High-speed Implementation of FFT-based Privacy Amplification on FPGA in Quantum Key Distribution

Qiong Li, Bing-Ze Yan, Hao-Kun Mao, Xiao-Feng Xue

Information Countermeasure Technique Institute, School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001 China.

Abstract: Privacy amplification (PA) is a vital procedure in quantum key distribution (QKD) to generate the secret key that the eavesdropper has only negligible information from the identical correcting key for the communicating parties. With the increase of repeat frequency of discrete-variable QKD (DV-QKD) system, the processing speed of PA has become the bottle neck restricting DV-QKD’s secure key rate. The PA using Toeplitz-based Hash function is adopted widely because of its simplicity and parallel feature. Because this algorithm can be accelerated with Fast Fourier Transform (FFT), an improved scheme PA for Field-programmable Gate Array (FPGA) based on this is proposed. This paper improves the custom FFT-based algorithm by reducing the number of computations and read/write memory operations significantly. The correctness is verified when implemented in a Xilinx Virtex-6 FPGA. Meanwhile, the processing speed of improved scheme can nearly double the classical Toeplitz Hashing scheme on FPGA through the actual experiment.

Index Terms: Quantum Key Distribution, Privacy Amplification, Fast Fourier Transform, Field-Programmable Gate Array.

1. Introduction
Quantum Key Distribution exploits quantum mechanics theorem to accomplish the secure key distribution. Since Bennet and Brassard proposed the first practicable protocol in 1984 [1], many protocols have been proposed successively. These protocols can divide into discrete variable (DV) protocols [2], [3] and continuous variable (CV) protocols [7], [8]. Because the DV-QKD is proposed earlier and the security proof of it is more complete, the development of DV-QKD drives to mature stage and many DV-QKD business systems have been developed [11], [12]. We focused on the research of DV-QKD and found that key generation rate and QKD system on chip are two important research points of DV-QKD at this stage. DV-QKD is divided into four phases: quantum communication, public discussion, key reconciliation, and privacy amplification. The first three phases make two distant legitimate parties, usually named Alice and Bob, obtain identical random sequence. However, in this process, some information is exposed inevitably to Eavesdropper, usually named Eve.

Privacy amplification (PA) eliminates the leaked information by distilling the final secret key from a long secret random sequence with universal Hash function [15], [17]. Several classes of Hash function has been applied to perform the PA [18], [19]. To reduce the finite size effect in distilling secure keys, the lengths of input blocks for PA should be at least $10^6$ [20], and this leads to large length of processing blocks. Zhang et al. choose a simple multiplicative universal class of Hash function to speed up PA process, and they construct an optimal multiplication algorithm with four
basic multiplication algorithms \cite{21}. The speed of this algorithm achieves 14.68 Mbps based on CPU, but this algorithm is an iterative algorithm meaning that it consumes large resources and it is unsuited for hardware implementation. Besides, Toeplitz hashing \cite{22} is widely used in PA process because of its simplicity and parallel feature. The authors in \cite{23} proposed block parallel algorithm to speed Toeplitz hashing. The authors in \cite{24, 25} proposed improved block parallel algorithm of Toeplitz hashing respectively. The algorithm in \cite{25} achieves 64 Mbps processing speed based on field-programmable gate array (FPGA) and reduces memory resources significantly. While this kind of algorithm has reach its speed limit constrained of its algorithmic complexity \(O(n^2)\). Fast Fourier transform (FFT) and fast number theory transform (FNTT) are efficient fast algorithms for Toeplitz hashing, which reduce the algorithmic complexity from \(O(n^2)\) to \(O(n \log n)\). The authors in \cite{26} firstly proposed a FFT based PA algorithm and implemented on Many Integrated Core (MIC). The process speed of algorithm reaches 60 Mbps with raw key length of 12.8M. The authors in \cite{27} proposed a FFT-based PA algorithm in CV-QKD based on graphic processing unit (GPU). The speed of privacy amplification is achieved over 1 Gbps. However, this algorithm is only suitable for CV-QKD, because it’s efficient in case of great raw key length and low compression ratio. Crucially, the GPU and MIC platform are both hard to be integrated for its volume and power consumption.

