Quantum Key Distribution Post-processing: A Heterogeneous Computing Perspective

He Li, Adrian Wonfor, Amanda Weerasinghe, Muataz Alhussein, Yupeng Gong and Richard Penty
Department of Engineering, University of Cambridge, Cambridge, UK

***Abstract*—Recent experimental demonstrations of quantum key distribution (QKD) have caught worldwide attention for their ability to transmit encryption keys. Current commercial QKD typically uses discrete optical components, resulting in systems which are typically sized to fit within telecommunications racks. Widespread deployment of QKD systems will require low-SWaP (*i.e.*, Size, Weight, and Power) QKD systems, thus Photonic Integration is seen as a key enabler of widespread adoption. QKD post-processing systems play a significant role in the distillation and high-speed generation of secure keys for discrete variable (DV) or continuous variable (CV) QKD protocols. Interest in using heterogeneous computing techniques for QKD post-processing is growing, owing to the emergence of domain-specific hardware accelerators for efficient quantum information processing applications. We hence discuss in a tutorial manner the principles and techniques of QKD post-processing. Characteristics of representative heterogeneous platforms for QKD post-processing implementations are compared, along with a review of the state-of-the-art QKD post-processing accelerators.**

Keywords—QKD, Post-processing, reconciliation, privacy amplification, hardware acceleration

I. Introduction

Quantum key distribution (QKD) meets the desire for cryptography that is more resistant to attack by tomorrow's quantum computers. A large-scale quantum computer will be able to tackle the mathematical complexity challenges that underpin public key cryptography, which is commonly used today for secure communications. QKD uses the principle that the quantum state of a single photon cannot be cloned, *i.e.* if an eavesdropper intercepts our signal, she will change the quantum state of the photon. QKD enables this change of state to be detected and thus action to be taken to keep our communication secure [1].

However, the cost, size, weight and power consumption of QKD equipment must be decreased to allow quantum cryptography to be competitive in future networks and a real alternative to the quantum safe algorithms being proposed today, which rely on computational complexity for their security. This is especially true when it comes to expanding QKD and quantum random number generators (QRNG) into new areas like the last-mile link to the consumer or the Internet of Things (IoT). The development of chip-based solutions is essential for allowing mass market applications, which are critical for realizing a quantum-ready economy [2]. For example, various companies have established solutions for shrinking the optical circuits used in QKD and QRNG down to the size of a single semiconductor chip. Not only are they smaller and lighter than their discrete component equivalents, but they also use less power. Most importantly, multiple modules can be fabricated on the same semiconductor wafer using industry-standard procedures, allowing them to be mass-produced in considerably higher quantities [3].

QKD systems consist of two main stages: quantum signal generation, transmission and detection through a quantum channel and post-processing through a classical channel. Development of QKD systems is difficult, owing to the combination of quantum physics, quantum information theory and computer engineering required to produce a successful QKD system. In particular, specialist skills are required to integrate different hardware components into a homogenous computing system. To address this issue, in this paper, we provide a tutorial for the design of full-stack QKD postprocessing systems, in a step-by-step procedure able to produce final keys. Three main steps in post-processing are discussed: 1) basis sifting for discrete variable QKD (DVQKD) protocols, or bit conversion from continuous variables to binary bit strings for continuous variable QKD (CVQKD) protocols, 2) error correction and 3) privacy amplification. To achieve high-speed QKD systems, efficient post-processing accelerations have been investigated by employing algorithmic and architectural optimisations on different hardware platforms including graphics processing units (GPUs) and field-programable gate arrays (FPGAs) [4]–[6]. We review the state-of-the-art post-processing designs and discuss their design features which will be beneficial for potential users.

The main contributions made in this paper are as follows:

- We give an overview of the physical processes of QKD systems and the subsequent processing required to produce secure keys.
- We then present a full-stack tutorial for efficient implementations of QKD post-processing, with principles and step-by-step procedures.
- We summarise the characteristics of CPU, GPU and FPGA hardware platforms for QKD post-processing implementations.
- We review the state-of-the-art QKD post-processing implementations with a focus on recent architectural optimisations and hardware acceleration.

We will now describe the operational principles of QKD systems, which will then enable us to describe their optimisations, both in terms of integration and computation in more details.

II. Quantum Key Distribution Principles

QKD technology can be divided into two main families: DVQKD and CVQKD. The main distinctions between DVQKD and CVQKD protocols lie in the optical transmission and detection techniques used. DVQKD takes advantage no-cloning properties of single photons, while CVQKD employs coherent states of light and their excess noise upon detection [1].

