

SYSTEM AND METHOD FOR RANDOMNESS EXTRACTION AND METHOD OF  
PERFORMING QUANTUM CRYPTOGRAPHY

FIELD OF INVENTION

- 5    The present invention relates broadly to a system and a method for randomness extraction for quantum cryptography and to a method of performing quantum cryptography between two or more parties.

BACKGROUND

- 10   Any mention and/or discussion of prior art throughout the specification should not be considered, in any way, as an admission that this prior art is well known or forms part of common general knowledge in the field.

15   All practical implementations of quantum cryptographic protocols have finite resources, and thus, suffer from statistical fluctuations in finite-size data. Such statistical fluctuations must be taken into account to ensure that the protocol is information-theoretically secure. This introduces the requirement that the size of the data collected has to be large. However, a large data size is detrimental for hashing, which is one of the necessary steps in both quantum key distribution (QKD) and quantum random number generation (QRNG). This is because a large data size severely slows down the execution of the hashing function, which has an execution  
20   time that increases superlinearly with respect to the data size. For the above reasons, hashing is widely known to be the bottleneck of QKD and QRNG implementations.

In more detail, implemented quantum cryptographic protocols are run for a finite duration, and therefore have finite resources. Because of the statistical fluctuations in finite-size data, randomness extraction has to be performed on large input sizes for practical implementations  
25   of quantum cryptographic protocols. For example, in QKD, finite-size analysis requires a block size of at least  $10^6$  bits. When one is considering protocols with weaker assumptions, such as in the measurement-device-independent setting and the device-independent setting, the block size required is even larger, around  $10^{11}$  bits.

If one takes device-independent QKD as an example, despite improvements from both  
30   theoretical and experimental aspects, the required block size is still larger than the input sizes of state-of-the-art implementations of Toeplitz hashing on FPGAs. This is because the hashing process becomes increasingly impractical as the input size increases. This can be attributed to two scenarios, which are a decrease in throughput or an increase in resources required. As the input size of hashing increases, the output speed naturally decreases, as a larger number of bits  
35   have to be processed to produce even a single secure bit. Alternatively, to prevent the throughput of hashing from decreasing, the FPGA can create duplicates of certain hardware

and execute them in parallel. However, this leads to a large increase in resource utilization, which restricts the quantum cryptographic system to the use of high-end FPGAs.

Embodiments of the present invention seek to address at least one of the above problems.

## 5 SUMMARY

In accordance with a first aspect of the present invention, there is provided a system for randomness extraction for quantum cryptography, comprising

a sampling module configured to sample a cryptographic key to generate one or more sub-blocks;

10 an entropy module configured to determine a lower bound of the entropy in each sub-block, wherein the determining of the lower bound comprises consideration of correlations between each sub-block and unsampled bits of the cryptographic key and/or other sub-blocks; and

15 a hashing module configured to hash each sub-block for randomness extraction on each sub-block.

In accordance with a second aspect of the present invention, there is provided a method of randomness extraction for quantum cryptography, comprising the steps of:

sampling a cryptographic key to generate one or more sub-blocks;

20 determining a lower bound of the entropy in each sub-block, wherein the determining of the lower bound comprises consideration of correlations between each sub-block and unsampled bits of the cryptographic key and/or other sub-blocks; and

hashing each sub-block for randomness extraction on each sub-block.

25 In accordance with a third aspect of the present invention, there is provided method of performing quantum cryptography between two or more parties, the method comprising announcing a uniform random variable  $V_i \in [1, N_s]$  identifying which bit of the cryptographic key is sampled into which sub-block.

## BRIEF DESCRIPTION OF THE DRAWINGS

30 Embodiments of the invention will be better understood and readily apparent to one of ordinary skill in the art from the following written description, by way of example only, and in conjunction with the drawings, in which:

FIG. 1 shows a schematic drawing illustrating the sequential model for the the generalized entropy-accumulation theorem (GEAT), where an initial state  $\rho_{R_0 E_0}$  undergoes a series of GEAT channels,  $\mathcal{M}_i$ , to generate the final state  $\rho_{X^N C^N R_N E_N}$ .

5 FIG. 2 shows a plot of the key rate per signal sent against the number of sampled sub-blocks (i.e.,  $1/p_S$ ), with and without the sampled sub-block hashing method according to an example embodiment applied.

FIG. 3 shows a plot of the expected key rate per unit time of randomness extraction with and without the sampled sub-block hashing method according to an example embodiment applied.

10 FIG. 4A shows a comparison of key rate per signal sent versus the number of sampled subblocks. The dashed line represents a target key rate per signal of 0.5. "Full" refers to hashing the sifted key directly, "splitting" refers to use of the sampled subblock hashing method according to an example embodiment, and "small block" refers to our running the QKD protocol for  $(N/N_S)$  rounds  $N_S$  times and performing parameter estimation and hashing for each of the  $N_S$  blocks separately.

15 FIG. 4B shows a comparison of key rate per unit time versus the number of sampled subblocks. The dashed lines represent the optimal  $N_S$  value, and its corresponding throughput, that achieves the target key rate per signal for the splitting ( $N_S = 17$ ) method and the small-block ( $N_S = 4$ ) method. "Full" refers to hashing the sifted key directly, "splitting" refers to use of our sampled subblock hashing method, and "small block" refers to our running the QKD protocol for  $(N/N_S)$  rounds  $N_S$  times and performing parameter estimation and hashing for each of the  $N_S$  blocks separately.

FIG. 5 shows results obtained from the NIST 800-22 statistical test suite for an example embodiment.

25 FIG. 6A shows Minimum P Value results obtained from the National Institute of Standards and Technology 800-22 statistical test suite for the concatenated output of 152.56 Mbits according to an example embodiment.

FIG. 6B shows Min Proportion results obtained from the National Institute of Standards and Technology 800-22 statistical test suite for the concatenated output of 152.56 Mbits according to an example embodiment.

30 FIG. 7 shows a schematic drawing illustrating a system 700 for randomness extraction for quantum cryptography according to an example embodiment.

FIG. 8 shows a flowchart 800 illustrating a method of randomness extraction for quantum cryptography, according to an example embodiment.

## DETAILED DESCRIPTION

A generic  $N$ -round protocol, with honest parties (or a single honest party) and an adversary typically has the following steps:

- 5 • (1) *Data generation*. The honest parties exchange classical and quantum information for  $N$  rounds, which could be attacked by the adversary. At the end of each round of exchange, the honest parties receive a raw bit string  $A_i$ , with the honest parties publicly exchanging information  $I_i$ , and with the adversary holding side information  $E_i$ . It is assumed that  $A_i$  is generated independently from any memory (from previous rounds) that the honest parties possesses.
- 10 • (2) *Statistical check*. Honest parties assign each round of the data generation as either a test round ( $T_i = 1$ ) or a data-generation round ( $T_i = 0$ ), with testing probability  $\gamma$ . For test rounds, the honest parties announce information  $I_i^{\text{test}}$ , which includes  $A_i$ , and compute statistics  $C_i$ . On the basis of the observed statistics  $C^N$ , the protocol aborts if they do not fall into the set of events  $\Omega$ .
- 15 • (3) *Additional information exchange*. Additional public information can be exchanged by the honest parties, labelled  $Y$ , which is not generated in a round-by-round manner (for instance, error correction).
- (4) *Sifting*. Honest users discard rounds that are inconclusive (for instance, due to loss or no detection) and part of the test rounds, arriving at a shorter sifted bit string  $A'$ .
- 20 • (5) *Randomness extraction*. Randomness extraction is performed on the sifted bit string  $A'$  to obtain output  $K = f_{RE}(A')$ , where  $f_{RE}$  is a hash function randomly chosen from a family of two-universal hash functions.

