user sets an initial value for the LFSR, also known as a random seed. The LFSR can generate sequences based on the seed. Among these, sequences generated by an LFSR with a primitive polynomial as the connection polynomial (m-sequence) satisfy Golomb's three randomness postulates.[5] Furthermore, due to its ease of implementation, LFSR is widely used.

In 1969, Massey proposed the B-M algorithm.[6] If the linear complexity of a sequence is n, then only $2n$ consecutive bits are needed to recover the entire sequence. Assume that the recursive formula for an n-th order LFSR is given by[7]

$$a_i = c_1 a_{i-1} \oplus c_2 a_{i-2} \oplus \cdots \oplus c_n a_{i-n}, \quad i \geq n.$$

We need to obtain $2n$ consecutive random number sequences to determine parameters c_1, $c_1, \ldots, c_n$. By expressing the known latter n random number sequences in terms of the former n random number sequences, we get a system of equations with $c_1, c_1, \ldots, c_n$ as the unknown variable

$$\begin{cases} a_{n+1} = c_1 a_n \oplus c_2 a_{n-1} \oplus \cdots \oplus c_n a_1, \\ a_{n+2} = c_1 a_{n+1} \oplus c_2 a_n \oplus \cdots \oplus c_n a_2, \\ \quad \vdots \\ a_{2n} = c_1 a_{2n-1} \oplus c_2 a_{2n-2} \oplus \cdots \oplus c_n a_n. \end{cases}$$

It can be proven that this system of equations is always full rank, and thus the unknown parameter $c_1, c_1, \ldots, c_n$ can be uniquely solved. Using the recursive formula, we can predict subsequent random number sequences with n consecutive random number sequences and the values of parameters $c_1, c_1, \ldots, c_n$.

Additionally, any pseudo-random sequence a of length N can be used to derive an LFSR using the B-M algorithm, so that the generated LFSR produces the same first N random numbers as the sequence a. Due to the linear nature of the generation algorithm, the linear complexity is low. For an N-stage LFSR, knowing its continuous N outputs and corresponding times allows for the entire sequence to be reconstructed.

Weak pseudo-random number generators are widely used. For instance, the A5-1 algorithm used in GSM communication employs a clock-controlled model to generate encryption randomness.[8] The E0 algorithm used for bluetooth encryption employs a nonlinear combination model.[9] However, the random numbers produced by weak pseudo-random number generators are not highly secure and can, under certain conditions, allow the reconstruction of the initial randomness source and the entire sequence. For the A5-1 and E0 algorithms, there are already well-established attack algorithms. So far, for strong pseudo-random numbers, we cannot predict subsequent random sequences based on the existing random sequences. Examples include random sequences generated by recursive call of a block cipher, counter with hash function, and RSA generator. For example, the pseudo-random number generator algorithm ChaCha in Linux kernel 4.8 or higher versions requires at least 2^{96} bits of data to predict the next random value for the 8-round ChaCha algorithm.[10]

4. The attack model of QKD system with pseudo-random sequences

In a QKD system, during the basis reconciliation process, the communicating parties will publish the basis information on a public channel, allowing eavesdroppers to easily

obtain all basis information. Since this information is acquired through a public channel, no error rate is introduced at this stage. Additionally, eavesdroppers can actively choose to perform an intercept–resend attack to obtain partial coding bit information. This type of attack introduces a certain error rate, but when the key is sufficiently long, the error rate introduced remains relatively low.

Through the above analysis, we can see that if a system uses a weak pseudo-random number generator to generate random sequences, under certain conditions, an eavesdropper can derive the entire sequence from partial pseudo-random sequence information, ultimately obtaining the generation details of the key vectors and encoded bits. The model proposed in this paper is only applicable to systems that use weak pseudo-random number generators and is ineffective for systems that use strong pseudo-random number generators.

In this security analysis, Eve will master the pseudo-random sequence generation algorithm adopted by QKD system of Alice and Bob, as well as the method of selecting basis bit and coding bit by pseudo-random sequence. There are three cases.

