Efficient Randomness Extraction in Quantum Random Number Generators

Maurício J. Ferreira[1], **Nuno A. Silva**[1], **Nelson J. Muga**[1]

[1]Instituto de Telecomunicações, University of Aveiro,
Campus Universitário de Santiago, 3810-193 Aveiro, Portugal

{mauricioferreira, nasilva, muga}@ua.pt

Abstract. *Randomness extraction algorithms play an essential role in Quantum Random Number Generators (QRNGs), where they are used to suppress unwanted classical noise and distill true randomness from their biased output. By employing the SHA-512 hash function and Toeplitz matrix multiplication, we analyse two suitable constructions based on different principles and reach postprocessing rates of* 8.69 Mbps *and* 3.68 Mbps*, respectively. Finally, we develop a length-compatible Toeplitz-hashing algorithm able to achieve rates of* 143.29 Mbps *in a parallelized GPU implementation.*

1. Introduction

Random Numbers (RNs) are currently indispensable in many cryptographic applications, where they are used as keys for authentication and encryption protocols [Herrero-Collantes and Garcia-Escartin 2017]. The security of these systems is thus inextricably linked with the statistical quality of the RN Generator (RNG) explored given that, if these values are predictable, the entire protocol is compromised [Kelsey et al. 1998]. So far, Pseudo RNGs (PRNGs) have been widely employed to suppress this demand as they can effortlessly achieve high throughput rates [Knuth 1998]. However, such solutions are entirely deterministic, and cannot yield true randomness since their output is inherently periodic. Therefore, they become predictable to an adversary with enough computational resources [Kelsey et al. 1998] and their use is generally unsuitable for a wide range of critical security applications such Quantum Key Distribution (QKD), where access to true randomness is a key assumption [Bouda et al. 2012].

Quantum RNGs (QRNGs) address these issues by exploring the probabilistic nature of quantum phenomena to obtain information-theoretically provable randomness. Several schemes were proposed, but most modern implementations explore the quantum properties of light. Among others, sources such as measuring amplified spontaneous emission [Guo et al. 2021], phase laser noise [Huang et al. 2020], or quadrature fluctuations of a vacuum state [Ferreira et al. 2021a] have been extensively explored, with generation rates up to several dozens of Gbps being reported [Nie et al. 2015]. However, in practice, the quantum noise is always mixed with additional contributions such as electrical noise, which opens security loopholes as all classical noise can be known to an eavesdropper [Ferreira et al. 2021b]. As a result, QRNGs frequently rely on the application of Randomness Extractors (REs), which deterministically extract almost-uniformly distributed data from a biased source by sacrificing part of the output sequence [Vadhan 2012]. Similarly to PRNGs, they may also require an initial random seed. Here, however, their output remains unpredictable as long as the input of the extractor is not entirely deterministic. Nonetheless, this process is computationally demanding and frequently constitutes the main limiting factor to the QRNG throughput [Ferreira et al. 2021a].

In this paper, we comparatively analyse two common approaches for the implementation of a RE, highlighting the advantages and limitations of each method. Specifically, we focus on the implementation of the nonuniversal SHA-512 cryptographic hash function and a seeded Toeplitz-hashing extractor. Finally, to increase the extraction efficiency, we implement a parallelized length-compatible version of the Toeplitz multiplication algorithm and demonstrate its performance on a Graphics Processing Unit (GPU).

2. Randomness extraction

A (n, m, k, ϵ) randomness extractor is defined as a mathematical function that converts n bits from a (n, k)-source into m bits with a distribution ϵ-close to the uniform distribution U_m over $\{0, 1\}^m$, which is the desired output space for a standard RNG [Vadhan 2012,Ma et al. 2013]. Conversely, two probability distributions X and Y, defined in the same domain $\mathcal{X}$, are said to be ϵ-close if their statistical difference is bound by [Vadhan 2012]:

$$d(X, Y) = \max_{x \in \mathcal{X}} |P_X(x) - P_Y(x)| \leq \epsilon, \tag{1}$$

where ϵ is called the security parameter. Each of these random distributions is said to be a (n, k) source if it outputs n-bit sequences with min-entropy $H_{\text{min}} \geq k$. Here, the parameter k defines the number of uniformly distributed bits that can be extracted from the original n-bit sequence, and thus quantifies the side-information introduced by classical noise sources. Its estimation should take into account the specificity of each QRNG [Haw et al. 2015]. Nonetheless, randomness can only be extracted if the input sequence already has some extractable entropy, and thus $m \leq k$ must necessarily hold.

