Efficient Key Integrity Verification for Quantum Cryptography Using Combinatorial Group Testing

Junbin Fang[a,b], Zoe L. Jiang[a], S. M. Yiu[a], Lucas C. K. Hui[a] and Zichen Li[c]

[a]Department of Computer Science, The University of Hong Kong,
Pokfulam Road, Hong Kong SAR, China
[b]School for Information and Optoelectronic Science and Engineering,
South China Normal University, Guangzhou University Town, 510006 China
[c]Department of Information Security,
Beijing Electronic Science and Technology Institute, 100070 China

ABSTRACT

In quantum cryptography, the key can be directly distributed to the communicating parties through the communication channel. The security is guaranteed by the quantum properties of the channel. However, the transmitted key may contain errors due to the noise of the channel. Key integrity verification is an indispensable step in quantum cryptography and becomes an important problem in higher speed systems. Computing only one hash value for the key string does not provide an effective solution as it may lead to dropping all the bits once the hash values on both sides do not agree. In this paper, we introduce a new idea of using the technique of combinatorial group testing, which seems to be an unrelated topic, to design a scheme to identify the error bits to avoid dropping all the bits. Our scheme can precisely locate the error bits if the number of error bits is within the maximum set by the scheme while the overhead is insignificant based on our experiments (additional bits: 0.1% of the key; time for computing the hash values: 16ms; verification time: 22 ms). Also, even if the number of error bits is higher than the maximum set by the scheme, only some correct bits may be misclassified as error bits but not the vice versa. The results show that we can still keep the majority of the correct bits (e.g. the bits discarded due to misclassification is only 5% of the whole string even if the number of error bits is 10 times of the maximum).

Keywords: Quantum cryptography, key integrity, combinatorial group testing, shifted transversal design

1. INTRODUCTION

Quantum cryptography (QC) provides an absolutely secure manner for distributing secret keys between two distinct parties.[1,2] It is expected to work along with one-time-pad cipher[3] to offer provably unbreakable encryption. Although the security of QC can be guaranteed by the quantum properties of channel connecting two parities, physical imperfections of channel may introduce noise into the messages passing through it, and thus the shared secret keys may contain error bits. Therefore, after a key has been transmitted via quantum channel, it is necessary for both remote parties, such as Alice and Bob, to reconcile the key bits of each other by public discussion to make them identical.[4]

The first step of key reconciliation is interactive error correction[5] and the few-percent error rate of quantum keys will be corrected down to the standard 10^{-9} error rate, typical in classical communication. Since error correction is always probabilistic unless all bits are revealed during the interactive process, there is a small possibility that Alice and Bob believe that they share identical bit sets, but in fact, they do not. Thus, before the error correction's end result can be used for further processes, such as privacy amplification,[6] it is essential to

Further author information: (Send correspondence to Zoe L. Jiang)
Junbin Fang: E-mail: junbinfang@gmail.com
S. M. Yiu, Lucas C. K. Hui: E-mail: {smyiu,hui}@cs.hku.hk
Zoe L. Jiang: E-mail: zoeljiang@gmail.com
Zichen Li: E-mail: lizc@besti.edu.cn

Quantum Information and Computation VIII, edited by Eric J. Donkor, Andrew R. Pirich, Howard E. Brandt,
Proc. of SPIE Vol. 7702, 77020F · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.853234

verify the integrity of the key, i.e., to check whether Alice and Bob have identical copies of a set of error-corrected bits in their local memories.

In the original BB84 protocol[7] – the first practical QC protocol, both Alice and Bob compute a hash value on the error-corrected bits of a key string individually and compare it publicly. If the hash values don't match, all the bits of the key string will be discarded. While this method is quit straightforward, it can only provide a yes-or-no answer about the integrity of the whole string. Even if there is only one error bit existing in key string, all the other correct bits have to be sacrificed. This method is suitable for previous QC systems, which operate at a very low rate of key production, about several hundred bits per second (bps). In those systems, the length of a key string to be verified at a time may be about several hundred bits.[8] Since the bit error rate has been downed to 10^{-9} by error correction, the event that such a key string of finite length cannot pass the integrity verification will occur with a very low frequency (once per 10^6 times) and the bits that have to be discarded for this rare event will be just a negligible fraction of all the key bits.

