Efficient error estimation in quantum key distribution*

Li Mo(李 默)[a)b)], Patcharapong Treeviriyanupab[c)], Zhang Chun-Mei(张春梅)[a)b)], Yin Zhen-Qiang(银振强)[a)b)†], Chen Wei(陈 巍)[a)b)‡], and Han Zheng-Fu (韩正甫)[a)b)]

[a)] *Key Laboratory of Quantum Information, Chinese Academy of Sciences, University of Science and Technology of China, Hefei* 230026, *China*
[b)] *Synergetic Innovation Center of Quantum Information & Quantum Physics, University of Science and Technology of China, Hefei* 230026, *China*
[c)] *Department of Computer Science, Faculty of Science and Technology, Phranakhon Rajabhat University, Bangkok, Thailand*

(Received 24 February 2014; revised manuscript received 25 July 2014; published online 25 November 2014)

In a quantum key distribution (QKD) system, the error rate needs to be estimated for determining the joint probability distribution between legitimate parties, and for improving the performance of key reconciliation. We propose an efficient error estimation scheme for QKD, which is called parity comparison method (PCM). In the proposed method, the parity of a group of sifted keys is practically analysed to estimate the quantum bit error rate instead of using the traditional key sampling. From the simulation results, the proposed method evidently improves the accuracy and decreases revealed information in most realistic application situations.

Keywords: error estimation, parity comparison, quantum key distribution

PACS: 03.67.Dd, 03.67.–a, 03.67.Hk **DOI:** 10.1088/1674-1056/24/1/010302

1. Introduction

Quantum key distribution (QKD) allows two parties, called Alice and Bob, to share a common string of secret keys even in the presence of an adversary, called Eve. Since the proposal in 1984,[1] there has been much significant progress towards its practical implementation.[2–4] In the QKD protocol, Alice and Bob can share the string of keys over quantum communication. However, there may exist some discrepancies between the shared keys due to the imperfect control of devices and the disturbance of environment, or the behavior of Eve. The error rate caused by these factors should be estimated because: (i) in order to fulfill the security requirements, the error rate needs to be estimated as an important reference for detecting the presence of Eve; (ii) in a practical QKD system, post-processing is inextricably needed to eliminate the discrepancies between the shared keys. The estimated error rate can be used to optimize the performance of post-processing, in particular at key reconciliation stage. Several methods for key reconciliation have been proposed, such as Cascade,[5] and LDPC.[6,7] In Cascade, the knowledge of error rate is helpful to set an optimal size of block length that significantly saves its time overhead in practice.[8] In LDPC, the error rate can help choose an appropriate code rate which can improve the efficiency and achieve higher key throughput.

For the purpose of clarity, we first categorize the error rates in QKD. Assume Alice and Bob share n bits sifted key, there are three kinds of error rates involved.

$e_{\mathrm{sift}}^{\mathrm{est}}$: It is the estimated error rate of the n bits sifted keys and generally obtained before key reconciliation.

e_{sift}: It is the error rate of the n bits sifted keys. If there are n_{e} bits discrepant between Alice's and Bob's sifted keys, e_{sift} is n_{e}/n, where e_{sift} can be precisely obtained once Alice and Bob have completed key reconciliation, by comparing the keys after reconciliation with the keys before.

e_{∞}: It is the asymptotic value of e_{sift} as the sifted keys' size n becomes infinite. However, because in the practical application the size of sifted keys is finite, e_{sift} is probably more or less deviated from e_{∞}.

As a summary, $e_{\mathrm{sift}}^{\mathrm{est}}$ is an estimation of e_{sift}, while e_{sift} is an estimation of e_{∞}, e_{sift} needs to be estimated instantly in each reconciliation round for the purpose of good post-processing performance. In this paper the term "error estimation" refers to the estimation of e_{sift} or in other words, the process of obtaining $e_{\mathrm{sift}}^{\mathrm{est}}$. We adopt the above notations throughout this paper.