4.1. Alice uses the same random sequence for both basis vectors and coding bits

First scenario: Alice generates pseudo-random sequence a_i based on the pseudo-random sequence generation algorithm $f(x)$ in her QKD system. According to a_i, Alice selects her basis bit and coding bit. It is assumed that the quantity of pseudo-random numbers that are required to restore the pseudo-random sequence a_i is N, that is, for any pseudo-random sequence a_i generated by $f(x)$, all information of a_i can be restored from any N positions in a_i. Alice prepares the quantum state according to her basis bit and coding bit. Bob then measures the quantum state and carries out sifting protocol and post processing with Alice.

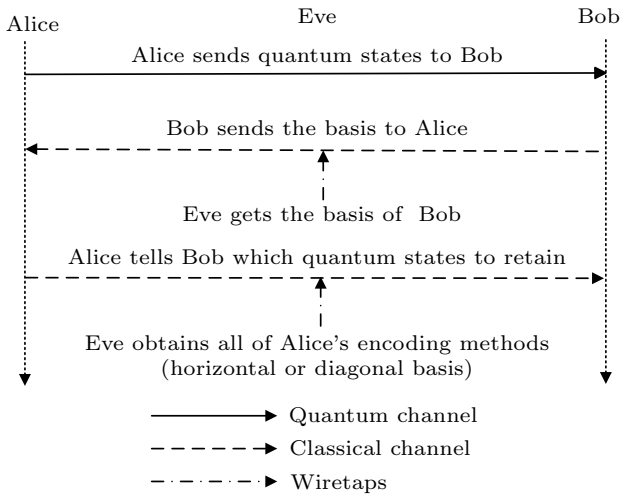


Fig. 1. The flow chart of the first scenario.

In the sifting step, Eve can obtain the basis bit of Alice and Bob. According to the relation between the pseudo-random sequence a_i and the basis bit, Eve obtains some correct value of random number in a_i.

Once the right amount Eve gets is over N, Eve can restore all the sequences a_i.

4.2. Alice uses different random sequences for basis vectors and coding bits

Alice generates pseudo-random sequences a_i^1 and a_i^2 by pseudo-random sequence generation methods $f(x)$ and $g(x)$ in her QKD system. According to a_i^1, Alice selects her basis bit, and according to a_i^2, Alice selects her own coding bit. It is assumed that the quantities of pseudo-random numbers that are required to restore the pseudo-random sequences a_i^1 and a_i^2 are N_1 and N_2, that is, for any pseudo-random sequence a_i^1 generated by $f(x)$, all information of a_i^1 can be restored from any N_1 positions in a_i^1; for any pseudo-random sequence a_i^2 generated by $g(x)$, all information of a_i^2 can be restored from any N_2 positions in a_i^2. Alice prepares the quantum state according to her basis bit and coding bit. Bob then measures the quantum state and carries out sifting protocol and post processing with Alice.

Second scenario: In two consecutive quantum state transmissions, if the encoding method of Alice's quantum state is from the same pseudo-random sequences, Eve can use the attack method in the first case. Eve restores pseudo-random sequence a_i^1 after the first sift step, and then deduces the basis bit of the second quantum state transmission. Eve uses the same basis bit as Alice to measure the quantum state in the second transmission. At this point, no quantum state will be changed, and no error bit will be generated.

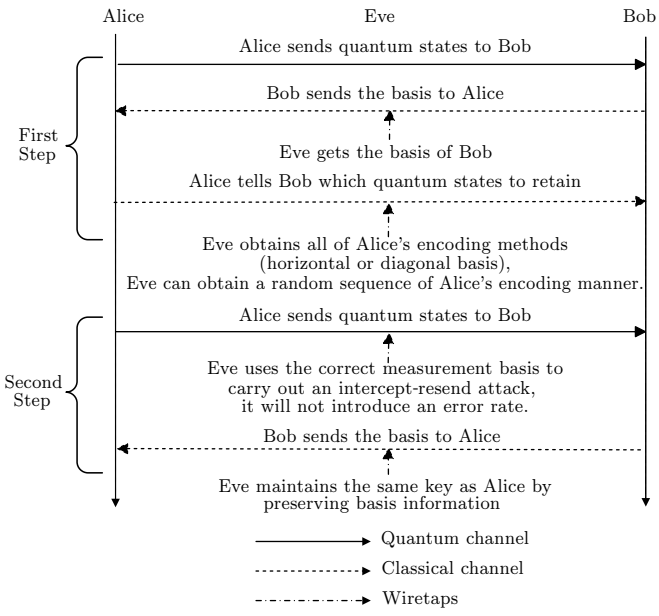


Fig. 2. The flow chart of the second scenario.