25 At the end of the protocol, the adversary will have information  $E'_N = T^N I^N (I^{\text{test}})^N E_N$ , along with  $Y$  and the hash function  $F_{RE}$ .

After randomness extraction, one ideally wants the hashed output  $K$  to be secret from the adversary. Accordingly, bounds on the penalty due to the announcements can be provided, for instance, with use of tighter min-entropy chain rules.

30 To ensure that (1) the quantum cryptographic protocol satisfies the finite-size effect; and (2) the hashing step remains practical, embodiments of the present invention implement a sampled sub-block hashing method to improve current hashing implementations.

In more detail, in example embodiments of the present invention, the large-input data on which randomness extraction is to be performed is randomly split (via sampling) into subblocks, and a lower bound is provided on the conditional smooth min-entropy of each subblock, noting that  
35 the correlation between different sub-blocks is considered in the entropy determination step.

In one example embodiment, for single round sampling, where the input data is reduced to one sampled sub-block, and randomness extraction is performed on that sub-block accounting for the correlation between different sampled sub-blocks, i.e. in this example embodiment between the bits randomly sampled into the one sub-block and the unsampled bits. This contrasts with

discarding the unsampled bits, which would effectively introduce a large loss of entropy, as has been recognised by the present inventors.

In another example embodiment, the input data is sampled into multiple sampled sub-blocks, and randomness extraction is performed on each sub-block independently, before all the outputs are concatenated together. In this example embodiment, the method can account for the correlation between different sampled sub-blocks. Thus, the present method is operable to perform hashing on all the sub-blocks and concatenate the outputs while still preserving the overall security.

In order to demonstrate the performance of example embodiments of the present invention, a theoretical assessment is made when the hashing method according to an example embodiment is performed on a protocol satisfying certain properties and whose security can be analysed with the recent generalized entropy-accumulation theorem (GEAT), which demonstrates that the hashing method according to an example embodiment introduces only a small (linear) loss in security while providing a linear improvement in the speed of randomness extraction and maintaining the same number of secure bits obtained. It is noted that to obtain this lower bound, i.e. the entropy determination, GEAT can be replaced using other methods that are, or may become to be available for determining the entropy while accounting for the correlation between one sampled sub-block and the unsampled bits and/or between different sampled sub-blocks.

An example embodiment of the present invention makes use of the generalised entropy accumulation theorem (GEAT) [T. Metger, O. Fawzi, D. Sutter, and R. Renner, in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society, Denver, CO, USA, 2022), p. 844], which will be first briefly described below, using QKD as an example. Suppose that a QKD protocol can be analysed with the GEAT, with the raw string having some min-entropy

$$H_{min}^{\epsilon}(X^n|E_n) \geq nh - c_1\sqrt{n} - c_0 \quad (1)$$

where

$$h = \min_{v \in \Lambda} f(v)$$

$$f(v) \leq \min_{\rho \in \Sigma_i(v)} H(X_i|E_i)_{\rho}$$

$$\Sigma_i(v) = \{\rho_{C_i X_i E_i \tilde{E}_{i-1}} = \mathcal{M}_i(\sigma_{R_{i-1} E_{i-1} \tilde{E}_{i-1}}) | \rho_{C_i} = v\},$$

and where  $n$  is the total number of rounds (or signals exchanged) during the protocol,  $f$  is the min-tradeoff function present in GEAT,  $C_i$  is the test statistic,  $\sigma_{R_{i-1} E_{i-1} \tilde{E}_{i-1}}$  is the state before the GEAT channel,  $\mathcal{M}_i$  is the GEAT channel,  $X_i$  is the sifted bit string, and  $\Lambda$  is the set of accepted statistics that would not trigger an abort of the protocol.

More specifically, the GEAT is used to prove the utility of example embodiments, and a brief description of GEAT is provided here. Suppose at the end of some protocol, one has the output quantum state  $\rho_{X^N C^N R_N E_N'}$ , which can be described as being generated by

applying a sequence of channels  $\mathcal{M}_1, \dots, \mathcal{M}_N$  (termed "GEAT channels"),  $\mathcal{M}_i: R_{i-1}E_{i-1} \rightarrow X_i R_i C_i E_i$ , on some initial state  $\rho_{R_0 E_0}$  (see FIG. 1  $X^N = X_1, \dots, X_N$  typically refers to the output (e.g., raw keys in QKD),  $C^N = C_1, \dots, C_N$  refers to the statistics used to select an event (e.g., whether the protocol should be aborted),  $R_i$  is the side information carried by Alice and Bob, and  $E_i$  is Eve's side information. The GEAT provides a fairly tight bound on the conditional smooth min-entropy of the output  $X^N$  conditioned on the side information  $E_N$ , i.e.,  $H_{min}^\varepsilon(X^N|E_N)$ . The entropy is evaluated on the state  $\rho_{X^N C^N R_N E_N|\Omega}$  where  $\Omega \subset C^N$  is a set of events. This bound is extremely useful as it is directly related to the final key length and the security parameter in quantum cryptographic protocols.

The GEAT states, that if the two conditions described below are satisfied, then

$$H_{min}^\varepsilon(X^N|E_N) \geq N h - v_1 \cdot \sqrt{N} - v_0, \quad (2)$$

where

$$\begin{aligned} h &= \min_{c^N \in \Omega} f(\text{freq}(c^N)), \\ v_0 &= \frac{\beta(\varepsilon)}{v^2} 2^{\frac{1-\varepsilon}{\varepsilon} v} \ln^3(2^v + e^2), v_1 = \gamma(\varepsilon)V, \\ v &= 2 \log_2(d_X) + \text{Max}(f) - \text{Min}(f), \\ V &= \log_2(2d_X^2 + 1) + \sqrt{2 + \text{Var}(f)}, \end{aligned}$$

with variables  $\xi$ ,  $\gamma$ , and  $\beta$ , which are independent of  $f$ , and  $\text{Var}(f)$ ,  $\text{Max}(f)$  and  $\text{Min}(f)$  as the variance, maximum value, and minimum value of some affine min-trade-off function  $f(q)$ . The two conditions are:

- (1) *Nonsignaling*. One requires that any side information that Eve can have about  $X_1, \dots, X_{i-1}$  should already be contained in  $E_{i-1}$ , and not stored in  $R_{i-1}$  and revealed in later rounds (for instance, to  $E_i$ ). More formally, for each GEAT channel  $\mathcal{M}_i$ , there exists a channel  $\mathcal{R}_i: E_{i-1} \rightarrow E_i$  such that  $\text{Tr}_{X_i R_i C_i} \circ \mathcal{M}_i = \mathcal{R}_i \circ \text{Tr}_{R_{i-1}}$ , where  $\circ$  represents the composition of the channels.
- (2) *Projective reconstructability*. One should be able to reconstruct statistics  $C^N$  by performing a projective measurement on  $X^N$  and  $E_N$ , and applying a function on the outcomes. More formally, there exists a channel  $\gamma$  such that  $\mathcal{M}_N \circ \dots \circ \mathcal{M}_1 = \gamma \circ \text{Tr}_{C_N} \circ \mathcal{M}_N \circ \dots \circ \text{Tr}_{C_1} \circ \mathcal{M}_1$ .