Based on the investigation result, FPGA is a suitable platform for DV-QKD system with the feature of high-parallelism and embeddable platform. More importantly, the energy consumption of FPGA is much lower than that of GPU and MIC. Nevertheless, existing PA algorithms on FPGA are all parallel block method of Toeplitz hashing with algorithmic complexity of \(O(n^2)\). FFT have great potential to accelerate the speed and reduce the consumption of PA algorithm on FPGA. Unfortunately, there is no practicable FFT-based PA scheme on FPGA. The main reason is that the \(10^6\) length of input blocks increases the difficulty and cost of implementation. To solve this problem, we proposed a FFT-based PA hardware accelerate algorithm and implemented it on Virtex-6 FPGA. The throughput of our algorithm reaches 116 Mbps with the raw key length \(n = 1M\). It’s nearly 2 times faster than the classical Toeplitz Hashing Algorithm on FPGA.

The rest of this paper is organized as follows. The principle of privacy amplification with modified Toeplitz we used is described in Section 2. In Section 3, the detail and the key improvements of our FFT-based PA hardware accelerate algorithm are stated. In Section 4, we present our PA hardware implementation module and experiment results, including the processing speed and the requirements of the hardware resources. A comparison between several FPGA-based hardware implementations is also presented. In Section 5, we provide a brief conclusion.

2. Related Work

2.1. Privacy Amplification

Privacy amplification is a process that allows two parties to distill a secret key from a secret random variable about which an eavesdropper has partial information \cite{15}. Before PA procedure in QKD, Alice and Bob share a random \(n\) bits binary string \(W\), while Eve learns a correlated random string \(V\) providing \(t\) \((t < n)\) bits of information about \(W\), i.e., \(H(W|V) \geq n - t\). Alice and Bob wish to publicly choose a compression function \(g : \{0, 1\}^n \rightarrow \{0, 1\}^r\) such that Eve’s partial information on \(W\) and her complete information on \(g\) give her arbitrarily little information about \(K = g(W)\). This procedure is indicated as Fig. 1.

Universal hash function \cite{22} is discovered to suit as the compression function \(g\) for privacy amplification \cite{15}. The mutual information between the distill key compressed by universal hash function and Eve’s information can be proved using Rényi entropy,

\[
I(K; g, V) \leq 2^{-s} / \ln 2
\]

(1)

Where \(s = n - t - r\) means the security coefficient of PA.

Considering the security threaten of finite-key effect, the input raw key length \(n\) of privacy amplification in DV-QKD should be larger than \(10^6\) \cite{20, 28}. Therefore, the calculation of hash function is very large, and the choice of hash function class is extremely important.
Fig. 1: The procedure of privacy amplification

2.2. Modified Toeplitz Matrix

The Toeplitz matrix is a particular class of the universal hash functions [22]. Because Toeplitz matrix is diagonal-constant matrix, Toeplitz matrix can be constructed by its first column and first row and calculated by FFT. Therefore, the required number of random bits can be reduced to $n + r - 1$ and the calculation complexity can be reduced to $O(n \log n)$. However, the input length of FFT would be $n + r - 1$ or $2n$ to calculate the Toeplitz matrix depending on the compression ratio, that would cost a lot of extra cost in hardware implementation.

Hayashi et al. proposed using modified Toeplitz matrix instead of Toeplitz matrix as the compression function and give the security proof [23]. The modified Toeplitz matrix is constructed by the concatenation of Toeplitz matrix and the identity matrix $(X, I)$. For instance, Eq. (2) is a modified Toeplitz matrix.

$$S_{r \times n} = \begin{bmatrix} 1 & V_{r-1} & V_{r-2} & \cdots & V_{n-2} & V_{n-1} \\ 1 & V_{r-2} & V_{r-1} & \cdots & V_{n-2} & V_{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & V_1 & V_{n-r-1} & V_{n-r} & \cdots & V_{n-2} & V_{n-1} \\ 1 & V_0 & V_1 & \cdots & V_{n-r-2} & V_{n-r-1} \end{bmatrix}$$

(2)

Using the modified Toeplitz matrix, the required quantity of random bits can be reduced to $n$ and the input length of FFT can be reduced to $n$. In this case, the length of FFT is only related to the input block length of the key other than the final key length. This feature notably reduced the design complexity of FFT-based PA.

