Symmetric Encryption on a Quantum Computer

David Garvin[†] Oleksiy Kondratyev[‡] Alexander Lipton[§]
Marco Paini[¶]

November 7, 2024

Abstract

Classical symmetric encryption algorithms use N bits of a shared secret key to transmit N bits of a message over a one-way channel in an information theoretically secure manner. This paper proposes a hybrid quantum-classical symmetric cryptosystem that uses a quantum computer to generate the secret key. The algorithm leverages quantum circuits to encrypt a message using a one-time pad-type technique whilst requiring a shorter classical key. We show that for an N-qubit circuit, the maximum number of bits needed to specify a quantum circuit grows as $N^{3/2}$ while the maximum number of bits that the quantum circuit can encode grows as N^2. We do not utilise the full expressive capability of the quantum circuits as we focus on second order Pauli expectation values only. The potential exists to encode an exponential number of bits using higher orders of Pauli expectation values. Moreover, using a parameterised quantum circuit (PQC), we could further augment the amount of securely shared information by introducing a secret key dependence on some of the PQC parameters. The algorithm may be suitable for an early fault-tolerant quantum computer implementation as some degree of noise can be tolerated. Simulation results are presented along with experimental results on the 84-qubit Rigetti Ankaa-2 quantum computer.

1 Introduction

The scientific literature has widely discussed the impact of quantum information and computation on communication and cryptosystems. On the one hand, quantum communication channels can transmit information with a complexity advantage [1] and can make quantum key distribution (QKD) more secure against eavesdropping attacks [2] – the latter thanks to fundamental quantum information principles such as the *no-cloning theorem* [3]. On the other hand, future fault-tolerant quantum computers may implement Shor's algorithm efficiently [4, 5], which would irreversibly

[†]Rigetti Computing, Email: dgarvin@rigetti.com
[‡]Imperial College London, Email: a.kondratyev@imperial.ac.uk
[§]Abu Dhabi Investment Authority (ADIA), Email: alexander.lipton@adia.ae
[¶]Rigetti Computing, Email: mpaini@rigetti.com

compromise the security of widely adopted public-key cryptosystems that rely on the computational complexity of semiprime integer factoring (Rivest-Shamir-Adleman, RSA) [6] and the discrete logarithm (Diffie-Hellman, DH) [7].

Public-key cryptosystems are called *asymmetric* as they utilize two *keys*: one is public and used for encryption, while the other is private and enables decryption. Let us assume that Alice wishes to establish a secure communication channel using RSA. To this aim, she first generates a key pair using two large prime integers and the modular exponentiation one-way function. Then, she shares one key publicly: Bob uses it to encrypt the plain text message for Alice. The resulting cypher text can be decrypted by Alice only using the other key, which is private to her. Digital signatures [8] and authentication [9] can be implemented with this method. Excluding the quantum computing threat, asymmetric cryptosystems are secure if the key pair has been correctly generated and the private key remains as such. In particular, knowledge of the public key cannot be leveraged to retrieve or compromise the private key, as no classical algorithm is known that can efficiently invert the one-way function [10]. The latter point is of practical advantage compared to the other major class of cryptosystems, called *symmetric*, in which a private key must be shared securely among the communicating parties.

Symmetric cryptosystems use one key only. Crucially, the key needs to be shared between Alice and Bob before any secure communication can commence, as it is used for both encryption of the plain text and decryption of the cypher text – as such, it is called the *secret*. This latter aspect requires establishing a physically secure channel for secret sharing and ensuring that none of the potentially many key copies is ever exposed: these are notable disadvantages as they are often difficult to guarantee and verify. In practice, symmetric-key algorithms are effectively used as subroutines to perform bulk plain text encoding, and a public-key cryptosystem is used to establish the physically secure channel for secret sharing.

With significant technological progress toward building commercial quantum computers [11, 12, 13], the relative balance in strengths and weaknesses between symmetric and asymmetric cryptosystems may be shifting. As mentioned, Shor's algorithms can efficiently invert RSA's one-way function with an asymptotic exponential speedup over the best known classical algorithm, the general number field sieve [14]. Public-key post-quantum cryptography algorithms are designed to be secure against known quantum computing attacks [15]; however, their vulnerability to other, currently unfeasible attacks is a matter of active study and critical undertaking of many leading initiatives [16]. Interestingly, widely adopted symmetric-key algorithms such as the Advanced Encryption Standard (AES) [17] are believed to be post-quantum since quantum computer attack times will remain large as long as key sizes are sufficiently big (at least 256 bits) [18]. With this specific focus on quantum computing, we move on to discuss the case of quantum symmetric cryptosystems in more detail.

The security of sending classical messages using one-time pads [19, 20] was investigated by Shannon who showed that N bits of a shared classical secret key are necessary and sufficient to transmit N bits of a message over a one-way classical

channel, in an information theoretically secure manner [21, 22]. Bennett et al. determined that $2N$ bits of shared classical secret key, along with N EPR pairs used for teleportation over a quantum channel, can be used to securely transfer N qubits of information [23]. Mosca et al. demonstrated that the transmission of N qubits securely over a quantum channel can be achieved using only $2N$ bits of a classical shared secret key [24]. Remote state preparation requires one bit of classical communication and one entangled qubit per qubit sent [25]. Encoding a classical message using quantum states allows for uncloneable encryption where an eavesdropper cannot clone the message for later decoding without alerting the message receiver [26]. The ability to detect whether eavesdropping has occurred allows reuse of a one-time pad without compromising security [27]. Protocols for transmitting classical data securely over quantum channels are presented in [28, 29].

This paper proposes a symmetric cryptosystem that uses a quantum computer to generate the secret. As far as the authors are aware, not much attention in the literature has been dedicated to *hybrid* symmetric encryption algorithms utilising quantum resources to encrypt and decrypt the message whilst using a one-way classical channel to transfer the information. Although we do not discuss its computational complexity rigorously, we put forward general considerations suggesting that our algorithm can be as secure as a one-time pad system – the latter being the only theoretically secure classical cipher [30]. The proposed algorithm leverages parameterised quantum circuits (PQC) [31, 32] employing particular circuit structures that render their classical sampling inefficient [33].

The algorithm enables us to encrypt a message using one-time pad-type techniques but requires the transmission of significantly less information to specify the key. We show that for a circuit containing N qubits, the maximum number of bits needed to specify its configuration grows as $N^{3/2}$ while the maximum number of bits encoded grows as N^2. However, we do not utilise the full expressive capability of the circuit in our implementation as we only use the second order Pauli expectation values. The potential exists to encode an exponential number of bits by using higher order Pauli expectation values. The amount of securely transmitted information can be increased by using a PQC where some gate parameters change over time according to a prescribed secret. In the following, we will use the terms quantum circuits and PQC interchangeably, as the protocol works by drawing random circuit parameters from families of circuits that share broad structure characteristics.

Paper contributions: The contributions made in this paper are:

- An illustrative example showing how symmetric encryption can be implemented utilising a gate-based quantum computer to generate the secret key.
- A general algorithm for performing symmetric encryption and decryption using a quantum computer to generate a form of one-time pad secret key. Due to the expressive power of the PQC, for large messages, a secret key of less than N classical bits is required to encrypt a message containing N classical bits. All the information shared via unsecure channels has no value to an eavesdropper attempting to decrypt the message.

- An algorithm which may be a candidate for the early fault-tolerant quantum computing era as some degree of noise can be tolerated. The algorithm requires high-quality gates on the quantum computer since deep circuits are needed. The algorithm requires measurements to be expectation values of Pauli observables. This adds noise resistance to the algorithm since error mitigation techniques can be applied.
- Simulation results on 5-qubit and 20-qubit circuits. Experimental results using the 84-qubit Rigetti Ankaa-2 quantum computer with an 84-qubit circuit.
- A method for specifying a PQC containing N qubits and M gate layers using a maximum of $(a + b)NM$ classical bits, with a-bit encoding of the gate type and b-bit encoding of the adjustable gate parameter (rotation angle).
- An analysis of the trade-off between sampling noise and the algorithm's encoding efficiency.

Section 2 provides a brief introduction to PQCs. Section 3 highlights a stylised example that illustrates the ideas behind the algorithm. Section 4 discusses additional security considerations. Section 5 gives a detailed algorithm description. Section 6 provides an example of the end-to-end encryption and decryption executed on a quantum simulator as well as on the quantum hardware (Rigetti's Ankaa-2 QPU) with the application of error detection and error correction techniques. Section 7 concludes and provides recommendations for future research. The Appendices provide additional information on plain text symbol to bitstring encoding, analysis of expectation values of second order Pauli measurements, determination of classical bits required to represent a PQC, and components of the symmetric encryption algorithm.