The min-trade-off function is an affine function  $f(q)$ , where for all  $i = 1, \dots, N$ ,

$$f(q) \leq \min_{\rho \in \Sigma_i(q)} H(X_i | E_i \widetilde{E_{i-1}})_\rho, \quad (3)$$

with  $\Sigma_i(q) = \{\rho | \rho_{X_i C_i R_i E_i \widetilde{E_{i-1}}} = \mathcal{M}_i(\omega_{R_i E_{i-1} \widetilde{E_{i-1}}}), \rho_{C_i} = q\}$  and  $\widetilde{E_{i-1}}$  as a purifying system of  $R_{i-1} E_{i-1}$ .

It is noted that any protocol that cannot be analyzed with the GEAT framework may still have utility if an alternative proof theory can be applied and satisfied. This includes quantum cryptography protocols that in general cannot be reduced to a sequential process that obeys the nonsignaling condition according to the GEAT.

- 5 According to example embodiments, a sampling method is considered where each round is sampled randomly, with some probability  $p_S$ . In this scenario, it is possible to define an appropriate GEAT channel to compute the min-entropy  $H_{min}^e(X_S|E_nS)$  of sub-block  $X_S$  by announcing  $S_i$  during the protocol and including only  $X_i$  that are in the sampled sub-block  $X_S$ .

- In more detail, a sampled subblock hashing protocol is used in an example embodiment.
- 10 One begins with the raw bit string  $A^N = (A_1, \dots, A_N)$  from a cryptographic key generation module, which is understood in the art and will not be described herein. By way of example, not limitation, reference is made to description of QKD key generation modules in a recent review of protocols and implementation in [Feihu Xu, Xiongfeng Ma, Qiang Zhang, Hoi-Kwong Lo, and Jian-Wei Pan, Rev. Mod. Phys. 92, 025002 (2020)], and to
- 15 description of QRNG key generation modules to a review of protocols and implementation in [Ma, X., Yuan, X., Cao, Z. et al. Quantum random number generation. npj Quantum Inf 2, 16021 (2016)]. The raw bit string  $A^N = (A_1, \dots, A_N)$  is supposed to undergo sifting and randomness extraction,  $p_S$  be the sampling probability and  $N_S = 1/p_S$  is chosen such that there are  $N_S \in \mathbb{N}$ . The sampled sub-block hashing
- 20 method according to an example embodiment comprises:

- (1) *Sampling*. For each round  $i \in [1, N]$ , the honest parties generate a uniform random variable  $V_i$ , which can take values  $v_i \in [1, N_S]$  (i.e.,  $V_i$  can take values from 1 to  $N_S$ , each with probability  $p_S$ ). In an example embodiment, if  $V_i = j$ , it means that bit  $A_i$  is “sampled” into the  $j$ th subblock, which is denoted as  $A_{S_j}$ , where the
- 25 indices for a set  $S_j = \{i: V_i = j\}$ . The value of  $V_i$  is then announced to the rest of the honest parties (if necessary).
- (2) *Sifting*. For each subblock  $S_j$ , sifting can be performed to discard rounds that are inconclusive or part of the test rounds, leaving sifted subblocks  $A'_{S_j}$ . In general, the length of these bit strings is not fixed. Therefore, to prevent an excessively long
- 30 bit string from slowing the Toeplitz hashing, in a preferred example embodiment one can choose to abort if the size of any sifted subblocks  $A'_{S_j}$  exceeds a predetermined threshold  $L_S^{UB}$ . In practice, this aborting probability would be very small.
- (3) *Randomness extraction*. If the protocol according to an example embodiment
- 35 does not abort, randomness extraction is performed on all the subblocks independently to yield output  $K_j = f_{RE,j}(A'_{S_j})$ , where  $f_{RE,j}$  are random hash functions chosen independently from a family of two-universal hash functions. For simplicity, the length of output  $K_j$  is taken to be  $\tilde{l}$  for every  $j$ .

- (4) *Concatenation*. The final output, denoted by  $K = (K_1, \dots, K_{N_S})$ , is obtained by concatenation of all the individual outputs of randomness extraction on each subblock.

It is noted that when the sampled subblock hashing according to an example embodiment is applied,  $V_i$  is additionally announced, along with the use of multiple hash functions,  $f_{RE,1} \dots f_{RE,N_S}$ , and would be accessible to the adversary, which is considered in the security assessment of the example embodiment.

In the example embodiment, when multiple hashed strings are concatenated, the min-entropy of each sub-block to be computed is  $H_{min}^{\varepsilon}(A'_{S_i} | E_N, V^N, Y, F_{RE}^{N_S}, A'_{S_{i+1}}, \dots, A'_{S_{N_S}})$  to account for correlations between sub-blocks. With appropriate GEAT channels announcing  $X_i$  for non-sampled rounds, the min-entropy value for each sub-block can be computed. Again, it is noted that to obtain this lower bound, i.e. the entropy determination, GEAT can be replaced using other methods that are, or may become to be available for determining the entropy while accounting for the correlation between one sampled sub-block and the unsampled bits and/or between different sampled sub-blocks.

Generally, for the security analysis demonstrating the utility of an example embodiment it is assumed that for each subprotocol, i.e. for each sub-block, the associated smooth min-entropy is of the form

$$H_{min}^{\varepsilon_{sm}}(A'_{S_j} | A'_{S_{j+1}} \dots A'_{S_{N_S}}, V^N, E_N) \quad (4),$$

which accounts for correlations between the sampled raw bit strings. Since the original protocol to which the sampled sub-block hashing method according to an example embodiment is applied is GEAT-analyzable, one is able to write down GEAT channels  $M_i$  and apply the EAT by defining some min-trade-off function  $f(q)$ . To construct the min-trade-off function for the sampled sub-block hashing method according to an example embodiment applied to the original protocol, one can adapt the EAT channel to output  $A'_{S_{j,i}}$  (note  $A'_{S_{j,i}} = 1$  is inserted if the  $i$ th round is discarded or not included in  $A'_{S_j}$ ) while forwarding the information  $A'_{S_{j+1,i}} \dots A'_{S_{N_S,i}}$  and  $V_i$  to Eve. Since only a fraction  $p_s$  of the bits  $A'_i$  is output as  $A'_{S_{j,i}}$ , the entropy and thus the min-trade-off function gain a factor of  $p_s = 1/N_S$ , giving the modification in the claim.