Third scenario: In two consecutive quantum state transmissions, if the encoding method of Alice's quantum state uses different pseudo-random sequences, we cannot predict the random number sequence used in Alice's encoding method. At this point, Eve can use partial "intercept–resend" to eavesdrop.

Eve obtains N_0 bits of quantum state information through an intercept-and-resend attack. Since Eve randomly selects the measurement basis, Eve may choose the incorrect basis, leading to errors in the N_0 bits of information. After Alice and Bob disclose their bases, Eve can determine which of her measurement bases are correct and only retain the quantum state information at those positions. Since the probability of Eve choosing the correct measurement basis is 50%, she can acquire $N_0/2$ bits of information about the random number sequence a_i^2. When the number of correct bits exceeds N_2, Eve can reconstruct a_i^2 and obtain all of Alice's encoded bit information.

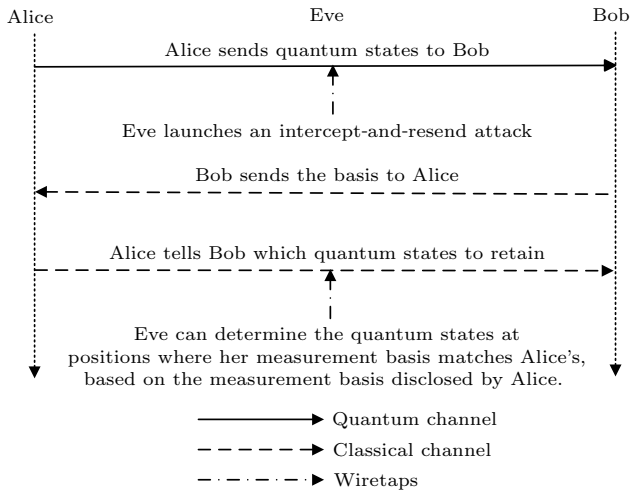


Fig. 3. The flow chart of the third scenario.

5. Performance analysis of the attack model

5.1. Validity

In the above attack model, Eve will get all the quantum state information as long as the correct number in the pseudo-random sequence she gets exceeds N. If Eve achieves this goal, this attack is effective. Now let's analyze how Eve can get N correct random numbers.

Suppose Alice and Bob disclose N_0 basis vector choices during basis comparison. Since basis comparison is conducted over a public channel, Eve can obtain N_0 bits of basis vector information. Thus, when N_0 satisfies

$$N_0 > N.$$

Eve gets N random numbers in a^i, Eve can obtain all of Alice's basis vector choices. In the first scenario, since the same random sequence is used, Eve does not need to take any action to obtain all quantum state information. In the second scenario, since Eve already knows the subsequent basis vector information, she can use the correct measurement basis to perform an intercept-and-resend attack. In this case, she correctly obtains all quantum state information without introducing any error rates.

In the third scenario, since the selection of the basis vector of Eve is also random, the probability that Eve and Alice select the same basis bit for the same quantum state is 50%. Assuming that Eve measures N_0 quantum states, when N_0 satisfies

$$N_0/2 > N.$$

Eve gets N random numbers in a_i^2. So just intercepting and resending $2N$ quantum states on average, Eve gets the N correct random numbers in sequence a_i^2.

5.2. Reliability

Reliability is considered from the aspect of bit error rate introduced by the attack method. If Eve carries out the attack without being found, this attack is reliable.

In the first scenario and second scenario, since Eve does not eavesdrop on the quantum channel, there is no extra bit error rate. Alice and Bob do not discover the presence of Eve.

In the third scenario, suppose the number of quantum states transmitted by Alice on the quantum channel is a_n, and the number of quantum states which Eve has handled is $2N$. Based on the bit error rate analysis of the protocol, in $2N$ quantum states, on average about N quantum states can cause error codes. Therefore, the increased bit error rate due to eavesdropping is N/a_n. So, the smaller the ratio of N to a_n, the smaller the error rate.

6. Case study

The core point of the attack model proposed in this paper is to predict subsequent sequences through existing random sequences. Therefore, as long as the system uses a weak pseudo-random number generator, we can launch an effective attack. For instance, the previously mentioned rand function and LFSR can be easily predicted to determine the subsequent random sequences, so further examples are not provided here.