A. Discrete-variable QKD

A DVQKD system performs the data communication by encoding the discrete bit values into the polarisation or phase of single photons. To decode the discrete data, a single photon detector is needed at the Bob side. The *de facto* DVQKD protocol is called BB84 proposed by Bennett and Brassard in 1984 [7], where a binary bit is encoded into photon polarisations using two non-orthogonal basis pairs (usually horizontal/vertical and diagonal/anti-diagonal, each pair represents 0 and 1). Alice and Bob each apply a sequence of randomly assigned bases at the transmit and receive modules. Only after Bob detects photons, does she tell Alice when they arrived and what basis she used to detect them, she does not disclose their values. Alice then tells Bob which of these detections correspond to a transmission event where they both used the same basis pair. If an eavesdropper (Eve) listened to this quantum channel she would have to randomly choose an basis pair herself for eavesdropping. This random choice of detection and retransmission by Eve, would result in an increased Quantum Bit Error rate (QBER) as her random choice of basis will result in a 50% QBER for each instance where she chooses wrongly. High QBERs in QKD systems are always attributed to eavesdropping attacks. Steps are taken to reduce the information which Eve can recover, by hashing the received error corrected signal to a smaller size secure key, effectively reducing Eve's recovered information to an arbitrarily small level.

As a key hardware device in DVQKD systems, a stable and accurate single photon source is required for security proof assumptions. However, high cost and technological difficulties still exist, making these single photon sources impractical. Consequently, researchers typically employ highly attenuated weak coherent pulses (WCPs) to reduce the average number of photons per pulse below 1 in most physical DVQKD systems (to limit the probability of transmitting multiple photons). Even so, some pulses can contain more than one photon, resulting in a potential photon-number splitting attack (PNS) [2]. To avoid the PNS attack, the decoy state BB84 protocol was proposed by using multiple intensity levels at the transmitter's source (one signal state and several decoy states), resulting in varying photon number statistics throughout the quantum channel. Alice announces publicly a specific intensity level that has been used for the transmission of each qubit. By monitoring QBERs associated with each intensity level associated with the decoy state protocol, Alice and Bob can detect a PNS attack. Furthermore, realistic implementations of QKD systems now face a new class of practical security issues, referred to as side-channel attacks such as detector blinding and control attack [9]. Many countermeasures against these have been proposed by adding either additional hardware to the system or using information processing to reveal side-channel attack events [9], [10].

B. Continuous-variable QKD

As an alternative approach to DVQKD, CVQKD employs coherent states of light and modulates both quadratures of the electromagnetic field in a similar way to those that used in classical high-speed optical communications [11]. The *de facto* standard of CVQKD protocol is GG02, proposed by Grosshans and Grangier [12]. A breakthrough of GG02 is the elimination of single-photon generation, detection and photon counting techniques, which allows CVQKD to be compatible with commercial off-the-shelf optical communication components. However, there are also several challenges that CVQKD has to face, such as dedicated post-processing algorithms and finite-size security analysis to decode final secure keys [13]. In comparison with DVQKD, CVQKD still has some open questions in both theory and experiments [13]. The security proof of the Gaussian modulation protocol has been established against collective attacks [14] and general attacks [15]. Recently, a finite-size analysis of a binary phase modulated CVQKD protocol is proposed to prove its security against general coherent attacks, based on proof techniques of DVQKD [13]. Finite-size analysis for other modulation forms of CVQKD is still an open research question in the QKD community. In practical systems, however, imperfections in transmitted Local oscillator (LO) CVQKD systems may result in loopholes which compromise their secure key [16]. The Local Oscillator enables coherent detection of the CVQKD signal. To overcome the weakness of transmitting the Local Oscillator from Alice, some have suggested using a Local Local Oscillator (LLO) or reference free CVQKD to nullify any possible side-channel attacks on LO CVQKD [16]. Table I illustrates the principal technical differences between DVQKD and CVQKD systems.

TABLE I. COMPARISONS BETWEEN DVQKD AND CVQKD SYSTEMS

Technical features	**QKD physical setups**	
	DVQKD	**CVQKD**
Detection tech.	Single photon detector/ photon counting	Coherent detectors (e.g., Homodyne detector)
Parameter estimation	QBER	Excess noise, trasnmittance
Max. distance	>300km [17]	~100km[18]
Max. data rate (distance)	~10 kbps (100km) [17]	~1Mbps (25km) [19]

III. QKD POST-PROCESSING TUTORIAL

DVQKD post-processing systems receive random bit sequences from the detection system, establish frame synchronization and extract a sifted key referring to these random bits. An alternative step is needed for CVQKD postprocessing to convert transmitted continuous variables into discrete or binary strings. The data transmission rate is of the order of Gb/s and thus the transmitter must be able to handle Gb/s of random input, therefore FPGAs promise an appropriate platform for this task. In CVQKD the symbol rates are generally lower, but they are transmitted and received in the analogue domain, requiring the use of high-speed digital-to-analog and analog-to-digital converters, with efficient processing at the receiver to convert these complex symbols into binary data sequences, another application for FPGAs.