The security condition of concern is

$$p_{\Omega} \Delta_f(\rho_{K_1^N E_N^N F_{RE}^{N_S}} |_{\Omega}, \tau_K \otimes \rho_{V^N E_N^N F_{RE}^{N_S}} |_{\Omega}) \leq \varepsilon', \quad (5)$$

where one can guarantee the bit string  $K$  is secret from the adversary, who has access to  $V^N E'_N F_{RE}^{N_S} Y$ . By introducing an intermediate state of the form  $\tau_{K_1 \dots K_j} \otimes \rho_{K_{j+1} \dots K_{N_S} V^N E'_N F_{RE}^{N_S} Y | \Omega}$ , one can apply the triangle inequality to arrive at

$$\begin{aligned} & p_{\Omega} \Delta_t(\rho_{K_1 \dots K_{N_S} V^N E'_N F_{RE}^{N_S} Y | \Omega}, \tau_K \otimes \rho_{V^N E'_N F_{RE}^{N_S} Y | \Omega}) \\ & \leq p_{\Omega} \sum_{j=1}^{N_S} \Delta_t\left(\rho_{K_j \dots K_{N_S} V^N E'_N F_{RE}^{N_S} Y | \Omega}, \tau_{K_j} \right. \\ & \quad \left. \otimes \rho_{K_1 \dots K_{j-1} V^N E'_N F_{RE}^{N_S} Y | \Omega}\right), \end{aligned} \quad (6).$$

5

Since subblock hashing according to an example embodiment computes each output string  $K_j = f_{RE,j}(A'_{S_j})$ , one can apply the quantum leftover-hash lemma [M. Tomamichel, C. Schaffner, A. Smith, and R. Renner, Leftover hashing against quantum side information, IEEE Trans. Inf. Theory 57, 5524 (2011)] [M. Tomamichel and A. Leverrier, A largely self-contained and complete security proof for quantum key distribution, Quantum 1, 14 (2017)] on each subblock. Therefore, one could set

10

$$\begin{aligned} \varepsilon' = p_{\Omega} \sum_{j=1}^{N_S} & \left\{ 2\varepsilon_{sm,j} \right. \\ & \left. + \frac{1}{2} 2^{-\frac{1}{2} [H_{\min}^{\varepsilon_{sm,j}}(A'_{S_j} | K_{j+1} \dots K_{N_S} V^N E'_N F_{RE}^{N_S} Y) - 1]} \right\} \end{aligned} \quad (7)$$

to achieve the required security level, where the min-entropy is evaluated on  $\rho_{A'_{S_j} K_{j+1} \dots K_{N_S} V^N E'_N F_{RE}^{N_S} Y | \Omega}$ , and refers to the set of hash functions used except for  $f_{RE,j}$ .

15 To determine the error, the important quantity to compute is the min-entropy. One can begin by lower-bounding the min-entropy using the data-processing inequality, since  $K_i = f_{RE,i}(A'_{S_i})$ , and  $A'_{S_i}$  can be computed from  $A_{S_i}$ ,

$$\begin{aligned}
H_{\min}^{\varepsilon_{\text{sm},j}}(A'_{S_j} | K_{j+1} \cdots K_{N_S} V^N E'_N F_{\text{RE}}^{N_S \setminus j} Y) \\
\geq H_{\min}^{\varepsilon_{\text{sm},j}}(A'_{S_j} | A'_{S_{j+1}} \cdots A'_{S_{N_S}} V^N E'_N F_{\text{RE}}^{N_S \setminus j} Y) \\
\geq H_{\min}^{\varepsilon_{\text{sm},j}}(A'_{S_j} | A_{S_{j+1}} \cdots A_{S_{N_S}} V^N E'_N Y),
\end{aligned} \tag{8}$$

where the second inequality stems from the fact that  $F_{\text{RE}}^{N_S \setminus j}$  is independent of all other terms in the min-entropy.

- 5 To analyze the min-entropy with the GEAT, one has to remove the conditioning on  $Y$ , which, by definition, cannot be expressed as being generated round by round. Depending on the nature of  $Y$ , there are many ways to remove the conditioning with the min-entropy chain rule. Here, one possible efficient method is described by way of example, if  $Y$  is generated by blocks according to an example embodiment. Suppose  $Y_j$  is generated from  
10 the information in rounds  $S_j$  only, i.e.,  $Y_j = g(A'_{S_j}, I_{S_j})$ . Then one could remove the conditioning with

$$\begin{aligned}
H_{\min}^{\varepsilon_{\text{sm},j}}(A'_{S_j} | A'_{S_{j+1}} \cdots A'_{S_{N_S}} V^N E'_N Y) \\
\geq H_{\min}^{\varepsilon_{\text{sm},j}}(A'_{S_j} | A'_{S^{\setminus j}} V^N E'_N Y) \\
\geq H_{\min}^{\varepsilon_{\text{sm},j}}(A'_{S_j} | A'_{S^{\setminus j}} V^N E'_N Y_j) \\
\geq H_{\min}^{\varepsilon_{\text{sm},j}}(A'_{S_j} | A'_{S^{\setminus j}} V^N E'_N) - \log_2 |\mathcal{Y}_j|,
\end{aligned} \tag{9}$$

- where  $S^{\setminus j} = \bigcup_{i: i \neq j} S_i$ . The second inequality makes use of the property that conditioning does not increase min-entropy, the third inequality uses the data-processing inequality  
15 on  $Y_i$  for  $i \neq j$ , since  $\tilde{O}(\tilde{O})$  and  $I_{S_i}$  (in  $E'_N$ ) are present, and the final inequality uses the chain rule. For consistency, however, the min-entropy expansion in the main text is used in the following analysis (incurs a penalty  $\log_2 |Y|$ ) but it will be appreciated that it is straightforward to generalize the result to the more-efficient case highlighted here.

- Focusing now on the min-entropy term  $H_{\min}^{\varepsilon_{\text{sm},j}}(A'_{S_j} | A'_{S_{j+1}} \cdots A'_{S_{N_S}} V^N E'_N)$ , which is now  
20 amenable to GEAT analysis, one can define the channel  $\widetilde{\mathcal{M}}_i^j: R_{i-1} E_{i-1} \rightarrow R_i E_i A'_{S_j, i} C_i$ , with  $E_i = E'_i A'_{S_{j+1}, i} \cdots A'_{S_{N_S}, i} V_i$ , and  $E'_i = E'_i A'_{S_{j+1}}^{i-1} \cdots A'_{S_{N_S}}^{i-1} V^{i-1}$

- (1) Implement EAT channel  $\mathcal{M}_i$ , which maps from  $R_{i-1} E'_{i-1}$  to  $R_i E'_i A'_i C_i$

- (2) Generate a uniformly random variable  $V_i$  (step 1 of sample and hash).
- (3) Define

$$A'_{S_{k,i}} = \begin{cases} A'_i, & V_i = k \\ \perp, & V_i \neq k \end{cases} \text{ for } k = j, \dots, N_S.$$

- 5 • (4) Announce the values of  $A'_{S_{k,i}}$  for  $k = j + 1, \dots, N_S$  and  $V_i$ , i.e., they will be known to the adversary.
- (5) Trace out  $A'_i$ .

