A Standardized Design for Sifting in Quantum Key Distribution Software

Omar Amer*‡ Vaibhav Garg† Walter O. Krawec‡

*Global Technology Applied Research
JPMorgan Chase Bank, N.A
New York, NY 10017 USA
omar.amer@jpmchase.com

†Comcast Cable
Philadelphia, PA, USA 19103
vaibhav_garg@comcast.com

‡University of Connecticut
Department of Computer Science and Engineering
Storrs, CT, USA 06269
walter.krawec@uconn.edu

***Abstract*—With 5G communication networks enjoying commercial adoption, work has begun towards defining future 6G networks. The emergence of commercially viable quantum communication technologies offer these future communication networks information-theoretic key agreement techniques enabling provably quantum resistant secure communications through quantum key distribution protocols. However, commercial quantum key distribution systems rely on independent software implementations of key agreement protocols. Thus, while these systems may be mathematically secure they may incur security vulnerabilities in software. To address this we design a standardized sifting module, which can be used to generalize the description and execution of the necessary sifting in most if not all discrete variable quantum key distribution protocols, including entanglement based protocols. We give a general description of the software stack underlying implementations of the post-processing stage of quantum key distribution protocols, including our formalization of the two-way sifting stage of said protocols and the machinery required to realize it in a completely general manner. Finally, we give some example configurations for some common quantum key distribution protocols.**

I. Introduction

Since the discovery of the BB84 protocol [1] the theory and practice of quantum key distribution (QKD) has seen significant development. On the side of theory we have seen, among other impressive advances, the development of a variety of rigorous methods for proving the security of QKD protocols [2, 3, 4, 5], the design of protocols using increasingly limited resources [6, 7, 8], the advancement of security models and protocols that remain secure even when the hardware implementing the protocols is untrusted [9, 10, 11] and the development of quantum key distribution networks making use of trusted nodes and quantum repeaters [12, 13]. On the side of practical QKD, we have of course seen the experimental implementation of many of the protocols discussed above, commercial implementations [14], and deployments of said protocols at industry scale and in a variety of contexts [15, 16, 17]. For a detailed surveys of QKD, we direct readers to [18, 14].

While the effort to standardize and deploy 5G communication technologies began too soon to include QKD and other quantum communication technologies, the field is now mature enough to warrant consideration for use in future 6G communication network [19, 20]. Indeed, in light of increasingly powerful quantum computers and improved quantum cryptanalysis techniques [21, 22, 23] as well as the ever-present possibility of both classical and quantum cryptanalysis attacks against "post-quantum cryptography" protocols, the ability to base security on non-complexity-theoretic assumptions, as QKD does on quantum mechanical principles, becomes increasingly relevant. As the technology continues to mature, with significant efforts towards miniaturizing QKD hardware [24, 25, 26, 27] and extending the reach of QKD protocols through both terrestrial [28, 29, 30] and free-space means [31, 32, 33], QKD becomes an increasingly attractive option for securing 6G communication networks.

However, while QKD offers security from a mathematical perspective, practical systems must also contend with vulnerabilities in software. As commercial solutions implement QKD protocols independently it is possible that security vulnerabilities may creep in. These vulnerabilities can potentially decrease confidence in the security of QKD solutions and thereby hinder adoption. One solution is to implement an open source standardized implementation for QKD protocols similar to OpenSSL for the TLS protocol. As with OpenSSL the security of this standardized software implementation can be assured by third party security researchers.

In this article, we report on our efforts to design a standardized software process to *sift* the measurement data from a quantum key distribution protocol into the the necessary statistics and raw key data for the classical post processing stages using two-way classical communication. We note that the method reported is general enough to work with most, if not all, discrete-variable QKD protocols, including entanglement based protocols. Indeed similar ideas likely could be extended to continuous-variable protocols as well. In [34] we discussed a modular software stack for conducting the post processing necessary for quantum key distribution protocols, in which the majority of the modules could be used for arbitrary protocols. The major protocol specific module in our stack was the sifting module. Our work in this article then is towards making that software, and other software like it, easily translatable for use with other QKD protocols. This would allow researchers to more rapidly test different QKD protocols, and to better understand the similarities of different QKD protocols. Moreover, this design may even allow for more flexible deployment of QKD protocols in 6G communication environments through the use of a standardized software stack.