After this initial step, we need to correct errors in the keys. Two types of error correction methods have been used: interactive and forward error correction code based [5]. Interactive error correction needs interactive communication between Alice and Bob, leading to throughput degradation when Alice and Bob are separated by long distances. Modern error correction utilises low-density parity-check (LDPC) code for high-speed QKD systems. Error correction exposes some data to Eve, but this is eliminated through privacy amplification (PA) to obtain a smaller, yet secure key. To obtain high throughout EC and PA process, accelerations via coprocessor, GPU and FPGAs have been widely investigated,

which will be discussed in Section IV. Table II lists the choice hardware platforms for each postprocessing step.

TABLE II. QUANTUM INFORMATION PROCESSING PLATFORMS

Name	Simulation	Acceleration
Sifting	CPU	FPGA
Bit conversion	CPU	FPGA
EC	CPU	Co-processor, GPU, FPGA
PA	CPU	GPU, FPGA

A. Sifting in DVQKD

The first process in DVQKD postprocessing is called sifting. Modern DVQKD systems use optical phase rather than polarization, as it is more resistant to perturbation when transmitting signals along optical fibres. In typical phase encoded BB84, Alice transmits random encoded data by one of four phases 0, π, π/2 and 3π/2 for each pulse. {0, π} and {π/2, 3π/2} correspond to two non-orthogonal bases. Bob receives Alice's signal and randomly chooses which basis he measures with (0 or π/2) [20]. The 0 and π/2 phases in the Mach-Zehnder interferometer correspond to {0, π} and {π/2, 3π/2} basis selections, respectively, and the detectors characterise the bit information {0,1}. Fig. 1 illustrates a step-by-step tutorial of sifting implementation.

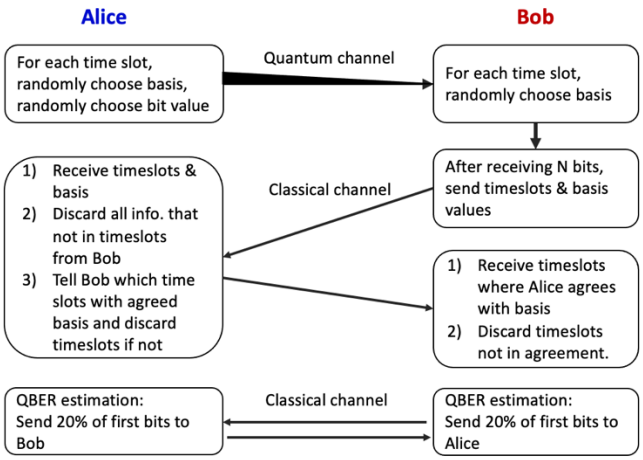

Fig. 1 A step-by-step procedure of sifting process.

B. Bit Conversion in CVQKD

Since DVQKD is based on a binary channel, sifting methods cannot be directly applied to CVQKD owing to continuous variables transmitted in the QKD link [11]. Consequently, before performing reconciliation in CVQKD schemes, conversion of continuous variables to discrete or binary strings is needed. Slice-based and multidimensional conversion methods are normally used for this task. The slice-based method can distil more than 1 bit per pulse and allows processing slices in parallel, even though, its transmission distance is limited to about 30 km [21]. Multidimensional method can extend the distance up to about 50-100 km [22], however, this method results in larger amounts of communication traffic through the classical channel. Fig. 2 illustrates how slice-based bit conversion works in CVQKD. The continuous variables at Bob and Alice are quantised and converted to slices.

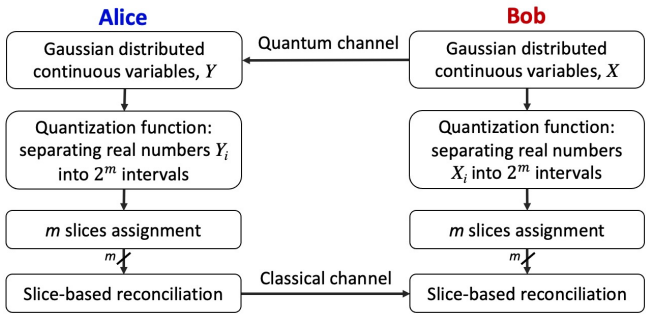

Fig. 2 A flowchart of slice-based conversion.