Applying this channel yields the state  $\rho_{A'_{S_j} \dots A'_{S_{N_S}} V^N C^N R_N E'_N}$  at the end of the protocol according to an example embodiment, with  $E_N = V^N A'_{S_{j+1}} \dots A'_{S_{N_S}} E'_N$  being known to the  
 10 adversary. To show that this is a valid EAT channel, one must check for the two conditions of projective reconstructability and nonsignaling discussed above.

The statistics generated in  $\{\widetilde{\mathcal{M}}_i^j\}_i, C^N$ , is no different from that in  $\{\mathcal{M}_i\}_i$ . Moreover, since  $C_i$  is generated from  $(I^{test})^N$ , which includes  $A_i$  by assumption, the same generation method can be used in analyzing  $\widetilde{\mathcal{M}}_i^j$ . Therefore, using 7 from the projective  
 15 reconstructability of the  $\mathcal{M}_i$  EAT channel, one can create  $\mathcal{T}'$ , which acts on the expanded  $E_N = E'_N V^N A'_{S_{j+1}} \dots A'_{S_{N_S}}$  with  $\mathcal{T}' = \mathcal{T} \otimes \text{id}_{V^N A'_{S_{j+1}} \dots A'_{S_{N_S}}}$ . It is clear that  $\mathcal{T}' \circ \text{Tr}_{C_N} \widetilde{\mathcal{M}}_N^j \circ \dots \circ \text{Tr}_{C_1} \widetilde{\mathcal{M}}_1^j = \widetilde{\mathcal{M}}_N^j \circ \dots \circ \widetilde{\mathcal{M}}_1^j$ , and thus  $\{\widetilde{\mathcal{M}}_i^j\}_i$  satisfies projective reconstructability.

The proof of nonsignaling for an example embodiment relies on the assumption that the  
 20 original EAT channel  $\mathcal{M}_i$  is nonsignaling and that  $A'_i$  is generated independently of any memory, i.e.,  $R_{i-1}$  of the EAT channel. Because of this independence, one could, in general, split the EAT channel  $\mathcal{M}_i$  into two parts. The first part  $\mathcal{E}_1: E'_{i-1} \rightarrow A'_i Q_i \bar{E}_i$  is responsible for the generation of  $A'_i$ , taking only inputs  $E'_{i-1}$  since it is independent of  $A'_i$ , with modifications to form  $\bar{E}_i$  and possible intermediate systems  $Q_i$  generated. The second  
 25 part  $\mathcal{E}_2: R_{i-1} \bar{E}_i A'_i Q_i \rightarrow R_i E'_i A'_i C_i$  includes the use of  $R_{i-1}$  to arrive at the final output systems of the  $\mathcal{M}_i$  channel. Since the first part does not depend on  $R_{i-1}$ , the nonsignaling condition will depend only on the second part, i.e., that there exists  $\widetilde{R}_i: A'_i Q_i \bar{E}_i \rightarrow E'_i$  such that  $\text{Tr}_{A'_i R_i} \circ \mathcal{E}_2 \circ \mathcal{E}_1 = \widetilde{R}_i \circ \mathcal{E}_1 \circ \text{Tr}_{R_{i-1}}$ .

Consider the channel  $\widetilde{\mathcal{M}}_i^j$  described above, which can be written as a series of  
 30 channels,  $\widetilde{\mathcal{M}}_i^j = \text{Tr}_{A'_i} \circ \mathcal{E}_{A'_i \rightarrow A'_i V_i A'_{S_{j,i}} \dots A'_{S_{N_S,i}}} \circ \mathcal{M}_i$ , where  $\mathcal{E}_{A'_i \rightarrow A'_i V_i A'_{S_{j,i}} \dots A'_{S_{N_S,i}}}$  describes the generation of  $V_i$  and the sampling step. The order of the maps  $\mathcal{E}_{A'_i \rightarrow A'_i V_i A'_{S_{j,i}} \dots A'_{S_{N_S,i}}}$  and  $\mathcal{E}_2$  is

interchangeable, as neither acts to alter the systems that are inputs to the other channel. Therefore, one has

$$\begin{aligned}
& \text{Tr}_{A'_{S_j,i} R_i} \circ \tilde{\mathcal{M}}_i^j \\
&= \text{Tr}_{A'_{S_j,i} R_i} \circ \text{Tr}_{A'_i} \circ \mathcal{E}_{A'_i \rightarrow V_i A'_i A'_{S_{j+1},i} \dots A'_{S_{N_S},i}} \circ \mathcal{E}_2 \circ \mathcal{E}_1 \\
&= \text{Tr}_{A'_i R_i} \circ \mathcal{E}_{A'_i \rightarrow V_i A'_i A'_{S_{j+1},i} \dots A'_{S_{N_S},i}} \circ \mathcal{E}_2 \circ \mathcal{E}_1 \\
&= \text{Tr}_{A'_i R_i} \circ \mathcal{E}_2 \circ \mathcal{E}_{A'_i \rightarrow V_i A'_i A'_{S_{j+1},i} \dots A'_{S_{N_S},i}} \circ \mathcal{E}_1 \\
&= \tilde{R}_i' \circ \text{Tr}_{R_{i-1}} \circ \mathcal{E}_{A'_i \rightarrow V_i A'_i A'_{S_{j+1},i} \dots A'_{S_{N_S},i}} \circ \mathcal{E}_1 \\
&= \tilde{R}_i' \circ \mathcal{E}_{A'_i \rightarrow V_i A'_i A'_{S_{j+1},i} \dots A'_{S_{N_S},i}} \circ \mathcal{E}_1 \circ \text{Tr}_{R_{i-1}},
\end{aligned} \tag{10}$$

where the second equality traces out the output  $A'_{S_j,i}$ , the fourth equality applies the nonsignaling property, and the final equality swaps the order of the trace over  $R_{i-1}$  since neither  $\mathcal{E}_1$  nor  $\mathcal{E}_{A'_i \rightarrow V_i A'_i A'_{S_{j+1},i} \dots A'_{S_{N_S},i}}$  acts on  $R_{i-1}$ . Noting that  $\mathcal{R}_i = \tilde{R}_i' \circ \mathcal{E}_{A'_i \rightarrow V_i A'_i A'_{S_{j+1},i} \dots A'_{S_{N_S},i}} \circ \mathcal{E}_1$  maps  $E_{i-1}$  to  $E_i = V_i E_i' A_{S_{j+1}}^{i-1} \dots A_{S_{N_S}}^{i-1} A'_{S_{j+1},i} \dots A'_{S_{N_S},i}$ , one can see that the channel  $\tilde{\mathcal{M}}_i^j$  is nonsignaling.