II. Background

The standardization of QKD protocols is by no means a novel concept — indeed there has recently been significant effort into the description and analysis of QKD protocols in a generalized fashion. We begin the description of our standardized sifting module by the exposition of the necessary context — the general QKD protocol. As a foundation we borrow a very useful formulation of the general QKD protocol used in [35], although we modify it in some places.

For our purposes, a QKD protocol contains within it seven sub-routines. The first two stages deal with the truly quantum stages of the protocol, whereas the latter five describe the classical post-processing steps. For simplicity we assume that the protocol only involves Alice and Bob, but our method could be extended to include parties which use an untrusted third party as well. We describe the seven steps below.

1) *State preparation and transmission:* Alice prepares an entangled quantum state $\rho_{AB}^{\otimes N}$ and sends Bob's systems to him.
2) *Measurement and data partitioning:* Alice and Bob measure each of their N entangled quantum states according to the specific protocol. For each measurement, indexed by i, they partition their measurement information into two data sets A_i^s, B_i^s and A_i^p, B_i^p, denoting the information that will remain private, and the information they will later announce publicly.
3) *Parameter estimation:* Alice and Bob announce both the public and private data for some random subset of rounds of size m in order to verify that the statistics fall within some previously agreed upon accepted set of statistics, otherwise they abort the protocol.
4) *Sifting:* Alice and Bob announce their public information, using it to determine which subset of the $N - m$ rounds should be used to arrive at their raw keys rk_a and rk_b. Alice and Bob each compute a mapping from those subsets of their own private information and the public information to obtain the raw keys rk_a and rk_b. The remaining private data, outside of the subset determined, is made public. This newly public data may be used to update the parameters arrived at in the previous step, which can be useful, for example, in the case of protocols which make use of mismatched measurement information [36, 8].
5) *Error correction:* Alice and Bob publicly engage in an information reconciliation protocol to attempt to reconcile errors between Alice's raw key rk_a and Bob's raw key rk_b, with Bob ultimately arriving at k_b, which matches rk_a with high probability.
6) *Key Rate Computation:* Using the statistics arrived at in the parameter estimation (potentially updated after the sifting step) and the amount of information leaked during error correction, Alice determines how many secret key bits can be extracted from the raw key.
7) *Privacy amplification:* Alice and Bob finally arrive at their secret keys by applying a two-universal hash function on rk_a and k_b, resulting in sk_a and sk_b.

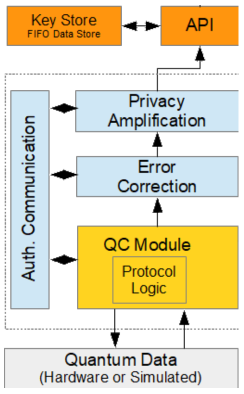

Fig. 1. (From [34]) The modules of the post-processing software implemented in [34], where the QC module includes both the key-rate equation and sifting process which are the only two protocol specific processes.

We note that for clarity we allow parameter estimation to act as its own stage, although in practice the sifting process can handle the totality of the statistics disclosure, and the key rate computation can determine whether or not the parties abort.

In [34] we describe a modular implementation, shown in Figure 1 of the four post-processing steps, for use with real quantum key distribution implementations. In that work, all stages save for the sifting and parameter estimation are generalized, and can be applied to all QKD protocols. In this work we provide a generalized sifting module design, and indeed the work towards generalized parameter estimation and key rate computation by numerical methods in [35] may allow us to move even closer to a truly general QKD implementation.