C. Error Correction

Prior to reconciliation, Alice has already sent her original message x over the quantum channel. Alice and Bob must undergo photon statistics evaluation at the end of the quantum exchange. Any baseline error rate is indistinguishable from the presence of an eavesdropper and must be treated as such. Therefore, an error estimation must be done and compared against a bound in order to ensure confidentiality. The most widely used error correction scheme, LDPC, is uniquely defined by its $m \times n$ matrix of sparse 1s, known as a parity-check matrix, where the original message has m bits, and the coded data has n bits, augmented with n-m parity bits. The coded data, if correctly received, will satisfy n-m parity checks, conventionally as even parity. The code rate in this error correction scheme is defined as m/n, satisfying the Shannon's channel coding theorem

$$\frac{m}{n} = 1 - h_2(e), \qquad (1)$$

where e is the error rate and $h_2(e)$ is the binary entropy function

$$h_2(e) = e \log_2\left(\frac{1}{e}\right) + (1-e)\log_2\left(\frac{1}{1-e}\right).$$

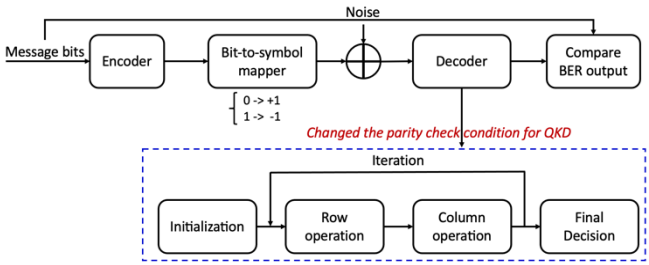

Fig. 3 Block diagram of LDPC-based reconciliation process.

Fig. 3 shows the general protocol for LDPC-based reconciliation process. Alice and Bob will have already agreed on what parity-check matrices to use for various QBERs, the value of QBER determines the dimensions of the LDPC matrix used (the higher the QBER, the larger the LDPC matrix required to correct it). Due to the channel noise, Alice receives a vector y similar to x but with some bits flipped. Bob first computes the syndrome of the original message as $s = \boldsymbol{H}x$, where $\boldsymbol{H}$ is the $m \times n$ parity-check matrix, and x is a size n column vector. She then sends s to Alice over a public channel, which Alice receives as a soft-decision vector. Using a decoding procedure, Alice then attempts to use $\boldsymbol{H}$ to produce an estimate $\hat{x}$ such that $\boldsymbol{H}\hat{x} = s$. As highlighted in Fig. 4, there are necessary modifications to the decoding algorithm for

LDPC-based error correction in QKD. The sign of computed value of δr_{mn} in the original LDPC decoding algorithm [23], when the parity bit is 1. Consequently, the decoding termination is determined by comparing newly calculated parities with those from the other side, rather than expecting a zero vector in the traditional LDPC decoding [23].

D. Privacy Amplification

The purpose of privacy amplification is to compress the error corrected data between Alice and Bob, so that Eve no longer has any information about the final key. This is performed by applying a one-way function, which produces a completely different output if any input bit is changed. Thus, unless Eve has the entire error corrected key, she cannot generate the final secure key after privacy amplification. Fig. 4 shows the flow of privacy amplification. Alice and Bob share a random t-bit binary string T, called the corrected key in QKD. Eve may learn a correlated random string Q with q bits, where q is smaller than p. Alice and Bob wish to publicly choose a compression function $f(T, \boldsymbol{M}): \{0,1\}^t \rightarrow \{0,1\}^q$, such that Eve can only obtain very little information of the secure key. To model the security of a QKD protocol, a tight finite-key security analysis has been provided by using two criteria, *i.e.*, correctness (ε) and secrecy (ϵ) [24]. A QKD protocol is ε-correct, if $\Pr(\hat{x} \neq x) \leq \varepsilon$. In some realistic applications, an incorrect decoding of the transmitted data would be detected. We could aim an extremely low decoding failure rate in reconciliation process with ε approaching to 0. A QKD protocol is ϵ-secret, if it outputs ω-secure keys with $(1 - p_{abort})\omega \leq \epsilon$, where p_{abort} is the probability that the protocol aborts [24]. The derivation of final secure key length ω has been presented in detail in [24].