In the traditional method of error estimation, the sampling of sifted keys is revealed directly from Alice to Bob or vice versa, which is around 10%–20% of sifted keys' size in general.[9–11] Then they can determine the $e_{\mathrm{sift}}^{\mathrm{est}}$ by comparing the sampling keys. However, this method cannot produce a completely accurate result when the size of sifted keys is not big, say several kbits, which is a common situation in some reconciliation protocol. For example, in the reconciliation protocol using short-length LDPC codes, the sifted keys are arranged into blocks of several-thousand bits depending on the optimal code rate. Increasing the amount of sampling is one solution for achieving higher accuracy, but that simultaneously means more waste of sifted keys. In a practical QKD

*Project supported by the National Basic Research Program of China (Grant Nos. 2011CBA00200 and 2011CB921200) and the National Natural Science Foundation of China (Grant Nos. 61101137, 61201239, and 61205118).

†Corresponding author. E-mail: yinzheqi@mail.ustc.edu.cn
‡Corresponding author. E-mail: kooky@mail.ustc.edu.cn

© 2015 Chinese Physical Society and IOP Publishing Ltd

system, the generation speed of sifted keys can be as high as only around 3 Mbits per second at the distance of 20 km,[12] because of which sifted keys are precious and should be reconciled with disclosure as little as possible.

In this paper, an efficient error estimation method based on parity comparison is proposed, which shows good property both at short and long key sizes. First we elaborate on how the proposed method works. Then several simulations related are conducted, for investigating the performance of the proposed method.

2. Parity comparison

In this section, the parity comparison method (PCM) is discussed with a schematic diagram. The traditional method (TM) and the PCM are drawn in Fig. 1.

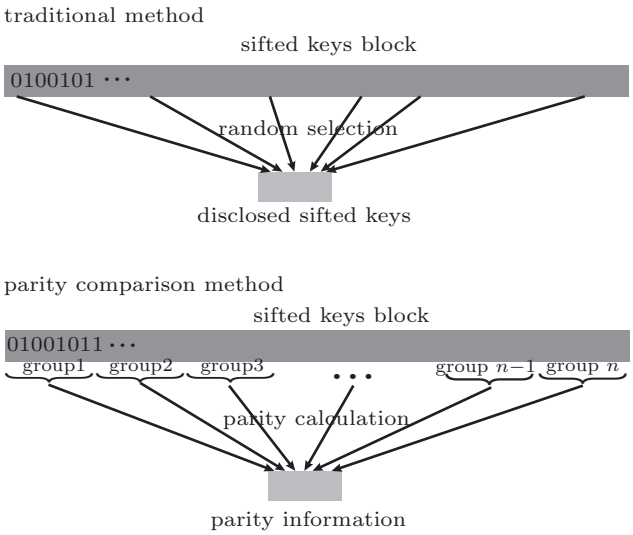


Fig. 1. The schematics of two methods. In the traditional method, the sent message (the disclosed sifted keys) is randomly chosen from the sifted keys. In the PCM method, the sent message (parity information) is calculated group by group from the sifted keys.

In the TM, a sample of the sifted keys is randomly selected and revealed, through which Alice and Bob can get an error rate e_{sample}, which is then taken to be $e_{\text{sift}}^{\text{est}}$ directly. In contrast, in the PCM, the sifted keys are divided into groups, and the parity bit of each group is calculated

$$b_{\text{parity}} = \left(\sum_{i \in \text{group}} b_i \right) \bmod 2, \tag{1}$$

where $b_i, b_{\text{parity}} \in \{0,1\}$. Then all the parity bits are revealed and compared between Alice and Bob. They can obtain the error rate of the parity bits: e_{parity}. However, e_{parity} will not be considered as $e_{\text{sift}}^{\text{est}}$ directly. $e_{\text{sift}}^{\text{est}}$ should be retrieved according to the following lemma.[13]