2 Background

Quantum computing is an emerging technology that employs properties of quantum mechanics to perform computations, in some cases exponentially more efficiently than classical calculations [4, 34, 35, 36]. *Superposition* enables qubits (quantum bits) to exist as a combination of both the $|0\rangle$ and $|1\rangle$ states simultaneously [37]. *Entanglement* statistically links the properties of one qubit with another beyond classical correlations. Quantum *interference* of probability amplitudes allows certain computations to be performed efficiently. *Measurement* of a qubit results in one bit of classical information, where the probability of measuring either "0" or "1" depends on the magnitude of the corresponding squared probability amplitudes of the qubit states, as per the Born rule [38].

Quantum algorithms are represented by quantum circuits – in the widespread gate-based computation model [39, 40]. These circuits are composed of single-qubit and multi-qubit quantum gates. Quantum gates are unitary linear operators that transform the quantum state (represented by a complex unit vector in a high-dimensional Hilbert space), thus implementing the computational protocol. Multi-qubit gates are used to create entanglement.

Some useful single-qubit gates are the Pauli X, Y and Z gates:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

and the rotation gates that rotate qubit states around the x, y and z axis by angle θ:

$$R_x(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix},$$

$$R_z(\theta) = \begin{bmatrix} \mathrm{e}^{-\mathrm{i}\theta/2} & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\theta/2} \end{bmatrix}.$$

Two-qubit gates that create entanglement are, e.g., the CZ and XY ($iSWAP$) gates:

$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad iSWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

A set of quantum gates that can be combined to implement any quantum (or classical) equivalent gate is called a universal gate set [39]. While there are many potential universal gate sets, the set of choice depends on the hardware modality and ease of physical implementation. A transpiler converts a quantum program into an efficient sequence of native gates, that is, with as few native gates as possible [41, 42, 43]. Quantum circuits may be hard to simulate on classical computers [44].

A parameterised quantum circuit is a type of quantum circuit where some (or all) gates can be specified by a set of adjustable parameters $\theta_1, \ldots, \theta_m$. Figure 1 shows a schematic representation of a PQC composed of single-qubit and multi-qubit quantum gates. The final quantum state, $|\psi_f\rangle$, is obtained after the application of the quantum circuit to the initial quantum state, $|\psi_0\rangle$:

$$|\psi_f\rangle = U_m(\theta_m) \ldots U_2(\theta_2) U_1(\theta_1) |\psi_0\rangle .$$

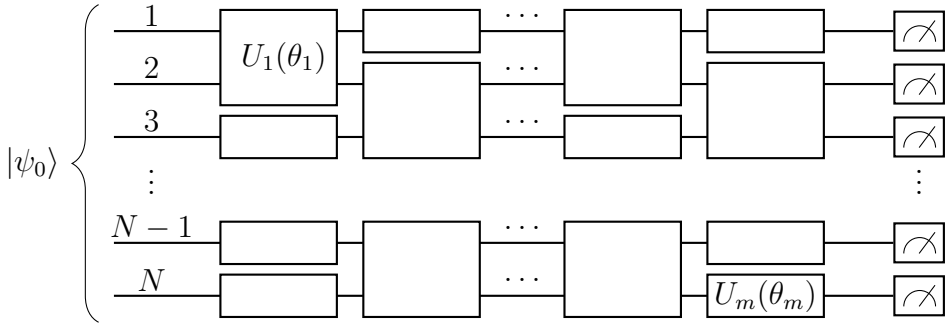

Figure 1: Schematic representation of an N-qubit PQC. Note that this illustrative example has an arbitrary choice of 1- and 2-qubit gate positions within the circuit.

The final state can be measured. The measurement produces a classical bitstring, which follows the probability distribution specified by the quantum state via the Born rule. Qubit measurement occurs in a specific basis. The computational basis is typically the z-basis (unless explicitly specified otherwise) and the measurement of the qubit state in the z-basis gives us either "0" (corresponding to the $+1$ eigenvalue of the Pauli Z operator) or "1" (corresponding to the -1 eigenvalue of the Pauli Z operator). The outcome is probabilistic, but running the same quantum circuit multiple times allows us to calculate the expectation value of Pauli Z on the corresponding qubit. Changing the z-basis to the x-basis or the y-basis allows us to calculate the expectation values of, respectively, Pauli X and Pauli Y operators. To measure these expectation values, "change of basis" gates are added to the quantum circuit before measurement [45]:

- To remain in the z-basis, add nothing (or the identity gate I) to the qubit.
- To transform the z-basis into the x-basis, add an H gate to the qubit.
- To transform the z-basis into the y-basis, add $HS^\dagger$ gates to the qubit (the $S^\dagger$ gate should be applied first).

The gates used to change the basis are as follows:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}.$$

3 Stylised Example

We now present a simplified example illustrating how a person, Alice, can securely transmit a message to her friend, Bob, using a hybrid quantum-classical encryption algorithm. This highlights the major components of our algorithm which are then covered in more detail in Section 5.

Alice and Bob, two trusted parties, wish to establish a secure communication channel. They meet in Alice's office – a secure place – where Alice generates a random PQC in the native gate set. An example of such a PQC is presented in Figure 2.

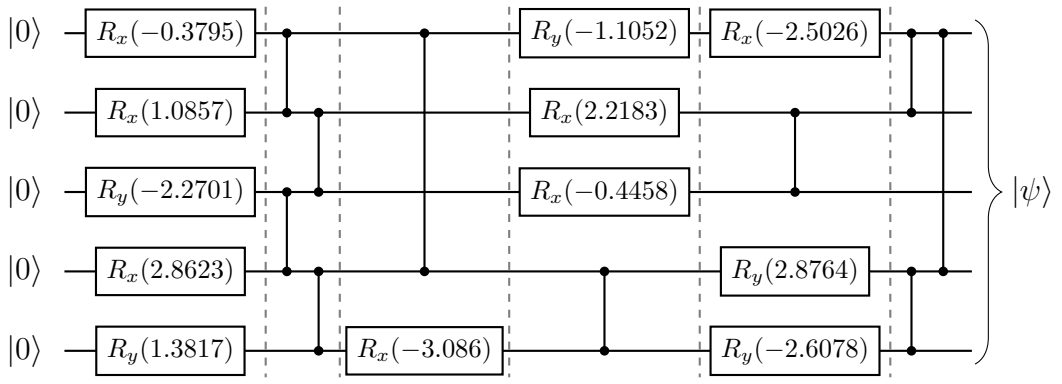

Figure 2: Randomly generated quantum circuit. All 1-qubit gate parameters (rotation angles) are expressed in radians.

In this stylised example, the randomly generated quantum circuit consists of 5 qubits and 6 layers of 1- and 2-qubit gates. The 1-qubit gates are random rotations around the x and y axes and the 2-qubit gates are CZ (controlled Z) gates that create entanglement.

Alice shares this circuit with Bob via a secure in-house communication channel. Now both Alice and Bob have the same quantum circuit on their laptops in the form of Python code. The Python code represents the circuit as a network of quantum gates applied to a set of qubits [46]. The algorithm is hardware agnostic enabling the code to be run on any quantum device once the circuit is represented using Python commands specific to that device [47].

Later, Alice wants to communicate with Bob. She decides to send him an encrypted message consisting of letter "K" in binary format:

Letter	Radio Signal	ASCII Binary	ICS Meaning
K	Kilo	1001011	"I wish to communicate with you"

Therefore, Alice needs to encrypt the bitstring [1001011].

Alice starts with generating a random string of Pauli operators X, Y and Z – the same length as the width of the quantum circuit she shared with Bob:

$$[X, Z, Y, Y, X].$$

This determines the bases in which qubits should be measured: the first and fifth qubits in the x-basis, the second qubit in the z-basis, and the third and fourth qubits in the y-basis. The objective is to calculate the expectation values of Pauli operators on various qubits. In this stylised example we have the following possibilities for measuring the expectation value of a tensor product of two Pauli operators P_k and P_l on qubits k and l, $\langle P_k \otimes P_l \rangle$:

$$\begin{gathered}
\langle X_1 \otimes Z_2 \rangle , \\
\langle X_1 \otimes Y_3 \rangle , \\
\langle X_1 \otimes Y_4 \rangle , \\
\langle X_1 \otimes X_5 \rangle , \\
\langle Z_2 \otimes Y_3 \rangle , \\
\langle Z_2 \otimes Y_4 \rangle , \\
\langle Z_2 \otimes X_5 \rangle , \\
\langle Y_3 \otimes Y_4 \rangle , \\
\langle Y_3 \otimes X_5 \rangle , \\
\langle Y_4 \otimes X_5 \rangle .
\end{gathered}$$

To measure these expectation values, Alice has to add "change of basis" gates to the quantum circuit before measurement (see Section 2) as shown in Figure 3:

- Add nothing (or identity gate I) to the second qubit as the computational basis is the z-basis.
- Add H gates to the first and fifth qubits to transform the z-basis into the x-basis.
- Add $HS^\dagger$ gates to the third and fourth qubits to transform the z-basis into the y-basis (the $S^\dagger$ gate should be applied first).