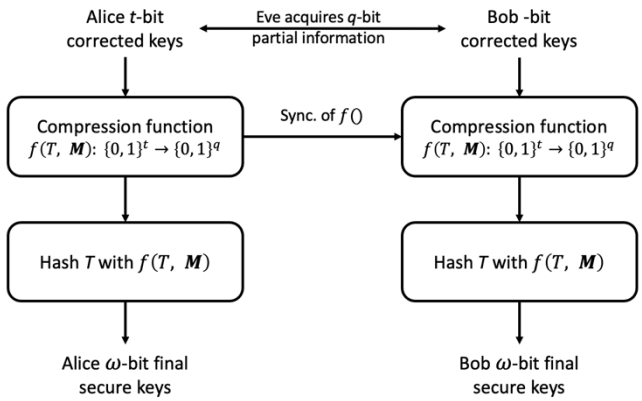

Fig. 4 A step-by-step procedure of privacy amplification.

IV. QKD Post Processing on Heterogeneous Hardware Platforms

A. Characteristics of QKD Post Processing on Different Hardware Platforms

The characteristics of representative platforms such as CPU, GPU and FPGA for QKD post-processing are summarized as shown in Table 3.

1) Development cost and difficulty: We first compare the development difficulty, development cycle and device cost for a QKD system implementation. A CPU is the most user friendly platform, as the x86 architecture is fully supported by various algorithm libraries, programming evironment, compilers, APIs and design manuals from the community. Therefore, its development cost is the lowest. The GPU has its own special architecture for parallel programming, however, users also need specialized design skills to make the most advantage of the GPU's parallel computation ability. Compared to CPUs and GPUs, the development cost and difficulty using an FPGA is the highest and longest, because users have to use a hardware description language (HDL) to design basic operations, instructions, and APIs, *etc.* Even though FPGA vendors, such as AMD (Xilinx) and Intel (Altera) have built their high-level synthesis (HLS) toolchains, dedicated design optimisations are needed to implement common post-processing algorithms *e.g.* FFT and large integer multiplication, using HLS.

2) Integration control for a complete QKD system: In the post-processing stage, there are two main tasks to address: signal control and calculation. An FPGA can complete both the system control and calculation tasks in an efficient manner, therefore, by the use of an FPGA it is easy to integrate optical and electronic devices into a system. However, the CPU, as a general-purpose processor can only complete the calculation task, and is difficult to use for system control tasks. Similarly, GPUs are designed for parallel computing accelerators, therefore system control tasks still require CPUs. Therefore, high speed system control is optimised with an FPGA, and CPUs should be used for complex, non time critical tasks.

3) Power consumption: GPUs can produce very high data throughputs, but at the cost of relatively low energy efficiency. FPGA implementations demonstrate high power efficiency [30], but are less flexible.

TABLE III. Characteristics of Different Platforms

Platform	CPU	GPU	FPGA
Development cost and difficulty	**Low**	Medium	High
Integration control for a complete QKD system	Fair	Hard	**Easy**
Power efficiency	**High**	Low	**High**

B. Hardware Acceleration for Error Correction

For high-speed quantum key distribution (QKD) post-processing, error correction is one of the most computationally intensive steps and is usually the system speed bottleneck. As the mostly used error correction method in the reconciliation process, LDPC linear codes demonstrate high throughput reconciliation, because LDPC codes have very low communication complexity and only require a single unidirectional message between Alice and Bob. Communication latency thus does not reduce the throughput, unlike Cascade reconciliation requiring many round-trip messages and performing poorly at long distances [4]. The structure of the decoding algorithm, combined with the simple communication complexity, also allows for the EC process to be readily parallelized. This is an important factor for achieving high decoding throughput, and also makes

LDPC decoders suitable for implementing using GPUs and FPGAs for very high throughput.

Error correction efficiency and operation throughput are the two most important performance parameters, but they are often compromised in actual realization due to the available hardware resources. Dixon *et al.* have previously realized a GPU implementation of LDPC-based reconciliation, where the authors presented an optimized memory access model with flood scheduling in GPUs, resulting in better performance [25]. Guo *et al.* employed the framework of open computing language (OpenCL) to accelerate the speed of reconciliation algorithms on a heterogeneous computing structure (GPU with a general-purpose CPU). Computation kernels that can be accelerated in parallel processing are assigned to GPUs, while other signal controlling tasks are performed by the CPU [6]. Yang *et al.* developed an LDPC-based EC accelerator employing FPGA's deep pipeline parallelism characteristics [26]. With careful design space exploration, multiplexing and non-multiplexing strategies have been investigated to achieve the trade-off between area and performance, leading to an up-to 100 million symbols per second throughput on a Xilinx Virtex-7 FPGA [26].