With  $\tilde{\mathcal{M}}_i^j$  as a valid EAT channel, one can now consider the min-trade-off function to use before the GEAT is applied. In this case, the min-trade-off function has to satisfy

$$f'(q) \leq \min_{\rho \in \Sigma_i'(q)} H(A'_{S_j,i} | A'_{S_{j+1},i} \dots A'_{S_{N_S},i} V_i E_i' \tilde{E}_{i-1})_\rho, \tag{11}$$

with  $\Sigma_i'(q) = \left\{ \rho \left| \rho_{A'_{S_j,i} R_i E_i \tilde{E}_{i-1}} = \tilde{\mathcal{M}}_i^j(\omega_{R_{i-1} E_{i-1} \tilde{E}_{i-1}}), \rho_{C_i} = q \right. \right\}$ , where  $E_i = E_i' A'_{S_{j+1},i} \dots A'_{S_{N_S},i} V_i$ . The von Neumann entropy can be simplified by expansion in terms of different  $v_i$  values. When  $v_i \neq j$ ,  $A'_{S_j,i} = \mathbb{1}$ , which has entropy 0 since it is a fixed value. When  $v_i = j$ ,  $A'_{S_j,i} = A'_i$  and  $A'_{S_k,i} = \mathbb{1}$  for  $k > j$ . Since  $v_i$  is fixed as  $j$ , with  $A'_{S_k,i}$  for  $k > j$  containing no information on  $A'_i$ , the conditional entropy can be reduced to  $H(A'_i | E_i' \tilde{E}_{i-1})$ . Since  $A'_{S_{j+1}}^{i-1} \dots A'_{S_{N_S}}^{i-1} V_i^{i-1}$  are not used in channel  $\tilde{\mathcal{M}}_i^j$ , they do not give any information on  $A'_i$  and one can obtain  $H(A'_i | E_i' \tilde{E}_{i-1}) = H(A'_i | E_i' \tilde{E}_{i-1})$ . Moreover, since the generation of  $A'_i$  and the generation of  $E_i'$  in  $\tilde{\mathcal{M}}_i^j$  are fully determined by the channel  $\mathcal{M}_i$ , the states to be optimized over reduce to  $\Sigma_i(q)$ , thus yielding the requirement on the min-trade-off function

$$f'(q) \leq \min_{\rho \in \Sigma_f(q)} p_S H(A'_i | E'_i \tilde{E}_{i-1})_\rho, \quad (12)$$

where the factor  $p_S$  is due to the probability of  $v_i=j$ . From the GEAT-analyzable original protocol, there exists a min-trade-off function  $f(q)$  that lower-bounds the right-hand side without the factor  $p_S$ . Therefore,  $f(q)=p_S f(q)$  is a valid min-trade-off function.

- 5 Applying the GEAT using this min – trade – off function, along with EAT channels  $\widetilde{\mathcal{M}}_i^j$ , one arrives at the result in the main text. Since this result applies to all min-entropy terms in Eq. (3) one has

$$\epsilon' = p_\Omega \sum_{i=1}^{N_S} \left\{ 2\epsilon_{sm,j} + \frac{1}{2} 2^{-\frac{1}{2} [N\tilde{H}' - v'_i \sqrt{N} - v'_0 - \log_2 |\mathcal{D}| - \tilde{H}]} \right\}. \quad (13)$$

- 10 If the smoothing parameters are  $\epsilon_{sm,j} = \widetilde{\epsilon}_{sm}$  set to be *equal* for all rounds,  $\epsilon_{sm,j} = \widetilde{\epsilon}_{sm}$ , the error reduces to

$$\epsilon' = N_S p_\Omega \left[ 2\widetilde{\epsilon}_{sm} + \frac{1}{2} 2^{-\frac{1}{2} [N\tilde{H}' - v'_i \sqrt{N} - v'_0 - \log_2 |\mathcal{D}| - \tilde{H}]} \right]. \quad (14)$$

- The present invention is demonstrated by employing it on the standard BB84 QKD protocol according to an example embodiment, with  $n = 4 \times 10^9$  number of rounds, assuming a phase and bit error rate of 1%. Optimising the key and test basis probabilities over a range of values of  $p_S$  using the GEAT, FIG. 2 shows a plot of the key rate per signal sent against the number of sampled sub-blocks (i.e.,  $1/p_S$ ), with and without the sampled sub-block hashing method according to an example embodiment applied. Additionally, FIG. 3 shows a plot of the expected key rate per unit time of randomness extraction with and without the sampled sub-block hashing method according to an example embodiment applied. From FIGS. 2 and 3, it is clear that using the sampled sub-block hashing method according to an example embodiment, there is a small loss in key rate per signal sent, but the speed up that is obtained is large.

- To illustrate the effects of the sampled subblock hashing method according to another example embodiment, a BBM92 protocol is considered with  $N=10^9$  rounds and secrecy parameter  $\epsilon_{sec}=1 \times 10^{-6}$ . The optimized testing probability  $p_X$  value for direct hashing is 0.0176, which is rounded to 0.02. This method of choosing  $p_X$  corresponds to the scenario where an optimized setup has already been achieved and one would like to modify the postprocessing step to increase the throughput. From FIG. 4, it is clear that with use of the sampled sub-block hashing method according to an example embodiment, there is also a small loss in key rate per signal sent, but a linear speedup is obtained. The loss in key rate from sampling, as shown in FIG. 4A, is expected, and is a result of the penalty from sampling. Interestingly, despite the loss, there is an advantage over starting with a smaller block. This may be because the analysis of the sampled sub-block hashing method

according to an example embodiment uses the overall statistics of  $N$  rounds to estimate the min-entropy of each subprotocol, whereas the small block scenario only uses each protocol's statistics of  $N/N_s$  rounds. In addition, as the time required scales more favorably for lower input size (based on the clock cycle equation obtained), a shorter subblock would increase the generation rate despite a loss in key length, as seen in FIG. 4B.

A hardware implementation of a method according to an example embodiment is performed with simulated datasets.  $p_s = 0.05$  was chosen, because at that point, the loss in extractable key length is small, but the increase in execution speed is large. In this example embodiment, the method is implemented on a Xilinx ZCU111 evaluation board and the results are tabulated below. First, Table 1 presents the results of Toeplitz hashing on large block sizes for the example BB84 protocol, without sampling. As expected, when the input size of randomness extraction increases, the speed of the randomness extraction decreases rapidly. This is because to generate even a single bit of output, a larger number of bits needs to be processed first.

|                |          |          |         |
|----------------|----------|----------|---------|
| Input size     | 10 Mbit  | 100 Mbit | 1 Gbit  |
| Output size    | 1 Mbit   | 10 Mbit  | 10 Mbit |
| Speed (bits/s) | 4,789.43 | 386.47   | 78.51   |

Table. 1

Table 2 presents the results of Toeplitz hashing on large block sizes for the example BB84 protocol, utilizing the method according to an example embodiment. Comparing Tables 1 and 2, it is clear that despite having a larger input size, randomness extraction with the method according to an example embodiment has an execution speed that is faster by almost an order of magnitude. This shows that the method according to an example embodiment allows one to satisfy the finite-size effect without too heavy a penalty on the speed of execution.

|                     |             |             |             |             |             |
|---------------------|-------------|-------------|-------------|-------------|-------------|
| Input size          | 80.919 Mbit | 121.96 Mbit | 410.07 Mbit | 823.08 Mbit | 1.6504 Gbit |
| Single block output | 1.3214 Mbit | 2.1976 Mbit | 8.7740 Mbit | 18.578 Mbit | 38.612 Mbit |
| Speed (bits/s)      | 11,537.93   | 7,750.65    | 2,329.31    | 1,162.22    | 579.96      |