However, this scenario has been changed due to the fact that the rate of key production in QC system has been increased to a few Mbps[9,10] and the chunk size of a key string to be verified should be adjusted to relatively long length, such as 100K∼1M bits, to fit the higher speed. Otherwise, comparing a great deal of hash values will add undue communication burden. In this case the yes-or-no answer for key integrity verification may not be good enough because such a longer length will raise the failure chance of integrity verification and increase the number of discarded key bits. For example, consider the experimental QC system in Ref. 11, where real-time video is encrypted by one-time-pad cipher. In this system, the rate of key production is 1 Mbps and the chunk size of key string is 1 Mbits. Once the chunk fails the integrity verification the whole chunk of 1 Mbits will be dropped. As a result the transmission of encrypted video will be interrupted or delayed for one second. With the rate of key production increasing more and more,[12] this problem will become more serious and the original method for integrity verification will be too preliminary and no longer appropriate for the high-speed QC system.[13,14]

To solve this problem, this paper proposes a new approach for key integrity verification by adapting the concept of combinatorial group testing (CGT). Our approach is about going beyond the yes-or-no answer afforded by the original method: given a n bits key string and any constant upper bound on the number of error bits, we try to enable differentiating between correct bits and rare error bits in a key string so as to avoid the great majority of correct bits discarded, at the same time keep the computation and information transmitted as little as possible.

1.1 Combinatorial group testing

CGT seems to be unrelated to quantum cryptography, but it turns out to be a useful technique in solving the key integrity problem. CGT was first proposed by Dorfman[15] during World War II when blood samples of millions of draftees were subject to identical analyses in order to detect a few thousand cases of syphilis. In the original scheme, tester first chooses some blood samples to form a group, extracts a few drops from each of these chosen samples and mixes the drops together, then the tester will test the mixed sample of the group instead of testing the samples one by one. When the outcome of the test is negative, then all samples in the group are clean (disease-free). Otherwise, it implies that this group contains at least one defective sample. From the outcomes of remarkably few such group tests, it is possible to infer which of a large set of n samples are clean and which are defective.

Here is a simple example. Assuming that a set C under test contains five samples: {1,2,3,4,5} and no more than one sample in the set is defective, a pattern of group testing can be designed as the following 3×5 binary matrix represents. $M = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$, where each column corresponds to a sample of C, each row corresponds to a group test and each cell (i, j) is 1 if and only if sample j is included in the test group i. In other words, in our example, for the test group 1, we mix samples 4 and 5 together. Therefore, with this testing pattern, if the group tests on {1,3,5} (test group 3) and {2,3} (test group 2) show that one of the samples is defective while the test on {4,5} (test group 1) indicates all samples in this test are clean, then we can conclude that sample at column 3 is the defective one. One can check for other possibilities that these three test groups

suffice to deduce the defective sample. In this case, only 3 group tests but not 5 individual tests are needed to find out the defective sample. In such a way, the cost of identifying which of n blood samples are tainted can be significantly decreased by applying tests to judiciously chosen subsets of the samples - given any upper bound on the number of tainted samples, a logarithmic (in n) number of tests suffices.

Due to the high efficiency of CGT, in the 1960s, Sobel et al.[16] revived the interest in it by giving many industrial applications in detecting chemical leakage and electrical blocking. At present, the applications of CGT are not limited to testing but extended to a vast variety of fields[17,18] including DNA library screening, communication, coding theory.

The key issue of CGT algorithm is how to design the testing groups, i.e., how to choose subsets of a set into groups, so as to minimize the total number of tests with guaranteed ability of locating the defective samples. A CGT algorithm can be non-adaptive or adaptive.[19] In non-adaptive CGT, all the subsets to be tested must be decided before any subset is tested. In adaptive CGT, by contrast, the next subset to be tested has to be chosen based on the outcomes of the previous tests. Although adaptive designs can require fewer tests, we are interested here in non-adaptive CGT because it allows parallelization and need not multi communication round-trips between Alice and Bob.