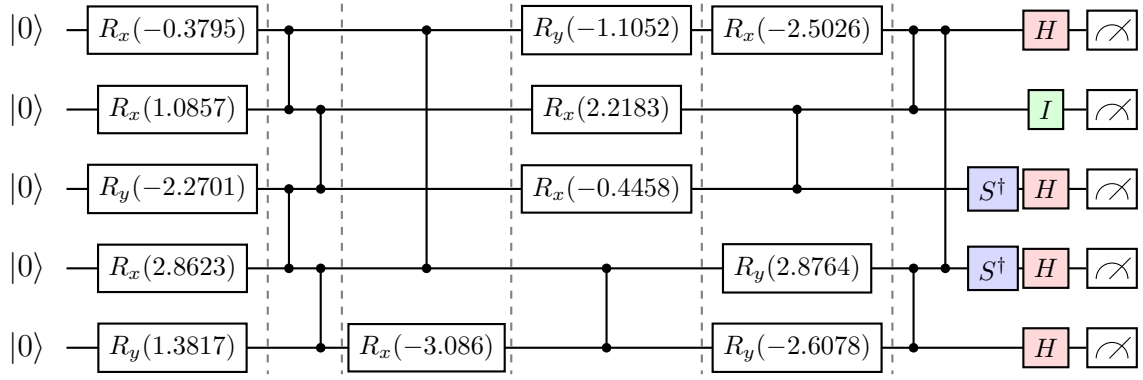

Figure 3: Measurement in different bases via inclusion of additional gates.

Next, Alice runs this quantum circuit 100,000 times and saves the measurement results in a $100,000 \times 5$ array. If "0" is measured, the eigenvalue of the corresponding Pauli operator is $+1$. If "1" is measured, the eigenvalue of the corresponding Pauli operator is -1. Therefore, the array entries are either $+1$ or -1. The number of rows is equal to the number of quantum circuit runs and the number of columns is equal to the number of qubits.

Then Alice computes the expectation values for the pairs of Pauli operators (tensor products of two Pauli operators). The expectation value $\langle P_k \otimes P_l \rangle \equiv \langle P_k P_l \rangle$ is the dot product of columns k and l, normalised by the number of rows. She gets:

$$\begin{aligned}
\langle X_1 Z_2 \rangle &= -0.76160 \\
\langle X_1 Y_3 \rangle &= 0.22724 \\
\langle X_1 Y_4 \rangle &= -0.00574 \\
\langle X_1 X_5 \rangle &= 0.13070 \\
\langle Z_2 Y_3 \rangle &= 0.81316 \\
\langle Z_2 Y_4 \rangle &= 0.02814 \\
\langle Z_2 X_5 \rangle &= -0.09922 \\
\langle Y_3 Y_4 \rangle &= -0.01034 \\
\langle Y_3 X_5 \rangle &= 0.12158 \\
\langle Y_4 X_5 \rangle &= 0.00368
\end{aligned}$$

- If the expectation value $\langle P_k P_l \rangle$ is larger than a chosen threshold value ϵ, the pair $P_k P_l$ can be used to encode "1".

- If the expectation value $\langle P_k P_l \rangle$ is smaller than $-\epsilon$, the pair $P_k P_l$ can be used to encode "0".

The value of ϵ depends on the circuit configuration and the number of runs. Alice sets it at $\epsilon = 0.01$. An analysis of the ϵ calculation and tradeoffs required is provided in Appendix B.

Therefore, Alice can use the following pairs of Pauli operators for encrypting her message:

$$
\begin{array}{rcll}
\langle X_1 Z_2 \rangle & \Rightarrow & 0 & \\
\langle X_1 Y_3 \rangle & \Rightarrow & 1 & \\
\langle X_1 Y_4 \rangle & \Rightarrow & - & \text{(expectation value falls within the } [-\epsilon, \epsilon] \text{ interval)} \\
\langle X_1 X_5 \rangle & \Rightarrow & 1 & \\
\langle Z_2 Y_3 \rangle & \Rightarrow & 1 & \\
\langle Z_2 Y_4 \rangle & \Rightarrow & 1 & \\
\langle Z_2 X_5 \rangle & \Rightarrow & 0 & \\
\langle Y_3 Y_4 \rangle & \Rightarrow & 0 & \\
\langle Y_3 X_5 \rangle & \Rightarrow & 1 & \\
\langle Y_4 X_5 \rangle & \Rightarrow & - & \text{(expectation value falls within the } [-\epsilon, \epsilon] \text{ interval)}
\end{array}
$$

Alice sends Bob a text message that reads: XZYYX.

Then Alice sends Bob an email that reads: (1,3) (1,2) (2,5) (1,5) (3,4) (2,3) (2,4).

After Bob receives the text message, he knows that he must change the basis on qubits 1, 3, 4 and 5 (add H to the first and fifth qubits and add $HS^\dagger$ to the third and fourth qubits).

After Bob receives the email, he knows what expectation values he needs to calculate and in what order.

Bob runs the quantum circuit (the same as Alice's) and gets the following expectation values (they would differ slightly from Alice's values due to the finite number of quantum circuit runs and some hardware noise):

$$
\begin{aligned}
\langle X_1 Y_3 \rangle &= 0.22762 \Rightarrow 1 \\
\langle X_1 Z_2 \rangle &= -0.76166 \Rightarrow 0 \\
\langle Z_2 X_5 \rangle &= -0.09430 \Rightarrow 0 \\
\langle X_1 X_5 \rangle &= 0.12724 \Rightarrow 1 \\
\langle Y_3 Y_4 \rangle &= -0.00986 \Rightarrow 0 \\
\langle Z_2 Y_3 \rangle &= 0.81240 \Rightarrow 1 \\
\langle Z_2 Y_4 \rangle &= 0.03230 \Rightarrow 1
\end{aligned}
$$

Bob correctly decrypts the message, reads the bitstring [1001011], and learns that Alice would like to get in touch with him.

Two unsecure channels (text and email) were used to transmit the message. Even if both channels are compromised it is impossible to decipher the message without knowledge of the quantum circuit that creates the quantum state in which the Pauli operators are measured. In other words, the quantum circuit plays the role of a symmetric key used to encrypt and decrypt messages between two trusted parties. It has been shown that it is exponentially difficult to learn quantum circuits of bounded 2-qubit gate numbers from their samples [48].

4 Additional Security

The stylised example in Section 3 contains two important security protocol elements that provide additional protection:

- No Pauli pairs are repeated.
- Only some of the possible Pauli pairs are used.

Our algorithm can be extended in multiple directions by including extra types of security. Below, we discuss the following additional elements of the protocol:

- Using a proprietary randomly generated symbol-to-bitstring encoding scheme for the plain text symbols.
- Making some of the 1-qubit gates vary with the corresponding adjustable parameters (rotation angles) depending on the time stamp (as a de facto random seed – structurally identical PQC can generate very different quantum states).
- Extending the protocol into a multiple-key encryption where two (or more) communicating parties are required to combine their keys to decrypt the message.
- Using the quantum key distribution protocol for secure exchange of the PQC instead of physical face-to-face communication.

If encrypting large amounts of data rather than a single ASCII symbol, we would face additional challenges. This means that the general encryption protocol based on measuring expectation values of second order Pauli operators should provide additional degrees of security. We cannot use ASCII encoding for the plain text symbols. To see why, let us look at how ASCII encodes letters. First, we notice that all upper-case letters (from "A" to "Z") start with "10" as leading bits in the 7-bit binary encoding, while all lower-case letters (from "a" to "z") start with "11" as the two leading bits. Secondly, the letter encodings have unequal numbers of "0"s and "1"s. For example, upper-case "A" is encoded as 5 "0"s and 2 "1"s: 1000001. This means that a relatively simple frequency analysis can help to break the encryption given a sufficiently large amount of text.

Therefore, we need to generate a random mapping of the plain text symbols to bitstrings of fixed length simultaneously with the generation of the quantum circuit. The key condition is that the number of "0"s and "1"s in each randomly generated bitstring should be equal to prevent any possible attempt at frequency analysis.

A sample random mapping that illustrates this principle is shown in Table 1 in Appendix A. There, each plain text symbol is represented by a 12-bit bitstring with six "0"s and six "1"s. The table is shared with the trusted parties together with the generated quantum circuit.

Another consideration concerns the architecture of the quantum circuit. Security can be dramatically improved if the quantum circuit is modified by making the 1-qubit gates in the first layer adjustable as shown in Figure 4.