2.1. Non-universal hashing extractor

Deterministic REs are defined as a function Ext : $\{0, 1\}^n \to \{0, 1\}^m$, thus forgoing additional randomness sources and only requiring the biased input sequence to be distilled. Unfortunately, no general deterministic extractor exists that is valid for an arbitrary input distribution [Herrero-Collantes and Garcia-Escartin 2017]. Nonetheless, several methods have been employed, such as XORing subsets of random sequences, taking the least significant bit, applying the von Neumann de-biasing algorithm, or feeding a linear feedback shift register [Ma et al. 2013]. These methods are frequently uncritically accepted as they require few computational resources but their application may, in fact, introduce unexpected correlations [Koç 2015]. Alternatively, one-way cryptographic hash functions project their input to a set of fixed length m such that the input values can not be determined solely from the output sequence. Consequently, their output is as close to uniformly distributed as possible, minimizing the probability of two different inputs resulting in the same hash value. Nevertheless, collisions still occur, and blindly applying a hash function does not suffice since the size of the input sequence n must be chosen so that it has enough entropy to assure the randomness of the extractor output.

Here, the SHA-512 cryptographic hash function was chosen to implement a deterministic RE by hashing subsets of the input data, for which well-tested and fast implementations exist [National Institute of Standards and Technology (NIST) 2015]. In this case, to guarantee a uniformly distributed output, n must be greater than 512 bit and thus should be chosen such that [Ferreira et al. 2021a]:

$$n = \lceil 512 \frac{H_{\text{t}}}{H_{\text{q}}} \rceil, \tag{2}$$

where H_{q} is the estimated entropy due to quantum fluctuations and H_{t} the total entropy of the raw output. The implemented algorithm can process the raw output at approximately 8.69 Mbps on an Intel i7-9700K CPU. Unfortunately, non-universal hashing still relies on computational assumptions and, ultimately, does not provide information-theoretically provable RNs [Ma et al. 2013]. Even worse, potential biases of the hashing function are inherited by the output RNs, even if the input is perfectly random [Koç 2015].

2.2. Toeplitz-hashing extractor

A (n, m, d, k, ϵ) seeded extractor is defined as Ext : $\{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$. It accepts a n-bit input sequence, and a perfectly random d-bit seed to output a m-bit sequence ϵ-close to U_m. In particular, an extractor is said to be strong if concatenating the output Ext(X, U_d) with the seed U_d yields a distribution ϵ-close to U_{m+d}, and thus maintains the randomness of the initial seed [Vadhan 2012]. Consequently, a QRNG output can be subdivided into blocks that are subsequently hashed with the same seed without compromising the security of the implementation. These constructions are especially attractive for providing some information-theoretically provable randomness extractors secure against quantum adversaries such as the Trevisan extractor. Unfortunately, this specific implementation is particularly slow [Ma et al. 2013]. An alternative method employs two-universal hashing functions, which are randomly chosen from a universal hashing family $\mathcal{H} = \{h : \mathcal{S} \rightarrow \mathcal{T}\}$, with collision probability [Ma et al. 2013]:

$$P_{h \in \mathcal{H}}\{h(x) = h(y)\} \leq \frac{1}{|\mathcal{T}|}, \forall x \neq y \in S. \tag{3}$$

One such promising implementation employs a $n \times m$ Toeplitz matrix and obtains m random bits by multiplication with the raw data vector. Since a seed of length $m + n - 1$ is required, no net randomness can be extracted. Fortunately, a Toeplitz extractor constitutes a strong extractor [Haw et al. 2015], and hence the initial seed can be recycled in each subsequent application. If its good statistical properties are assured, the uniformity of the output is information-theoretically guaranteed by the leftover hash lemma, which states that given a two-universal hashing family $\mathcal{H} = \{h : \{0,1\}^n \rightarrow \{0,1\}^m\}$, and a probability distribution $X \in \{0,1\}^n$ with $H_{\text{min}}(X) \geq k$, if [Vadhan 2012]:

$$m = k - 2\log_2\left(\frac{1}{\epsilon}\right), \tag{4}$$

then for $x \in X$, $h \in H$, and $\epsilon > 0$, Ext$(x, h) := h(x)$ is a (k, ϵ) strong extractor. In other words, the statistical distance $d\Big(\big(\text{Ext}(X, U_d), U_d)\big), \big(U_d, U_m\big)\Big) \leq \epsilon$ [Vadhan 2012].