III. Design

We provide now our design for the standardized sifting module, as it fits into the context of the general QKD protocol. We design the module so that Alice and Bob should run their own modules independently, with the two communicating at set times and in pre-set manners throughout the process. Alice's module takes as input her preparation choices, as well as the results of any measurements she makes, while Bob's similarly takes as input his choices and measurement outcomes. We configure the sifting module for a given protocol with three λ-functions for each Alice and Bob, $(\lambda_p^x, \lambda_s^x, \lambda_k^x)$ for $x \in \{a, b\}$ denoting Alice and Bob's descriptions, respectively. The first set of functions parse the input data into the necessary information for sifting – the "public" data for both parties; The second set uses the public data to determine which rounds will contribute to the raw key; and finally the third partitions the data into the raw key and the discarded, now public data. In this section we will formalize the input and output of the three stages, as well as the input and output of the total module. The first stage we name the parsing stage, in which the two parties parse the data for a round into private and public data, sending the public data over the public communication channel. In the second stage of the pipeline, Alice and Bob conduct the actual sifting, deciding based on their public and private inputs if they should keep the round for key distillation, and doing so only if both agree on the public channel. In this stage, data that is not kept may also be used to update various statistics to be used for parameter estimation and the key rate computation in other parts of the post-processing pipeline. Finally, in the third round, they apply their respective key mapping functions to transform the data from kept iterations into their respective raw keys.

Throughout this section, we will motivate our design based on the BB84 [1] protocol, and later we will provide additional configurations for other protocols. For clarity, we include a brief description of the BB84 protocol now. In the BB84 protocol, Alice selects two random bits on each round, θ_a, x_a and sends to Bob the state $H^{\theta_a} |x_a\rangle$ where H is the Hadamard operator. Bob chooses his own random bit θ_b and applies the operator H^{θ_b} to the received state, before measuring in the computational basis and seeing output x_b (equivalently he measures in the basis dictated by θ_b). Alice and Bob publicly disclose their basis choices and keep as the raw key their prepared and measured states from rounds in which their basis choices agreed.

A. Input

We refer to these data sets moving forward as Alice and Bob's quantum data, and we denote them respectively as A and B, with A_i and B_i denoting the data from round i. Without loss of generality, we assume that the data consists of N l-tuples from some finite alphabet $\mathcal{A}$. That is to say that each round produces a set of data in $D = \mathcal{A}^l$ and the entire protocol produces a data-set in D^N. Of course for many protocols, Alice and Bob do not have identical data distributions, but without loss of generality we can assume that they belong to the same set. Each round of the protocol produces $A_i, B_i \in D$ and the protocol as a whole produces the two data sets $A = \prod_i^N A_i$ and $B = \prod_i^N B_i$.

In the case of BB84, Alice and Bob's input data is symmetric. For a given round, Alice has two bits of information representing her basis choice and prepared state, which we denote by the tuple $A_i = (\theta_{a_i}, x_{a_i})$, both elements of $\{0, 1\}$. Bob's is much the same, with $B_i = (\theta_{b_i}, x_{b_i})$ denoting his basis choice and observed state.

B. Parsing

For the purposes of our discussion, we assume that each round's data is parsed independently of the data from other rounds. For some protocols, such as the COW protocol [37], this may not be the case, but our method should easily extend to consider blocks of data, or indeed the entire data-set as a whole. Whatever the block size used, we iterate the parsing, sifting, and key mapping stage N/n times, where n is the sifting block size (again we take n to be one for BB84 and many common protocols).

For each protocol then, we let λ_p^x be the protocol specific function which determines what information is made public by party x in order to determine how to sift the information. More formally:

$$\lambda_p^x : (\mathcal{A}^l)^n \mapsto \mathcal{A}^k,$$

where the input to this stage is the data from some n rounds and the output is k symbols of data from processed data, where $k \leq l * n$. The output of this stage from each party is sent to the other party, for input into the next stage. In practice, it will often be sufficient for only one party to communicate data to the other party on this stage, in which case the non-communicative party's function would be trivial.

For BB84, on each round Alice's λ-function outputs $A_p = \lambda_p^a(\theta, x) := \theta$ and Alice sends this value to Bob. Bob's is trivial, and outputs $B_p = \lambda_p^b(\theta, x) = \perp$, which he sends to Alice.

C. Sifting

This stage determines, based on the output of the last stage, what action to take for the data from each round, of which there are two possible. Each round's data can either be kept, for use in the next stage in arriving at a raw key, or it can be discarded, in which case it can be made public without harm to the security of the protocol, and used to determine the various statistics necessary for parameter estimation. As input, this stage takes the full data for one party, and the publicized data of the other. As output, it simply outputs a Boolean value for the data from each round. Formally, for a block of n rounds:

$$\lambda_s^x : (\mathcal{A}^l)^n \times \mathcal{A}^k \mapsto \{0, 1\}^n.$$

Both parties send the other the output from this stage, keeping a line if both agree to keep it, and discarding it otherwise. Once again, for many protocols, for one party this function may be trivial — in which case they would always output 1, and the action would be determined by the other party. Moreover, this stage of the sifting module may be used to determine which rounds, that may have otherwise been used to contribute to the raw key, will instead be published and discarded. This discarded information is used to update the statistics of each of the possible events in the protocol and the accompanying sample size information for use in computing confidence intervals.

