Eur. Phys. J. Plus (2024) 139:602
https://doi.org/10.1140/epjp/s13360-024-05337-2

THE EUROPEAN PHYSICAL JOURNAL PLUS

Regular Article

Simulation of an entanglement-based quantum key distribution protocol

L. Mariani[1,a], L. Salatino[1], C. Attanasio[1,2], S. Pagano[1,2,3], R. Citro[1,2,3]

[1] Department of Physics "E. R. Caianiello", University of Salerno, Via Giovanni Paolo II 132, 84014 Fisciano, SA, Italy
[2] CNR - SPIN, c/o University of Salerno, Via Giovanni Paolo II 132, 84014 Fisciano, SA, Italy
[3] INFN, Gruppo collegato di Salerno, Via Giovanni Paolo II 132, 84014 Fisciano, SA, Italy

Received: 3 January 2024 / Accepted: 1 June 2024

Abstract Since the 80 s when it was first proposed, Quantum Key Distribution (QKD) elicited great interest in the field of cryptography as a unique procedure for key generation that could in principle guarantee unconditionally secure communication "by the laws of Physics". In the last fifteen years commercial solutions have started appearing on the market, showing that practical implementations of the protocol were not only possible but also competitive in terms of security and achievable secret-key rate. In this work we describe a simulation of the historical QKD protocol E91 on the *IBM Quantum* platform, making use of the qubit formalism to represent the quantum states received by two communicating nodes. Having implemented also the post-processing steps for the error correction and the privacy amplification, this model can represent a simple stand-alone tool to study the performance not only of one-to-one communication but of more complex systems that rely on QKD for security, one above all QKD networks.

1 Introduction

Quantum Key Distribution (QKD) made its first appearance in the cryptography landscape in 1984, when the first protocol was developed by Bennett and Brassard [1] upon an original idea of Wiesner [2]. Its main claim was to provide a method for generating a key shared between two parties (conventionally named Alice and Bob) that is unconditionally secure by the laws of Physics. This means that—assuming that the predictions of quantum Physics are true—any third undesired party eavesdropping on the key exchange could be detected with any degree of confidence, regardless of the computational power at their disposal. For this reason QKD is considered to be a resource against the so-called *quantum threat*: the public-key cryptography schemes that currently represent the standard in information security (such as RSA [3]) rely on the computational complexity of specific mathematical problems, and could be broken by quantum computers with sufficient computational power [4–10].

In 1991 Arthur Ekert designed independently from Bennett and Brassard an alternative protocol [11], which requires the exchange of particles bound by quantum entanglement and makes use of the CHSH Bell's inequality (named after Clauser, Horne, Shimony and Holt who proposed it [12]) to detect the possible presence of eavesdroppers. This approach opened the way for a number of entanglement-based protocols, in which—as opposed to prepare-and-measure protocols such as BB84—the value of the bit that will possibly be part of the shared key is not encoded in the travelling quantum state, but is defined only after the measurement by Alice and Bob: this eliminates a number of vulnerabilities that instead may affect the experimental setup in prepare-and-measure protocols. It must be noted though that compared to these (a) a more complex apparatus is needed in order to work with entanglement and (b) the secret-key rate is subject to a more severe decay as the distance between Alice and Bob increases.

Quantum computing is another technology that took big steps forward in the last decade. Among the several companies that invested in the field, IBM stands out: in addition to carrying out cutting-edge research both hardware- and software-wise [13–17], they launched in 2016 the online platform *IBM Quantum* (IBM Q), that allows the general public to execute programs on the company's quantum computers [18]. Along with this, IBM developed and released in 2017 the SDK *Qiskit*, an open-source set of tools based on the Python language, with the purpose of simplifying the interface with such machines. Qiskit also gives the possibility to run these quantum programs—or quantum circuits—on a range of noiseless classical simulators and compose them with some noise models that can be conveniently tuned by means of the component `qiskit.Aer`. The simulators firstly display the possibilities that a quantum computer will be offering with the improvement of error correction (currently the error rate on a real machine is of the order of the percent). Secondly, they are a useful tool for the designing step of quantum algorithms, allowing to run the programs locally on a personal computer and with less limiting constraints in the number of available qubits.

