Cheng et al. *EPJ Quantum Technology* (2025) 12:137
https://doi.org/10.1140/epjqt/s40507-025-00433-3

EPJ Quantum Technology
a SpringerOpen Journal

RESEARCH Open Access

An efficient large-scale privacy amplification scheme exceeding 10G bits for quantum key distribution

Xi Cheng[1], Haokun Mao[1], Hongwei Xu[1] and Qiong Li[1*]

*Correspondence:
qiongli@hit.edu.cn
[1]Faculty of Computing, Harbin Institute of Technology, Xidazhi Street, Harbin, 150006, China

Abstract

Privacy Amplification (PA) is indispensable in Quantum Key Distribution (QKD), ensuring security against eavesdropping by eliminating information leakage. For Discrete-Variable QKD (DV-QKD) protocols, a large input block size exceeding 10^8 bits is preferred to achieve the secure key rate approaching the asymptotic limit. However, in state-of-the-art quantum key distribution systems operating at multi-GHz pulse rates, PA becomes a critical bottleneck due to the conflicting requirements of large input block sizes and high throughput. To address this challenge, we propose a novel PA algorithm utilizing a newly constructed universal hash family DM3H and prove its cryptographic security rigorously. Based on the PA algorithm, we design and implement an efficient PA scheme which is capable of processing input block sizes up to 10^{10} bits while achieving high throughput performance. For an input block size of 10^{10} bits, the implementation on the i9-14900 platform demonstrates a throughput of 112 Mbps with a retention ratio of 0.33. This breakthrough significantly enhances the secure key rate and maximum transmission distance of DV-QKD systems.

Keywords: Quantum key distribution; Privacy amplification; Central processing unit

1 Introduction

Quantum Key Distribution (QKD) is a secure communication technology based on quantum mechanics that enables two legitimate parties, named Alice and Bob, to generate a shared key that is information-theoretically secure [1]. In recent years, the performance and practicality of QKD systems have been significantly improved [2–9], with transmission distances exceeding 1000 kilometers [4] and the highest secure key rate exceeding 115 Mbps [7]. QKD is increasingly demonstrating its pivotal role in safeguarding communications.

Practical QKD systems consist of two main components: the quantum communication subsystem and the post-processing subsystem [10]. The postprocessing subsystem ensures the correctness and security of the shared keys between Alice and Bob, primarily through two critical steps: Information Reconciliation (IR) and Privacy Amplification (PA). As the final step of the QKD system, PA plays an indispensable role since it reduces the possi-

ble information leaked to Eve to a negligible level, thereby transforming partially secure reconciled keys into information-theoretically secure keys [11, 12].

For the DV-QKD system, PA is required to support a large input block size and a high retention ratio [7]. The input block size of the PA scheme is critical to the security of the QKD system due to the finite-size effect. An input block size exceeding 10^8 bits is considered necessary to efficiently mitigate the finite-size effect to obtain a high secure key rate [13–15]. Meanwhile, the maximum retention ratio determines the proportion of secure key bits that can be extracted from each bit of sifted key, which directly influences the overall key rate. To achieve a high-rate DV-QKD system while meeting the dual requirements of large input block sizes and high retention ratios, the PA module also demands high throughput, which directly determines the final secure key rate. However, existing PA schemes remain insufficient to simultaneously support large input block sizes, high retention ratios, and high throughputs required for practical applications.

Numerous PA schemes [7, 13, 16–23] for DV-QKD have been proposed on CPU platforms, among which several PA implementations with input block sizes exceeding 10^8 bits have been developed. In 2018, Takahashi et al. accelerated the multiplication of the Toeplitz matrix using the Number Theoretic Transform (NTT) algorithm, achieving a throughput of 108.77 Mbps at an input block size of 10^8 bits [18]. B. Y. Tang et al. used a split-and-merge strategy for the Toeplitz matrix, implementing block-wise matrix multiplication acceleration through the Fast Fourier Transform (FFT), which enabled their PA scheme to handle the input block size of 10^{10} bits. However, their throughput dropped below 1 Mbps under such large input sizes [19]. B. Z. Yan et al. accelerated modular arithmetic hashing using the Schönhage-Strassen algorithm [24], achieving a throughput of 140.96 Mbps at an input block size of 10^8 bits [20]. Their subsequent MMH-MH PA algorithm further extended the input block size to a maximum 2×10^8 bits while maintaining 140 Mbps throughput [21]. Based on this work, W. Li et al. achieved an average throughput of 308.8 Mbps at an input block size of 10^8 bits, reporting a QKD system capable of delivering secret keys at rates exceeding 115 Mbps [7]. In 2025, B. Q. Yan et al. utilized the square-modular-arithmetic hash function to achieve 571.63 Mbps at an input block size approaching 10^9 bits [25]. However, this scheme attained a retention ratio of only 0.09, making it more suitable for CV-QKD systems than DV-QKD systems.

For DV-QKD PA schemes, since a high retention ratio is necessary, achieving large input block sizes and high throughput simultaneously remains a critical challenge. Existing PA schemes rarely achieve input block sizes exceeding 10^9 bits, and even when input sizes reach 10^8 bits, maintaining high throughput becomes substantially constrained. The processing load increases prohibitively with large input block sizes, making it difficult to improve PA throughput in high-rate QKD systems. To our knowledge, no existing PA scheme has been able to simultaneously achieve all three requirements: large input block size of 10^9 bits, high retention ratio exceeding 0.3, and high throughput of a hundred-megabit.