Another characteristic in QKD error correction lies in its time-varying quantum channel. In order to guarantee high error correction efficiency, rate compatible reconciliation scheme should be adopted. Recently, Zhu *et al.* have proposed a rate-compatible reconciliation algorithm based on quasi-cyclic (QC) low-density parity-check (LDPC) codes. Prototyping tool of high-level synthesis (HLS) is used targeting Zynq Ultrascale+ ZCU102 development board [4]. Experimental results show that the maximum throughput rate of the implemented reconciliation module can reach 136 Mbps, while the efficiency factor is kept lower than 1.32 across the error rate range of 1.7% and 10.6% [4]. However, in their method, a particular set of IEEE 802.16E standard LDPC codes are used because these LDPC matrices have regular structure leading to an easy mapping using on-chip memory blocks on FPGAs.

C. Hardware Acceleration for Privacy Amplification

Hardware accelerators for privacy amplification largely focus on algorithmic and architectural optimisations by employing the modern hardware architectures provided by GPUs and FPGAs. The simplest approach for PA is direct matrix multiplication, which has computational complexity of $O(n^2)$. To reduce the computational complexity from $O(n^2)$ to $O(n \log_2 n)$, Takahashi *et al.* first speeded up the Toeplitz matrix in PA with number theoretical transform (NTT), achieving a PA throughput of 28.22 Mb/s at a dataset size of 10^8 bits on CPUs [27]. Further improvement of PA throughput has been demonstrated by exploiting massive parallelism offered by Intel Xeon Phi™ co-processors. This co-processor is able to perform fast matrix multiplication due to several programming techniques, such as vectorization for matrix transpose, butterfly computation of NTT, suitable instruction set regarding cache hit ratio, loop unrolling to butterfly computation for reducing the number of iterations and parallelization by multi-thread processing of data input and output [5]. Another algorithmic optimization is to use an FFT-based PA algorithm, which is also performed in the computational complexity of $O(n \log_2 n)$. An FFT enhanced high-speed and large-scale PA scheme was first evaluated on a commercial CPU platform [28]. When input scale is 128Mbits, the speed can reach at least 39.15Mbps. Except for a Toeplitz hash, other universal hash families, such as modular arithmetic hash which is suitable for low-cost CPU platform have been investigated [29]. Yan *et al.*'s research focus on the acceleration of large module multiplication in this hash computation by using the GNU multiple precision arithmetic library (GMP) in CPU, and 69 Mbps at the block sizes of10^8 has been achieved [29].

Mirroring CPU software implementations, many GPU and FPGA-based PA accelerators have been investigated. Constantin *et al.* [30] and Yang *et al.* [31] both proposed improved block algorithm of PA with Toeplitz matrix on FPGAs, and achieved 41Mbps and 65.443Mbps, respectively. However, this architecture requires a large amount of on-chip storage. The computational power is limited by memory size, and cannot perform long-input-block-length privacy amplification directly. Thanks to the FPGA's flexible adaptability, Walenta *et al.* optimised FPGA-based PA using pipeline and stored the data in off-chip memory instead of on-chip block RAMs to be able to process enough bits following the finite-size security proof [25]. Moreover, Wang *et al.* proposed a attenuation-compatible method to satisfy the sufficient key length requirement of different attenuation [32]. This method is implemented by dividing the long input length into short block length, and then performing the privacy amplification separately, finally merging the corresponding results together. To make full use of the computing resources, the short blocks can be calculated in parallel leading to a speed around 1Gbps in PA process [32]. A review of FPGA-accelerated EC and PA on different hardware platforms has been summarised in Table IV.

TABLE IV. REPRESENTATIVE IMPLEMENTATIONS OF POST PROCESSING ON HETEROGENOUS PLATFORMS

Stage	Authors (Year)	Hardware platform	Features
EC	Guo et al. (2020) [6]	Intel CPU i7-7700k & Nvidia Tesla K40C	Heterogeneous system of CPU+GPU and the framework of OpenCL
	Dixon et al. (2014) [25]	Nvidia M2090 GPU	Flood scheduling and fast floating-point arithmetic cores
	Zhu *et al*. (2022) [4]	Zynq Ultrascale+ ZCU102 FPGA	HLS for LDPC, and rate-compatible EC
	Yuan *et al*. (2018) [5]	Custom-made FPGA	Integration of optics control, sifting, and LDPC EC.
	Yang *et al*. (2020) [26]	Virtex-7 VC709 FPGA	Fully pipelined, non-multiplexed and multiplexed LDPC
Stage	**Authors (Year)**	**Hardware platform**	**Features**
PA	Takahashi et al. (2016) [27]	Intel Xeon E5-2620v2 CPU	NTT for Toeplitz matrix multiplication
	Yan et al. (2018) [29]	Intel i9-9900k CPU	Modular arithmetic hash with GMP