Table. 2

The results in Table 2 are for single round sampling, where the input data is reduced to 1 sampled sub-block, and randomness extraction is performed on that sub-block. In the next

setting according to another example embodiment, a concatenation approach is considered, where the input data from the BB84 example protocol is sampled into three different sub-blocks, by way of example, not limitation, and randomness extraction is performed on each sub-block independently, before all the outputs are concatenated together. The results are presented in Table 3.

| Input size          | 40.034 Mbit | 121.96 Mbit | 410.07 Mbit | 823.08 Mbit |
|---------------------|-------------|-------------|-------------|-------------|
| Concatenated output | 1.494 Mbit  | 6.588 Mbit  | 26.322 Mbit | 55.728 Mbit |
| Speed (bits/s)      | 22,982.49   | 7,804.48    | 2,332.75    | 1,163.09    |

Table. 3

Comparing the results from Tables 2 and 3, there is a further improvement in terms of speed when the method is performed with multiple sub-blocks. This is because the time taken for sampling, which only needs to be executed once, is averaged over a greater output size. For a further assessment of an implementation of an example embodiment, the concatenated output of approximately 55.728 Mbit is fed into the NIST 800-22 statistical test suite [A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, and J. Dray, NIST 800-22 A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2010).39] to test for uniformity. The results are shown in FIG. 5.

A P-value greater than 0.0001 indicates that the sequence under test passes that particular test for uniformity. From FIG. 5, the concatenated output passes all the tests in the test suite. In addition, as the security is maintained by the above-described theoretical model, it can be confidently certified that the output is uniform and secret, even from the adversary.

In another implementation, again  $p_s = 0.05$  is chosen, because at that point the loss in extractable key length is small but the increase in execution speed is large. A Xilinx ZCU111 evaluation board is used as implementation platform and the results of the BBM92 example protocol are presented in Table 4. As expected, when the input size of randomness extraction increases, the speed of the randomness extraction decreases linearly. This is because to generate even a single bit of output, a larger number of bits needs to be processed first. From comparison between direct hashing (data labelled "traditional method" in table 4) and the example embodiment, the method according to the example embodiment has an execution speed that is faster by almost 20 times. This shows that the example embodiment allows one to satisfy the finite-size effect without too great a penalty on speed. In particular, for the cases where the input size is 960.40 and 1920 Mbits, traditional Toeplitz hashing would have taken approximately 413 and 1695 h, respectively, but our method reduced the timing to 20.73 and 84.99 h, respectively, therefore greatly improving the practicality of large-input hashing.

| Input size<br>(Mbits) | Output size<br>(Mbits) | Speed (bits/s)        |                       |
|-----------------------|------------------------|-----------------------|-----------------------|
|                       |                        | Traditional<br>method | Example<br>embodiment |
| 96.040                | 6.054                  | 498.60                | 9899.80               |
| 192.08                | 13.302                 | 249.31                | 4972.47               |
| 288.12                | 20.736                 | 166.24                | 3318.63               |
| 960.40                | 74.406                 | ...                   | 997.05                |
| 1920.00               | 152.56                 | ...                   | 498.62                |

Table 4

For further assessment of an implementation according to an example embodiment , the concatenated output of 152.56 Mbits is fed into the National Institute of Standards and Technology 800-22 statistical test suite [A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, and J. Dray, NIST 800-22 A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2010).] to test for uniformity. In FIG. 6A, the P values of each individual test in the test suite are shown. A P value greater than  $10^{-4}$  indicates that the sequence under test passes that particular test for uniformity. Additionally, in FIG. 6B, the proportion result of each individual test are shown. The proportion result shows the proportion of samples from the sequence under test that passes that particular test. A proportion greater than 0.9657, illustrated by the horizontal line 500 in FIG. 5B, indicates that the sequence under test passes that particular test for uniformity. The combined results show that the concatenated output passes all the tests in the test suite. In addition, as the security is guaranteed by the theoretical model described above, it can again be confidently certified that the output is uniform and secret, even from the adversary.

Table 5 presents the resource utilisation results from the implementation of the method according to an example embodiment on the ZCU111 evaluation kit. The utilisation of look-up table (LUT), LUTRAM, flip-flop (FF), block RAM (BRAM), and digital signal-processing (DSP) blocks are low. The low utilisation percentage also indicates that the entire project can fit on platforms which are smaller and have lower cost.

| Resource | Utilisation | Utilisation (%) |
|----------|-------------|-----------------|
| LUT      | 60,950      | 14.33           |
| LUTRAM   | 6,184       | 2.90            |
| FF       | 78,685      | 9.25            |
| BRAM     | 32.5        | 3.01            |
| DSP      | 39          | 0.91            |

Table. 5

As described above, the method according to an example embodiment improves the speed of hashing while accounting for finite-size effects that impose the requirement of large block sizes. The operations required are advantageously straightforward to implement, and the overall security of the entire process is information-theoretic and preserved. Additionally, the method according to an example embodiment can also be implemented on resource-constrained platforms due to its low resource utilization.

FIG. 7 shows a schematic drawing illustrating a system 700 for randomness extraction for quantum cryptography according to an example embodiment, comprising a sampling module 702 configured to sample a cryptographic key to generate one or more sub-blocks; an entropy module 704 configured to determine a lower bound of the entropy in each sub-block, wherein the determining of the lower bound comprises consideration of correlations between each sub-block and unsampled bits of the cryptographic key and/or other sub-blocks; and a hashing module 706 configured to hash each sub-block for randomness extraction on each sub-block.

The cryptographic key may comprise either a raw key or a sifted key.

The sampling module 702 may be configured to sample the cryptographic key into the one or more sampled sub-blocks in a random manner based on a pre-determined probability.

The system 700 may comprise a sifting module 708 configured to sift the cryptographic key or to sift the one or more sampled sub-blocks.

The sampling module 702 may be configured to apply a thresholding on the length of bit strings representing the respective sub-blocks and to abort the method if the length of the bit string for any sub-block exceeds a predetermined threshold.

The system 700 may comprise a concatenation module configured to concatenate the hashed sub-blocks to generate a secret key.

The entropy module 704 may be configured to determine the lower bound based on the generalized entropy-accumulation theorem.

FIG. 8 shows a flowchart 800 illustrating a method of randomness extraction for quantum cryptography, according to an example embodiment. At step 802, a cryptographic key is sampled to generate one or more sub-blocks. At step 804, a lower bound of the entropy in each sub-block is determined, wherein the determining of the lower bound comprises consideration of correlations between each sub-block and unsampled bits of the cryptographic key and/or other sub-blocks. At step 806, each sub-block is hashed for randomness extraction on each sub-block.

The cryptographic key may comprise either a raw key or a sifted key.

The step of sampling the raw key into the one or more sub-blocks may be performed in a random manner based on a pre-determined probability.

The method may comprise sifting the cryptographic key or sifting the one or more sampled sub-blocks.

The method may comprise applying a thresholding on the length of bit strings representing the respective sub-blocks and aborting the method if the length of the bit string for any sub-block exceeds a predetermined threshold.

The method may comprise concatenating the hashed sub-blocks to generate a secret key.