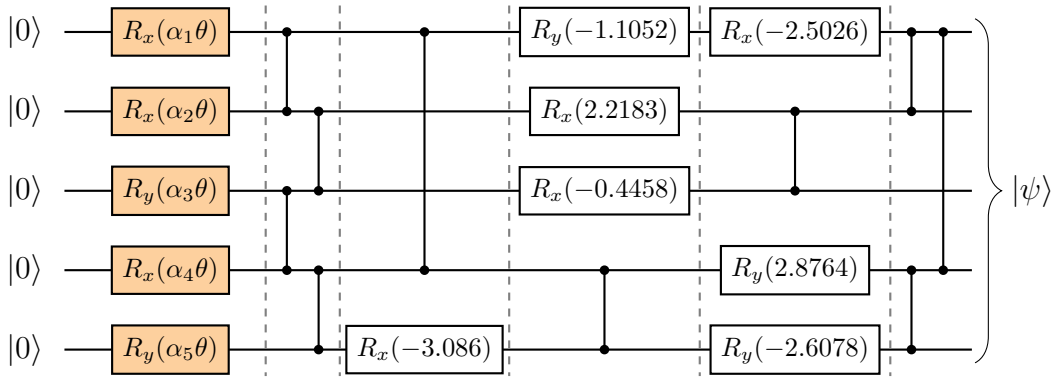

Figure 4: Quantum circuit with adjustable gates in the first layer. These gates are used as a "seed" to generate circuits that create different quantum states.

Making the first layer adjustable turns the generated quantum circuit into a de facto family of quantum circuits, where the configuration of adjustable parameters plays the role of a "seed" that can be set according to some rules. For example, it can be derived from the time stamp of some process as explained below. Without this functionality we will be limited in how much text we can encode before we start reusing the same pairs of Pauli operators with the same expectation values. With this functionality, assuming that we perform a regular reset of the seed, the same pairs of Pauli operators will have different expectation values for different seeds. This should prevent any meaningful attempt at analysis of large amounts of intercepted encrypted text as long as the seed is changed before the Pauli pairs are repeated. In Figure 4, the 1-qubit gates in the first layer are rotations around the x, y and z axes by angle $\alpha_i\theta$, where i indicates the qubit number. Coefficients $\alpha_i \in [-1, 1]$ are fixed. Parameter θ is a function of the time stamp of the generated vector of Pauli operators. As an illustration, we can adopt the following scheme for setting the value of θ.

Let us write down the time stamp in the format: YYMMDDhhmmss. For example, 13:03:46 on 27 November 2023 would be represented by the following integer number: 231127130346. Then the value of parameter θ is set equal to YYMMDDhhmmss modulo 2π. In our example it would be $\theta = 1.32392129$. The value of θ is unique for each time stamp. At the same time the knowledge of θ will be of no use unless the whole quantum circuit is known. It means that the time stamp can also be shared between the trusted parties using the unsecure channel – the same channel that is used for communicating the vector of Pauli operators. Making the parameters of

the first layer's gates depend on the time stamp ensures that it is fruitless to try to establish the mapping between the pairs of Pauli operators and their binary values: each random vector of Pauli operators that are used to encode the message will be accompanied by a unique quantum circuit given by the unique time stamp. This would allow us to reuse the same quantum circuit multiple times since every configuration of adjustable parameters in the first layer will lead to the corresponding unique quantum state in which Pauli operators are measured. The proposed symmetric encryption algorithm relies on the fact that the knowledge of which Pauli operators have been measured reveals nothing about their expectation values unless the quantum state in which they have been measured is also known.

It is also possible to transform the proposed algorithm into a multiple key encryption protocol. For example, Alice splits the quantum circuit into two parts: the first M_1 layers and the second M_2 layers ($M_1 + M_2 = M$). Then Alice shares the first M_1 layers with Bob and the second M_2 layers with Charlie. It would only be possible to perform decryption if both Bob and Charlie combine their quantum sub-circuits into a single complete quantum circuit.

The stylised example described at the start of Section 3 requires a face-to-face meeting of two trusted parties in order to exchange the PQC – a symmetric encryption/decryption key. In some cases this may be inconvenient if not altogether impossible. A natural solution that relies on the same type of security provided by the laws of quantum mechanics is to use one of the QKD protocols. The Bennett-Brassard (BB84) protocol [2, 49] is a potential choice.

QKD is a secure communication method that enables two parties to produce a shared random secret key known only to them, which then can be used to encrypt and decrypt messages. The main point here is that the secret key is a *random* bitstring of any desired length. Appendix C shows that the PQC (our secret key) consisting of N qubits and M layers of 1- and 2-qubit gates can be encoded as a bitstring of length $(a + b)NM$, with a-bit encoding of the gate type and b-bit encoding of the gate parameter (rotation angle). Equivalently, a random bitstring of length $(a + b)NM$ can be translated into the corresponding PQC.

5 General Algorithm

We can now formulate the general hybrid quantum-classical symmetric encryption/decryption Algorithm 1.

Algorithm 1: Symmetric Encryption

1: Generation of a random quantum circuit on N qubits with M layers of 1-qubit and 2-qubit gates (e.g., $M = \text{int}(2\sqrt{N})$). The 1-qubit gates are random rotations around the x, y and z axes; the 2-qubit gates are suitable native gates (e.g., CZ or $iSWAP$). The first layer of the quantum circuit consists of 1-qubit gates with parameters (rotation angles) of the form $\alpha_i\theta$, where all $\alpha_i \in [-1, 1]$, $i = 1, \ldots, N$, are randomly generated coefficients and the parameter θ is the same for all 1-qubit gates in the first layer.

2: Generation of a random mapping scheme that would provide a one-to-one mapping between any desired plain text symbol and a $2n$-digit bitstring with exactly n "0"s and exactly n "1"s.

3: Sharing of the generated quantum circuit and the plain text symbol mapping scheme with the trusted party. The quantum circuit is a symmetric key that can be used by the trusted parties to encrypt and decrypt data in binary format.

4: When one trusted party (Alice) wants to communicate with another trusted party (Bob), the protocol is as follows:

a) Alice converts the plain text she wants to encrypt into the bitstring using the generated mapping scheme.

b) Alice generates a random vector of Pauli operators $\{X, Y, Z\}$. The length of the vector should be equal to the width of the quantum circuit, with one-to-one mapping between the elements of the vector of Pauli operators and the qubits (e.g., $[Z_1, Y_2, X_3, \ldots, Y_N]$).

c) Alice saves the time stamp of the generated random vector of Pauli operators as an integer number in the format YYMMDDhhmmss. This number modulo 2π (double precision) is the value of parameter θ in the first layer of the generated quantum circuit.

d) Alice adds "change of basis" gates to each qubit in accordance with the Pauli operator which is assigned to the qubit:

- nothing to qubits with Pauli Z (z-basis);
- H to qubits with Pauli X (transformation from the z-basis to the x-basis);
- $HS^\dagger$ to qubits with Pauli Y (transformation from the z-basis to the y-basis).

e) Alice executes L runs of the quantum circuit and saves the measurement results in an $L \times N$ array:

- **if** "0" is measured
  then the eigenvalue of the corresponding Pauli operator is $+1$;
- **if** "1" is measured
  then the eigenvalue of the corresponding Pauli operator is -1.

Therefore, the array entries are either $+1$ or -1. The number of rows is equal to the number of quantum circuit runs and the number of columns is equal to the number of qubits.

f) Alice randomly selects two qubits i and j (without replacement). She calculates the expectation value of Pauli operators $\langle P_i P_j \rangle$ as the dot product of columns i and j, normalised by the number of rows:

- **if** the value of $\langle P_i P_j \rangle$ is larger (closer to $+\infty$) than ϵ
  then this pair of qubits can be used to encode "1";
- **if** the value of $\langle P_i P_j \rangle$ is smaller (closer to $-\infty$) than $-\epsilon$
  then this pair of qubits can be used to encode "0".

The value of ϵ is one of the algorithm's parameters. The broad condition is $\epsilon \geq m\sigma$, where σ is the estimated average standard deviation of expectation values. The value of m can be chosen from some general considerations.

4: g) Alice takes the first bit from the bitstring she wants to encrypt:
if the bit value is "1" and $\langle P_i P_j \rangle > \epsilon$ **or** the bit value is "0" and $\langle P_i P_j \rangle < -\epsilon$
then Alice encrypts this bit with the pair (i, j)
else the pair (i, j) is either discarded (if $-\epsilon \leq \langle P_i P_j \rangle \leq \epsilon$) or put on hold (for the next suitable bit), and Alice tries another random pair of Pauli operators (without replacement).

h) This process continues (steps (f) and (g) are repeated) until Alice encrypts the first K bits with the pairs of qubit indices. The broad condition is $K < N(N-1)/2$.

i) After that Alice generates a new random vector of Pauli operators, saves its time stamp as an integer number in the format YYMMDDhhmmss, and repeats the above procedure for the next K bits.

j) This process continues until Alice encrypts the whole bitstring (the whole dataset in binary format). If the bitstring length is B, then we end up with D vectors of Pauli operators, D time stamps, and D cycles of quantum circuit runs: $(D-1)K < B \leq DK$.