The purpose of this work is to describe an implementation of the E91 protocol, developed on IBM Q exploiting the `qasm_simulator` backend, which allows for building circuits consisting of a maximum of 32 qubits (see https://quantum-

[a] e-mail: lmariani@unisa.it (corresponding author)

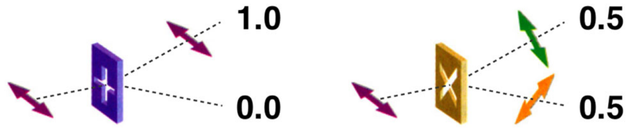


Fig. 1 Schematization of a photon hitting a beamsplitter with angle $\theta = 0$ (left) and $\theta = \pi/4$ (right), along with the probabilities of each path. Adapted from [19]

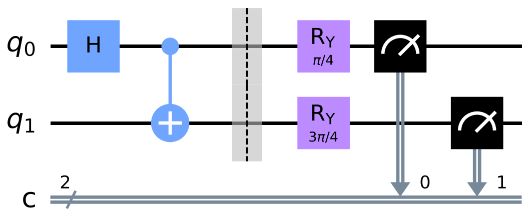


Fig. 2 One of the quantum circuits used to simulate the E91 protocol. In this instance the R_y gates (after the dashed line) applied to the entangled 2-qubit state correspond to the polarization angles $\theta_A = \pi/8, \theta_B = 3\pi/8$

computing.ibm.com/lab/docs/iql/manage/simulator) against the 7 qubits of the quantum devices with open access (https://www.ibm.com/quantum/access-plans). Having programmed also on the same platform the *information reconciliation* and *privacy amplification* steps, we were able to analyse the performance of the protocol under different conditions. The choice of IBM Q was driven by some of its valuable features: it provides a simple interface and a compact formalism to work with qubits and to visualize the results of a quantum circuit; being implemented in a high-level programming language such as Python, it makes it possible to keep the script compact and manageable. Last but not least, being by now a mainstream resource for quantum computing it can rely on a vast global community and an active research team for debugging and keeping the platform alive and up-to-date. In the next section we will shortly describe the quantum part of the original E91 protocol and how it translates to a quantum circuit. We will then address the post-processing, namely two different error-correction algorithms—that we compared in terms of efficiency and performance—and the privacy amplification step. We will report and discuss the results in terms of the secret fraction and the secret key rate that were achieved while varying the length of the keys and the noise level affecting the quantum channel. Finally we will explain possible future developments and use cases of this work.

2 Methods

2.1 E91 protocol

The E91 protocol [11] establishes that pairs of entangled particles (let us consider photons) are emitted from a source that is not necessarily trusted, then separated and singularly measured by Alice and Bob. The portion of those pairs whose polarization is measured in the same basis by the two parties gives perfectly correlated results, that become bits of the shared *raw keys*. Of the discarded bits, part is used to verify the violation of the CHSH inequality (1), as a proof that the sent qubits were actually entangled and had not been somehow counterfeited:

$$|S| > 2 \quad , \quad S = E(\vec{a}_1, \vec{b}_1) - E(\vec{a}_1, \vec{b}_3) + E(\vec{a}_3, \vec{b}_1) + E(\vec{a}_3, \vec{b}_3) \tag{1}$$

where $E(\vec{a}_i, \vec{b}_j)$ is defined as:

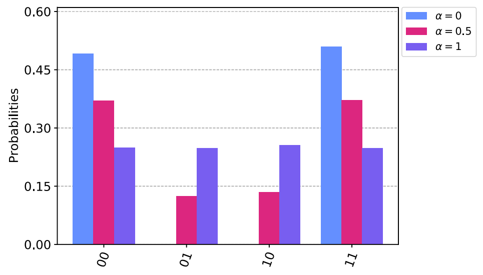
$$E(\vec{a}_i, \vec{b}_j) = P_{++}(\vec{a}_i, \vec{b}_j) + P_{--}(\vec{a}_i, \vec{b}_j) - P_{+-}(\vec{a}_i, \vec{b}_j) - P_{-+}(\vec{a}_i, \vec{b}_j) \tag{2}$$

and $\vec{a}_i$, $\vec{b}_i$ with $i = 1, 2, 3$ is the couple of measurement bases, i.e. the directions along which the polarizations of the two individual particles are measured. In Ekert's protocol these directions are identified by the angles θ_A, θ_B randomly extracted among $\{0, \pi/8, \pi/4\}$ and $\{\pi/8, \pi/4, 3\pi/8\}$ respectively, which in an experimental setup would define the orientations respect to the vertical of a pair of polarizing beamsplitters, each positioned before a single-photon detector. The probabilities of the two possible measures of the polarization depend on θ_A and θ_B and follows the laws of quantum Physics (see Fig. 1). These particular sets of angles are chosen so that the expected value of $|S|$ for maximally entangled particles is maximum and equal to $2\sqrt{2}$.

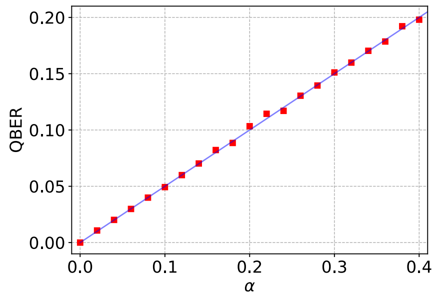
2.2 Quantum circuit in IBM Quantum

In order to transfer this protocol on IBM Q each photon is represented by a qubit, so that the protocol can be translated into the multiple iteration of a 2-qubit quantum circuit. A possible realization of such a circuit is shown in Fig. 2.