There are several known methods for non-adaptive CGT design, e.g. set packing designs,[20] direct constructions.[21] In Ref. 22, Ngo et al. have made a survey on CGT algorithms and also have given a taxonomy of non-adaptive CGT designs. In 2006, an efficient non-adaptive combinatorial group construction, the Shifted Transversal Design (STD) was proposed by Thierry-Mieg,[23] which is highly flexible and can be tailored to function robustly. It will be used in our application.

1.2 Our Contributions

In this paper, CGT algorithm is used to solve the key integrity problem in QC system such that the integrity of the correct bits in key string can be efficiently verified and the rare error bits can be accurately identified. It is necessary to claim that the paper does not focus on developing novel CGT algorithm theoretically, but on singling out the most appropriate CGT algorithm and adapting it to help solving the problem. The following shows the details of our contributions.

- We propose a new idea by adapting CGT algorithm for verifying the integrity of key bits in QC system. Our design is based on STD construction which is the most efficient approach for designing test groups in a non-adaptive CGT algorithm up to now. Thus, the cost for key integrity verification can be as little as possible.
- The implemented scheme provides a guaranteed ability of identifying all bits accurately and detecting the error bits provided that the number of such error bits is no more than the threshold we set in the design. Even if the number of error bits excesses the threshold, this scheme can still verify a great majority of correct bits.
- By comparing the performance of the scheme with the original method, it shows that the scheme is more efficient than the original method. The significant improvement is that the scheme can save the correct bits when rare error bits exist in a key string while the original method has to sacrifice all the bits in this situation.

1.3 Outline

The organization of the rest of the paper is as follows. In Sec. 2, we will explain our idea of adapting CGT to solve the key integrity problem and review the CGT construction by STD, especially the parameter design. Our experimental results are shown and explained in Sec. 3. Sec. 4 concludes the paper.

2. KEY INTEGRITY VERIFICATION USING CGT APPROACH

In this section, we first introduce the idea of adapting CGT into key integrity verification. Then by reviewing the selected construction - STD, some specific concerns in terms of integrity verification are also considered.

2.1 Adapting CGT to solve the key integrity problem

CGT originated from a very simple requirement of cost saving,[15] so it is mainly applied to situations where it is very expensive to test every single item individually of a large number of items and it is necessary to pinpoint the defective items precisely. The basic idea of CGT is to combine the members of a set under test into groups and run tests on the groups rather than on individual entries. When CGT is introduced into our application to deal with the key integrity problem, an analogy should be made first. Assuming that a correct bit in a key string is a clean item and an error (or mismatched) bit is a defective one, grouping a subset of a n-bits key string to form a chain of bits and computing a cryptographic hash for this chain can be analogous to a mix of blood drops from a subset of n blood samples respectively. In our approach, Alice first computes such a hash value for a chain of Alice's key bits and sends it to Bob. Bob also computes a hash value for the corresponding chain of Bob's key bits independently. When Bob receives the hash value, he compares the hash value from Alice with the one from himself. The comparing process is analogous to applying a blood test to the mix of blood drops. A mismatch between the hash value computed from Alice and that from Bob indicates that there is at least one error bit in the chain of bits corresponding to that hash value. From the results of all the comparisons, Alice (or Bob) can precisely identify each bit as correct or error according to the decoding algorithm of CGT. Once Alice (or Bob) makes out the position of an error bit clearly, she (or he) can simply eliminate that bit or reverse it to correct the error.