Lemma Let Alice and Bob share a string of L sifted key bits. The probability for each corresponding bit in Alice's string and Bob's string to be different is e_{sift} and independently and identically distributed. Then the probability that Alice and Bob have different parities of the string is

$$p = \frac{1 - (1 - 2e_{\text{sift}})^L}{2}. \tag{2}$$

Proof Alice and Bob have different parities of the string when there is an odd number of wrong bits in Bob's string relative to Alice's. Thus the probability can be expressed as

$$p = \sum_{i=\text{odd}}^{L} \binom{L}{i} e_{\text{sift}}^i (1 - e_{\text{sift}})^{L-i}. \tag{3}$$

To prove that equation (2) is equivalent to Eq. (3), we just simply rewrite the right-hand side of Eq. (2) as

$$\begin{aligned} &\frac{1 - (1 - 2e_{\text{sift}})^L}{2} \\ &= \frac{(1 - e_{\text{sift}} + e_{\text{sift}})^L - (1 - e_{\text{sift}} - e_{\text{sift}})^L}{2}, \end{aligned} \tag{4}$$

and expand it in binomial series, which will obviously lead to Eq. (3).

It is easily deduced that an odd number of wrong bits in a group in the PCM will lead to a wrong b_{parity}. Let the group size be L as in the lemma. The relation $e_{\text{parity}} = p_{\text{odd}}$ is clear. Thus our goal is to obtain e_{sift} with e_{parity} through the above lemma. However, equation (2) is difficult to solve backwards. It needs to be done in another way. Actually, a look-up table of Eq. (2) can adequately accomplish the task. We can in advance store the e_{parity} values with respect to all e_{sift} values. The e_{sift} can be thus easily retrieved. In practice, once the e_{parity} is acquired, the corresponding e'_{sift} in the table which gives the nearest e'_{parity} to e_{parity} is thought to be $e_{\text{sift}}^{\text{est}}$. The table can be made with arbitrary accuracy.

A rough security analysis is given here. Let N be the size of sifted keys. In the TM, if the ratio of sampling is α, the amount of leaked information of the sifted keys to Eve can be taken to be αN. In the PCM, if the size of a group is n, the amount of leaked information to Eve can be taken to be N/n. Thus, under the condition $\alpha = 1/n$, the security levels of the two methods are identical.

3. Simulation

For evaluating the estimation method, we need to investigate its average deviation from the real value. The average deviation is $\Delta = \overline{|e_{\text{sift}}^{\text{est}} - e_{\text{sift}}|}$. The simulation of Δ for the TM and PCM is given in Fig. 2. The simulation is conducted with the sifted key size being 5000 bits and the result is averaged over 3000 times of simulations. In the TM, the sample ratio is $1/10$, while in the PCM the group size is 15. From the previous section, it is known that the PCM leaks less information than the TM.

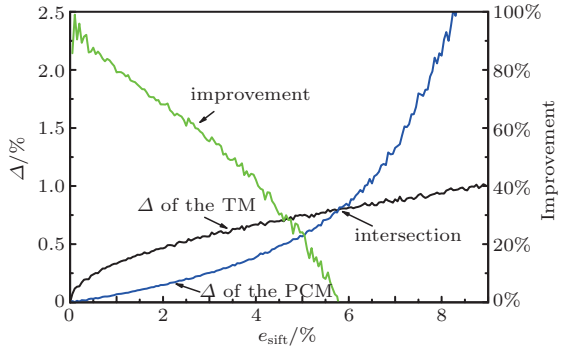


Fig. 2. (color online) Simulation results of the PCM and TM. The sifted keys' length for simulation is 5000 bits. The two lines illustrate the deviations of the two methods averaged over 3000 times. PCM outperforms the TM when the error rate is not very high, less than 6%.