First the Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is obtained by applying the well-known sequence of a Hadamard gate and a CNOT gate to the two qubits in the default state $|0\rangle$ (Fig. 2, before the dashed barrier). After that, considering that in IBM Q qubits are measured along the z axis by default, we equivalently rotate the quantum states just before measuring them. In particular, if we map each qubit into a vector of the Bloch sphere according to the formalism of quantum information [20, sec. 1.2], each of these physical rotations



(a) Frequencies of the possible outcomes of the measurement, in the noiseless ($\alpha = 0$) and maximum-noise ($\alpha = 1$) scenarios and in an intermediate case ($\alpha = 0.5$).



(b) QBER as a function of the parameter α, exhibiting the expected direct proportionality.

Fig. 3 Effects of the depolarizing error channel on the Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$

can be performed using a $R_y(\gamma)$ gate, whose parameter γ must be twice the physical angle θ_j. Finally, in order to simulate the interaction between photons and environment, we chose to apply a *depolarizing error* channel [20, sec. 8.3.4] (see also https://qiskit.org/documentation/stubs/qiskit.providers.aer.noise.depolarizing_error.html), tunable through a parameter α, to the 2-qubit CNOT gate in the quantum circuit. This is equivalent to append—with probability α—a gate randomly chosen among X, Y and Z to at least one of the two qubits, and the net effect is a partial loss of entanglement between them (in Fig. 3a a comparison between the results for different values of α is shown). A non-zero quantum bit error rate (QBER) with an expected value equal to $\alpha/2$ consequently arises (see also Fig. 3b), which gives us the possibility of conveniently controlling the simulated disturbance on the quantum channel.

For a sequence of L entangled couples of photons, this circuit is replicated L times, every time randomly extracting a couple of bases and performing a measurement of the two qubits. The two strings of bits resulting from this step represent Alice's and Bob's raw keys: from them all the bits such that $\theta_A \neq \theta_B$ are discarded. This process reproduces the *sifting* of the keys, namely the phase in which Alice and Bob communicate to each other on a public channel the sequence of bases they have chosen and keep only the bits for which the two bases match. The steps up to this point represent the only part of the program that is run on an actual quantum computer (or on a simulator): the remaining instructions, as in the actual protocol, are classical algorithms and can be programmed in any language and run locally on any machine.

2.3 Post-processing

The post-processing consists of two steps, called *information reconciliation* (IR) and *privacy amplification*, common to all QKD protocols. The former aims to correct any mismatching between the strings of Alice and Bob, may they be caused by a malfunctioning of the equipment, by a degrading of the signal in the quantum channel or by a third party "Eve" that is eavesdropping; the latter instead compresses the strings in shorter, safer sequences, namely the final keys that will be used to encrypt the message.

In order to perform IR, part of the key is sacrificed for the estimation of the QBER through comparison on the public channel of the bits of Alice and Bob. The error rate is a fundamental quantity in this phase because it determines the size of the blocks in which the remaining key is split. In this work we will also compare the performance of the BBBSS protocol [21] and of the Cascade protocol [22], two historically relevant IR algorithms summarized in "Appendix 1". In particular, Cascade is still considered a benchmark for performance [23–25] and some of its variants are proposed as IR protocols [25–27] in recent literature because it represents a good compromise between the amount of information that is given away in the error correction process and the computational effort required, being also pretty simple to implement. In order for the whole protocol to succeed, the two parties must share two identical *reconciled* keys after IR.

Finally, *privacy amplification* is performed on the key so that any information about physical bits that Eve may have gained until now is turned into information about parity bits. In order to do so, a new key is generated: its bits are parity bits of randomly extracted subsets of the *reconciled* key. If at this point of the protocol the leaked information in an R bits-long reconciled key is estimated to consist of l bits, then the recommended minimum length F of the final key is $R - l - s$ bits, where s is a security parameter: increasing s decreases exponentially Eve's total information [20, 21].