5: The encrypted dataset consists of three arrays:

a) $D \times N$ array of Pauli operators (D rows, N columns), where each kth row is the vector of Pauli operators used to encrypt $[(k-1)K+1, kK]$ section of the bitstring.

b) $D \times 1$ array of time stamps (a time stamp for each vector of Pauli operators).

c) $D \times K$ array of pairs of qubit indices (D rows, K columns), where each kth row is the vector of qubit index pairs used to encrypt $[(k-1)K+1, kK]$ section of the bitstring.

Alice sends the first and second arrays to Bob via their preferred unsecure communication channel (e.g., email). Then Alice sends the third array to Bob via the same or a different unsecure channel (e.g., second email).

6: After receiving the arrays from Alice:

a) Bob takes the first value from the time stamp array as an integer number in the format YYMMDDhhmmss. This number modulo 2π is the value that should be assigned to the parameter θ in the first layer of the quantum circuit.

b) Bob takes the first row from the Pauli operator array and adds the corresponding basis transformation gates to the quantum circuit – exactly the same procedure as done by Alice. Bob runs the quantum circuit L times and saves the measurement results $(+1, -1)$ in an $L \times N$ array.

c) Bob takes the first element (i, j) from the index pairs array and calculates the dot product of columns i and j from the $L \times N$ array:
if the value of the dot product of columns i and j is positive
then Bob decrypts (i, j) as "1"
else Bob decrypts (i, j) as "0".
Bob continues this procedure until he reaches the end of the first row of the index pairs array (i.e., until he decrypts first K bits).

6: d) Bob switches to the second row of the Pauli operators array and repeats steps (a) and (b). He continues until the whole bitstring is decrypted.
e) Bob translates the decrypted bitstring into plain text using the same mapping scheme as Alice.

Components of the algorithm are illustrated in Figure 16 (Appendix D).

Remark: With $N = 2,500$, $M = 100$, $L = 500,000$, $K = 2,500,000$, $\epsilon = 0.01$, $m = 6$, and $n = 6$, Algorithm 1 would allow encryption of 65 full pages of dense normal text per single vector of Pauli operators (1 page = 40 lines $\times$ 80 symbols per line $\times$ 12 bits per symbol). The 6-sigma threshold is equivalent to 1 typo per about 26,000 pages of dense normal text (see Appendix B for the determination of ϵ).

6 Encryption and Decryption Example

The following example illustrates the application of Algorithm 1 to the encryption and decryption of a meaningful amount of plain text. The sample message is taken from the short story by Edgar Allan Poe, *The Gold-Bug* [50]:

```
A good glass in the bishop's hostel in the devil's seat.

Forty-one degrees and thirteen minutes northeast and by north.

Main branch seventh limb east side.

Shoot from the left eye of the death's-head.

A bee line from the tree through the shot fifty feet out.
```

In the novel, this message was encrypted by Scottish privateer Captain Kidd using simple substitution cipher and successfully decrypted more than a century later by the novel's protagonist, William Legrand, using frequency analysis[1].

We will encrypt and then decrypt this message first on a quantum simulator and then on the actual quantum computer (Rigetti's Ankaa-2 QPU). The message is 258 characters long (counting spaces) and should be translated into the 3,096-bit string using the mapping Table 1 (Appendix A) before encryption.

6.1 Quantum Simulator

Figure 5 displays embedding of a 20-qubit quantum circuit on the square-grid QPU graph. Even though we can easily realise the "all-to-all" qubit connectivity on a quantum simulator, our objective is to stay within the restrictions imposed by the limited connectivity of existing quantum processors. The corresponding quantum circuit is shown in Figure 6. It consists of three layers of 1-qubit gates with adjustable parameters (R_x, R_y) and three layers of fixed 2-qubit gates ($iSWAP$).

[1]The punctuation has been changed from its original version to make the message more readable.

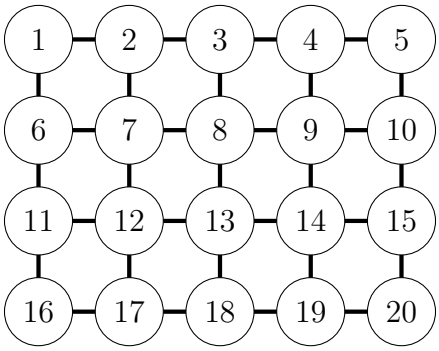

Figure 5: Embedding of a 20-qubit quantum circuit on the square-grid QPU graph. Each circle represents a qubit. The connecting lines show the physical connections between qubits on the device. The 2-qubit gate operations can be applied directly to these physically connected qubits.

In total, the quantum circuit is configured by 100 adjustable parameters. Each 1-qubit rotation angle ϕ_i^j is defined on the interval $[-\pi/2, \pi/2]$, thus attempting to provide sufficiently uniform coverage of the qubit states on the Bloch sphere.

Since our task is to encode 3,096 bits and a single cycle (single set of basis gates) can produce at most $(20 \cdot 19)/2 = 190$ unique Pauli pairs, we have to run multiple cycles when working with the 20-qubit circuit. Also, not all Pauli pairs will be eligible for encoding purposes due to the finite number of quantum circuit runs as explained in Appendix B. The analysis presented there and shown in Figure 12 indicates that with 10,000 quantum circuit runs we need to set the expectation value threshold parameter ϵ to 0.05 in order to ensure the 6 standard deviations confidence level. This is a demanding requirement that would push the number of cycles up significantly unless we increase the number of quantum circuit runs and, therefore, decrease the threshold value.

However, we can decrease the value of ϵ without increasing the number of quantum circuit runs and without sacrificing the accuracy by applying an error detection and error correction technique suggested by the very nature of the proposed encryption protocol. The encryption protocol requires every plain text character to be represented by a bitstring with equal numbers of "0"s and "1"s. For example, six "0"s and six "1"s as per the mapping Table 1. If the decrypted bitstring has unequal number of "0"s and "1"s, then we immediately know that the bitstring has been encrypted/decrypted incorrectly. As long as the error rate remains low, it is highly unlikely that two errors (two bit-flips) would happen for the same bitstring and, therefore, all occurred errors can be detected.

The encryption protocol also suggests an obvious error correction mechanism. For example, if the decrypted bitstring has seven "0"s and five "1"s, we can try to flip each "0" into "1" one by one and test the obtained seven new bitstrings with equal number of "0"s and "1"s as to whether they encode a plain text character. This is the approach we are taking in our experiment with the 20-qubit circuit given by Figure 6.