The above mentioned E0 algorithm is an international standard bluetooth encryption algorithm. The random number generated by the encryption has good randomness. In the third scenario, Alice uses different random number generators for the basis vector information and the quantum bit information, and uses the pseudo-random sequences generation algorithm in E0 algorithm to generate pseudo-random sequences. Each number in a_i^2 decides one coding bit option.

Table 1 is the performance index needed to restore the random number generated by E0 algorithm.[11–14] If Eve employs a BDD attack, when she wiretaps in the average 2^8 quantum states, she can get 2^7 random numbers in a_i^2 and restores a_i^2. If Alice and Bob negotiate the quantum state of 1M bits, the bit error rate introduced by Eve eavesdropping is

$$2^7/2^{20} = 2^{-13}.$$

Table 1. Simulation parameters.

Attack method	The size of the N	Time complexity	Memory complexity	Required quantum state
Algebraic attack	$2^{23.07}$	$2^{67.5}$	$2^{46.14}$	$2^{24.07}$
Fast algebraic attack	$2^{23.4}$	2^{49}	2^{37}	$2^{24.4}$
BDDattack	2^{7}	2^{81}	2^{81}	2^{8}
Fast correlation attack	2^{37}	$2^{35.1}$	2^{32}	2^{38}

Since the QKD system generally sets the security upper limit of bit error rate to 11%, at this point, the error bit rate can be ignored.

In the first scenario and second scenario, because Eve can get part of the pseudo-random sequence information in the public channel during sifting protocol, once Alice and Bob disclose the number of measurement bases exceeds 2^7, Eve can restore pseudo-random number sequence through BDD attack, and get all of the information of the key.

7. Conclusion

In quantum key distribution, to ensure system security, true random sequences with good randomness properties should be used. In pseudo-random sequences, the correlation between individual random numbers allows an eavesdropper to reconstruct the entire sequence from partial information. The analysis presented in this paper sets clear requirements for the quality of random numbers for the practical security evaluation of quantum key distribution protocols.

From the previous analysis, it is evident that in quantum key distribution systems, random sequences generated by weak pseudo-random number generators, such as the rand function and linear feedback shift register, should not be used. However, strong pseudo-random number generators can be used with a slight compromise in security, provided that random seeds are frequently changed, such as the pseudo-random number generator algorithm ChaCha in Linux operating systems.

References

[1] Shannon and Claude E 1949 *Bell System Technical Journal* **28** 656

[2] Bennett C H and Brassard G 1984 *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, December 10–12, 1984, Bangalore, India, p. 175

[3] Priv.-Doz.Dr. W Schindler 1999 *Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators* (Bonn: BSI) pp. 5–11

[4] Knuth D E 1997 *The Art of Computer Programming*, 3nd edn. (Boston: Addison-Wesley) pp. 10–26

[5] Golomb and Solomon W 1972 *AIEE Transactions on Information Theory* **18** 80

[6] Massey and James L 1969 *IEEE Transactions on Information Theory* **15** 122

[7] Jin C H, Zheng H R and Zhang S W 2009 *Cryptography* (Beijing: Higher Education Press) pp. 86–88 (in Chinese)

[8] Biham E and Dunkelman O 2000 *Progress in Cryptology: INDOCRYPT* 2000 (Heidelberg: Springer-Verlag) pp. 43–51

[9] Lu Y and Vaudenay S 2004 *Advances in Cryptology-ASIACRYPT* 2004 (Heidelberg: Springer-Verlag) pp. 483–499

[10] Coutinho M and Passos I 2022 28*th International Conference on the Theory and Application of Cryptology and Information Security*, December 5–9, 2022, Taipei, Taiwan, pp. 256–286

[11] Armknecht F and Krause M 2003 23*th Annual International Cryptology Conference*, August 17–21, 2003, Santa Barbara, USA, pp. 162–175

[12] Hawkes P and Rose G 2024 24*th Annual International Cryptology Conference*, August 15–19, 2004, Santa Barbara, USA, pp. 390–406

[13] Krause M and Stegemann D 2006 *Proceedings of the* 13*th International Workshop on Fast Software Encryption* (Heidelberg: Springer-Verlag) pp. 163–178

[14] Zhang B and Feng D G 2011 *Science China Information Sciences* **54** 1635