In this paper, we propose a highly efficient large-scale PA scheme capable of processing input block sizes up to 10^{10} bits while maintaining a throughput exceeding 100 Mbps. To support the large input sizes and high retention ratio, we introduce a new hash family - Dynamic Multi-stage Multilinear-Modular Hashing (DM3H) family based on Multilinear-Modular Hashing (MMH) and prove its universal properties. Utilizing this hash family, we design an optimized PA algorithm and implement it on the CPU platform. Experimental results demonstrate that this PA scheme can support large input block sizes up to 10^{10} bits

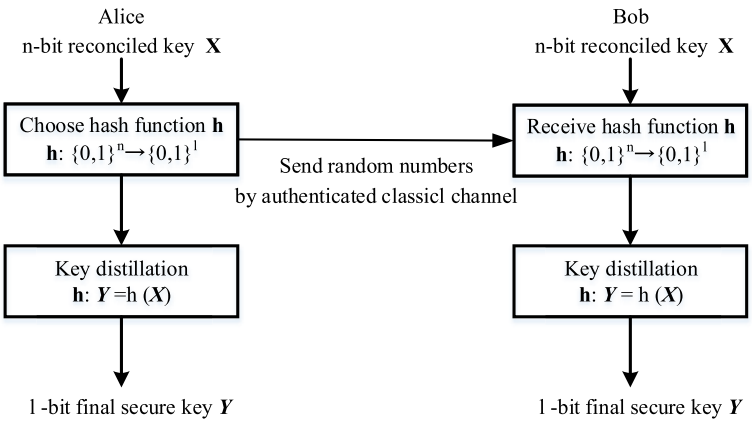

Figure 1 The process of privacy amplification

while maintaining high throughput. Implementation on the i9-14900 platform achieves a throughput of 112 Mbps at input block sizes of 10^{10} bits with a retention ratio of 0.33. To our knowledge, our work is the first CPU-based PA implementation capable of processing 10^{10} bit input block sizes while achieving hundred-megabit throughput.

The rest of this paper is organized as follows. Section 2 reviews the foundational concepts of PA for DV-QKD, including the PA process and finite-size analysis. In Sect. 3, we propose a new universal hash family and prove its security in PA. Based on this hash family, a novel hybrid PA algorithm is designed. The implementation and the experiment results are presented in Sect. 4. In Sect. 5, we give a comprehensive conclusion of this paper.

2 Preliminary

As the final step in the QKD system, PA plays a vital role because it enables Alice and Bob to distill an information-theoretically secure key from a partially secure reconciled key. In this section, we first introduce the typical PA process based on universal hash families [26], and analyze the advantages and disadvantages of PA algorithms based on different universal hash families. Then, we give the finite-size analysis for the decoy-state QKD system, demonstrating the importance of the large input block size of PA.

2.1 Privacy amplification by universal hash families

After the sifting and information reconciliation stages, Alice and Bob share a n-bit consistent but partially secure reconciled key X. A typical process of PA can be represented as Fig. 1. Using an authenticated public classic channel, Alice randomly selects a hash function h from the universal hash family $\mathcal{H}$, then transmits the random selection to Bob. Alice and Bob subsequently apply this hash function h to the reconciled key X, yielding the final secure key $Y = h(X)$. A universal hash family ensures that the keys distilled by the PA process are highly secure, such that any eavesdropper (Eve) has only a negligible probability of successfully guessing the secure key.

A universal hash family is formally defined as follows:

Definition 1 A universal hash family $\mathcal{H}$ is a set of functions from X to Y satisfying the condition: For any distinct inputs $x_1, x_2 \in X, x_1 \neq x_2$, $|h \in \mathcal{H} : h(x_1) = h(x_2)| \leq \frac{|\mathcal{H}|}{|Y|}$.

Currently, the two most commonly used universal hash families in PA are Toeplitz hashing and modular arithmetic hashing. Toeplitz hashing PA and modular arithmetic hashing PA present distinct advantages and limitations in their application to large-scale QKD systems.

Toeplitz matrix hashing [27] is the most widely used universal hash family in PA. An $l \times n$ binary Toeplitz matrix can be defined as follows, it can transform n-bit input key $\mathbf{X}$ into l-bit secure key $\mathbf{Y}$:

$$\begin{aligned}\mathbf{Y} &= \mathbf{T}_{l\times n}\cdot\mathbf{X}\\ &= \begin{pmatrix} t_l & t_{l+1} & \cdots & t_{n+l-2} & t_{n+l-1}\\ t_{l-1} & t_l & \cdots & t_{n+l-3} & t_{n+l-2}\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ t_2 & t_3 & \cdots & t_n & t_{n+1}\\ t_1 & t_2 & \cdots & t_{n-1} & t_n \end{pmatrix}\cdot\begin{pmatrix} x_1\\ x_2\\ \vdots\\ x_{n-1}\\ x_n \end{pmatrix}\end{aligned} \tag{1}$$

The elements in the Toeplitz matrix on each diagonal line are constant, therefore, the matrix can be represented by $(n + l - 1)$-bit random numbers. Toeplitz hashing PA offers the advantage of flexibly splitting and processing large input block sizes. However, in large-scale applications, the computation of binary Toeplitz matrix multiplication remains inefficient despite optimizations like FFT and NTT.

Modular arithmetic hashing [26] is another important universal hash family used in PA. Its mathematical formulation is formally defined as follows:

Definition 2 Let α and β be two positive integers with $\beta < \alpha$. Define the modular-arithmetic hash (MH) family from Z_{2^α} to Z_{2^β}:

$$\mathrm{MH} \coloneqq \{h_{c,d} : \mathbb{Z}_{2^\alpha} \to \mathbb{Z}_{2^\beta} \mid c, d \in \mathbb{Z}_{2^\alpha},\ \gcd(c, 2) = 1\} \tag{2}$$

For each choice of parameters $c, d \in \mathbb{Z}_{2^\alpha}$, the functions $h_{c,d}$ are defined by:

$$h_{c,d}(x) \coloneqq \left\lfloor \frac{(cx + d) \bmod 2^\alpha}{2^{\alpha-\beta}} \right\rfloor \tag{3}$$

Modular arithmetic hashing PA improves multiplication efficiency by processing the input key as a single large integer rather than a binary sequence. However, hardware constraints the scalability of large integer multiplication, thus limiting the maximum input block size for PA. Besides, the functions in the modular arithmetic hash family require $(2n - 1)$-bit random numbers to represent.

2.2 Finite-size analysis for privacy amplification

In practical QKD systems, Alice and Bob cannot exchange arbitrarily large key blocks for final key computation due to physical constraints. In the finite-size regime, parameter estimation must be conducted over finite-length samples, therefore, statistical fluctuations must be taken into consideration. Consequently, this imposes a tighter security bound for realistic QKD systems, thus larger input block sizes are required to mitigate the impact of finite-size effect. In the following, we investigate the finite-size effects in the PA process of decoy-state QKD protocols.