3 Results

In the following we analyse and compare, under different conditions, two figures of merit of the BBBSS and Cascade reconciliation algorithms:

- the *secret fraction* is here defined, in analogy with [28, 29], as the ratio between the amount of secret information shared by the two parties after the execution of the IR protocol and the original length of the raw keys:

$$\text{secret fraction} = \frac{\langle R - N_{\text{leak}} \rangle}{L} \tag{3}$$

  R is the length of the reconciled key and is considered non-zero only if Alice's and Bob's keys are identical, N_{leak} is the number of bits leaked during IR. In the BBBSS algorithm a bit is discarded every time that the parity of a block is disclosed on the public channel, exactly to prevent Eve from obtaining knowledge about the keys, so in this case $N_{\text{leak}} = 0$; the Cascade algorithm requires instead that every bit is preserved (see the supplemental document for details) so N_{leak} is taken equal to the number of parities declared on the public channel.
- the *secret key rate* is here defined as the ratio between the average amount of secret bits in the reconciled key and the average value of the time interval Δt taken to execute just the reconciliation phase:

$$\text{secret key rate} = \frac{\langle R - N_{\text{leak}} \rangle}{\langle \Delta t \rangle} \tag{4}$$

  Even though many efforts have been addressed in the past decades to improve the secret fraction, it is essential to also keep in consideration the secret key rate when designing or choosing an IR algorithm. An error correction routine that is efficient but requires an intense computational effort (and then a long runtime) could represent a bottleneck for the QKD protocol: in a time-sensitive task such as real-time data encryption the users would be forced to discard part of the raw key, reducing the actual throughput [23].

Both of these quantities have been obtained for keys of different lengths and for several values of the QBER; the simulation have been executed on a Macbook (early 2015, 1.1GHz dual-core Intel Core M processor).

In order to observe how these figures of merit vary with L, the depolarizing noise parameter has been maintained constant at a value of $\alpha = 0.2$, for which QBER ≈ 0.1. In Fig. 4a we can observe that for the same number of iterations and the same noise level, the secret fraction shows opposite behaviours for the two protocols: BBBSS performs best on short keys, while Cascade's secret fraction increases as the keys get longer, seemingly converging to a plateau. This is probably due to a substantial difference between the two: BBBSS executes a limited number of parity checks between corresponding blocks of the two raw keys, regardless of the length of such keys. This is enough to correct all errors in short keys, but as L increases, so does the number of blocks in which the key is split: as a consequence it is more and more probable that a lucky error passes undetected all the iterations, causing the key reconciliation to fail. The Cascade algorithm, on the contrary, retains memory of the block parities of the previous iterations and can use that information to correct more than one error for a single parity bit publicly disclosed: detecting even just one error during any iteration triggers the cascade effect after which the protocol is named. The probability of this event is given by the probability of having an odd number of errors in at least one of the blocks, which grows with the number of blocks and then makes the error correction process more and more effective as the keys gets longer. For these reasons we expect that for values of L beyond this window the secret fraction keeps declining for BBBSS and increasing for Cascade.

Moreover, we are interested in finding the best conditions to conduct a real-time encryption, where the post-processing must be executed on many consecutive trains of quantum states. For this reason we looked for the optimal length L^* of the raw key that maximizes the secret key rate. We observed (Fig. 4b) that for both IR protocols the rate indeed peaks in correspondence of $L_B^* \approx 2000$ for BBBSS and $L_C^* \approx 5000$ for Cascade, and appears to drop drastically for longer keys.

Since reducing the QBER in the quantum channel under the order of the percent is still a challenge in real-life applications, it is worth to test how robust these algorithms are with respect to noise. In order to also make a fair comparison between the two while doing this, for each algorithm we kept L constant at the approximate optimal value (in particular, we set $L = 2000$ for BBBSS and $L = 5000$ for Cascade), while sweeping the α parameter over the $0 \div 0.4$ range. However, the results are plotted as a function of QBER that is a quantity of more practical relevance than α, the relation between the two having already been discussed above. As for the interval, we chose it to be centered around a QBER of 10% because typically that is the threshold value around which the transmission ceases to be considered unconditionally secure: for example in the case of the BB84 protocol it is proven that if the error rate is over 11% it is not possible to extract a safe key [30, 31]. Even though in this case the two curves of the secret fraction are very similar for a QBER between 2 and 12% (Fig. 4c), Cascade is again performing better when it comes to key rates (Fig. 4d). Interestingly though, Cascade's performance starts suffering from noise approximately at QBER $= 0.15$, which is the threshold value beyond which—as proven in [22]—the protocol does not guarantee a safe communication any more.