In the implementation of this algorithm, the raw bits were subdivided into sequences of 2^{12} bits ($n = 4096$) and m was chosen so that [Ferreira et al. 2021a]:

$$m = \lfloor 2^{12} p \frac{H_{\text{q}}}{H_{\text{t}}} \rfloor, \tag{5}$$

where p was arbitrarily chosen as 0.9 to account for any potential entropy overestimation. With this method, raw bits can be processed at approximately 3.68 Mbps for an input length of 32 Mbits, which, despite the security offered, can severely limit the output rate of a QRNG. Furthermore, the algorithmic complexity of the Toeplitz-vector multiplication is $O(n^2)$, and thus this rate quickly decreases for larger input lengths.

2.2.1. Length-compatible Toeplitz-hashing

Consequently, to improve the speed of this implementation, a multiplication algorithm for Toeplitz matrices that reduces the complexity to $O(nlog(n))$ was implemented [Wang et al. 2018]. It explores a fast method for multiplication of $n \times n$ circulant matrices, C_n, which can be solely characterized by their first column $\vec{a}_n$, and are diagonalized by the discrete Fourier transform matrix F_n such that:

$$C_n = F_n^{-1}\text{diag}(F_n\vec{a}_n)F_n, \tag{6}$$

where diag$(\cdot)$ represents the diagonal matrix. Consequently, its multiplication with a given vector $\vec{x}$ yields [Hayashi and Tsurumaru 2016]:

$$C_n\vec{x} = \mathcal{F}^{-1}\{\mathcal{F}(\vec{a}_n) \odot \mathcal{F}(\vec{x}_n)\} = \mathcal{F}^{-1}\{\vec{v} \odot \vec{y}\}, \tag{7}$$

where $\odot$ represents the Hadamard product and $\mathcal{F}^{-1}$ the inverse Fourier transform. An arbitrary $n \times m$ Toeplitz matrix, $T_{n\times m}$, can be embedded into a circulant matrix of size $n + m$ simply by concatenating extra elements. In fact, if $\vec{a}_n = [a_0, \cdots, a_{n-1}]$ is its first column and $\vec{b}_m = [a_0, \cdots, a_{-(m-1)}]$ is the first row, then the Toeplitz matrix can be contained in a circulant described by [Hayashi and Tsurumaru 2016]:

$$\vec{a}_{n+m} = [a_0, a_1, \cdots, a_{n-1}, a_0, a_{-(m-1)}, \cdots, a_{-1}]. \tag{8}$$

As such, it is possible to transform the Toeplitz hashing into a multiplication of a circulant matrix with a vector $\vec{r}$ containing the raw binary output by following the steps:

1. Construct $\vec{a}_{n+m}$ from the elements of the Toeplitz matrix, as described by (8).
2. Append a vector $\vec{0}$ of size m to $\vec{r}$ and compute $\vec{y} = \mathcal{F}\{[\vec{r}, \vec{0}]\}$.
3. Compute $\vec{u} = \mathcal{F}^{-1}\{\mathcal{F}(\vec{a}_{n+m}) \odot \vec{y}\}$.
4. Extract the first m entries of $\vec{u}$, which constitutes the solution for $T_{n\times m}\vec{r}$.

This length-compatible algorithm was thus implemented on a GeForce RTX 3070 GPU using *Matlab*'s GPU computing support. To avoid exhausting its memory, this multiplication problem was subdivided into smaller matrices with 4 Mbits input blocks, which were serially processed. Since the computational precision required to retrieve accurate results from the Fourier transform increases with the input length, the blocks were further subdivided into parallelized smaller batches to allow the use of single-precision calculations [Wang et al. 2018]. In Table 1, these results are compared with a non-parallelized implementation on an Intel i9-10900k Central Processing Unit (CPU) for different batch sizes. As can be seen, the algorithmic speed decreases for longer input sequences. However, batches of 2 Mbits showed better performance and were thus chosen for the final algorithm implementation, yielding an average postprocessing rate of 143.29 Mbps. We are thus able to increase the throughput by an order of magnitude when compared with the non-parallelized extractor. Further efforts to increase this postprocessing rate should be the focus of future work, for example, by employing Field-programmable Gate Arrays.

Table 1. Algorithmic speed (in Mbps) of the length-compatible Toeplitz-hashing algorithm for different input lengths and batch sizes, when comparatively implemented in an Intel i9-10900k CPU and in a GeForce RTX 3070 GPU.