Follow Ref. [15], the secure key rate for the 1-decoy QKD protocol can be calculated by:

$$l \leq s^l_{Z,0} + s^l_{Z,1}(1 - h(\phi^u_Z)) - \lambda_{EC} - 6\log_2\left(\frac{19}{\epsilon_{sec}}\right) - \log_2\left(\frac{2}{\epsilon_{cor}}\right) \tag{4}$$

Where $s^l_{Z,0}$ and $s^l_{Z,1}$ are lower bounds on the vacuum state and single-photon contributions in sifted keys; ϕ^u_Z is the upper bound on the phase error rate; $h(.)$ denotes the binary entropy; λ_{EC} is the average information leaked during the error correction process; ϵ_{sec} and ϵ_{cor} are the secrecy and correctness parameters, respectively.

Notably, three parameters $(s^l_{Z,0}, s^l_{Z,1}, \phi^u_Z)$ in the above equation are sensitive to the finite-size effect. This limitation motivates our investigation of input block size optimization strategies. As demonstrated in [15], increasing the input block size N not only improves the final secure key rate but also extends the maximum transmission distance.

3 Large-scale and high-speed privacy amplification algorithm

In this section, we present a new universal hash family for PA, called the Dynamic Multi-stage Multilinear-Modular Hashing family (DM3H). Then we prove its universality, thereby establishing its theoretical security for PA in QKD systems. Using the DM3H family, we design a large-scale and high-speed PA algorithm and present its main process.

3.1 Dynamic multi-stage multilinear-modular hashing

Previous PA schemes typically employ Toeplitz matrix hashing or modular arithmetic hashing for distillation. However, each of these schemes suffers from inherent limitations when processing input blocks exceeding 10^8 bits. Although Toeplitz matrix hashing enables parallel processing by splitting the input key into sub-blocks, it struggles to achieve high throughput. Modular arithmetic hashing achieves higher throughput, but is constrained in processing input key sequences larger than 10^8 bits due to limitations in handling large integer arithmetic.

Combining the advantages of the Toeplitz matrix and modular arithmetic hashing, the MMH was first applied to PA algorithms in [21]. The definition of the MMH family [28] is given as follows.

Definition 3 Let p be a prime and let k be an integer $k > 0$. Define a family MMH of functions from $\mathbb{Z}_p^k$ to $\mathbb{Z}_p$:

$$\text{MMH} \coloneqq \{h_A : \mathbb{Z}_p^k \to \mathbb{Z}_p \mid \mathbf{X} \in \mathbb{Z}_p^k, \mathbf{A} \in \mathbb{Z}_p^k\} \tag{5}$$

For any input key sequence $\mathbf{X} = \langle x_1, \ldots, x_k \rangle \in \mathbb{Z}_p^k$ and random numbers $\mathbf{A} = \langle a_1, \ldots, a_k \rangle \in \mathbb{Z}_p^k$, the functions h_A are defined by:

$$h_A(X) \coloneqq \sum_{i=1}^{k} a_i \cdot x_i \bmod p \tag{6}$$

This construction operates within the finite field $\mathbb{Z}_p$. The family of hash functions comprises all multilinear functions defined over $\mathbb{Z}_p^k$, where k is an integer.

However, a significant limitation of the MMH family in PA is the constrained output size. Specifically, the maximum retention ratio achievable by MMH is only $1/k$, which is insufficient for DV-QKD systems.

According to the definition of MMH, the MMH family is a set of functions mapping from $\mathbb{Z}_p^k$ to $\mathbb{Z}_p$, where both p and k can be freely chosen. However, in practical implementations of PA, the conversion from binary sequences to the finite field $\mathbb{Z}_p$ constrains p to be a Mersenne prime $p = 2^\gamma - 1$. This enables the input sequence of PA, i.e., a binary sequence of length $\gamma \times k$, to be efficiently represented as an element of $\mathbb{Z}_p^k$.

The use of Mersenne primes simplifies the implementation of PA but simultaneously imposes a limitation on the retention ratio of MMH. When expressed in vector form, MMH can be represented as:

$$\begin{aligned}\mathbf{Y}_{1\times 1} &= \mathbf{A}_{1\times k}^T \cdot \mathbf{X}_{k\times 1} \\ &= \begin{pmatrix} a_1 & a_2 & \cdots & a_k \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}\end{aligned} \tag{7}$$

From the Equation (7), the output secure key sequence can be expressed as $\mathbf{Y} \in \mathbb{Z}_p$, where the maximum output length l equals the exponent γ of the corresponding Mersenne prime. So far, the largest known Mersenne prime is $2^{136,279,841} - 1$, so the theoretical maximum output length of MMH is 136,279,841 bits, approximately 1.4×10^8. However, when the input block size reaches 10^9 bits, the retention ratio is less than 0.14, and when the input block size increases to 10^{10} bits, the retention ratio further decreases to below 0.014. These results demonstrate that for large input block sizes exceeding 10^9 bits, the retention ratio of MMH cannot meet the requirements of DV-QKD systems, which are typically around 0.3.

To address the retention ratio limitations of the MMH family, an optimized universal hash family DM3H is proposed. The new hash family is constructed by concatenating multiple MMH functions. Furthermore, we use the construction method proposed in [29] to further reduce the length of the random numbers.

The definition of DM3H is as follows:

Definition 4 Let p be a prime and let k and m be an integer, $k \geq m > 0$. For any $\mathbf{X} = \langle x_1, \ldots, x_k \rangle \in \mathbb{Z}_p^k$, and $\mathbf{A} = \langle a_1, \ldots, a_{k-1} \rangle \in \mathbb{Z}_p^{k-1}$, define the DM3H family $\mathcal{H}$ as follows:

$$\mathcal{H} \coloneqq \{h_A : \mathbb{Z}_p^k \to \mathbb{Z}_p^m \mid X \in \mathbb{Z}_p^k, A \in \mathbb{Z}_p^{k-1}\} \tag{8}$$

The result of the hash function $h_A(X)$ is the concatenation of the results of functions $f_1(X)$ to $f_m(X)$. In this formula, '$\|$' represents the concatenation symbol.

$$h_A(\mathbf{X}) \coloneqq f_1(\mathbf{X}) \| f_2(\mathbf{X}) \| \ldots \| f_m(\mathbf{X}) \tag{9}$$

In the equation above, for any input key sequence $\mathbf{X} = \langle x_1, \ldots, x_k \rangle \in \mathbb{Z}_p^k$ and random numbers $\mathbf{A} = \langle a_1, \ldots, a_{k-1} \rangle \in \mathbb{Z}_p^{k-1}$, the MMH functions $f_1, \ldots, f_m$ are defined as follows:

$$f_i(X) \coloneqq \left(\sum_{j=m}^{k} a_{j-i+1} \cdot x_j + x_i\right) \mod p \tag{10}$$