Applying CGT in key integrity problem has the following advantages. Firstly, CGT can accurately distinguish between correct bits and error bits in a key string if the number of error bits is within the guaranteed bound of the CGT design, while the original yes-or-no method of verification can only give a coarse answer for the whole key string. Thus, by keeping the correct bits unchanged and reversing the indicated error bits, our CGT approach can preserve all bits of the key string, which could be discarded totally in the original scheme of key integrity verification with answer NO. Secondly, even if the actual number of error bits excesses the number guaranteed by the CGT design, our approach can still identify the great majority of correct bits by the decoding algorithm of CGT. The experimental results in Sec. 3 will show this point. Thirdly, since the number of compared hash values equals the number of group tests, the communication burden required for transmitting the hash values will be as little as possible by applying CGT.

2.2 CGT Construction by Shifted Transversal Design (STD)

Described more mathematically, CGT is an especially efficient algorithm aiming at the problem of identifying relatively rare events from a large collection of items by applying basic yes-or-no tests. Thus, a CGT construction with n items and t groups is usually represented by a $t \times n$ binary matrix where each cell (i, j) has a 1-entry if and only if item j is contained in test i, shown in Fig. 1. Therefore, design the pattern of group testing can be transferred to the problem of constructing a testing matrix with minimized t. In our scheme, STD, a novel construction for CGT proposed in 2006, is used to provide higher efficiency than other existing non-adaptive designs.

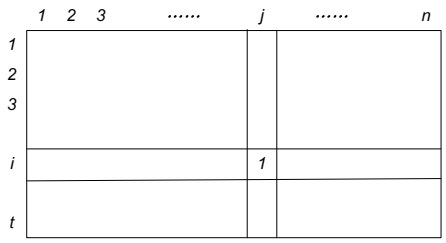

Figure 1. An illustration of a $t \times n$ matrix

The following subsections review the technical material on how to design the testing groups (that is, how to group the key bits into chains) and choose design parameters.

2.2.1 Construction algorithm of STD

STD-based construction starts with the specification of the number of total items (n) and the maximum number of expected defective items (d). These input parameters (n, d) are used to choose the design parameters of STD construction q and k, where q must be a prime number. More precisely, STD is a layered construction with k layers, each of size $q \times n$. After the design parameters are selected, the construction algorithm will produce a $t \times n$, $0-1$ matrix $M = STD(n; q; k)$, where $t(= q \times k)$ rows represent the group tests to be performed and n columns represent the items (bits) to be tested. Since each item appears only once in each layer, each of the n columns has k 1's in it and each row has usually $\lceil n/q \rceil$ 1's in it. While a detailed description and proof of the construction is available in the original STD paper,[23] we present how the algorithm works here.

Given the design parameters q and k, the matrix $M = STD(n; q; k)$ can be constructed as follows. The STD has a layered construction consisting of k layers of $q \times n$ boolean matrices. For all $j \in \{0, \cdots, k-1\}$, let M_j be a $q \times n$ boolean matrix representing layer $L(j)$, with columns $C_{j,0}, \cdots, C_{j,n-1}$.

Let the circular shift operator, σ_q, be defined as for all $(x_1, \cdots, x_q) \in \{0,1\}^q$, $\sigma_q \left[x_1\ x_2\ \cdots\ x_q\right]^T = \left[x_q\ x_1 \cdots x_{q-1}\right]^T$ and $C_{0,0} = \left[1\ 0\ \cdots\ 0\right]^T$. Note that σ_q is a cyclic function and when applied q times maps $\{0,1\}^q$ onto itself, $\sigma_q^s \left[x_1\ x_2\ \cdots\ x_q\right]^T = \left[x_1\ x_2\ \cdots\ x_q\right]^T$, $s = q$. To design a layer $L(j)$, for all $i \in \{0, \cdots, n-1\}$, construct $C_{j,i} = \sigma_q^{s(i,j)} C_{0,0}$ where if $j < q : s(i,j) = \sum_{c=0}^{\Gamma} j^c \lfloor i/q^c \rfloor$, if $j = q : s(i,j) = \lfloor i/q^\Gamma \rfloor$. The layers $L(j)$ are put together to form M by $STD(n; q; k) = \bigcup_{j=0} L(j)$.

2.2.2 Choosing design parameters of STD

The detailed procedures of mapping the experimental parameters n and d to the design parameters q and k are listed below.