2.3. Modified Toeplitz Matrix Calculation by FFT

FFT algorithm is a common algorithm to calculate the Toeplitz matrix due to the $O(n \log n)$ calculation complexity of FFT. A general calculative process of the modified Toeplitz is provided in this section. $X_n = [X_0, X_1, \ldots, X_{n-1}]'$ is the input sequence of PA. $Y_r = [Y_0, Y_1, \ldots, Y_r]'$ is the final key sequence. Eq. (3) is the calculative process of the final key with the modified Toeplitz matrix $S_{r \times n}$.

$$Y_r = S_{r \times n} \times X_n = [I_{r \times r}, V_{r \times (n-r)}] \times [X_r] = [I_{r \times r} \times X_r] + [V_{r \times (n-r)} \times X_{n-r}] = X_r + Y_r'$$

(3)

Make up the Toeplitz matrix $V_{r \times (n-r)}$ to the cyclic matrix, then the calculation of $Y_r'$ can be accelerated by FFT shown as Eq. (4).

$$Y_r' = \begin{bmatrix} P_{(n-r) \times (n-r)} & X_{n-r} \end{bmatrix} \times \begin{bmatrix} 0 \\ V_r \times (n-r) \times X_{n-r} \end{bmatrix} = \begin{bmatrix} P_{(n-r) \times r} & P_{(n-r) \times (n-r)} \end{bmatrix} \times \begin{bmatrix} P_{(n-r) \times r} & V_r \times (n-r) \end{bmatrix} \times \begin{bmatrix} X_{n-r} \\ 0 \end{bmatrix}$$

(4)

$$= V_n \otimes X'_n = \text{IFFT}(\text{FFT}(V_n) \cdot \text{FFT}(X'_n))$$
\[ V_n = [V_0, V_1, \ldots, V_n] \] is the description of Toeplitz matrix, \( X'_n = [0, 0, \ldots, X_{n-r}, \ldots, X_{n-1}] \)' is a part of the input sequence. The matrix \( P \) aims to complement the Toeplitz matrix \( V_{r \times (n-r)} \) to cyclic matrix.

## 3. High Speed FFT-based Privacy Amplification Hardware Scheme

A high speed PA hardware scheme for FPGA implementation is proposed in this section. An overall process of the scheme is given based on FFT algorithm. Furthermore, three points of Algorithm optimization are given in accordance with the feature of privacy amplification.

### 3.1. Overall Process of FFT-based PA Hardware Schemes

The overall process of FFT-based PA Hardware Scheme is indicated as Fig. 2. The pre-processing phase divides the input sequence \( X_n = [X_0, X_1, \ldots, X_{n-1}] \) into \( X_r = [X_0, X_1, \ldots, X_{r-1}, 0, \ldots, 0] \) and \( X_{n-r} = [0, 0, \ldots, 0, X_r, \ldots, X_{n-1}] \). The dot operational character means the dot product of the two FFT results. The post-processing phase rounds the result of IFFT to Boolean sequence. The final key sequence of PA is the XOR result of \( X_{n-r} \) and the result of the Toeplitz cyclic convolution.

![Fig. 2: The Overall Process of FFT-based PA Hardware Scheme](image)

The calculation of FFT/IFFT needs the greatest computation in the entire process. Although special hardware circuits for FFT/IFFT is provided in FPGA, the input length of these circuits can't satisfy the request of PA. Concerning this issue, two-dimensional long FFT algorithm [30] is adopted to calculate long FFT with small point FFT core. The procedure of custom 2-D long FFT algorithm is described as Algorithm 1.

In this way the calculation of FFT and IFFT can be accomplished with multiple small point FFT cores at high speed. However, this algorithm needs matrix transposition and storage repeatedly, that leads to large amount of memory read and write operations. Thus, the data transfer rate of memory is the bottleneck of the entire PA algorithm. Aiming at such shortcomings, several optimizations are proposed to reduce the amount of storage data and improve the processing speed according to the feature of PA.

### 3.2. Real-valued FFT Acceleration

The input sequence \( X_n \) and the description of Toeplitz \( V_n \) are both real sequences, but the FFT hardware circuits are designed to compute the FFT of a complex sequence. Most FFT-based PA scheme regards the input sequence as the real part and sets imaginary part to 0 directly. This method leads to a waste of computing resource and storage resource. A real-valued FFT algorithm [31] is introduced in our method to solve this problem. With this algorithm, two real-valued FFT calculations can be accomplished by one complex-valued FFT. In our scheme, we need the FFT results of the input sequence and the Toeplitz sequence. We can get the results in one complex-valued FFT with the method described as below.

\[
z(n) = x(n) + i \cdot v(n)
\]

\[
Z(k) = FFT(z(n))
\]
This optimization accomplishes the FFT of two sequence with one complex-valued FFT operation, improving the processing rate and saving nearly half of the computing resource and storage resource.