More specifically, we can represent the DM3H function h_A as a $m \times k$ matrix as follows:

$$\begin{aligned}\mathbf{Y}_{m\times 1} &= \mathbf{T}_{m\times k} \cdot \mathbf{X}_{k\times 1} \\ &= \begin{pmatrix} 1 & & & a_m & a_{m+1} & \cdots & a_{k-1} \\ & 1 & & a_{m-1} & a_m & \cdots & a_{k-2} \\ & & \ddots & \vdots & \vdots & \ddots & \vdots \\ & & 1 & a_1 & a_2 & \cdots & a_{k-m} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}\end{aligned} \tag{11}$$

Our construction retains the core advantages of the MMH framework while expanding the maximum retention ratio to m/k, significantly improving its applicability to input block sizes larger than 10^8 bits.

In contrast to MMH, which maps from $\mathbb{Z}_p^k$ to $\mathbb{Z}_p$, the DM3H functions are defined as mappings from $\mathbb{Z}_p^k$ to $\mathbb{Z}_p^m$, as shown in Equation (11). For a chosen Mersenne prime $p = 2^\gamma - 1$, the parameters k and m are determined based on the input block size n and the output length l.

For example, if $\gamma = 136{,}279{,}841$ is chosen, each sub-block size is approximately 1.4×10^8 bits. In a QKD system with an input block size of 10^9 bits, assuming the output length l is calculated to be 2.5×10^8 bits according to the secure bound, we can obtain $k = \lceil n/\gamma \rceil = 8$ and $m = \lceil l/\gamma \rceil = 2$. In contrast, for an MMH-based PA algorithm, the maximum output length is limited to γ bits, yielding a maximum retention ratio of $\gamma/n \approx 0.14$, as shown in Equation (7). This constraint results in a substantial loss of potential secure key.

Compared with MMH, DM3H imposes no limitation on the maximum output length. Therefore, by adjusting the parameters k and m, it can support adaptive retention ratios ranging from 0 to 1. This design provides a significant improvement in flexibility compared to MMH.

To incorporate the DM3H family into the PA algorithm, it is necessary to ensure the security of the PA algorithm. We prove that DM3H is a universal hash family as follows:

Theorem 1 *The DM3H family $\mathcal{H}$ is a universal hash family.*

Proof Consider the situation of hash collision, for $\mathbf{X_1}, \mathbf{X_2} \in \mathbb{Z}_p^n, \mathbf{X_1} \neq \mathbf{X_2}, h \in \mathcal{H}$ the following result holds:

$$h(\mathbf{X_1}) = h(\mathbf{X_2}) = \mathbf{Y}, \quad \mathbf{Y} \in \mathbb{Z}_p^m \tag{12}$$

The necessary condition for the above equation to hold is $h(\mathbf{X_1}) = h(\mathbf{X_2})$. Let $\mathbf{X_1} - \mathbf{X_2}$ be denoted as $\mathbf{Z}$, the above formula can be rewritten as: $h(\mathbf{X_1} - \mathbf{X_2}) = h(\mathbf{Z}) = 0$. Since $\mathbf{X_1}$ is not equal to $\mathbf{X_2}$, we have $\mathbf{Z} \neq \mathbf{0}$.

For fixed $\mathbf{Z} \in \mathbb{Z}_p^k$, we can rewrite the formula as a system of linear equations.

$$\begin{pmatrix} a_m & a_{m+1} & \cdots & a_{k-1} \\ a_{m-1} & a_m & \cdots & a_{k-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_{k-m} \end{pmatrix} \cdot \begin{pmatrix} x_{m+1} \\ x_{m+2} \\ \vdots \\ x_k \end{pmatrix} + \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \mathbf{0} \tag{13}$$

Let matrix $\mathbf{A}$ denote the coefficient matrix in Equation above, we define the subvectors $\mathbf{X_A} = \langle x_{m+1}, \ldots, x_k \rangle$ and $\mathbf{X_I} = \langle x_1, \ldots, x_m \rangle$. Given $\mathbf{Z} \neq 0$, the vectors $\mathbf{X_A}$ and $\mathbf{X_I}$ cannot simultaneously vanish. Consequently, when $\mathbf{X_A} = 0$, it necessarily follows that $h(\mathbf{Z}) \neq 0$. Therefore, the analysis can be restricted to the case where $\mathbf{X_A} \neq 0$.

In order to obtain the number of hash functions $h \in \mathcal{H}$ that satisfy the condition above, we can rewrite the linear equations as follows:

$$\begin{pmatrix} 0 & 0 & \cdots & & x_{m+1} & x_{m+2} & \cdots & x_{k-1} & x_k \\ 0 & 0 & \cdots & x_{m+1} & x_{m+2} & \cdots & \cdots & x_k & 0 \\ \vdots & \vdots & \iddots & \iddots & \iddots & \iddots & \iddots & \vdots & \vdots \\ 0 & x_{m+1} & x_{m+2} & \cdots & x_{k-1} & x_k & \cdots & 0 & 0 \\ x_{m+1} & x_{m+2} & x_{m+3} & \cdots & x_k & 0 & \cdots & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{k-2} \\ a_{k-1} \end{pmatrix} = - \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{14}$$

Since $\mathbf{X_A} \neq 0$, the rank of the coefficient matrix of this system of linear equations is m, which restricts m variables, leaving $k-m-1$ variables free. In the finite field, each variable can take p distinct values, the total number of solutions to this system of equations is determined by the number of possible combinations of values for the free variables, which is p^{k-m-1}. So we have the following result:

$$|\{h_A \,:\, h(X_1) = h(X_2) \,|X_1 \neq X_2\}| \leq p^{k-m-1} = \frac{|\mathcal{H}|}{\left|\mathbb{Z}_p^m\right|} \tag{15}$$

According to the definition of the universal hash family, the DM3H is a universal hash family. □

3.2 Optimized privacy amplification algorithm

Based on the DM3H function, we propose an optimized PA algorithm that supports large input block sizes and high retention ratios while sustaining high throughput. By dynamically splitting the input key sequence into multiple sub-blocks, the optimized PA algorithm can flexibly adjust sub-block sizes and quantities to accommodate different input block lengths and retention ratio requirements of the QKD system.