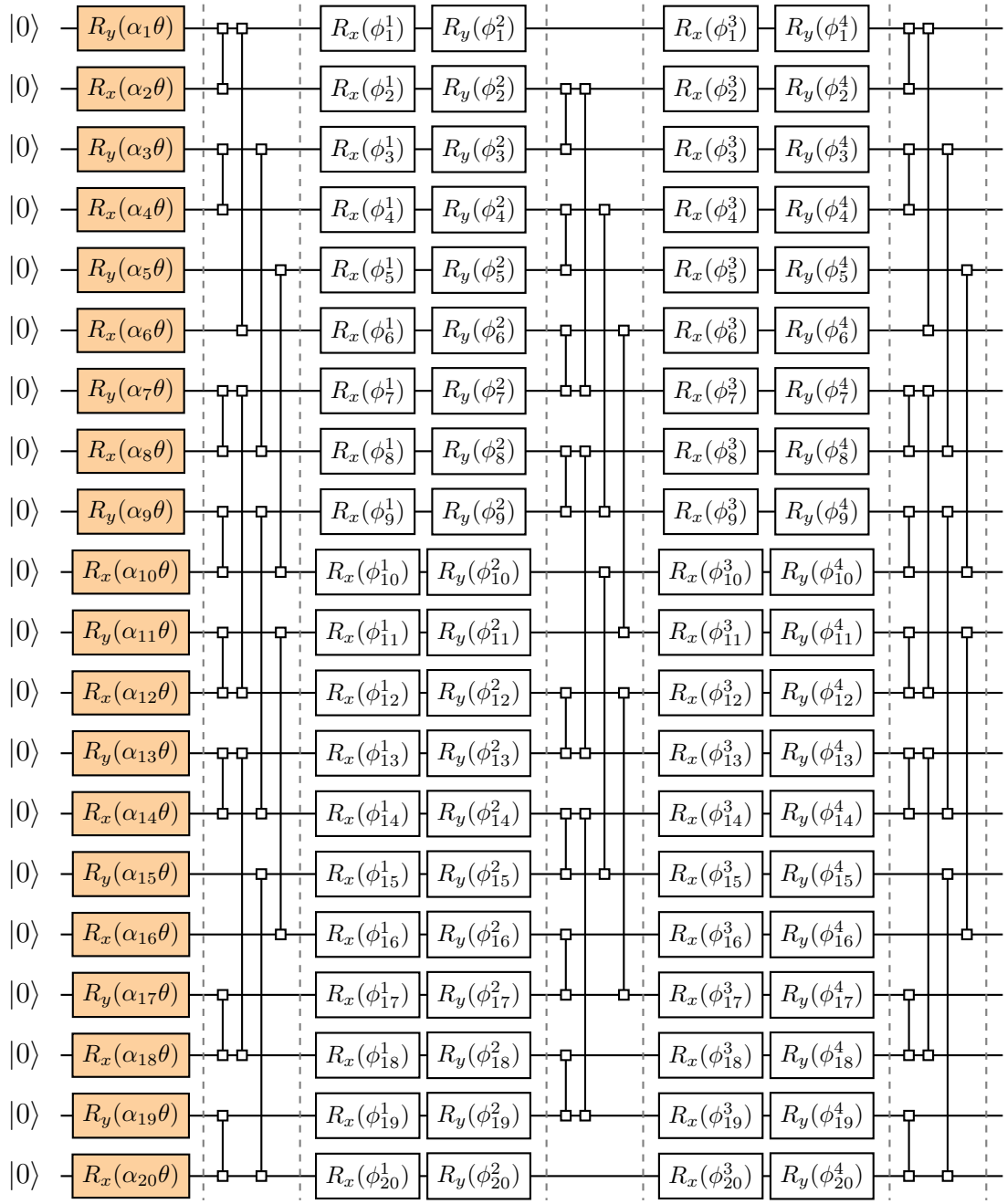

Figure 6: 20-qubit quantum circuit with 100 configurable parameters. The fixed 2-qubit gates are *iSWAP* gates. The 2-qubit gates are applied between qubits which have physical connections on the device (see Figure 5).

The numerical experiment consists of executing the encryption and decryption protocol end-to-end as specified by Algorithm 1 with 10,000 quantum circuit runs while varying the values of the threshold parameter ϵ.

In our experiment, we observed error-free encryption and decryption for $\epsilon > 0.035$.

Once the value of ϵ went below 0.035 we started to see instances of incorrect encryption and decryption. A typical decrypted message would then look as follows:

```
A good glass in the bishop's hostel in the devil's seat.

Forty-one degrees and thirteen minutes northeast and by north.

Main branch sev$nth limb east side.

Shoot from the left eye of the death's-head.

A bee line from the tree through the $hot fifty feet out.
```

Here, symbol `$` indicates detected incorrectly encrypted/decrypted character. The application of the error correction mechanism outlined above restores the message to its original form:

```
A good glass in the bishop's hostel in the devil's seat.

Forty-one degrees and thirteen minutes northeast and by north.

Main branch seventh limb east side.

Shoot from the left eye of the death's-head.

A bee line from the tree through the shot fifty feet out.
```

The first incorrect bitstring has been error-corrected to the bitstring representing character "e" and the second incorrect bitstring has been error-corrected to the bitstring representing character "s". In some cases the proposed error correction method can suggest several valid substitutions. To ensure uniqueness, the mapping between plain text characters and the corresponding bitstrings should be organised in such a way that the Hamming distance between any two bitstrings is > 2. This may require longer bitstrings (e.g., 14-bit long) but would significantly increase the number of eligible Pauli pairs – often by a substantial multiple for relatively small number of quantum circuit runs.

For $\epsilon < 0.03$ we observe instances of more than one error per bitstring and, therefore, can no longer rely on the error correction. In this case, the solution is to increase the number of quantum circuit runs.

6.2 Rigetti Ankaa-2 QPU

The 20-qubit circuit used in the numerical experiments on a quantum simulator can be seen as a "proof of concept" exercise. However useful, the quantum simulator run on a classical hardware has obvious limitation: memory. Increasing the qubit count from 20 to 30 would slightly more than double the number of eligible Pauli pairs per cycle but would increase the number of probability amplitudes of the quantum state (and, therefore, the memory requirements) by three orders of magnitude. Going beyond 70 qubits becomes prohibitively expensive from the computational perspective even for large high performance computing (HPC) systems.

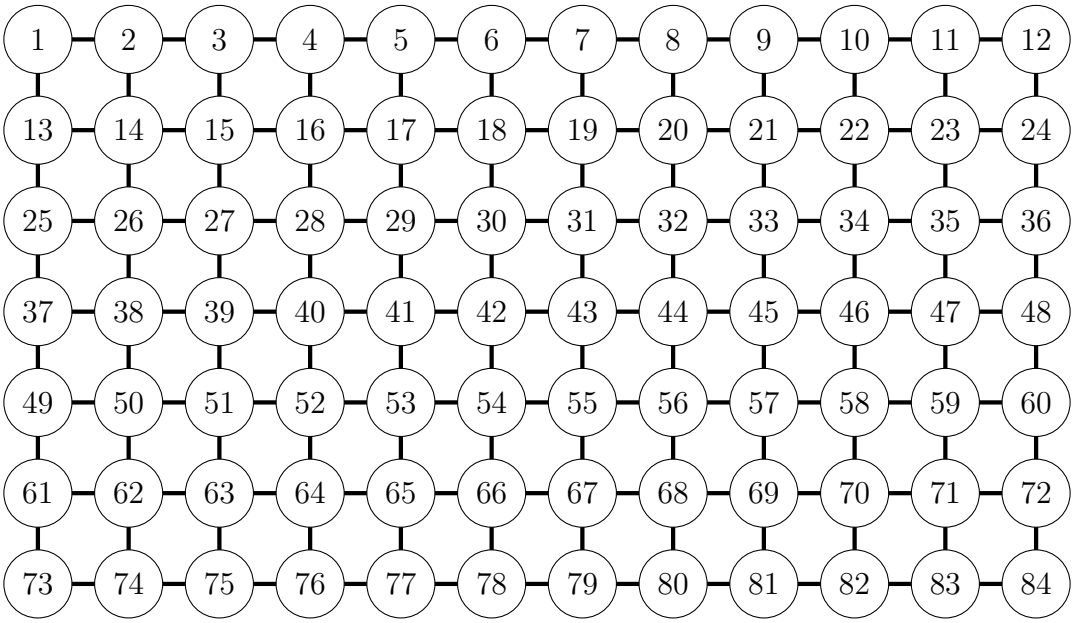

Figure 7: Embedding of an 84-qubit circuit on Ankaa-2 QPU graph. Each circle represents a qubit. The connecting lines show the physical connections between qubits on the device. The 2-qubit gate operations can be applied directly to these physically connected qubits.

This is why experiments with 70+ qubit circuit executed on the actual quantum hardware is so important – we are entering the realm of quantum computing that cannot be simulated classically. We also aim to verify the possibility of executing the proposed symmetric encryption protocol on NISQ computers. Figure 7 shows embedding of an 84-qubit circuit on Rigetti's Ankaa-2 square-grid QPU graph.

The 84-qubit circuit is structurally similar to the 20-qubit circuit we used in numerical experiments executed on a quantum simulator. The circuit consists of three layers of 1-qubit gates (R_x, R_y) and three layers of 2-qubit gates ($iSWAP$). The total number of configurable parameters is 420.

We are interested in finding the level of critical threshold for the given number of quantum circuit runs – for any threshold value larger than critical we should not expect to see more than one error per bitstring. As long as there is only one bit-flip in any given bitstring we can easily detect it and apply the error correction mechanism described above. But once we start observing two or more bit-flips per bitstring, the encryption/decryption process breaks down completely.

Figure 8 shows the values of critical threshold, ϵ^*, as a function of the number of quantum circuit runs. Results are shown for three separate experiments, as represented by the green, orange and blue points/lines. These curves are purely indicative as they depend on the specific quantum circuit architecture and are inherently noisy. However, they provide valuable insight into the feasibility of executing the proposed symmetric encryption protocol on NISQ hardware. Only three experiments could be performed due to limited access to the QPU and the time required to complete each experiment.