Input length (Mbits)	CPU Implementation		GPU Implementation		Gain (%)	
	1 Mbit Batch	2 Mbit Batch	1 Mbit Batch	2 Mbit Batch	1 Mbit Batch	2 Mbit Batch
4	68.67	74.28	125.12	143.29	82.13	92.91
8	72.33	89.19	126.48	143.18	74.87	60.53
16	58.73	83.42	111.52	137.35	89.89	64.65
32	38.28	62.59	83.01	116.21	116.85	85.67

3. Conclusion

In conclusion, we have implemented and analysed two distinct randomness extraction methods. While deterministic extractors such as the SHA-512 cryptographic hash function offer fewer limitations to the throughput of a QRNG, their implementation is not reliable for every input distribution and fails to assure information-theoretic security. Meanwhile, the use of seeded extractors is frequently impracticable in implementations aiming for practical viability. This observation is corroborated by the implemented algorithms, given that postprocessing rates of 8.69 Mbps and 3.68 Mbps were obtained, respectively, for the SHA-512 and Toeplitz-hashing algorithms. Nonetheless, these constructs clearly benefit from high-parallelized implementations, as demonstrated by the length-compatible implementation explored, which employs circulant matrix multiplications to reach postprocessing rates of 143.29 Mbps.

This work was supported in part by Fundação para a Ciência e a Tecnologia (FCT) through national funds, by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework, under the projects QuantumPrime (PTDC/EEI-TEL/8017/2020) and UIDB/50008/2020-UIDP/ 50008/2020 (actions DigCORE and QuRunner). The authors also acknowledge support from the Programme New Talents in Quantum Technologies of the Gulbenkian Foundation (Portugal).

References

Bouda, J., Pivoluska, M., Plesch, M., and Wilmott, C. (2012). Weak randomness seriously limits the security of quantum key distribution. *Phys. Rev. A*, 86:062308.

Ferreira, M. J., Silva, N. A., Pinto, A. N., and Muga, N. J. (2021a). Characterization of a quantum random number generator based on vacuum fluctuations. *Applied Sciences*, 11(16).

Ferreira, M. J., Silva, N. A., Pinto, A. N., and Muga, N. J. (2021b). Homodyne noise characterization in quantum random number generators. In *2021 Telecoms Conference (ConfTELE)*, pages 1–6, Leiria, Portugal.

Guo, Y., Cai, Q., Li, P., Jia, Z., Xu, B., Zhang, Q., Zhang, Y., Zhang, R., Gao, Z., Shore, K. A., and Wang, Y. (2021). 40 gb/s quantum random number generation based on optically sampled amplified spontaneous emission. *APL Photonics*, 6(6):066105.

Haw, J. Y., Assad, S. M., Lance, A. M., Ng, N. H. Y., Sharma, V., Lam, P. K., and Symul, T. (2015). Maximization of extractable randomness in a quantum random-number generator. *Phys. Rev. Applied*, 3:054004.

Hayashi, M. and Tsurumaru, T. (2016). More efficient privacy amplification with less random seeds via dual universal hash function. *IEEE Transactions on Information Theory*, 62(4):2213–2232.

Herrero-Collantes, M. and Garcia-Escartin, J. C. (2017). Quantum random number generators. *Rev. Mod. Phys.*, 89:015004.

Huang, M., Chen, Z., Zhang, Y., and Guo, H. (2020). A phase fluctuation based practical quantum random number generator scheme with delay-free structure. *Applied Sciences*, 10(7).

Kelsey, J., Schneier, B., Wagner, D., and Hall, C. (1998). Cryptanalytic attacks on pseudorandom number generators. In Vaudenay, S., editor, *Fast Software Encryption*, pages 168–188, Berlin, Heidelberg. Springer Berlin Heidelberg.

Knuth, D. E. (1998). *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, 3 edition.

Koç, Ç. (2015). *Open Problems in Mathematics and Computational Science*. Springer International Publishing.

Ma, X., Xu, F., Xu, H., Tan, X., Qi, B., and Lo, H.-K. (2013). Postprocessing for quantum random-number generators: Entropy evaluation and randomness extraction. *Phys. Rev. A*, 87:062327.

National Institute of Standards and Technology (NIST) (2015). Secure Hash Standard (SHS) (FIPS PUB 180-4). *Federal Information Processing Standards Publication*, 180-4(August):36.

Nie, Y.-Q., Huang, L., Liu, Y., Payne, F., Zhang, J., and Pan, J.-W. (2015). The generation of 68 Gbps quantum random number by measuring laser phase fluctuations. *Review of Scientific Instruments*, 86(6):063105.

Vadhan, S. P. (2012). Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336.

Wang, X., Zhang, Y., Yu, S., and Guo, H. (2018). High-speed implementation of length-compatible privacy amplification in continuous-variable quantum key distribution. *IEEE Photonics Journal*, PP:1–1.