Algorithm 1 Custom 2-Dimensional Long FFT of \( x \)

**Input:** \( X_n = x_0, x_1, \cdots, x_{n-1} \)

**Output:** \( Y_n = \text{FFT}(X_n) \)

1: Convert one-dimensional input sequence \( X_n \) into two-dimensional matrix \( A_{k \times k} \)

2: \( A' = \text{Transposed}(A) \) // Transposed(A) is the transpose function of the matrix A

3: for \( i = 0 \) to \( k - 1 \) do

4: \( A_1[i][0 : k - 1] = \text{FFT}(A'[i][0 : k - 1]) \)

5: end for

6: for \( i = 0 \) to \( k - 1 \) do

7: for \( j = 0 \) to \( k - 1 \) do

8: \( A_2[i][j] = A_1[i][j] \times W[i \times j] \) //W is the multiply rotation factor

9: end for

10: end for

11: \( A_2' = \text{Transposed}(A_2) \)

12: for \( i = 0 \) to \( k - 1 \) do

13: \( A_3[i][0 : k - 1] = \text{FFT}(A_2'[i][0 : k - 1]) \)

14: end for

15: \( A_3' = \text{Transposed}(A_3) \)

16: for \( i = 0 \) to \( k - 1 \) do

17: for \( j = 0 \) to \( k - 1 \) do

18: \( Y[i \times k + j] = A_3'[i][j] \)

19: end for

20: end for

3.3. A Modified 2D-FFT for PA

The matrix transposition and storage operation of long FFT algorithm will cost an amount of time. Nevertheless, in the PA, the unconditional secure key is the final result of PA and the security of the key is not affected by the input sequence of the origin key and the export order of the final key. Therefore, the input and output order of FFT algorithm result is also out of consideration in PA algorithm. Without regard to the order of FFT algorithm, the procedure of long FFT/IFFT algorithm mentioned earlier (3.1) can be simplified as follow:

In this way the matrix transformation and storage that the long FFT/IFFT algorithm needs will significantly decrease. Taking the one-million points PA algorithm as an example, the output sequence of the PA algorithm with the natural order FFT and the unnatural order FFT is indicated as fig.3.

Because our one-million points PA algorithm is based on the two-dimensional FFT algorithm, the input sequence will be loaded into a 1024-dimension matrix. If the matrix is processed row by row in the two-dimensional FFT algorithm, the input and output sequence of the PA algorithm with the natural order FFT is identical to that in Fig.3(a). Meanwhile, the input and output sequence of the PA algorithm with the unnatural order FFT is shown in Fig.3(b) in the column-by-column
**Algorithm 2** Modified 2-D FFT for PA

**Input:** $X_n = x_0, x_1, \cdots, x_{n-1}$

**Output:** $Y_n = FFT(X_n)$

1. Convert one-dimensional input sequence $X_n$ into two-dimensional matrix $A_{k \times k}$
2. for $i = 0$ to $k - 1$
3. $A_1[i][0 : k-1] = FFT(A[i][0 : k-1])$
4. end for
5. for $i = 0$ to $k - 1$
6. for $j = 0$ to $k - 1$
7. $A_2[i][j] = A_1[i][j] \times W[i \times j]$ // $W$ is the multiply rotation factor
8. end for
9. end for
10. $A' = Transposed(A_2)$ // Transposed(A) is the transpose function of the matrix A
11. for $i = 0$ to $k - 1$
12. $A_3[i][0 : k-1] = FFT(A'_{2}[i][0 : k-1])$
13. end for
14. for $i = 0$ to $k - 1$
15. for $j = 0$ to $k - 1$
16. $Y[i \times k + j] = A_3[i][j]$
17. end for
18. end for

![Fig. 3: the Input/Output Sequence Order Diagram of FFT](image)

Although modified 2D-FFT algorithm has decreased the matrix transposition times to twice, the matrix transposition still spends a lot of time. A high effective matrix transposition method is introduced in our scheme. Due to the access mechanism of DDR-SDRAM, the row span access operation will reduce the access speed of DDR. However, the matrix transposition needs a large amount of the row span access operations. The high effective matrix transposition method uses...
the matrix partitioning technology to reduce the times of the row span access operation. Taking the one-million points PA algorithm as an example, the main process of the common matrix transposition based on the DDR memory model is indicated as Fig. 4.