Yuan *et al*. (2018) [5]	Intel Xeon Phi™ co-processor	Vectorization for matrix transpose, butterfly computation and parallel computing insturctions
Constantin et al. (2017) [30]	Virtex-6 LX240T FPGA	Use of off-chip memory (DDR2 SDRAM)
Yang *et al*. (2017) [31]	Vertex-7 FPGA	Rhomboid-block for matrix multiplication
Tang et al. (2019) [28]	Nvidia TITAN Xp GPU	Length-compatible for differentlength requirement

V. Conclusion.

In this paper, we provide a systematic tutorial for QKD post-processing with step-by-step procedures. We also review the state-of-the-art QKD post-processing implementations on heterogenous CPU, GPU and FPGA platforms, with a particular focus on domain-specific hardware accelerators for LDPC-based error correction and privacy amplification. The user-friendly design environment of the CPU is helpful for software-based simulations, even though the use of CPU's is inadequate in performing parallel computation. GPU's provide parallel processing capabilities, however, high power consumption hinders their potential deployment in embedded systems. FPGAs can fulfil a key role in QKD post-processing. They are ideally suited to the relatively simple, yet high speed tasks required at the first stages of post processing. Their outputs can be interfaced with general purpose CPUs and specialised co-processors to perform more mathematically complex functions such as error-correction and privacy amplification.

Acknowledgment

This work has been funded by the UK EPSRC via the Quantum Communications Hub (EP/T001011/1).

References

[1] P. K. Lam and T. C. Ralph, 'Continuous improvement', *Nat. Photonics*, vol. 7, no. 5, pp. 350–352, May 2013.

[2] B. Huttner, N. Imoto, N. Gisin, and T. Mor, 'Quantum cryptography with coherent states', *Phys. Rev. A*, vol. 51, no. 3, pp. 1863–1869, Mar. 1995.

[3] Toshiba Europe, 'Toshiba shrinks quantum key distribution technology to a semiconductor chip'. 2021. [Online]. Available: https://www.toshiba.eu/pages/eu/Cambridge-Research-Laboratory/toshiba-shrinks-quantum-key-distribution-technology-to-a-semiconductor-chip

[4] M. Zhu, K. Cui, S. Li, L. Kong, S. Tang, and J. Sun, 'A Code Rate-Compatible High-Throughput Hardware Implementation Scheme for QKD Information Reconciliation', *J. Light. Technol.*, pp. 1–1, 2022.

[5] Z. Yuan *et al.*, '10-Mb/s Quantum Key Distribution', *J. Light. Technol.*, vol. 36, no. 16, pp. 3427–3433, Aug. 2018.

[6] D. Guo, C. He, T. Guo, Z. Xue, Q. Feng, and J. Mu, 'Comprehensive high-speed reconciliation for continuous-variable quantum key distribution', *Quantum Inf. Process.*, vol. 19, no. 9, p. 320, Sep. 2020.

[7] C. H. Bennett and G. Brassard, 'Quantum cryptography: Public key distribution and coin tossing', in *International Conference on Computers, Systems and Signal Processing*, 1984, vol. 1, pp. 175–179.

[8] M. Alhussein and K. Inoue, 'Differential phase shift quantum key distribution with variable loss revealing blinding and control side-channel attacks', *Jpn. J. Appl. Phys.*, vol. 58, no. 10, p. 102001, Oct. 2019.

[9] M. Alhussein, K. Inoue, and T. Honjo, 'Monitoring coincident clicks in differential-quadrature-phase shift QKD to reveal detector blinding and control attacks', *Jpn. J. Appl. Phys.*, vol. 58, no. 1, p. 012006, Jan. 2019.

[10] A. R. Dixon *et al.*, 'Quantum key distribution with hacking countermeasures and long term field trial', *Sci. Rep.*, vol. 7, no. 1, p. 1978, Dec. 2017.

[11] T. Hirano, 'Introduction to Continuous Variable Quantum Key Distribution', in *Optical Fiber Communication Conference (OFC) 2022*, San Diego, California, 2022, p. Tu3I.1.

[12] F. Grosshans and P. Grangier, 'Continuous Variable Quantum Cryptography Using Coherent States', *Phys. Rev. Lett.*, vol. 88, no. 5, p. 057902, Jan. 2002.

[13] T. Matsuura, K. Maeda, T. Sasaki, and M. Koashi, 'Finite-size security of continuous-variable quantum key distribution with digital signal processing', *Nat. Commun.*, vol. 12, no. 1, p. 252, Dec. 2021.