(1) Choose a prime number q, with $q < n$. Start with the smallest prime, 2.

(2) Find the *compression power*, $\Gamma(q, n) = min\{\gamma | q_{\gamma+1} \geq n\}$, therefore $\Gamma = \lceil \log n / \log q \rceil - 1$. Set $k = d\Gamma + 1$.

(3) Check if this choice of q and k satisfies the *guarantee* requirements of identifying d defective elements, using the inequality, $k \leq q + 1$.

(4) If the inequality is satisfied, continue to step (5), else choose the next prime in step (1) and repeat steps (2) and (3).

(5) For each q, find its corresponding compression power Γ that satisfies $q \geq n^{1/(\Gamma+1)}$, and calculate the number of tests (t) needed by each (q, k) pair from $t = q \times k$.

(6) Choose the q and k pair to produce the desirable number of tests.

In general group testing problem, the most desirable number of tests should be the least one in numbers produced by all possible combinations of q and k, since the aim of group testing concept is to minimize the number of tests required to identify all the defective items. Nevertheless, as for key integrity problem, a special consideration on the numbers should be taken since the problem has a different setting: we need to consider minimizing the number of hash values as well as minimizing the time to compute all hash values. In the traditional application of blood testing case, a test of a group of one hundred individuals has the same (computational) costs as a group of ten individuals. But in hash value computation, the cost of computing a hash value of one hundred bits will be more than that of ten bits. In Sec. 3, two sets of design parameters of STD will be demonstrated with experimental results for comparison.

2.3 Decoding algorithm of CGT

This subsection shows how to interpret the outcomes of tests. That is, after comparing the hash value from Alice with the one from Bob for each group (chain of bits), how can we identify which bit is correct and which bit is error depending on all results of comparisons.

With the designed testing pattern, M, all the bits under test can be tagged by CGT according to the following decoding algorithm: all the bits present in at least one clean group (the test result shows that the hash value matches for the group of bits) are tagged correct; any bit present in at least one defective group (in which the hash value of the group do not match) where all other bits have been tagged correct, is tagged error. STD guarantees that the design will be able to correctly identify up to d error bits as it uses a combinatorial procedure to ensure that no two bits are mixed together more than a minimum number of times (here is 1), to prevent ambiguous decoding results. Also, the number of bits mixed in each test is roughly the same, ensuring the computation required for each chain is similar.

It is worthy to remark that if there actually exist d' error bits and $d' > d$, some bits may not be tagged at all while all the other bits can still be tagged inerrably by the above algorithm. Since the untagged bits may include both correct bits and error bits, once the decoding algorithm of STD detects such a situation (the existence of d' is discovered by the failure of tagging some bits), some further operations need to be taken, such as discarding those bits or retesting them if completeness is sought. Fortunately, with STD's well-chosen testing groups, the number of untagged bits should be a small percent of total unless the number of error bits is much larger than expected maximum d. Furthermore, it has been shown[24] recently that STD is also capable of detecting much larger number of defective items than that it guarantees, with a more sophisticated decoding algorithm. In one word, using CGT and STD, we will not mistake any error bit as correct.

3. EXPERIMENTAL RESULTS

A series of experiments have been performed to test the performance of key integrity verification using CGT approach and compare it with that using original method in terms of (1) the number of bits can be verified in a fixed-length chunk of key string which contains error bit, (2) the number of hash values needed for verification and (3) the time required to compute the hash values and decode the results of tests. We developed a software tool applying CGT and STD to generate hash information for key integrity and to verify the integrity of key bits depending on the hash information from both parties. We measure the cost of the actual communication required for transmitting the hash values for integrity verifying, the actual running time for computations to generate the values and the additional time for decoding the results of comparing the values with those from other party. The experimental computer running our software is a desktop PC configured with an Intel® Core™ 2 CPU (E6750 at 2.66GHz) and 1.97 GB RAM.