Our PA algorithm represents each sub-block $x \in \mathbb{Z}_p$ as a large integer. This design enables accelerated large-integer multiplications through the Schönhage-Strassen algorithm, achieving enhanced multiplication performance. In order to map the input key sequence $\mathbf{X} \in \mathbb{Z}_2^\gamma$ to a prime field $\mathbb{Z}_p$, we specifically select a Mersenne prime $p = 2^\gamma - 1$ as the large prime modulus. This mechanism ensures a one-to-one correspondence between γ-bit binary sequences and elements of $\mathbb{Z}_p$, eliminating the need for additional encoding or modular arithmetic and thus enabling efficient field mapping. Notably, input sequences equal to $2^\gamma - 1$ are discarded during key generation. Given the extremely large value of γ, the probability of encountering such sequences becomes negligible.

The high retention ratio primarily results from the structural characteristics of DM3H, as illustrated in Equation (11). The main parameters of the algorithm include the Mersenne prime $p = 2^\gamma - 1$, the number of input sub-blocks k, and the number of output sub-blocks m. Once γ is determined, k and m can be obtained from the input block

size n and the required output length l. In MMH-based PA algorithms, when γ is fixed, the maximum achievable output length l is limited to γ, leading to a rapid decrease in the retention ratio γ/n as n increases. In contrast, the DM3H-based PA algorithm overcomes this limitation, with its output length not bounded by γ. This allows adaptive configuration of the transformation matrix according to the input block size n and the output length l, thereby enabling an adaptive retention ratio ranging from 0 to 1.

In addition to improving the retention ratio, the proposed PA algorithm also achieves high throughput. The high throughput mainly arises from two factors. First, the algorithm significantly reduces computational complexity. This reduction arises from the optimized DM3H structure and fewer multiplication operations. Taking the binary Toeplitz matrix $\mathbf{T}_{l\times n}$ shown in Equation (1) as a baseline, and assuming the same sub-block size γ, it can be divided into $m \times k$ sub-blocks, resulting in $m \times k$ matrix multiplications. In contrast, the DM3H structure in Equation (11) needs only $m \times (k-m)$ large integer multiplications. Hence, as the retention ratio increases, the computational savings become more significant.

Furthermore, each sub-block operation in the DM3H-based PA algorithm exhibits lower complexity than in Toeplitz PA. Toeplitz sub-block operations involve $\gamma \times \gamma$ matrix-vector multiplications, accelerated via convolution and NTT. In contrast, the DM3H-based PA algorithm computes the product of two large integers represented by γ bits binary sequences. These sequences can be expressed in base-2^B form and divided into $\lceil \gamma/B \rceil$ binary segments of length B for NTT acceleration. Using the number of butterfly units, denoted as U, to evaluate the computational complexity, we can obtain:

$$\frac{U_{\text{Toeplitz}}}{U_{\text{DM3H}}} = \frac{C_{\text{Toeplitz}}\ \gamma \log \gamma}{C_{\text{DM3H}}\ \lceil \gamma/B \rceil \log \lceil \gamma/B \rceil} \approx \frac{C_{\text{Toeplitz}}\ B}{C_{\text{DM3H}}} \tag{16}$$

Here, C_{Toeplitz} and C_{DM3H} are comparable constants corresponding to NTT-based convolution and polynomial multiplication, respectively. Since B typically ranges from 16 to 64, the approximation implies that large-integer multiplication is substantially more efficient than Toeplitz convolution. This finding is also noted in [21]. Overall, the DM3H-based PA algorithm requires substantially less computation than the Toeplitz-based PA algorithm.

Second, the proposed PA algorithm is highly parallelizable. As shown in Equation (11), the computations corresponding to different rows of $\mathbf{T}_{m\times k}$ are independent, allowing concurrent multi-threaded execution. Moreover, the multiplications within each row are also independent, enabling further parallelization. The high degree of computational parallelism ensures that the algorithm achieves high throughput even under large input block sizes and high retention ratios.

Based on the above analysis, we can roughly derive the throughput trend of the proposed PA algorithm. Following the previous notation, let the input block size be n, the retention ratio be $R = l/n$, where l denotes the output length. Denoting the processor's computational speed as v, the processing time t for a single round can be expressed as follows:

$$t = \frac{\text{total work}}{v} = \frac{(k-m)m \times c\lceil \gamma/B \rceil \log \lceil \gamma/B \rceil}{v} \tag{17}$$

Subsequently, we can obtain the specific throughput. Assuming the input key sequence is not zero-padded, the throughput is given by:

$$\begin{aligned}\text{Throughput}_{\text{PA}} &= \frac{\text{Input block size}}{\text{Time}} = \frac{n}{t} \\ &= \frac{nv}{(k-m)m \times c\lceil \gamma/B \rceil \log \lceil \gamma/B \rceil} \\ &= \frac{nv}{k^2(1-R)R \times c\lceil \gamma/B \rceil \log \lceil \gamma/B \rceil} \\ &\approx \frac{B\gamma v}{n(1-R)R \times c \log \lceil \gamma/B \rceil} \\ &= \frac{B}{c(1-R)R} \times \frac{\gamma v}{n \log \lceil \gamma/B \rceil}\end{aligned} \tag{18}$$

From the equation above, we can analyze the throughput trend. When the constants c, B, and the retention ratio R are fixed, the following behavior is observed: For an input block size $n < 10^8$ bits, the sub-block size γ generally maintains a linear relationship with the growth of n. In this range, m typically remains 1, resulting in low parallelism. The processing speed v increases slowly with the number of sub-blocks k. Consequently, the throughput fluctuates with the growth of γ but remains generally stable.

When the output length $l = n \times R \geq \gamma_{\max}$, the parameter γ can no longer increase. As a result, the throughput gradually decreases with increasing n. However, since $m \geq 1$ in this regime, the increased parallelism leads to a growth in the processor speed v, this mitigates the decline in throughput. When both the input block size n and the number of matrix rows m continue to increase, the parallel processing capacity reaches its limit, which is $v = v_{\max}$, the throughput becomes inversely proportional to the input block size n.