\[
T_{\text{row-span}} = T_{\text{write}} + T_{\text{read}} = 1024 + 1024 \times 1024 = 1049600 \tag{11}
\]

The high effective matrix transposition method uses the matrix partitioning technology to balance the row span access times of the read operation and the write operation. This method can reduce the total row span access times and increase the data rates significantly. The main process of the high effective matrix transposition is indicated as Fig. 5.

In this case, each row of the matrix is transformed to a 32-dimension matrix. The row span access operation times of the high effective matrix transposition method is calculated as follow:

\[
T_{\text{row-span}} = T_{\text{write}} + T_{\text{read}} = 32 \times 1024 + 32 \times 1024 = 65536 \tag{12}
\]

This method reduces the row span access times obviously. This method is experimented with the DDR3-SDRAM to prove its improvement on the data rates of matrix transposition. The comparison experiment result of the two methods is indicated in TABLE I.

| Data Size | Matrix Format | Operation time | Average rate |
|-----------|---------------|----------------|--------------|
| The common method | 64Mb | 1024 x 1024 | 8.959 us | 7.14Gbps |
| The high effective method | 64Mb | 1024 x 1024 | 3.242 us | 19.74Gbps |

According to the experiment results, the high effective Matrix Transposition method can double the data rate of matrix transposition.
4. Implementations and Results

Our PA scheme is implemented on the ML605 Evaluation Kit. The kit is based on a Virtex-6 XC6VLX240T-1FFG1156 FPGA with 241,152 logic cells. The kit also contains a 512MB DDR3 SDRAM to support our scheme. The overall structure of our PA scheme is shown in Fig. 6.

The input data buffer is designed to store the input key and the Toeplitz random sequence. On output, it also converts the data to the floating-points for the FFT convolution. The PA Control module is the core controller of the PA module. It controls the FFT convolution unit to accomplish the PA computational tasks. The FFT convolution unit is the key unit in PA module. It contains five major parts. The FFT core is designed to calculate the FFT on each row of the matrix. Two FFT IP-Cores provided by Xilinx are used to meet the speed requirements. The length of the FFT IP-Core is set as 1024 in one million points PA scheme. Similarly, the IFFT core is designed to calculate the IFFT on each column or row of the matrix. Two FFT IP-Cores are used and their length also set as 1024. The multiply unit completes the computational task of the real-valued FFT accelerated optimizing. The rotator factory multiply unit point-wisely multiply rotation factors by the result of FFT/IFFT core to accomplish the two-dimensional FFT algorithm. The data distribute unit distributes the data for the calculation units and exchanges data with SDRAM controller.

We simulated the scheme using the simulation tool Modelsim and verified the correctness of the scheme with the result on Matlab. Then the scheme was implemented on a ML605 Evaluation Kit and the result is accord with the simulation. The resource utilization of the PA scheme in hardware
is shown in TABLE II. According to the resource utilization above, there is enough spare resource for other modules to constitute the post-processing system on one chip with the PA module.

| The scheme used | XC6VLX240T Available | Utilization rate |
|-----------------|----------------------|------------------|
| Number of LUTs  | 37203                | 150720           | 24%              |
| Number of 36K BRAMs | 164                  | 416              | 39%              |
| Number of 18K BRAMs | 66                   | 832              | 7%               |
| Number of DSP48E1s | 360                  | 768              | 46%              |

The comparison of several FPGA-based implementations of PA scheme is indicated in TABLE III. The schemes in [23], [24], [25] are all based on the linear feedback shift register (LFSR) to calculate the Toeplitz matrix. This kind of scheme can be high speed and resource-saving when the compression ratio of PA is low and fixed. When the compression ratio of PA is high or floated, the resource consumption of PA will rapidly increase. The FFT-based PA algorithm implementation isn’t affected by the compression ratio, so it’s easy to implement the wide-range variable compression ratio with our PA scheme. Unlike the LFSR-based PA algorithm, the processing speed of the FFT-based PA algorithm is mainly limited by the memory transfer rate instead of the resource of FPGA. The processing speed of our PA scheme can reach 116Mbps and it can increase sharply with faster memory (e.g. DDR4-DRAM).