It can be seen that although with less information leakage, the PCM still outperforms the TM in the error region less than around 6%. The improvement is defined as

$$\text{improvement} = \frac{\Delta_{\text{TM}} - \Delta_{\text{PCM}}}{\Delta_{\text{TM}}}. \tag{5}$$

As shown in Fig. 2, a 50% improvement at around $e_{\text{sift}} = 3\%$ is achieved. When it is higher than 6%, the PCM is less accurate than TM. However, this inaccuracy at high error rate does not influence PCM's practicability substantially, because nowadays the QKD system is typically running under an error rate of less than 3%,[12,14] and what is more, few keys are generated when the system is running under an error rate higher than 6%. For the above reasons, the PCM is of much better practicability at short key sizes such as several kbits.

The above simulation uses a short key size of 5000 bits for error estimation, which is often needed when adopting LDPC as the error reconciliation method. However, if the reconciliation is not carried out by LDPC, for example Cascade, or if the error estimation is to only serve for a security parameter, the length of the string should be long enough, typically 1 Mbits, in order to eliminate the finite-size effect. Here we study the performance of the two methods when the sifted keys' length increases from 5000 bits to 1 Mbits. Recall that in Fig. 2 the two lines intersect at around 6%, which means the TM outperforms the PCM at this error rate. In other words, the intersection is the maximum error rate that the PCM is better than the TM. Therefore we can investigate the intersection at different sizes of sifted keys, which can indicate the tendency of accuracy shift of the two methods.

Figure 3 clearly shows that the TM outperforms the PCM at a smaller and smaller error rate as the sifted keys' block length grows, which means that the PCM is less and less advantageous. However, this does not destroy PCM's advantage over the TM, because the QKD system nowadays often runs at an error rate lower than 3% and what is more, in the reconciliation based on LDPC, the code length needed for error estimation or error correction is normally less than 1 Mbit.[15,16] So the PCM still has the advantage over the TM in the realistic situation.

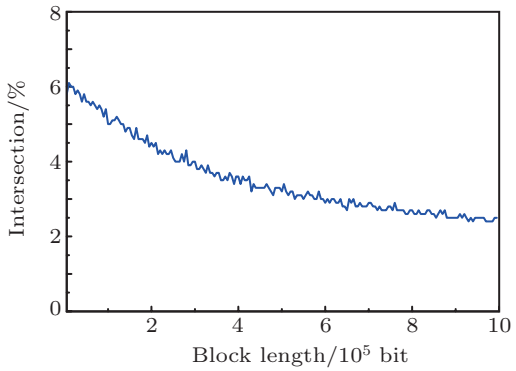


Fig. 3. (color online) Simulation results of the shift of the intersection of the PCM and TM. It indicates that the intersection point decreases as the length grows, which means that the PCM becomes less and less advantageous.

Finally, we investigate the effect of different group sizes on the PCM. The group size changes from 5 to 30 while the sifted key length is fixed at 1 Mbits. The sample ratio of the TM is fixed at 1/10. Similar to the previous simulation, we study the position of the intersection.

Figure 4 shows that as the group size increases, the intersection becomes smaller and smaller. This is reasonable because as mentioned above, a bigger group size corresponds to smaller disclosure. However, it is easily understood that to achieve a higher accuracy more information needs to be disclosed. Therefore bigger group size shows worse performance. Nevertheless, it can be seen that the PCM can still outperform the TM in the error rate region smaller than 2% even when the group size is 30, corresponding to 1/3 of the disclosed amount of the TM, which is very economic. Thus PCM is very meaningful and practical in real situations.