In BB84 it is this stage in which Alice's λ-function in this stage is trivial, outputting and sending to Bob $A_s = \lambda_s^a(A, B_p) := 1$. Bob's λ-function is simply $B_s = \lambda_s^b(B, A_p) := \theta == A_p$. Both parties announce the output of this stage, proceeding to the next stage if $A_s \wedge B_s$, and publishing A and B otherwise for potential use in parameter estimation and advancing to parsing stage of the next round otherwise.

D. Key Mapping

In the final stage of the sifting module, the parties transform their un-discarded data into their respective raw keys, potentially using the public data disclosed by the other in the process. Formally, this processing is of the form:

$$\lambda_k^x : (\mathcal{A}^l)^n \times \mathcal{A}^k \mapsto \{0,1\}^{r(n)}$$

where $r(n)$ is a function dictating how many bits of raw key can be generated from n rounds of the protocol, and n is again the block size. This stage may happen as part of the ongoing loop, alongside the parsing and sifting, or it may occur once the entire key has been sifted, after the N rounds have been otherwise processed.

For BB84 Alice and Bob's λ-functions in this stage are again both simple, and in this case identical. Party p arrives at their raw key bit using $rk_p = \lambda_k^p(\theta, x) := x$.

E. Output

The output of the module, after the N rounds are completed, is the raw key string output by the Key Mapping stage, and the count of the various discarded events, output by the Sifting module. The raw key, of course, takes the form of $x \in \{0,1\}^{r(n)}$, while the statistics and sample sizes output are a subset of $[0,1]^{lk/n} \times [0,N]^{lk/n}$ where lk/n is an upper-bound of the number of possible events, in terms of disparate private and public choices and outcomes, that can occur. These statistics, gathered from phases that are not used to generate key-material, are crucial for the key rate computation necessary to ensure the privacy of the final, secret key.

IV. Case Studies

In this section we showcase the configuration of our framework for two additional protocols in order to demonstrate the versatility of our approach. Previously we gave a detailed walk-through the process for BB84, but in the interest in brevity we give only the specification for the other protocols, except when further explanation is necessary. Below, where the party to whom the λ-function defined belongs to is exceedingly clear, we may omit the superscript denoting ownership. We use $x == y$ to denote the equality of two symbols, x and y. We note that, in all cases discussed below and indeed in most QKD protocols, the sifting stage is a relatively cheap procedure in terms of computational cost — consisting mainly of classical communication and string processing. Indeed it is mainly the error correction and privacy amplification stages which can have non-trivial computational cost depending on the block-sizes, and therefore usually key-rates, desired. Similarly, where there may be failure probabilities associated with error correction and privacy amplification, in these examples sifting is an inherently resilient process. One notable yet unavoidable requirement is the requirement to store the quantum data and sifted key information — which of course linearly with the number of rounds. In most cases, it is possible to engage in block-wise sifting to periodically parse and discard extraneous quantum data and reduce overall storage requirements.