Finally, we report a visualization of the values of the key rate for Cascade when varying both L and α (Fig. 4e). This shows that the optimal length L^* of the raw keys changes with the noise level, so the parameters of the communication between Alice and Bob should be adapted accordingly to the detected QBER. In particular the results show that excluding the trivial noiseless case $\alpha = 0$—in which the characteristic "cascade dynamic" is never triggered and the Cascade protocol behaves very similarly to the BBBSS from which it evolves—the optimal parameter L^* diminishes as the noise parameter α increases, and tends to saturate in the considered interval. Recalling that for a specific noise the length of the blocks is $k \propto \alpha^{-1}$ (see "Appendix 1"), the non-monotonous behaviour of the rate as a function of L shown in Fig. 4b may probably be explained through the competition of two mechanisms: (i) on one hand, keys that are too short are split into a lower number $n_{bl} = \lceil L/k \rceil$ of blocks and therefore do not fully exploit the

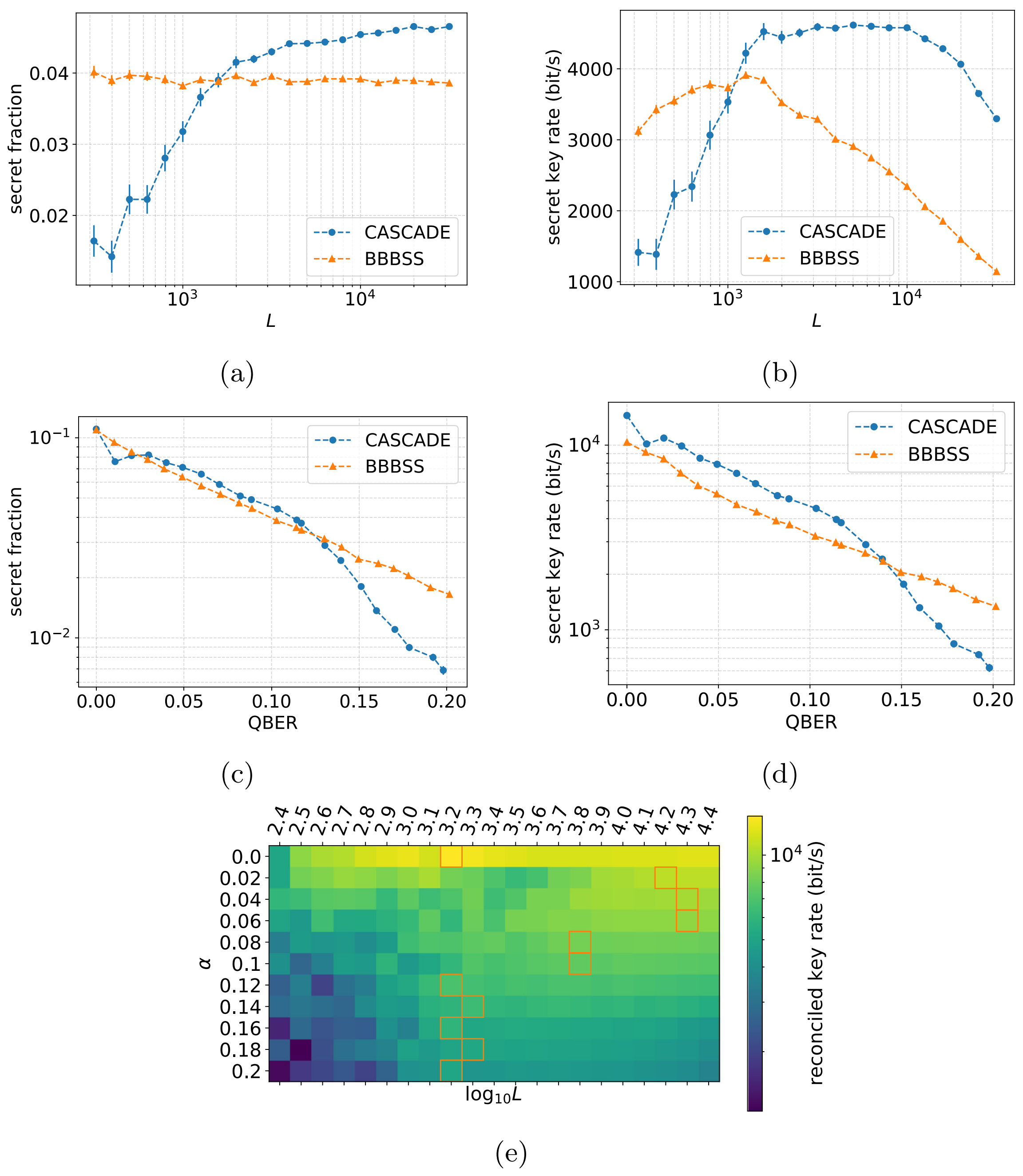


Fig. 4 Figures of merit of the two information reconciliation algorithms. In the top row, the secret fraction (a) and the reconciled key rate (b) are displayed for different values of the length L of the raw key, while maintaining a constant $\alpha = 0.2$. In (c), (d) the same quantities are plotted as functions of the QBER with a fixed $L = 1000$. Error bars have been computed as the standard deviation of the mean; dashed lines are just a guide for the eye. In (e) a heatmap of the rates achieved by Cascade algorithm when varying independently L and α: for a given noise, the value of L that optimizes the secret key rate is highlighted with orange borders. (All the plots in this figure have been obtained using the Python library Matplotlib 3.7.1, documentation available at: https://matplotlib.org/3.7.1/index.html)