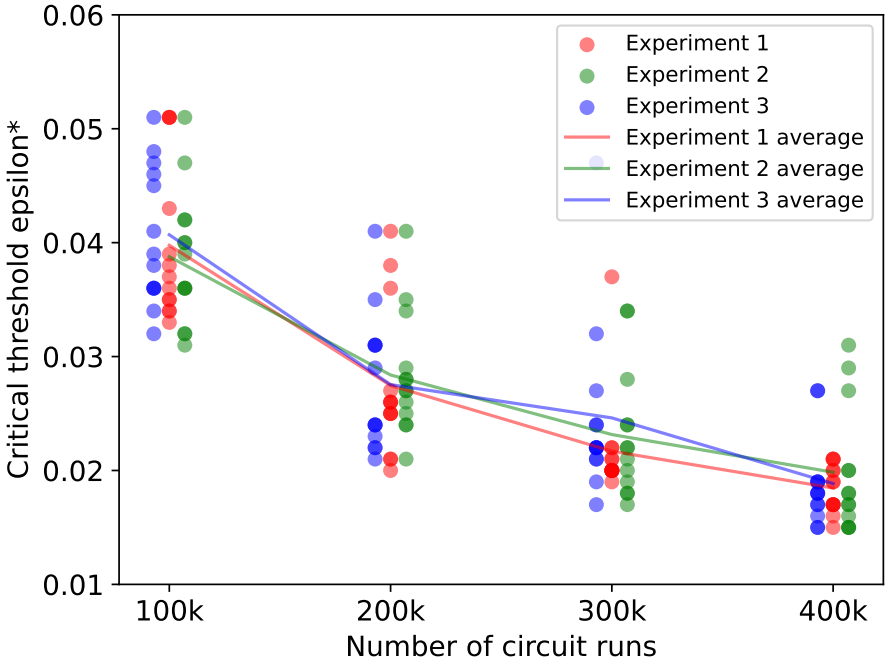

Figure 8: Critical threshold ϵ^* for various numbers of quantum circuit runs. Three separate experiments are represented by the green, orange and blue points respectively. Each experiment has thirteen ϵ^* points (one for each cycle) plotted at each of the number of circuit runs. The lines are a visual aid connecting the average of the thirteen points, for each of the number of circuit runs.

Figure 9 illustrates the share of ineligible Pauli pairs as a function of the number of quantum circuit runs, corresponding to the points in Figure 8. Noise of the QPU contributes significantly to attenuation of the magnitude of the expectation values and, therefore, to the percentage of ineligible Paulis. We cannot derive the percentage for a noiseless case at this scale, and discrimination between the effects of circuit size and noise to the magnitude of the expectation values is an important question for future work. Appendix B.2 contains results for a 5-qubit circuit showing lower numbers of ineligible Pauli pairs in simulated results compared to the QPU results.

To reduce the impact of QPU calibration drift with time, we developed an additional error mitigation technique. Each expectation value was calculated four times, each using 100,000 shots. Averaging one, two, three or four of these values generated estimates of the expectations using 100K, 200K, 300K and 400K shots respectively. The standard deviations were also calculated. Expectation values with large standard deviations could be discarded as these had a higher likelihood of being inaccurate, potentially changing sign and thereby leading to errors. This mitigation technique could be applied when encrypting data bits. However, it can't be used to exclude expectation values when decrypting as the decryption algorithm has to use the expectations specified by the encryption process (via the list of pairs). The decryption algorithm can use the standard deviation information in situations where an error occurs and results in more than one potential decoded character. The character

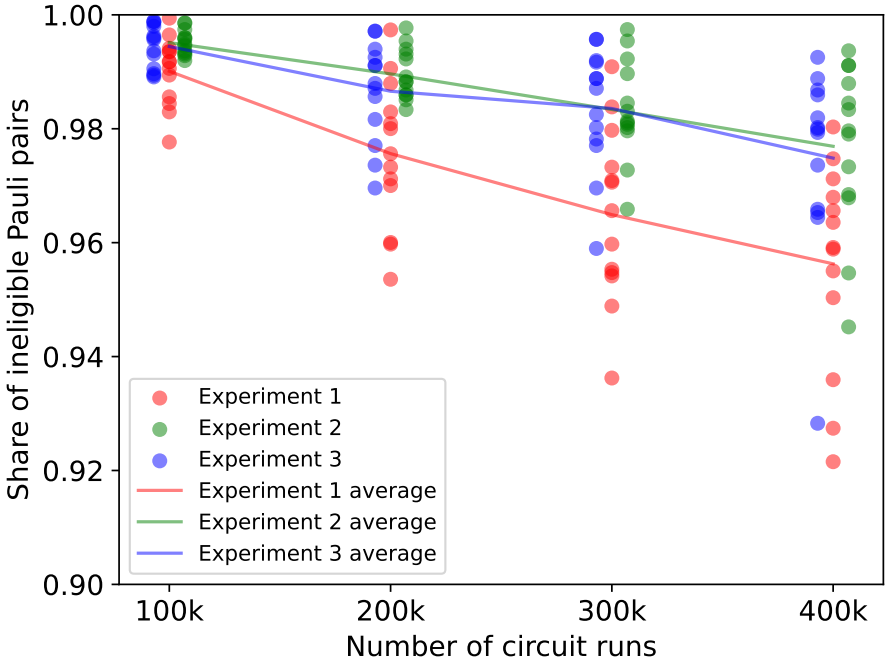

Figure 9: Share of ineligible Pauli pairs for various numbers of quantum circuit runs. Three separate experiments are represented by the green, orange and blue points respectively. Each experiment has thirteen points (one for each cycle) plotted at each of the number of circuit runs. The lines are a visual aid connecting the average of the thirteen points, for each of the number of circuit runs.

resulting from flipping the bit which has the largest standard deviation is likely to be the error that occurred.

Where possible, sampling without replacement was used. As long as there were at least 12 unique expectations then these could be used to encode a character (needing 12 bits) and accurately decode it even in the presence of a single expectation value error.

Note that since Pauli values on separate qubits commute, for unique Pauli operators P_k and P_l on qubits k and l, the second order Pauli $P_k \otimes P_l$ is equal to $P_l \otimes P_k$. Therefore, we can define the pair (k, l) where $k < l$ to represent the expectation value E of $P_k \otimes P_l = P_l \otimes P_k$ and we can define pair (l, k) where $k < l$ to represent the negated expectation value $-E$. This means that each expectation value measured can be used to represent either a "0" or a "1". This removes any issues related to unequal numbers of positive and negative Pauli expectations.

7 Conclusions and Outlook

This paper proposes a hybrid quantum-classical symmetric encryption algorithm. Encoding and decoding the message relies on computing expectation values of a random parameterised quantum circuit. Some non-secret information is sent via an unsecured, one-way classical channel. The PQCs enable long messages to be

transmitted securely with compact shared secrets, which are the PQC structure and the symbol cypher. A stylised example was provided, followed by the detailed general algorithm.

Initial experiments were performed using a 5-qubit PQC run on a quantum simulator and Rigetti Ankaa-2 quantum hardware. Using 100,000 quantum circuit runs of the simulator, it was found that about 20% of the second order Pauli expectation values measured were ineligible to use as their magnitude was less than six standard deviations of the noise. The same experiment on the quantum hardware resulted in about 57% ineligible second order Pauli expectation values as the noise levels were higher. There was an almost equal number of measured positive and negative expectation values, meaning that a similar amount of "1"s and "0"s could be encrypted.

Further "proof of concept" experiments were conducted on a quantum simulator (20-qubit circuit) and quantum hardware using the Rigetti Ankaa-2 QPU (84-qubit circuit). Error detection and error correction techniques were applied in order to increase the percentage of eligible Pauli pairs for the given number of quantum circuits runs, which further improves the efficiency of the proposed protocol. We note that it is beyond the capabilities of most existing classical computers to simulate the 84-qubit quantum circuit.

Future work will continue to investigate running the algorithm on quantum hardware. This will enable wider and deeper PQC to be utilised. The impact of noise will be observed and quantified. Using simulation and quantum hardware, the relationship of Figure 10 will be examined and verified for larger quantum circuits.

While fault-tolerant quantum computers may be required to run circuits using thousands of qubits, initial results could be obtained for hundreds of qubits on Noisy Intermediate Scale Quantum (NISQ) era computers [51]. The algorithm requirement for calculating expectation values of Pauli observables means that error mitigation techniques can be utilised [52]. The availability of more qubits is beneficial as it provides additional second order Pauli observables to encode the classical bits thereby improving the efficiency (see Appendix C for details).

Some theoretical aspects that were touched upon by this work are left for future investigation.

First, we consider the distribution of the 2-qubit Pauli observables. In our example, they are a small subset drawn randomly and without replacement, and their expectation depends on the random PQC. Therefore, one may conjecture that these expectation values should be approximately evenly distributed over their support. However, certain types of PQC may suffer from the *barren plateau* phenomena, which dictates the concentration of expectation values and their exponential reduction of the expectation's variances as the qubit count increases [53]. If our circuits were to be affected, this may complicate the choice of second-order Pauli operators as we add more qubits. Although we did not investigate this phenomenon, a possible mitigation is to reduce the asymptotic depth growth of the circuit as a logarithm of the qubit number, for which barren plateaus are expected not to occur.

Second, one of the possible attacks on this protocol entails eavesdropping on the unsecured communication channel and performing a frequency attack. Although we did not explicitly address this case, a recent result states that it is exponentially difficult to learn quantum circuits of bounded 2-qubit gate numbers from their samples [48].

Third, Pauli expectation values can be classically simulated efficiently if the generating circuit contains only a small number of non-Clifford gates [54]. We did not study if this aspect may pose a security threat to our algorithm or make it more susceptible to specific attacks. However, this should not raise immediate concern as the number of non-Clifford gates can be chosen freely within the depth-scaling constraint. Additionally, the output of the circuit is privy to the communicating parties only.