A. Extended B92 [38]

Protocol Description: The original B92 protocol [39] makes use of only two non-orthogonal states in order for Alice and Bob to establish a secret key. Here we give a sifting formulation for the extended variant, in which a second computational state is used in order to achieve better key rates. Alice prepares one of the two computational states or the third, non-orthogonal state and sends it to Bob. Bob measures randomly in either the computational basis or a basis containing the non-orthogonal state. Alice discloses to Bob only whether or not she sent the $|1\rangle$ state — when she doesn't, the round may result in a key round. Bob informs Alice whether or not his measurement result was "conclusive" i.e. whether or he observed one of the states which she does not send on a key round. On a conclusive key round, Bob may infer which basis Alice chose, constituting their raw key bit.
Data Format: Alice's data consists of simply the value $A_i = S_i \in \{0,1,2\}$ where a value of 0 or 1 denotes her preparing the corresponding computational state, and 2 denotes her preparing the third, non-orthogonal state. Bob's consists of two bits $B_i = (\theta_i, x_i)$ denoting his basis choice and measurement outcome.
Parsing: Alice sends $A_p = \lambda_p(A_i) := S_i \neq 1$ to Bob; Bob's function and message are again trivial.
Sifting: Bob sends $B_s = \lambda_s(A_p, B_i) := A_p \wedge (\theta_i \neq x_i)$. Alice's function is trivial.
Key Mapping: Alice's key mapping function outputs $rk_a = \lambda_k^a(A_i) := S_i/2$ while Bob's outputs $rk_b = \lambda_k^b(B_i) := 1 - \theta_i$

B. AK19 [8]

Protocol Overview: Here we give the configuration for sifting of a high noise-tolerance semi-quantum key distribution protocol. This protocol is unique (among the ones we discuss) in that Alice makes two measurements, due to the fact that semi-quantum protocols regularly use a two-way quantum channel, and also in that Alice may use three bases. Briefly, Alice chooses one of her possible bases, and a state from that basis, and sends it to Bob. Bob chooses to either measure the state in the computational basis, and send the result (as a new quantum state) to Alice, or reflect the state he received back to

	Alice	Bob
X_i	(θ_{a_i}, x_{a_i})	(θ_{b_i}, x_{b_i})
$\lambda_p^X(X_i)$	θ	$\perp$
$\lambda_s^X(X_i, Y_p)$	1	$\theta == A_p$
$\lambda_k^X(X_i, Y_p)$	x_{a_i}	x_{b_i}

TABLE I
Summarizing data format and functions for sifting in the BB84 protocol, where for each party X_i denotes their personal data and Y_p denotes the public data received.

	Alice	Bob
X_i	$S_i \in \{0,1,2\}$	(θ_{b_i}, x_{b_i})
$\lambda_p^X(X_i)$	$S_i \neq 1$	$\perp$
$\lambda_s^X(X_i, Y_p)$	1	$A_p \wedge (\theta_i \neq x_i)$
$\lambda_k^X(X_i, Y_p)$	$S_i/2$	$1 - \theta_i$

TABLE II
Summarizing data format and functions for sifting in the B92 protocol.

Alice. Alice again chooses a random basis, and measures the returned state in that basis. Candidate key-rounds are those in which Alice chose the Z basis in both instances, observed the same state she sent, and in which Bob chose to measure and resend. With some probability, key rounds are still discarded for testing.

Data Format: In this protocol, Alice's data consists of the tuple $A_i = (b_1, k_a, b_2, r, t)$ with $b_1, b_2 \in \{0, 1, 2\}$ denoting the basis choices of Alice when she prepares the qubit and when she measures the returned qubit, $k_a \in \{0, 1\}$ denoting the state prepared, r denoting the state she measured when the qubit is returned to her, and t denoting whether or not this round will be used for testing purposes. Bob's data is simpler, consisting only of $B_i = (c, k_b)$ where $c, k_b \in \{0, 1\}$ denote whether or not Bob choose to measure or reflect and the result of a measurement, respectively. For simplicity, we omit round indices in the labeling of these variables description, and due to length constraints we omit a summarizing table for this protocol.

Parsing: In this protocol, neither Alice nor Bob need make public any information in this stage.

Sifting: Alice sends $A_s = \lambda_s^a(A) := (b_1 == b_2 == 0) \wedge (k_a == r) \wedge \neg t)$ Bob sends $B_s = \lambda_s^b(B) := c$

Key Mapping: Alice's key mapping function is $rk_a = \lambda_k^a(A_i) := k_a$ while Bob's is $rk_b = \lambda_k^b(B_i) := k_b$

V. Closing Remarks

In this article we give a simple solution for the implementation of general sifting modules that may be reused, with little configuration effort, for most QKD protocols. We demonstrated it's versatility by giving simple formulations for a number of different QKD protocols, including one semi-quantum protocol. The integration of this design into existing QKD software stacks would allow for the rapid adaptation of the software for use with novel QKD protocols, greatly simplifying the testing workflow in practical QKD research efforts. This flexibility will play an important role in making QKD software more accessible for quantum-enabled 6G communication research, design, and implementation efforts, and this design may even assist in ensuring that the software in eventual QKD deployments in 6G communication environments can be easily adapted for use with different underlying QKD protocols.