strength of the error correction algorithm. To clarify this, let us consider the extreme case in which only two errors are left in the key: as L grows, n_{bl} grows, and it becomes more and more unlikely for these errors to be randomly assigned to the same block, so the probability of isolating and correcting them actually approaches 1. (ii) On the other hand, as L grows we verified that the time taken to distill a key increases more than linearly (as expected given the recursive nature of the Cascade algorithm), so this brings the rate down even though—as commented above and displayed in Fig. 4a—the secret fraction appears to improve. In conclusion, an increment in α results in a larger $n_{bl} \propto \alpha$, so as α grows the necessity of having longer keys explained in (i) becomes less urgent and the optimal tradeoff consequently “moves” towards shorter blocks.

4 Conclusions

By exploiting the IBM Quantum environment, we have been able to implement a simulation of the entire E91 QKD protocol on one platform. In particular we used the `aer_simulator` with an appropriate noise model (the depolarizing channel), that are both built-in features of Qiskit, to run the quantum circuit and simulate the execution of the protocol between two parties in a scenario of entanglement decay.

We also described the performance of two information reconciliation algorithms through two figures of merit: the secret fraction and the secret key rate have been computed while varying the number of qubits and the noise level. The comparison highlighted the superiority of the Cascade algorithm over the BBBSS algorithm under most of the conditions tested, and identified for each of them an optimal L that maximizes the achievable secret key rate. Finally, in terms of resilience to noise, the results reproduced the prediction in [22] according to which Cascade's performance would be heavily affected by a QBER greater than 0.15.

As simple as it is, and with no claims of competitiveness, this project could serve as a stand-alone resource for modelling a system of parties communicating through QKD-generated keys. One of the most commonly proposed use cases, rather than a one-to-one interaction, is indeed the realization of a Metropolitan Area Network (MAN) [28, 32] to perform secret communication on an urban scale across a chain of trusted nodes, since the inter-node distances that allow for an effective communication without the aid of the so-called *quantum repeaters* are of the order of the tens of kilometers [33, 34]. Not only would such a network permit to mitigate the rate-distance trade-off, one of the most strict limits to this technology, but the same approach could also be followed to build a blockchain system in which QKD would unconditionally guarantee secrecy and security, instead of the asymmetric cryptography schemes currently used that are vulnerable to the "quantum threat" [5–7]. Moreover, given the long-term objective of building a quantum network on a continental scale, the research is becoming more and more active in this field [35–37] and several initiatives are currently being supported by the European Union and by the industry towards this direction, including—but not limited to—the European Quantum Communication Infrastructure (EuroQCI), the Quantum Secure Networks Partnership (QSNP) and the European Quantum Internet Alliance (QIA). The model and the results presented in this work could then be a tool to study the performances and consequently assess the feasibility of such projects in the very first steps of the design.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1140/epjp/s13360-024-05337-2.

Author Contributions Conceptualization: LM, LS, RC; formal analysis: LM; investigation: LM; methodology: LM, LS, CA, SP, RC; software: LM; supervision: CA, RC and SP; writing (original draft): LM; writing (review and editing): LM, LS, CA, SP, RC. All authors have read and agreed to the published version of the manuscript.

Funding Open access funding provided by Università degli Studi di Salerno within the CRUI-CARE Agreement.

Data Availability Statement Data underlying the results presented in this paper are not publicly available at this time but may be obtained from the corresponding author upon reasonable request. The manuscript has associated data in a data repository.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Appendix 1: Algorithm description

In order to describe in detail the methods for this work we will first introduce here some primitives defined in [21] and then proceed to describe the two IR protocols that we used.

BINARY The `BINARY` primitive is an error-correction method based on a divide-and-conquer approach: given a couple of strings with mismatching parities, this string are divided in halves, whose parities are computed and compared to find in which half the error is. Then the procedure is repeated recursively until the length of the sub-strings with mismatching parity is 1 and then the error is identified and corrected by flipping the bit in one of the two strings (say Bob's). Please note that this primitive is interactive and it requires the public exchange of a certain number of parity bits equal to $\lceil \log n \rceil$ if n is the length of the original string.

CONFIRM When applied to a couple of strings, this primitive will always return a positive outcome if the strings are identical. Instead, if there is disagreement between the two strings, it will report the presence of an error with only 50% probability. It consists in 1) extracting a random subset of bits (each bit is included in the subset with 50% probability), 2) comparing the parities and 3) raising a flag if there is a mismatch. This implies that if CONFIRM does not detect a mismatch between the parities of the sub-blocks for k consecutive times, there is a $1 - 2^{-k}$ probability that the two strings coincide.