The step of determining the lower bound may be based on the generalized entropy-accumulation theorem.

In one embodiment, a method of performing quantum cryptography between two or more parties is provided, the method comprising announcing a uniform random variable  $V_i \in [1, N_S]$  identifying which bit of the cryptographic key is sampled into which sub-block.

The method may use the system for randomness extraction for quantum cryptography of any example embodiment described herein.

The method may comprise using the method for randomness extraction for quantum cryptography of any example embodiment described herein.

Industrial applications of example embodiments

The method according to example embodiments can be useful in all quantum cryptographic protocols. The speed up obtained, together with accounting for finite-size security, are useful features to have in, for example, both QKD and QRNG implementations. As such, the method according to example embodiments can be utilised as a general approach for randomness extraction in both QKD and QRNG systems.

The method according to example embodiments are believed to be especially useful in the measurement-device-independent scheme, as well as the device-independent protocols. This is

because such protocols require a large block size to account for finite-size effects, as described herein.

5 In addition, chip-based implementations of QKD and QRNG systems, which are of keen interest to the industry, can also greatly benefit from the method according to example embodiments. This is because on-chip systems have tight resource-constraints, which would lead to a decrease in throughput due to the finite-size effect. The method according to example embodiments can resolve this issue, allowing finite-size effects to be addressed for on-chip systems without heavy penalties to the speed, with the addition of non-complex operations.

10 Aspects of the systems and methods described herein, such as the sampling module, the entropy module, the hashing module, and the concatenation module described herein, including in the claims, may be implemented on computing device(s), including cloud-based computing device(s) and/or Internet-of-Things computing device(s), for example as functionality programmed into any of a variety of circuitry, including programmable logic devices (PLDs), such as field programmable gate arrays (FPGAs), programmable array logic (PAL) devices, 15 electrically programmable logic and memory devices and standard cell-based devices, as well as application specific integrated circuits (ASICs). Some other possibilities for implementing aspects of the system include: microcontrollers with memory (such as electronically erasable programmable read only memory (EEPROM)), embedded microprocessors, firmware, software, etc. Furthermore, aspects of the system may be embodied in microprocessors having 20 software-based circuit emulation, discrete logic (sequential and combinatorial), custom devices, fuzzy (neural) logic, quantum devices, and hybrids of any of the above device types. Of course, the underlying device technologies may be provided in a variety of component types, e.g., metal-oxide semiconductor field-effect transistor (MOSFET) technologies like complementary metal-oxide semiconductor (CMOS), bipolar technologies like emitter-coupled logic (ECL), polymer technologies (e.g., silicon-conjugated polymer and metal-conjugated polymer-metal structures), mixed analog and digital, etc. 25

The various functions or processes disclosed herein may be described as data and/or instructions embodied in various computer-readable media, in terms of their behavioral, register transfer, logic component, transistor, layout geometries, and/or other characteristics. 30 Computer-readable media in which such formatted data and/or instructions may be embodied include, but are not limited to, non-volatile storage media in various forms (e.g., optical, magnetic or semiconductor storage media) and carrier waves that may be used to transfer such formatted data and/or instructions through wireless, optical, or wired signaling media or any combination thereof. When received into any of a variety of circuitry (e.g. a computer), such 35 data and/or instruction may be processed by a processing entity (e.g., one or more processors).

It will be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive. 40 Also, the invention includes any combination of features described for different embodiments,

including in the summary section, even if the feature or combination of features is not explicitly specified in the claims or the detailed description of the present embodiments.

In general, in the following claims, the terms used should not be construed to limit the systems and methods to the specific embodiments disclosed in the specification and the claims but  
5 should be construed to include all processing systems that operate under the claims. Accordingly, the systems and methods are not limited by the disclosure, but instead the scope of the systems and methods is to be determined entirely by the claims.

Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense as  
10 opposed to an exclusive or exhaustive sense; that is to say, in a sense of "including, but not limited to." Words using the singular or plural number also include the plural or singular number respectively. Additionally, the words "herein," "hereunder," "above," "below," and words of similar import refer to this application as a whole and not to any particular portions of this application. When the word "or" is used in reference to a list of two or more items, that  
15 word covers all of the following interpretations of the word: any of the items in the list, all of the items in the list and any combination of the items in the list.

## CLAIMS

1. A system for randomness extraction for quantum cryptography, comprising  
a sampling module configured to sample a cryptographic key to generate one or more  
5 sub-blocks;  
an entropy module configured to determine a lower bound of the entropy in each sub-block, wherein the determining of the lower bound comprises consideration of correlations between each sub-block and unsampled bits of the cryptographic key and/or other sub-blocks;  
and  
10 a hashing module configured to hash each sub-block for randomness extraction on each sub-block.
2. The system of claim 1, wherein the cryptographic key comprises either a raw key or a sifted key.
3. The system of claims 1 or 2, wherein the sampling module is configured to sample the  
15 cryptographic key into the one or more sampled sub-blocks in a random manner based on a pre-determined probability.
4. The system of any one of the preceding claims, comprising a sifting module configured to sift the cryptographic key or to sift the one or more sampled sub-blocks.
5. The system of any one of the preceding claims, wherein the sampling module is  
20 configured to apply a thresholding on the length of bit strings representing the respective sub-blocks and to abort the method if the length of the bit string for any sub-block exceeds a predetermined threshold.
6. The system of any one of the preceding claims, comprising a concatenation module configured to concatenate the hashed sub-blocks to generate a secret key.
7. The system of any one of the preceding claims, wherein the entropy module is  
25 configured to determine the lower bound based on the generalized entropy-accumulation theorem.
8. A method of randomness extraction for quantum cryptography, comprising the steps of:  
30 sampling a cryptographic key to generate one or more sub-blocks;  
determining a lower bound of the entropy in each sub-block, wherein the determining of the lower bound comprises consideration of correlations between each sub-block and unsampled bits of the cryptographic key and/or other sub-blocks; and

hashing each sub-block for randomness extraction on each sub-block.

9. The method according to claim 8, wherein the cryptographic key comprises either a raw key or a sifted key.
10. The method according to claims 8 or 9, wherein the step of sampling the raw key into the one or more sub-blocks is performed in a random manner based on a pre-determined probability.
11. The method of any one of claims 8 to 10, comprising sifting the cryptographic key or sifting the one or more sampled sub-blocks.
12. The method of any one of claims 8 to 11, comprising applying a thresholding on the length of bit strings representing the respective sub-blocks and aborting the method if the length of the bit string for any sub-block exceeds a predetermined threshold.
13. The method of any one of claims 8 to 12, comprising concatenating the hashed sub-blocks to generate a secret key.
14. The method of any one of claims 8 to 13, wherein the step of determining the lower bound is based on the generalized entropy-accumulation theorem.
15. A method of performing quantum cryptography between two or more parties, the method comprising announcing a uniform random variable  $V_t \in [1, N_S]$  identifying which bit of the cryptographic key is sampled into which sub-block.
16. The method of claim 15, using the system for randomness extraction for quantum cryptography of any one of claims 1 to 7.
17. The method of claims 15 or 16, comprising using the method for randomness extraction for quantum cryptography of any one of claims 8 to 14.