Security consideration. On one hand, the constructed matrix should be kept private only between two communicating parties. Otherwise, once the eavesdropper steal the matrix, he may have chance to get into some key information with the help of both hash values in public channel and the matrix. Simply swapping columns or rows can make the matrix different. On the other hand, it is preferred by QC to expose as little information as possible during the public-discussion phase. Therefore, in our experiment, only 16 bits of a hash value is enough to be transmitted from one party to the other so as to do the comparison, which enhance the security of the system, as well as transmission efficiency.

3.1 Experimental parameters and testing matrices for CGT

Following the procedures described in STD algorithm, we set the parameters n and d as 10^6 and 1 respectively. Choosing these parameters means that each chunk of key strings to be verified contains 10^6 bits, and the number of error bits contained in each chunk is expected at most 1, i.e., the maximum probability for a chunk which contains error bit is 10^{-6}. Note that this probability is far more overestimated than the common one, 10^{-9}, which is used typically in quantum communication systems.[25]

As described in Sec. 2.2.1, with same input experimental parameters, different sets of design parameters can be chosen for STD to accommodate to various applications. Since the performance of CGT algorithm depends on the matrices (how the bits grouped into chains), it is essential to study the effect of different parameters

and the corresponding matrices in our application. Thus, we choose two sets of STD parameters to construct matrices for some practical considerations. Using the chosen parameters, the software tool can handily construct a CGT matrix with STD. The selected design parameters and the corresponding testing matrices are:

$M_a = STD(10^6; 11; 6)\ :\ n = 10^6; q_a = 11; k_a = 6; t = 66;$

$M_b = STD(10^6; 101; 3)\ :\ n = 10^6; q_b = 101; k_b = 3; t = 303;$

Once the testing matrix of CGT is prepared, it can be applied to each chunk of key string to compute hash values. The cost of transmissions and computations can also be deduced from the matrix. The number of hash values to be computed and transmitted per n bits will be $q \times k$ for CGT with matrix $M = STD(n; q; k)$. Each of such hash values will be computed from a chain of $\left\lceil \frac{n}{q} \right\rceil$ bits and all $q \times k$ hash values will be computed from $k \times n$ bits because every bit exists once and only once in each layer.

For M_a, the constructed matrix has only 66 rows and requires the minimum transmissions of only 66 hash values for every 10^6 bits. The shortcoming of M_a is that it has 6 layers and has to generate the 66 hash values from 6×10^6 bits (each bit needs to be computed 6 times), such that the computational time will increase. On the contrary, M_b has fewer number of layers and requires much fewer computations (each bit needs to be computed only twice) and so as the computational time, while it requires more transmissions of hash values (303 hash values per 10^6 bits). Besides, since M_b is more sparse – each row in M_b contains only $9,901$ 1-entries while each row in M_a contains $90,910$, it can enhance the accuracy of identifying the correct bits when the number of error bits excesses the threshold set in advance, which will be shown in our experimental results. The problem of M_b is that for a security concern exposing more information about the quantum key on public channel may shorten the final key length. Therefore, there is a tradeoff of computation versus transmission which could be considered in specific scenarios.

3.2 Results and discussion

In our first experiment, 100 random key strings, each of which contains 10^6 bits, are prepared. For every key string, we purposely modify 1 bit in advance as an error bit. Our goal is to test whether CGT approach with our selected parameters M_a and M_b can precisely identify the error bit. Besides, the experiment also consider the computation time T_c, verifying/decoding time T_v and the number of bits of hash values to be transmitted N_h. As a comparison, results for original scheme (computing one hash) are also given. Please refer to Table 1.

Table 1. Time and transmission needed for CGT scheme with M_a, M_b and for original method

	$T_c(ms)$	$T_v(ms)$	$N_h(bits)$
CGT(M_a)	16.0	23.0	66×16
CGT(M_b)	8.3	20.0	303×16
Original method	2.5	$\times$	1×64

As we expected, CGT approach with either M_a or M_b can precisely identify the purposely modified bit in each key string and conclude that the other bits are correct. However, the original method can only give an answer No about the integrity of the key string and fail to verify the other correct bits in the string. The better thing is that in this example, we can further correct the error bit. From the efficiency of transmission point of view, using M_a in our approach yields $66 \times 16 = 1056$ hash bits to be transmitted, which is only about 0.1% of the total key bits and is better than using M_b. Concerning the efficiency of computation, using M_b requires less computational time, 8.3ms, and less verifying time, 20ms, than using M_a. Both of the two constructions are practical regarding the time consuming. And different applications should adopt different constructions to achieve better performance.