BICONF The $\texttt{BICONF}^s$ primitive has been defined in [22] but the original Cascade protocol there proposed does not make use of it. It consists in s consecutive runs of CONFIRM, combined with the execution of a BINARY every time that an error is detected.

BBBSS Based on the QBER estimated in the IR step, the sifted key is divided in blocks containing $k = a/QBER$ bits (no recommendation regarding the value of a was given in the original paper [21], but the choice of $a \approx 0.73$ for the similar Cascade protocol was later found to be optimal [22]). After that, BINARY is executed on each block, always discarding the last bit of any sub-block whose parity is publicly declared, so as to give up no information to Eve in this step. The procedure is iterated on the remaining key, dividing the key in blocks twice as long as in the first iteration. After that, as a further guarantee that the reconciliated keys are identical, $\texttt{BICONF}^1$ can be iterated (again sacrificing a bit after every pass) until it highlights no parity mismatches for s consecutive times: this provides a $1 - 2^{-s}$ confidence that no errors survived the protocol. For this work we took $s = 10$.

Cascade The first iteration (or pass) of this protocol [22] is identical to the first iteration on BBBSS, except for the fact that—as in the rest of this protocol—no bits are discarded. Eve will then get some information about sub-block parities but this allows for the correction of more than one error per parity exchange. For any i-th pass after the first one, the block size k_i is doubled, and whenever an error is found the following routine is executed: after that Bob receives Alice's parities and finds a disagreement when comparing them with its own, he executes BINARY to correct the mismatching bit B_l. Then he builds a list $\mathcal{K}$ of all the blocks that contained B_l in the previous iterations and performs BINARY on the shortest of such blocks, correcting another bit $B_{l'}$. At this point $\mathcal{K}$ is updated:

$$\mathcal{K} \rightarrow \mathcal{K}' = (\mathcal{B} \cup \mathcal{K}) \setminus (\mathcal{B} \cap \mathcal{K}) \tag{5}$$

$\mathcal{B}$ being the list of blocks containing $B_{l'}$. This corresponds to adding to $\mathcal{K}$ the blocks containing $B_{l'}$ while subtracting the ones containing both B_l and $B_{l'}$. The previous steps are repeated until $\mathcal{K} = \emptyset$. If any other mismatch is found between Alice's and Bob's parities of the i-th iteration these steps are repeated. Finally another iteration can be performed if considered necessary.

References

1. C.H. Bennett, G. Brassard, Quantum cryptography: public key distribution and coin tossing. Theoret. Comput. Sci. **560**, 7–11 (2014)
2. S. Wiesner, Conjugate coding. ACM SIGACT News **15**, 78–88 (1983)
3. R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**, 120–126 (1978)
4. V. Mavroeidis, K. Vishi, M. Zych, A. Jøsang, The impact of quantum computing on present cryptography. Int. J. Adv. Comput. Sci. Appl. **9**, 405–414 (2018)
5. T.M. Fernández-Caramès, P. Fraga-Lamas, Towards post-quantum blockchain: a review on blockchain cryptography resistant to quantum computing attacks. IEEE Access **8**, 21091–21116 (2020)
6. I. Stewart, D. Ilie, A. Zamyatin, S. Werner, M.F. Torshizi, W.J. Knottenbelt, Committing to quantum resistance: a slow defence for Bitcoin against a fast quantum computing attack. R. Soc. Open Sci. **5**, 180410 (2018)
7. D. Yaga, P. Mell, N. Roby, K. Scarfone, Blockchain technology overview, Technical Report. NIST IR 8202, National Institute of Standards and Technology, Gaithersburg (2018)
8. L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, D. Smith-Tone, Report on post-quantum cryptography, Technical Report. NIST IR 8105, National Institute of Standards and Technology (2016)
9. M. Mosca, Cybersecurity in an era with quantum computers: Will we be ready? IEEE Secur. Privacy **16**, 38–41 (2018)
10. M. Campagna, L. Chen, Ö. Dagdelen, J. Ding, J.K. Kernick, N. Gisin, D. Hayford, T. Jennewein, N. Lütkenhaus, M. Mosca, B. Neill, M. Pecen, R. Perlner, G. Ribordy, J.M. Schanck, D. Stebila, N. Walenta, W. Whyte, Z. Zhang, Quantum safe cryptography and security—an introduction, benefits, enablers and challenges, Technical Report, ETSI (European Telecommunications Standards Institute) (2015)
11. A.K. Ekert, Quantum cryptography based on Bell's theorem. Phys. Rev. Lett. **67**, 661–663 (1991)
12. J. Clauser, M. Horne, A. Shimony, R. Holt, Proposed experiment to test local hidden-variable theories. Phys. Rev. Lett. **23**, 880–884 (1969)
13. A.D. Córcoles, J.M. Chow, J.M. Gambetta, C. Rigetti, J.R. Rozen, G.A. Keefe, M. Beth Rothwell, M.B. Ketchen, M. Steffen, Protecting superconducting qubits from radiation. Appl. Phys. Lett. **99**, 181906 (2011)
14. A.D. Córcoles, E. Magesan, S.J. Srinivasan, A.W. Cross, M. Steffen, J.M. Gambetta, J.M. Chow, Demonstration of a quantum error detection code using a square lattice of four superconducting qubits. Nat. Commun. **6**, 6979 (2015)
15. K. Temme, S. Bravyi, J.M. Gambetta, Error mitigation for short-depth quantum circuits. Phys. Rev. Lett. **119**, 180509 (2017)
16. A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J.M. Chow, J.M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature **549**, 242–246 (2017)
17. S. Bravyi, D. Gosset, R. König, Quantum advantage with shallow circuits. Science **362**, 308–311 (2018)
18. R. Mandelbaum, Five years ago today, we put the first quantum computer on the cloud. Here's how we did it. (2021). https://research.ibm.com/blog/quantum-five-years
19. D.R. Hjelme, L. Lydersen, V. Makarov, Quantum cryptography. arXiv:1108.1718 [quant-ph] (2011)
20. M.A. Nielsen, I.L. Chuang, *Quantum Computation and Quantum Information*, 10th edn. (Cambridge University Press, Cambridge, 2010)
21. C.H. Bennett, F. Bessette, G. Brassard, L. Salvail, J. Smolin, Experimental quantum cryptography. J. Cryptol. **5**, 3–28 (1992)
22. G. Brassard, L. Salvail, Secret-key reconciliation by public discussion, in *Advances in Cryptology—EUROCRYPT '93*, vol. 765, ed. by T. Helleseth (Springer, Berlin, 1994), pp.410–423
23. J. Martinez-Mateo, D. Elkouss, V. Martin, Key reconciliation for high performance quantum key distribution. Sci. Rep. **3**, 1576 (2013)
24. J. Martinez-Mateo, C. Pacher, M. Peev, A. Ciurana, V. Martin, Demystifying the information reconciliation protocol cascade. arXiv:1407.3257 [quant-ph] (2014)