WOK would like to acknowledge support from the National Science Foundation under grant number 2143644.

Disclaimer

References

[1] Charles H Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*. Vol. 175. New York. 1984.

[2] Renato Renner. "Security of quantum key distribution". In: *International Journal of Quantum Information* 6.01 (2008), pp. 1–127.

[3] Niek J Bouman and Serge Fehr. "Sampling in a quantum population, and applications". In: *Annual Cryptology Conference*. Springer. 2010, pp. 724–741.

[4] Keegan Yao, Walter O. Krawec, and Jiadong Zhu. *Quantum Sampling for Optimistic Finite Key Rates in High Dimensional Quantum Cryptography*. 2020. arXiv: 2012.04151 [quant-ph].

[5] Omar Amer and Walter O Krawec. "High-Dimensional Quantum Conference Key Agreement". In: *arXiv preprint arXiv:2202.00140* (2022).

[6] Wei Zhang et al. "A single-state semi-quantum key distribution protocol and its security proof". In: *arXiv preprint arXiv:1612.03087* (2016).

[7] Michel Boyer, Dan Kenigsberg, and Tal Mor. "Quantum Key Distribution with Classical Bob". In: *Phys. Rev. Lett.* 99 (14 Oct. 2007), p. 140501. DOI: 10.1103/PhysRevLett.99.140501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.99.140501.

[8] Omar Amer and Walter O. Krawec. "Semiquantum key distribution with high quantum noise tolerance". In: *Phys. Rev. A* 100 (2 Aug. 2019), p. 022319. DOI: 10.1103/PhysRevA.100.022319. URL: https://link.aps.org/doi/10.1103/PhysRevA.100.022319.

[9] Hoi-Kwong Lo, Marcos Curty, and Bing Qi. "Measurement-device-independent quantum key distribution". In: *Physical review letters* 108.13 (2012), p. 130503.

[10] Antonio Acın et al. "Device-independent security of quantum cryptography against collective attacks". In: *Physical Review Letters* 98.23 (2007), p. 230501.

[11] Wei Zhang et al. "A device-independent quantum key distribution system for distant users". In: *Nature* 607.7920 (July 2022), pp. 687–691. DOI: 10.1038/s41586-022-04891-y.

[12] Piotr K Tysowski et al. "The engineering of a scalable multi-site communications system utilizing quantum key distribution (QKD)". In: *Quantum Science and Technology* 3.2 (2018), p. 024001.

[13] Omar Amer, Walter O Krawec, and Bing Wang. "Efficient routing for quantum key distribution networks". In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE. 2020, pp. 137–147.

[14] Omar Amer, Vaibhav Garg, and Walter O Krawec. "An Introduction to Practical Quantum Key Distribution". In: *IEEE Aerospace and Electronic Systems Magazine* 36.3 (2021), pp. 30–55.

[15] Teng-Yun Chen et al. "Metropolitan all-pass and intercity quantum communication network". In: *Optics express* 18.26 (2010), pp. 27217–27225.

[16] Chip Elliott. "The DARPA quantum network". In: *arXiv preprint quant-ph/0412029* (2004, unpublished).

[17] Farzam Toudeh-Fallah et al. "Paving the Way towards 800 Gbps Quantum-Secured Optical Channel Deployment in Mission-Critical Environments". In: *arXiv preprint arXiv:2202.07764* (2022).

[18] Christopher Portmann and Renato Renner. "Security in quantum cryptography". In: *Rev. Mod. Phys.* 94 (2 June 2022), p. 025008. DOI: 10.1103/RevModPhys.94.025008. URL: https://link.aps.org/doi/10.1103/RevModPhys.94.025008.

[19] Chonggang Wang and Akbar Rahman. "Quantum-Enabled 6G Wireless Networks: Opportunities and Challenges". In: *IEEE Wireless Communications* 29.1 (2022), pp. 58–69.