Our proposed PA algorithm consists of four main steps:

Step 1: Choose parameter γ. As established in our prior analysis, $p = 2^\gamma - 1$ represents a Mersenne prime. For typical decoy-state QKD systems operating with input block sizes exceeding 10^8 bits, the suitable γ values are constrained to the following set:

$$\begin{aligned}\gamma \in \{&25{,}964{,}951, \quad 30{,}402{,}457, \quad 32{,}582{,}657, \quad 37{,}156{,}667, \\ &42{,}643{,}801, \quad 43{,}112{,}609, \quad 57{,}885{,}161, \quad 74{,}207{,}281, \\ &77{,}232{,}917, \quad 82{,}589{,}933, \quad 136{,}279{,}841\}\end{aligned} \tag{19}$$

When the output length l does not exceed the largest currently known Mersenne prime (136,279,841), we select the minimal $\gamma > l$ from this set. For output lengths greater than 136,279,841, we employ $\gamma = 136{,}279{,}841$.

Step 2: Padding and split input key. To align with the constructions of the DM3H, the input key sequence $\mathbf{X}$ (with length denoted as n) must be extended to a length that is a multiple of γ. This extended input sequence, denoted $\mathbf{X}'$, is then mapped to the finite field $\mathbb{Z}_p$.

Step 3: Construct DM3H function. We can construct the DM3H function using the input random number $\mathbf{A}$. Analogous to Step 2, random numbers of length $(k-1) \times \gamma$ are mapped to $k-1$ large integers to form the matrix $\mathbf{T}_{m \times k}$.

Algorithm 1 Proposed PA Algorithm

Input:

Input key sequence: $X \in \{0,1\}^n$.

Random Seeds: $A \in \mathbb{Z}_p^{k-1}$

Output:

Output key sequence $Y \in \{0,1\}^l$

if $l \leq 136{,}279{,}841$ **then**
    $\gamma = \min\{\gamma | \gamma \in \text{Mersenne prime} \wedge \gamma > l\}$
else
    $\gamma = 136{,}279{,}841$
end if
$X' = (X, \text{padding } 0)$, $X' \in \{0,1\}^{\gamma \times k}$// Padding
$X' = \langle x_1, \ldots, x_k \rangle$ // Split data X
$A = \langle a_1, \ldots, a_{k-1} \rangle$ // Split data A
if $x_i = 2^\gamma - 1$ for any $i = 1, \ldots, k$ **then**
    Reload data x_i
end if
$X_I = \langle x_1, \ldots, x_m \rangle$
for $i = 1$ to m **do**
    for $j = m$ to k **do**
        $y_i = y_i + x_j \times a_{j-i+1}$
    end for
end for
$Y_A = \langle y_1, \ldots, y_m \rangle$
return $Y = X_I \oplus Y_A$

Step 4: Distill the secure key. The final secure key $\mathbf{Y}$ is obtained through $\mathbf{Y} = \mathbf{T}_{m\times k} \cdot \mathbf{X}'$. The large number of γ enables efficient computation where only $(k-m) \times m$ large-integer multiplications are required, significantly reducing computational complexity.

The more mathematically formal expression of the proposed PA algorithm is illustrated in Algorithm 1.

4 Implementation and results

To evaluate the performance of our proposed PA scheme, we implement it on two typical CPU architectures: an Intel i7-10700 platform to assess algorithmic efficiency, and an Intel i9-14900 platform to evaluate high-throughput capabilities. The i7-10700 platform is equipped with 8 physical cores operating at 2.90 GHz and 16 GB of memory, while the i9-14900 platform provides 24 physical cores at 3.20 GHz with 64 GB of memory, offering substantially greater parallelized processing capacity.

The Intel i7-10700 platform is selected as the baseline to ensure consistency with the MMH-based PA scheme reported in [7], which was implemented on the same CPU platform. Under these identical hardware conditions, the proposed DM3H-based PA scheme can accommodate a larger input block size, reflecting its scalability within the same computational environment. The Intel i9-14900 platform, equipped with enhanced computational and memory resources, is utilized to further investigate the scalability of the proposed scheme in high-parallelism scenarios. Leveraging its increased number of physical

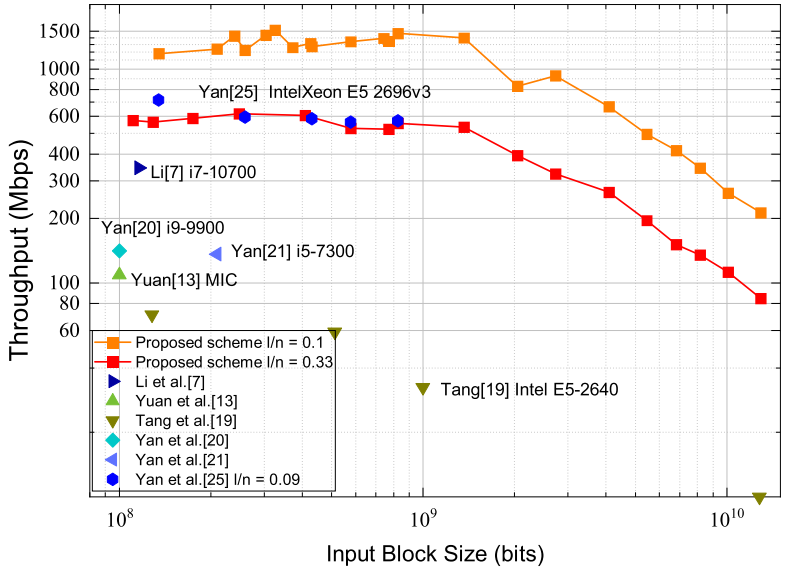

Figure 2 Comparison of input block size and throughput exceeding 10^8 bits with retention ratios of 0.1 and 0.33 on the i9-14900 CPU platform

cores and larger memory capacity, we evaluate the achievable throughput improvement enabled by parallel processing. This analysis also provides a basis for future extensions of our work on server-grade CPU and GPU platforms.

To evaluate the performance of our proposed scheme, we adopt a retention ratio of 0.33, which aligns with the parameters of practical high-rate DV-QKD systems [7, 13]. For a comprehensive comparison with existing PA schemes, we also report results with a retention ratio of 0.1.

Subsequent evaluations focus on two critical aspects: input block size, which is directly related to the secure key rate and transmission distance of QKD systems, and throughput, defined as the processing speed of reconciled keys. We compare throughput performance with all documented CPU-based PA schemes at input block sizes exceeding 10^8 bits.