The second experiment performed is to test the robustness of CGT approach when the actual number of error bits, d', excesses the guaranteed d (here d is set to 1). 100 groups of key strings, each of which contains 100 key

strings, are prepared. Same as the first experiment, each string has 10^6 bits. The difference is that a number of randomized bits d', not limited to 1, are modified purposely as error bits. The average numbers of untagged bits including all error bits, N_f, using M_a and M_b are shown in Table 2. For example, if there are 2 bits are purposely modified as error bits, b_1 and b_2 respectively, using M_a will identify $(10^6 - 28)$ correct bits while leave 28 bits untagged which includes b_1 and b_2. However, there are 26 correct bits which are wrongly identified and will be sacrificed. With d' increasing from 1 to 10, N_f increases from 0 to 52187 (about 5% of the key) using M_a. Note that the increment of N_f with M_b is much slower than that with M_a because M_b is more sparse. Even d' is ten times of d, the number of sacrificed bits is about 0.09% of total bits.

Table 2. Robustness test for CGT scheme

d'	1	2	3	4	5	6	7	8	9	10
$n_f(M_a)$	0	28	240	1020	2762	6745	13463	23656	39056	52187
$n_f(M_b)$	0	8	25	61	113	196	303	450	623	866

4. CONCLUSIONS

The key integrity verification becomes a more and more important problem in quantum cryptography system with the increasing rate of key production. This paper relates the problem to a seemingly unrelated technique, called combinatorial group testing. We develop a scheme to verify the integrity of key bits efficiently by locating the correct bits and error bits in the key string precisely so as to avoid getting rid of all the key bits in the original method due to extremely rare error bit in key string. The experimental results show the good performance in both efficiency and robustness. Future work includes designing more efficient algorithm to construct matrix with larger n to meet future requirement, as well as optimizing the parameters of CGT.

ACKNOWLEDGMENTS

The work described in this paper was partially supported by the General Research Fund from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. RGC GRF HKU 713009E), and a HKU Seed Funding Programme for Basic Research Grant No. 200811159155.

REFERENCES

[1] Bennett, C. H., Bessette, F., Brassard, G., Salvail, L., and Smolin, J., "Experimental quantum cryptography," *Journal Of Cryptology* **5**(1), 3–28 (1992).

[2] Zbinden, H., Bechmann-Pasquinucci, H., Gisin, N., and Ribordy, G., "Quantum cryptography," *Applied Physics B-lasers And Optics* **67**, 743–748 (1998).

[3] Vernam, G., "Cipher printing telegraph systems for secret wire and radio telegraphic communications," *J. Am. Inst. Electr. Eng.* **45**, 109–115 (1926).

[4] Brassard, G. and Salvail, L., "Secret-key reconciliation by public discussion," in [*Advances in Cryptology EUROCRYPT 93*], 410–423 (1994).

[5] Elliott, C., "Quantum cryptography," *IEEE security & privacy* **2**(4), 57–61 (2004).

[6] Bennett, C. H., Brassard, G., Crepeau, C., and Maurer, U. M., "Generalized privacy amplification," *IEEE Transactions on Information Theory* **41**(6), 1915–1923 (1995).

[7] Bennet, C. H. and Brassard, G., "Quantum cryptography: Public key distribution and coin tossing," in [*IEEE International Conference on Computers, Systems and Signal Processing*], 175–179 (1984).

[8] Nakassis, A., Bienfang, J. C., and Williams, C. J., "Expeditious reconciliation for practical quantum key distribution," in [*Quantum Information and Computation II*], **5436**, 28–35, SPIE, Orlando, FL, USA (2004).