[20] Hasan Abbas Al-Mohammed and Elias Yaacoub. "On the use of quantum communications for securing IoT devices in the 6G era". In: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE. 2021, pp. 1–6.

[21] C. Ryan-Anderson et al. *Implementing Fault-tolerant Entangling Gates on the Five-qubit Code and the Color Code*. 2022. DOI: 10.48550/ARXIV.2208.01863. URL: https://arxiv.org/abs/2208.01863.

[22] Andrew W Cross et al. "Validating quantum computers using randomized model circuits". In: *Physical Review A* 100.3 (2019), p. 032328.

[23] Élie Gouzien and Nicolas Sangouard. "Factoring 2048-bit rsa integers in 177 days with 13 436 qubits and a multimode memory". In: *Physical Review Letters* 127.14 (2021), p. 140503.

[24] Gwenaelle Mélen et al. "Integrated quantum key distribution sender unit for daily-life implementations". In: *Advances in Photonics of Quantum Computing, Memory, and Communication IX*. Vol. 9762. SPIE. 2016, pp. 31–36.

[25] Taofiq K Paraıso et al. "A modulator-free quantum key distribution transmitter chip". In: *npj Quantum Information* 5.1 (2019), pp. 1–6.

[26] Gwenaelle Vest et al. "Design and evaluation of a handheld quantum key distribution sender module". In: *IEEE journal of selected topics in quantum electronics* 21.3 (2014), pp. 131–137.

[27] Hyunchae Chun et al. "Handheld free space quantum key distribution with dynamic motion compensation". In: *optics express* 25.6 (2017), pp. 6784–6795.

[28] Yang Liu et al. "Experimental twin-field quantum key distribution through sending or not sending". In: *Physical Review Letters* 123.10 (2019), p. 100505.

[29] Philip A Hiskett et al. "Long-distance quantum key distribution in optical fibre". In: *New Journal of Physics* 8.9 (2006), p. 193.

[30] Jiu-Peng Chen et al. "Sending-or-not-sending with independent lasers: Secure twin-field quantum key distribution over 509 km". In: *Physical review letters* 124.7 (2020), p. 070501.

[31] Sheng-Kai Liao et al. "Satellite-relayed intercontinental quantum network". In: *Physical review letters* 120.3 (2018), p. 030501.

[32] Robert Bedington, Juan Miguel Arrazola, and Alexander Ling. "Progress in satellite quantum key distribution". In: *npj Quantum Information* 3.1 (2017), pp. 1–13.

[33] M Avesani et al. "Full daylight quantum-key-distribution at 1550 nm enabled by integrated silicon photonics". In: *npj Quantum Information* 7.1 (2021), pp. 1–8.

[34] Omar Amer et al. "A Modular Quantum Key Distribution Software Stack for Rapid Experimental Prototyping". In: *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 2021, pp. 467–468. DOI: 10.1109/QCE52317.2021.00081.

[35] Ian George, Jie Lin, and Norbert Lütkenhaus. "Numerical calculations of the finite key rate for general quantum key distribution protocols". In: *Phys. Rev. Research* 3 (1 Mar. 2021), p. 013274. DOI: 10.1103/PhysRevResearch.3.013274. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.3.013274.

[36] Walter O Krawec. "Quantum Key Distribution with Mismatched Measurements over Arbitrary Channels". In: *Quantum Information and Computation* 17.3 and 4 (2017), pp. 209–241.

[37] Damien Stucki et al. "Fast and simple one-way quantum key distribution". In: *Applied Physics Letters* 87.19 (2005), p. 194108.

[38] Marco Lucamarini, Giovanni Di Giuseppe, and Kiyoshi Tamaki. "Robust unconditionally secure quantum key distribution with two nonorthogonal and uninformative states". In: *Phys. Rev. A* 80 (3 Sept. 2009), p. 032327. DOI: 10.1103/PhysRevA.80.032327. URL: https://link.aps.org/doi/10.1103/PhysRevA.80.032327.

[39] Charles H. Bennett. "Quantum cryptography using any two nonorthogonal states". In: *Phys. Rev. Lett.* 68 (21 May 1992), pp. 3121–3124. DOI: 10.1103/PhysRevLett.68.3121. URL: http://link.aps.org/doi/10.1103/PhysRevLett.68.3121.