[14] M. Navascués, F. Grosshans, and A. Acín, 'Optimality of Gaussian Attacks in Continuous-Variable Quantum Cryptography', *Phys. Rev. Lett.*, vol. 97, no. 19, p. 190502, Nov. 2006.

[15] A. Leverrier, R. García-Patrón, R. Renner, and N. J. Cerf, 'Security of Continuous-Variable Quantum Key Distribution Against General Attacks', *Phys. Rev. Lett.*, vol. 110, no. 3, p. 030502, Jan. 2013.

[16] S. Ren, S. Yang, A. Wonfor, I. White, and R. Penty, 'Demonstration of high-speed and low-complexity continuous variable quantum key distribution system with local local oscillator', *Sci. Rep.*, vol. 11, no. 1, p. 9454, Dec. 2021.

[17] B. Korzh *et al.*, 'Provably secure and practical quantum key distribution over 307 km of optical fibre', *Nat. Photonics*, vol. 9, no. 3, pp. 163–168, Mar. 2015.

[18] D. Huang, P. Huang, D. Lin, and G. Zeng, 'Long-distance continuous-variable quantum key distribution by controlling excess noise', *Sci. Rep.*, vol. 6, no. 1, p. 19201, May 2016.

[19] D. Huang *et al.*, 'Continuous-variable quantum key distribution with 1 Mbps secure key rate', *Opt. Express*, vol. 23, no. 13, p. 17511, Jun. 2015.

[20] M. Lucamarini *et al.*, 'Efficient decoy-state quantum key distribution with quantified security', *Opt. Express*, vol. 21, no. 21, p. 24550, Oct. 2013.

[21] G. VanAssche, J. Cardinal, and N. J. Cerf, 'Reconciliation of a Quantum-Distributed Gaussian Key', *IEEE Trans. Inf. Theory*, vol. 50, no. 2, pp. 394–400, Feb. 2004.

[22] A. Leverrier, R. Alléaume, J. Boutros, G. Zémor, and P. Grangier, 'Multidimensional reconciliation for a continuous-variable quantum key distribution', *Phys. Rev. A*, vol. 77, no. 4, p. 042325, Apr. 2008.

[23] D. J. C. MacKay and R. M. Neal, 'Near Shannon Limit Performance of Low Density Parity Check Codes', *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.

[24] M. Tomamichel, C. C. W. Lim, N. Gisin, and R. Renner, 'Tight finite-key analysis for quantum cryptography', *Nat. Commun.*, vol. 3, no. 1, p. 634, Jan. 2012.

[25] A. R. Dixon and H. Sato, 'High speed and adaptable error correction for megabit/s rate quantum key distribution', *Sci. Rep.*, vol. 4, no. 1, p. 7275, May 2015.

[26] S.-S. Yang, Z.-G. Lu, and Y.-M. Li, 'High-Speed Post-Processing in Continuous-Variable Quantum Key Distribution Based on FPGA Implementation', *J. Light. Technol.*, vol. 38, no. 15, pp. 3935–3941, Aug. 2020.

[27] R. Takahashi, Y. Tanizawa, and A. R. Dixon, 'High-Speed Implementation of Privacy Amplification in Quantum Key Distribution', in *6th International Conference on Quantum Cryptography*, 2016.

[28] B.-Y. Tang, B. Liu, Y.-P. Zhai, C.-Q. Wu, and W.-R. Yu, 'High-speed and Large-scale Privacy Amplification Scheme for Quantum Key Distribution', *Sci. Rep.*, vol. 9, no. 1, p. 15733, Dec. 2019.

[29] B. Yan, Q. Li, H. Mao, and X. Xue, 'High-Speed Privacy Amplification Scheme Using GMP in Quantum Key Distribution', *IEEE Photonics J.*, vol. 12, no. 3, pp. 1–13, Jun. 2020.

[30] J. Constantin *et al.*, 'An FPGA-Based 4 Mbps Secret Key Distillation Engine for Quantum Key Distribution Systems', *J. Signal Process. Syst.*, vol. 86, no. 1, pp. 1–15, Jan. 2017.

[31] S.-S. Yang, Z.-L. Bai, X.-Y. Wang, and Y.-M. Li, 'FPGA-Based Implementation of Size-Adaptive Privacy Amplification in Quantum Key Distribution', *IEEE Photonics J.*, vol. 9, no. 6, pp. 1–8, Dec. 2017.

[32] X. Wang, Y. Zhang, S. Yu, and H. Guo, 'High-Speed Implementation of Length-Compatible Privacy Amplification in Continuous-Variable Quantum Key Distribution', *IEEE Photonics J.*, vol. 10, no. 3, pp. 1–9, Jun. 2018.