[9] Mink, A., Ma, L., Nakassis, T., Xu, H., Slattery, O., Hershman, B., and Tang, X., "A quantum network manager that supports a one-time pad stream," in [*Quantum, Nano and Micro Technologies, 2008 Second International Conference on*], 16–21 (2008).

[10] Tang, X., Ma, L., Mink, A., Nakassis, A., Xu, H., Hershman, B., Bienfang, J., Su, D., Boisvert, R. F., Clark, C., and Williams, C., "Quantum key distribution system operating at sifted-key rate over 4 mbit/s," in [*Quantum Information and Computation IV*], **6244**, 62440P–8, SPIE, Orlando (Kissimmee), FL, USA (2006).

[11] Mink, A., Tang, X., Ma, L., Nakassis, T., Hershman, B., Bienfang, J. C., Su, D., Boisvert, R., Clark, C. W., and Williams, C. J., "High speed quantum key distribution system supports one-time pad encryption of real-time video," in [*Quantum Information and Computation IV*], **6244**, 62440M–7, SPIE, Orlando (Kissimmee), FL, USA (2006).

[12] Dixon, A. R., Yuan, Z. L., Dynes, J. F., Sharpe, A. W., and Shields, A. J., "Gigahertz decoy quantum key distribution with 1 mbit/s secure key rate," *Opt. Express* **16**(23), 18790–18979 (2008).

[13] Mink, A., Bienfang, J. C., Carpenter, R., Ma, L., Hershman, B., Restelli, A., and Tang, X., "Programmable instrumentation and gigahertz signaling for single-photon quantum communication systems," *New Journal of Physics* **11**(4), 045016 (2009).

[14] Ma, L., Nam, S., Xu, H., Baek, B., Chang, T., Slattery, O., Mink, A., and Tang, X., "1310 nm differential-phase-shift qkd system using superconducting single-photon detectors," *New Journal of Physics* **11**(4), 045020 (9pp) (2009).

[15] Dorfman, R., "The detection of defective members of large populations," *Annals of Mathematical Statistics* **14**, 436–440 (1943).

[16] Sobel, M. and Groll, P. A., "Group testing to eliminate efficiently all defectives in a. binomial sample," *The Bell Systems Technical Journal* **38**, 1179–1253 (1959).

[17] Charles, J. C. and Paul, C. v. O., "Applications of combinatorial designs in computer science," *ACM Comput. Surv.* **21**(2), 223–250 (1989). 66446.

[18] Fang, J., Jiang, Z. L., Yiu, S. M., and Hui, L. C. K., "Hard disk integrity check by hashing with combinatorial group testing," in [*The 2nd International Conference on Computer Science and its Applications (CSA 2009)*], **1**, 569–574 (2009).

[19] Du, D.-Z. and Hwang, F. K., [*Combinatorial Group Testing and Its Applications*], World Scientific, Singapore, 2nd ed. (2000).

[20] Balding, D. J. and Torney, D. C., "Optimal pooling designs with error detection," *Journal of Combinatorial Theory, Series A* **74**(1), 131–140 (1996).

[21] Macula, A. J., "A simple construction of d-disjunct matrices with certain constant weights," *Discrete Math.* **162**(1-3), 311–312 (1996).

[22] Ngo, H. Q. and Du, D.-Z., "A survey on combinatorial group testing algorithms with applications to dna library screening," in [*Discrete Mathematical Problems with Medical Applications DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*], **55**, 171–182, American Mathematical Society, Providence, RI (2000).

[23] Thierry-Mieg, N., "A new pooling strategy for high-throughput screening: the shifted transversal design," *Bmc Bioinformatics* **7**, 13 (2006).

[24] Thierry-Mieg, N. and Bailly, G., "Interpool: interpreting smart-pooling results," *Bioinformatics* **24**(5), 696–703 (2008).

[25] Gisin, N., Ribordy, G., Tittel, W., and Zbinden, H., "Quantum cryptography," *Rev. Mod. Phys.* **74**, 145–195 (Mar 2002).