25. H. Yan, T. Ren, X. Peng, X. Lin, W. Jiang, T. Liu, H. Guo, Information reconciliation protocol in quantum key distribution system, in *2008 Fourth International Conference on Natural Computation*, vol. 3, pp. 637–641 (2008)
26. M. Mehic, M. Niemiec, M. Vozňák, Calculation of the key length for quantum key distribution. Elektronika ir Elektrotechnika **21**, 81–85 (2015)
27. W.T. Buttler, S.K. Lamoreaux, J.R. Torgerson, G.H. Nickel, C.H. Donahue, C.G. Peterson, Fast, efficient error reconciliation for quantum cryptography. Phys. Rev. A **67**, 052303 (2003)
28. V. Scarani, H. Bechmann-Pasquinucci, N.J. Cerf, M. Dusek, N. Lutkenhaus, M. Peev, The security of practical quantum key distribution. Rev. Mod. Phys. **81**, 1301–1350 (2009)
29. L. Sheridan, L. Thinh, V. Scarani, Finite-key security against coherent attacks in quantum key distribution. New J. Phys. **12**, 123019 (2010)
30. P.W. Shor, J. Preskill, Simple proof of security of the BB84 quantum key distribution protocol. Phys. Rev. Lett. **85**, 441–444 (2000)
31. N. Lütkenhaus, Security against individual attacks for realistic quantum key distribution. Phys. Rev. A **61**, 052304 (2000)
32. European Telecommunications Standards Institute, *Quantum Key Distribution; Use Cases*, vol. ETSI GS QKD 002 (European Telecommunications Standards Institute) (2010)
33. M. Takeoka, S. Guha, M.M. Wilde, Fundamental rate-loss tradeoff for optical quantum key distribution. Nat. Commun. **5**, 5235 (2014)
34. M. Lucamarini, Z.L. Yuan, J.F. Dynes, A.J. Shields, Overcoming the rate–distance limit of quantum key distribution without quantum repeaters. Nature **557**, 400–403 (2018)
35. V. Brosco, L. Pilozzi, C. Conti, Paths in quantum communication networks, in *2022 IEEE 15th Workshop on Low Temperature Electronics (WOLTE)*, pp 1–4 (2022)
36. R. Yehia, S. Neves, E. Diamanti, I. Kerenidis, Quantum city: simulation of a practical near-term metropolitan quantum network (2022)
37. F. Centrone, F. Grosshans, V. Parigi, Cost and routing of continuous variable quantum networks. Phys. Rev. A **108**(4), 042615 (2022)