We present the implementation results of our PA scheme on the Intel i9-14900 platform in Fig. 2. As illustrated in the figure, the maximum input block size supported by most existing PA schemes remains below 10^9 bits, indicating a scalability limitation. Our approach extends the input block size to 10^{10} bits while maintaining a hundred-megabit throughput. In particular, at an input block size of 10^{10} bits, we achieve a throughput of 112 Mbps. Compared to the prior large-scale PA scheme [19], which attains 0.44 Mbps under the input block size of 1.28×10^{10} bits, our approach demonstrates a significant throughput advantage. The experimental results show a great improvement in throughput through optimized construction of DM3H and parallelized computation strategies.

For the Intel i7-10700 platform, we achieve a throughput of 44.2 Mbps with an input block size of 10^{10} bits, as shown in Fig. 3. This performance still meets the requirements of realistic QKD systems in terms of both input block size and throughput.

In conclusion, the experimental results show the high efficiency of our PA scheme in large-scale scenarios. The experimental results on different CPU platforms demonstrate that our proposed PA scheme can extend the input block sizes to 10^{10} bits while maintaining hundred-megabit throughputs.

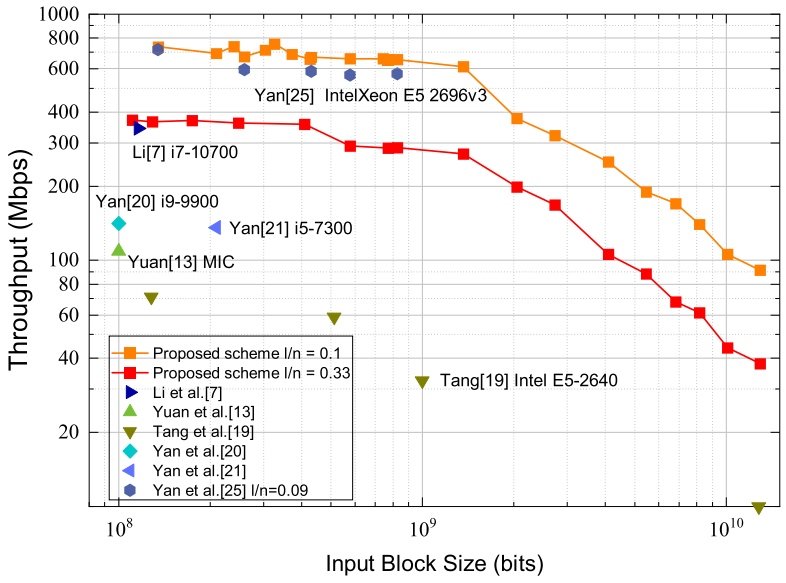

Figure 3 Comparison of input block size and throughput exceeding 10^8 bits with retention ratios of 0.1 and 0.33 on the i7-10700 CPU platform

However, the proposed scheme also exhibits certain limitations, which highlight a fundamental design trade-off in PA algorithms. Achieving a large input block size, high throughput, and low resource cost simultaneously appears to present a design trilemma. Our approach prioritizes the former two objectives, which inevitably necessitates greater computational and memory resource consumption. This trade-off is also validated in our experiments.

For example, as illustrated by the performance curve with a retention ratio of 0.33 in Fig. 2, the throughput remains relatively stable and the resource utilization is contained when the input block size is below 4×10^8 bits. As the input block size increases exceeding 4×10^9 bits, the throughput becomes approximately inversely proportional to the input block size, indicating that the parallelization capability has reached its limit and that both processor and memory usage are approaching their maximum capacity. This observation is consistent with the theoretical trend estimated in Eq. (18).

5 Conclusion

This work significantly advances PA in DV-QKD systems through several key contributions. First, we introduce a newly constructed universal hash family DM3H, and prove its security for PA process. Second, using this hash family, we propose a large-scale and high-efficient PA scheme capable of handling input blocks as large as 10^{10} bits for DV-QKD systems. Through the optimized construction of DM3H and parallelized computation strategies, our approach not only supports large-scale input block sizes, but also maintains high throughput. Experimental results on Intel i9-14900 platforms demonstrate its high performance under ultra-long ($\geq 10^8$ bits) input block sizes. Specifically, our scheme achieves throughput of 112 Mbps at input block sizes of 10^{10} bits with a retention ratio of 0.33. To our knowledge, this is the first CPU-based PA scheme to reach hundred-megabit throughput under input block sizes of 10^{10} bits. This work enables secret key rates ap-

proaching asymptotic limits and extends transmission distances, demonstrating a significant advance in high-rate DV-QKD systems.

Acknowledgements
This work is supported by National Natural Science Foundation of China (Grant No. 62471161), National Cryptography Science Fund (Grant No. 2025NCSF02051), Innovation Program for Quantum Science and Technology (Grant No. 2021ZD0300701).

Author contributions
Xi Cheng came up with the original idea and drafted the manuscript. Haokun Mao participated in the proof process and improved the manuscript. Hongwei Xu and Qiong Li discussed and refined the proof process. Additionally, Qiong Li guided the overall direction of the manuscript and revised it. All authors reviewed and approved the final manuscript.

Funding information
This work is supported by National Natural Science Foundation of China (Grant No. 62471161), National Cryptography Science Fund (Grant No. 2025NCSF02051), Innovation Program for Quantum Science and Technology (Grant No. 2021ZD0300701).

Data availability
No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 22 July 2025 Accepted: 16 October 2025 Published online: 01 December 2025