| TABLE III: the comparison of several FPGA-based PA schemes |
|-----------------------------------------------------------|
| This Work | Yang et al. [25] | Constantin et al. [24] | Zhang et al. [23] |
| Devices | Xilinx Virtex-6 | Xilinx Virtex-7 | Xilinx Virtex-6 | Cyclone III |
| Length of the final key | 0-1,000,000 | 0-1,000,000 | 0-995,328 | 76,800 |
| LUTs | 37,203 | 26,571 | 15,604 | 1,902 |
| BRAM | 7,092kb | 100kb | 0kb | 656kb |
| Clock frequency | 200MHz | 100MHz | 125MHz | 40MHz |
| Max. processing speed | 116Mbps | 64Mbps | 41Mbps | 0.7Mbps |

5. Conclusions and Outlook

This paper provides a high-speed PA hardware scheme and its implementation in FPGA based on the FFT. The verification is accomplished on the Virtex-6 FPGA and the processing speed of the scheme can reach 116Mbps. Compared with other work, the proposed PA scheme supports wide-range and variable compression ratio and can reach faster processing speed with faster memory. The optimizing proposed in this paper can also improve the FFT-based PA algorithm on other platforms, such as CPU and GPU. In the future, we will research on the relationship between the precision of FFT and the safety of PA and try to replace the floating-point FFT with fixed-point FFT in PA algorithm to reduce the resource consumption.