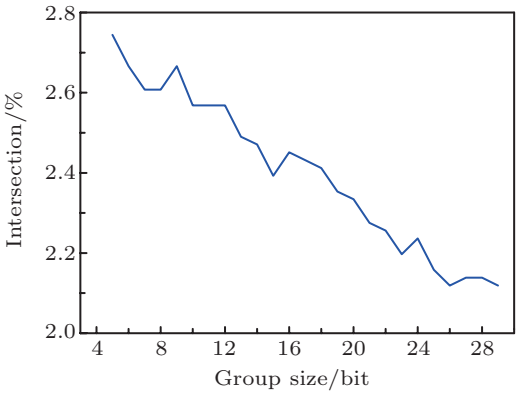


Fig. 4. (color online) The position of intersections when the group size of PCM changes while the sample ratio of the TM is fixed at 1/10. The sifted keys' length is fixed at 1 Mbit.

4. Conclusion

We propose a simple but efficient method, called PCM, of key reconciliation for QKD. The PCM can outperform the TM

in most realistic scenarios. Furthermore, the PCM can meet the same accuracy level with much less information leakage about the sifted keys, saving more sifted keys for generating secure keys. It should be emphasized that the $e_{\text{sift}}^{\text{est}}$ is not used for privacy amplification. It is e_{∞} that should substitute the error rate in the security analysis and final key generation.[17] Once Alice and Bob have finished reconciling their keys, they can accurately get the e_{sift} as stated in the introduction section. Based on enough e_{sift}, a good estimated e_{∞} can be achieved.

References

[1] Bennet C H and Brassard G 1984 *Processings of IEEE International Conference on Computers, Systems and Singnal Processing,* p. 175

[2] Peev M, Pacher C and Alléaume R 2009 *New J. Phys.* **11** 075001

[3] Xu F X, Chen W, Wang S, Yin Z Q, Zhang Y, Liu Y, Zhou Z, Zhao Y B, Liu D, Han Z F and Guo G C 2009 *Chin. Sci. Bull.* **54** 2991

[4] Fröhlich B, Dynes J F, Lucamarini M, Sharpe A W, Yuan Z L and Shields A J 2013 *Nature* **501** 69

[5] Brassard G and Salvail L 1994 *Advances in Cryptology–EUROCRYPT'*93 (Berlin, Heidelberg: Springer-Verlag) p. 410

[6] Pearson D 2004 *AIP Conference Proceedings* **734** 299

[7] Jesus M M, Elkouss D and Martin V 2013 *Sci. Rep.* **3** 1576

[8] Cui K, Wang J, Zhang H F, Luo C L, Jin G and Chen T Y 2013 *IEEE T. Inf. Foren. Sec.* **8** 184

[9] Gisin N, Ribordy G, Tittel W and Zbinden H 2002 *Rev. Mod. Phys.* **74** 145

[10] Poppe A, Fedrizzi A, Ursin R, Böhm H R, Lorünser T, Maurhardt O, Peev M, Suda M, Kurtsiefer C, Weinfurter H, Jennewein T and Zeilinger A 2004 *Opt. Express* **12** 3865

[11] Scarani V, Bechmann-Pasquinucc H, Cerf N J, Dušek M, Lütkenhaus N and Peev M 2009 *Rev. Mod. Phys.* **81** 1301

[12] Dixon A R, Yuan Z L, Dynes J F, Sharpe A W and Shields A J 2008 *Opt. Express* **16** 18790

[13] Gallager R 1963 *Low-Density Parity-Check Codes* (Ph.D. Thesis) (Massachusetts Institute of Technology)

[14] Wang S, Chen W, Guo J F, Yin Z Q, Li H W, Guo G C and Han Z F 2012 *Opt. Lett.* **37** 1008

[15] Mo X F, Lucio-Martinez I, Chan P, Healey C, Hosier S and Tittel W 2011 *Opt. Express* **19** 17729

[16] Tanaka A, Fujiwara M, Yoshino K, Takahashi S, Nambu Y, Tomita A, Miki S, Yamashita T, Wang Z, Sasaki M and Tajima A 2012 *IEEE J. Quantum Electron.* **48** 542

[17] Gottesman D, Lo H K, Lütkenhaus and Preshill J 2004 *Quant. Inf. Comput.* **5** 325