References

1. Bennett CH, Brassard G. Quantum cryptography: public key distribution and coin tossing. In: Proceedings of IEEE international conference on computers, systems and signal processing. IEEE; 1984. p. 175–9.
2. Liao S-K, Cai W-Q, Liu W-Y, Zhang L, Li Y, Ren J-G, Yin J, Shen Q, Cao Y, Li Z-P, et al. Satellite-to-ground quantum key distribution. Nature. 2017;549(7670):43–7.
3. Wang S, Yin Z-Q, He D-Y, Chen W, Wang R-Q, Ye P, Zhou Y, Fan-Yuan G-J, Wang F-X, Chen W, et al. Twin-field quantum key distribution over 830-km fibre. Nat Photonics. 2022;16(2):154–61.
4. Liu Y, Zhang W-J, Jiang C, Chen J-P, Zhang C, Pan W-X, Ma D, Dong H, Xiong J-M, Zhang C-J, Li H, Wang R-C, Wu J, Chen T-Y, You L, Wang X-B, Zhang Q, Pan J-W. Experimental twin-field quantum key distribution over 1000 km fiber distance. Phys Rev Lett. 2023;130:210801. https://doi.org/10.1103/PhysRevLett.130.210801.
5. Grünenfelder F, Boaron A, Resta GV, Perrenoud M, Rusca D, Barreiro C, Houlmann R, Sax R, Stasi L, El-Khoury S, et al. Fast single-photon detectors and real-time key distillation enable high secret-key-rate quantum key distribution systems. Nat Photonics. 2023;17(5):422–6.
6. Zhou L, Lin J, Jing Y, Yuan Z. Twin-field quantum key distribution without optical frequency dissemination. Nat Commun. 2023;14(1):928.
7. Li W, Zhang L, Tan H, Lu Y, Liao SK, Huang J, Li H, Wang Z, Mao HK, Yan B, Li Q, Liu Y, Zhang Q, Peng CZ, You L, Xu F, Pan JW. High-rate quantum key distribution exceeding 110 Mb s–1. Nat Photonics. 2023;17(5):416–21.
8. Li Y, Cai W-Q, Ren J-G, Wang C-Z, Yang M, Zhang L, Wu H-Y, Chang L, Wu J-C, Jin B, et al. Microsatellite-based real-time quantum key distribution. Nature. 2025. 1–8.
9. Pittaluga M, Lo YS, Brzosko A, Woodward RI, Scalcon D, Winnel MS, Roger T, Dynes JF, Owen KA, Juárez S, et al. Long-distance coherent quantum communications in deployed telecom networks. Nature. 2025;640(8060):911–7.
10. Fung C-HF, Ma X, Chau HF. Practical issues in quantum-key-distribution postprocessing. Phys Rev A. 2010;81:012318. https://doi.org/10.1103/PhysRevA.81.012318.
11. Bennett CH, Brassard G, Robert J-M. Privacy amplification by public discussion. SIAM J Comput. 1988;17(2):210–29. https://doi.org/10.1137/0217014.
12. Bennett CH, Brassard G, Crepeau C, Maurer UM. Generalized privacy amplification. IEEE Trans Inf Theory. 1995;41(6):1915–23. https://doi.org/10.1109/18.476316.
13. Yuan Z, Plews A, Takahashi R, Doi K, Tam W, Sharpe AW, Dixon AR, Lavelle E, Dynes JF, Murakami A, Kujiraoka M, Lucamarini M, Tanizawa Y, Sato H, Shields AJ. 10-mb/s quantum key distribution. J Lightwave Technol. 2018;36(16):3427–33. https://doi.org/10.1109/JLT.2018.2843136.
14. Lucamarini M, Patel KA, Dynes JF, Fröhlich B, Sharpe AW, Dixon AR, Yuan ZL, Penty RV, Shields AJ. Efficient decoy-state quantum key distribution with quantified security. Opt Express. 2013;21(21):24550–65. https://doi.org/10.1364/OE.21.024550.
15. Rusca D, Boaron A, Grünenfelder F, Martin A, Zbinden H. Finite-key analysis for the 1-decoy state QKD protocol. Appl Phys Lett. 2018;112(17):1–8. https://doi.org/10.1063/1.5023340. arXiv:1801.03443.

16. Zhang C-M, Li M, Huang J-Z, Li H-W, Li F-Y, Wang C, Yin Z-Q, Chen W, Han Z-F, Treeviriyanupab P, Sripimanwat K. Fast implementation of length-adaptive privacy amplification in quantum key distribution. Chin Phys B. 2014;23(9):090310. https://doi.org/10.1088/1674-1056/23/9/090310.
17. Liu B, Zhao B, Yu W, Wu C. FiT-PA: fixed scale FFT based privacy amplification algorithm for quantum key distribution. J Internet Technol. 2016;17(2):309–20. https://doi.org/10.6138/JIT.2016.17.2.20150703e.
18. Takahashi R, Tanizawa Y, Dixon AR. High-speed implementation of privacy amplification in quantum key distribution. In: 6th int. conf. quantum cryptography. 2016.
19. Tang B-Y, Liu B, Zhai Y-P, Wu C-Q, Yu W-R. High-speed and large-scale privacy amplification scheme for quantum key distribution. Sci Rep. 2019;9(1):15733.
20. Yan B, Li Q, Mao H, Xue X. High-speed privacy amplification scheme using GMP in quantum key distribution. IEEE Photonics J. 2020;12(3):1–13. https://doi.org/10.1109/JPHOT.2020.2987611. arXiv:1910.04429.
21. Yan B, Li Q, Mao H, Chen N. An efficient hybrid hash based privacy amplification algorithm for quantum key distribution. Quantum Inf Process. 2022;21(4):130. https://doi.org/10.1007/s11128-022-03462-4. arXiv:2105.13678.
22. Bai E, Jiang X-q, Wu Y. Memory-saving and high-speed privacy amplification algorithm using lfsr-based hash function for key generation. Electronics. 2022;11(3):377.
23. Lu Y, Bai E, Jiang X-q, Wu Y. High-speed privacy amplification algorithm using cellular automate in quantum key distribution. Electronics. 2022;11(15):2426.
24. Schönhage A, Strassen V. Fast multiplication of large numbers. Computing. 1971;7:281–92.
25. Yan B-Q, Wu J-B, Fan F, Xu B-J, Zhang X-L, Gu M. Quantum proof and high-speed privacy amplification for quantum key distribution. Phys Rev Appl. 2025;23:054004. https://doi.org/10.1103/PhysRevApplied.23.054004.
26. Carter JL, Wegman MN. Universal classes of hash functions. J Comput Syst Sci. 1979;18(2):143–54. https://doi.org/10.1016/0022-0000(79)90044-8.
27. Krawczyk H. Lfsr-based hashing and authentication. In: Desmedt YG, editor. Advances in cryptology — CRYPTO '94. Berlin: Springer; 1994. p. 129–39.
28. Halevi S, Krawczyk H. Mmh: software message authentication in the gbit/second rates. In: Biham E, editor. Fast software encryption. Berlin: Springer; 1997. p. 172–89.
29. Hayashi M, Tsurumaru T. More efficient privacy amplification with less random seeds via dual universal hash function. IEEE Trans Inf Theory. 2016;62(4):2213–32. https://doi.org/10.1109/TIT.2016.2526018. arXiv:1311.5322.

Publisher's note