Acknowledgements

References

[1] C. H. Bennett and G. Brassard, “Quantum cryptography: Public key distribution and coin tossing,” Theoretical Computer Science, vol. 560, no. P1, pp. 7–11, 2014.
[2] D. Stucki, N. Brunner, N. Gisin, V. Scarani, and H. Zbinden, “Fast and simple one-way quantum key distribution,” Applied Physics Letters, vol. 87, no. 19, pp. 1–3, 2005.
[3] V. Scarani, A. Acín, G. Ribordy, and N. Gisin, “Quantum Cryptography Protocols Robust against Photon Number Splitting Attacks for Weak Laser Pulse Implementations,” Physical Review Letters, vol. 92, no. 5, p. 4, 2004.
[4] K. Inoue, E. Waks, and Y. Yamamoto, “Differential phase shift quantum key distribution,” Physical Review Letters, vol. 89, no. 3, pp. 379-021–379-023, 2002.
[5] D. Bruß, “Optimal eavesdropping in quantum cryptography with six states,” Physical Review Letters, vol. 81, no. 14, pp. 3018–3021, 1998.
[6] C. H. Bennett, “Quantum cryptography using any two nonorthogonal states,” Physical Review Letters, vol. 68, no. 21, pp. 3121–3124, 1992.
[7] N. J. Cerf, M. Lévy, and G. Van Assche, “Quantum distribution of Gaussian keys using squeezed states,” Physical Review A, Atomic, Molecular, and Optical Physics, vol. 63, no. 5, pp. 523111–523115, 2001.
[8] F. Grosshans and P. Grangier, “Continuous variable quantum cryptography using coherent states,” Physical Review Letters, pp. 1–6, 2002.
[9] S. Pirandola, S. Mancini, S. Lloyd, and S. L. Braunstein, “Continuous-variable quantum cryptography using two-way quantum communication,” Nature Physics, vol. 4, no. 9, pp. 726–730, 2008.
[10] M. SUN, X. PENG, Y. SHEN, and H. GUO, “Security of a New Two-Way Continuous-Variable Quantum Key Distribution Protocol,” International Journal of Quantum Information, vol. 10, no. 05, p. 1250059, 2012.
[11] S. Wang, W. Chen, J.-F. Guo, Z.-Q. Yin, H.-W. Li, Z. Zhou, G.-C. Guo, and Z.-F. Han, “2 GHz clock quantum key distribution over 260 km of standard telecom fiber,” Optics Letters, vol. 37, no. 6, p. 1008, 2012.
[12] A. Muller, J. Braguet, and N. Gisin, “Experimental Demonstration of Quantum Cryptography Using Polarized Photons in Optical Fibre over More than 1 km,” Europhysics Letters, vol. 23, no. 6, pp. 383–388, 2007.
[13] P. D. Townsend and I. Thompson, “A quantum key distribution channel based on optical fibre,” Journal of Modern Optics, vol. 41, no. 12, pp. 2425–2433, 1994.
[14] C. Z. Peng, J. Zhang, D. Yang, W. B. Gao, H. X. Ma, H. Yin, H. P. Zeng, T. Yang, X. B. Wang, and J. W. Pan, “Experimental long-distance decoy-state quantum key distribution based on polarization encoding,” Physical Review Letters, vol. 98, no. 1, 2007.
[15] C. H. Bennett, G. Brassard, C. Crépeau, U. M. Maurer, and S. M., “Generalized privacy amplification,” Information Theory, IEEE Transactions on, vol. 41, no. 6, pp. 1915–1923, 1995.
[16] C. H. Bennett, G. Brassard, and J.-M. Robert, “Privacy Amplification by Public Discussion,” SIAM Journal on Computing, vol. 17, no. 2, pp. 210–229, 1988.
[17] R. Impagliazzo, L. a. Levint, and M. Luby, “Pseudo-random generation from one-way functions ( Extended Abstract ),” STOC ‘89 Proceedings of the twenty-first annual ACM symposium on Theory of computing, pp. 12–24, 1989.
[18] M. N. Wegman and J. L. Carter, “New hash functions and their use in authentication and set equality,” Journal of Computer and System Sciences, vol. 22, no. 3, pp. 265–279, 1981.
[19] J. L. J. Carter and M. M. N. M. Wegman, “Classes of Hash Functions,” Journal of computer and system sciences, vol. 18, pp. 143–154, 1979.
[20] R. Y. Cai and V. Scarani, “Finite-key analysis for practical implementations of quantum key distribution,” New Journal of Physics, vol. 11, pp. 1–17, 2009.
[21] C. M. Zhang, M. Li, J. Z. Huang, H. W. Li, F. Y. Li, C. Wang, Z. Q. Yin, W. Chen, Z. F. Han, P. Treeviyanupab, and K. Sriphranwat, “Fast implementation of length-adaptive privacy amplification in quantum key distribution,” Chinese Physics B, vol. 23, no. 9, pp. 1–6, 2014.
[22] H. Krawczyk, “LFSR-based Hashing and Authentication,” Advances in Cryptology CRYPTO ’94, vol. 10598, pp. 129–139, 1994.
[23] H. F. Zhang, J. Wang, K. Cui, C. L. Luo, S. Z. Lin, L. Zhou, H. Liang, T. Y. Chen, K. Chen, and J. W. Pan, “A real-time QKD system based on FPGA,” Journal of Lightwave Technology, vol. 30, no. 20, pp. 3226–3234, 2012.
[24] J. Constantin, R. Houlmann, N. Preyss, N. Walenta, H. Zbinden, P. Junod, and A. Burg, “An FPGA-Based 4 Mbps Secret Key Distillation Engine for Quantum Key Distribution Systems,” Journal of Signal Processing Systems, vol. 86, no. 1, pp. 1–15, 2017.
[25] 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 Journal, vol. 9, no. 6, 2017.
[26] B. Liu, B. Zhao, W. Yu, and C. Wu, “Fit-PA: Fixed scale FFT based privacy amplification algorithm for quantum key distribution,” Journal of Internet Technology, vol. 17, no. 2, pp. 309–320, 2016.
[27] Xiangyu Wang, Yichen Zhang, Song Yu and H. Guo, “High-Speed Implementation of Privacy Amplification in Quantum Key Distribution," vol. xx, no. xx, pp. 1–10, 2016.
[28] V. Scarani and R. Renner, “Quantum cryptography with finite resources: Unconditional security bound for discrete-variable protocols with one-way postprocessing,” Physical Review Letters, vol. 100, no. 20, pp. 1–4, 2008.
[29] T. Ururamaru, M. Hayashi, and S. Member, “Dual Universality of Hash Functions and Its Applications to Quantum Cryptography,” vol. 59, no. 7, pp. 4700–4717, 2013.
[30] B. Lerner, “Parallel implementation of fixed-point ffts on tigersharc processors,” IEEE Transactions on Signal Processing, vol. 35, no. 6, pp. 849–863, 1987.
[31] W. U. Qinwen, “High Efficiency Matrix Transposition Method Based on FPGA and DDR,” 2017.
[32] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus, “Real-Valued Fast Fourier Transform Algorithms,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 35, no. 6, pp. 849–863, 1987.