Fourth, our explicit example demonstrates that a hybrid quantum-classical symmetric algorithm is within reach of current quantum computation – in our view, this is a key aspect of this work. Moreover, some elements of the algorithm may help achieve secure communication with fewer bits of information exchanged as secrets. It is of future interest to assess whether or not this algorithm could be more efficient in an end-to-end case than a specific classical alternative, such as the classical one-time pad.

Our hope is that cryptanalysis experts can take these ideas further, enabling the creation of a new efficient symmetric encryption algorithm.

Bibliography

[1] Niraj Kumar, Iordanis Kerenidis, and Eleni Diamanti. Experimental demonstration of quantum advantage for one-way communication complexity surpassing best-known classical protocol. *Nature Communications*, 10(1), September 2019.

[2] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Proceedings of International Conference on Computers, Systems and Signal Processing*, page 175–179, 1984.

[3] Isaac L. Chuang Michael A. Nielsen. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 10 anv edition, 2011.

[4] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[5] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, April 2021.

[6] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the ACM*, pages 120–126, 1978.

[7] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[8] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[9] Paul Grassi, Elaine Newton, James Fenton, Ray Perlner, Andrew Regenscheid, William Burr, Justin Richer, Naomi Lefkovitz, Jamie Danker, Yee-Yin Choong, Kristen Greene, and Mary Theofanos. Digital identity guidelines: Authentication and lifecycle management. *NIST special publication 800-63b*, 2017.

[10] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[11] IBM. IBM debuts next-generation quantum processor & IBM Quantum System Two, extends roadmap to advance era of quantum utility. `https://newsroom.ibm.com/2023-12-04-IBM-Debuts-Next-Generation-Quantum-Processor-IBM-Quantum-System-Two,-Extends-Roadmap-to-Advance-Era-of-Quantum-Utility`, December 2023.

[12] IonQ. How we achieved our 2024 performance target of #aq 35. `https://ionq.com/posts/how-we-achieved-our-2024-performance-target-of-aq-35`, January 2024.

[13] Rigetti. Rigetti investor presentation. `https://investors.rigetti.com/static-files/fbac3801-223f-4f0f-a207-47d25084a1d7`, November 2023.

[14] Carl Pomerance. The number field sieve. *Proceedings of Symposia in Applied Mathematics*, 48, 1994.

[15] Daniel J. Bernstein. *Introduction to post-quantum cryptography*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[16] NIST. Post-quantum cryptography pqc - selected algorithms 2022, July 2022.

[17] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. Advanced encryption standard (aes), Nov 2001.

[18] Ray Perlner and David Cooper. Quantum resistant public key cryptography: A survey. *IDtrust '09: Proceedings of the 8th Symposium on Identity and Trust on the Internet*, April 2009.

[19] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, 1996.

[20] Steven M. Bellovin. Frank miller: Inventor of the one-time pad. *Cryptologia*, 35(3):203–222, 2011.

[21] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[22] Claude E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[23] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, Mar 1993.

[24] Michele Mosca, Alain Tapp, and Ronald de Wolf. Private Quantum Channels and the Cost of Randomizing Quantum Information. *arXiv e-prints*, pages quant–ph/0003101, March 2000.

[25] Charles H. Bennett, Patrick Hayden, Debbie W. Leung, Peter W. Shor, and Andreas Winter. Remote preparation of quantum states. *arXiv e-prints*, pages quant–ph/0307100, July 2003.

[26] Daniel Gottesman. Uncloneable Encryption. *arXiv e-prints*, pages quant–ph/0210062, October 2002.

[27] Jonathan Oppenheim and Michał Horodecki. How to reuse a one-time pad and other notes on authentication, encryption, and protection of quantum information. *Phys. Rev. A*, 72(4):042309, October 2005.

[28] A. Beige, B. G. Englert, Ch. Kurtsiefer, and H. Weinfurter. Secure Communication with a Publicly Known Key. *Acta Physica Polonica A*, 101(3):357, March 1999.

[29] Kim Boström and Timo Felbinger. Deterministic secure direct communication using entanglement. *Phys. Rev. Lett.*, 89:187902, Oct 2002.

[30] Alexander Lipton and Adrien Treccani. *Blockchain and Distributed Ledgers: Mathematics, Technology, and Economics*. World Scientific, 2022.

[31] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, Nov 2019.

[32] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. Expressive power of parameterized quantum circuits. *Physical Review Research*, 2(033125), 2020.

[33] Aram W. Harrow and Ashley Montanaro. Quantum computational supremacy. *Nature*, 549(7671):203–209, September 2017.

[34] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.

[35] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 59–68, New York, NY, USA, 2003. Association for Computing Machinery.

[36] Scott Aaronson. How Much Structure Is Needed for Huge Quantum Speedups? *arXiv e-prints*, page arXiv:2209.06930, September 2022.

[37] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[38] Max Born. Zur Quantenmechanik der Stoßvorgänge. *Zeitschrift fur Physik*, 37(12):863–867, December 1926.

[39] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, Nov 1995.

[40] Tycho Sleator and Harald Weinfurter. Realizable universal quantum logic gates. *Phys. Rev. Lett.*, 74:4087–4090, May 1995.

[41] Robert S. Smith, Eric C. Peterson, Mark G. Skilbeck, and Erik J. Davis. An open-source, industrial-strength optimizing compiler for quantum programs. *Quantum Science and Technology*, 5(4):044001, October 2020.

[42] Fei Hua, Meng Wang, Gushu Li, Bo Peng, Chenxu Liu, Muqing Zheng, Samuel Stein, Yufei Ding, Eddy Z. Zhang, Travis S. Humble, and Ang Li. QASMTrans: A QASM based Quantum Transpiler Framework for NISQ Devices. *arXiv e-prints*, page arXiv:2308.07581, August 2023.

[43] Francisco J. R. Ruiz, Tuomas Laakkonen, Johannes Bausch, Matej Balog, Mohammadamin Barekatain, Francisco J. H. Heras, Alexander Novikov, Nathan Fitzpatrick, Bernardino Romera-Paredes, John van de Wetering, Alhussein Fawzi, Konstantinos Meichanetzidis, and Pushmeet Kohli. Quantum Circuit Optimization with AlphaTensor. *arXiv e-prints*, page arXiv:2402.14396, February 2024.

[44] Daniel Gottesman. The Heisenberg Representation of Quantum Computers. *arXiv e-prints*, pages quant–ph/9807006, July 1998.

[45] Antoine Jacquier and Oleksiy Kondratyev. *Quantum Machine Learning and Optimisation in Finance: On the Road to Quantum Advantage*. Packt, 2022.

[46] Daniel Koch, Laura Wessing, and Paul M. Alsing. Introduction to Coding Quantum Algorithms: A Tutorial Series Using Pyquil. *arXiv e-prints*, page arXiv:1903.05195, March 2019.

[47] Ryan LaRose. Overview and Comparison of Gate Level Quantum Software Platforms. *Quantum*, 3:130, March 2019.

[48] Haimeng Zhao, Laura Lewis, Ishaan Kannan, Yihui Quek, Hsin-Yuan Huang, and Matthias C. Caro. Learning quantum states and unitaries of bounded gate complexity, 2023.

[49] Chris Bernhardt. *Quantum Computing for Everyone*. MIT Press, 2019.

[50] Edgar Allan Poe. *The Complete Tales and Poems of Edgar Allan Poe*. Vintage, 1975.

[51] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[52] Andrew Arrasmith, Andrew Patterson, Alice Boughton, and Marco Paini. Development and Demonstration of an Efficient Readout Error Mitigation Technique for use in NISQ Algorithms. *arXiv e-prints*, page arXiv:2303.17741, March 2023.

[53] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), November 2018.

[54] Tomislav Begušić, Kasra Hejazi, and Garnet Kin-Lic Chan. Simulating quantum circuit expectation values by clifford perturbation theory, 2023.

[55] Rigetti. Ankaa-2 quantum processor. `https://qcs.rigetti.com/qpus`, March 2024.

A Plain Text Symbol to Bitstring Encoding Scheme

Symbol	Encoding	Symbol	Encoding	Symbol	Encoding
A	111100100010	a	100110001011	0	101010110100
B	111101010000	b	100001100111	1	001001101011
C	011101101000	c	100000101111	2	001010110101
D	011011001010	d	101100101010	3	100010101101
E	110001110010	e	101000001111	4	101000110011
F	110100010101	f	011110101000	5	010100011011
G	110100111000	g	100111101000	6	011010001110
H	010010101101	h	001011000111	7	011111000010
I	001110011010	i	000100101111	8	011100001011
J	011110010100	j	001110001101	9	110100001101
K	001011010101	k	100101010011	+	010101100101
L	100001110110	l	110010011010	-	011001011010
M	110101010010	m	100010101011	*	001011110001
N	011011000110	n	010101101100	/	011001001110
O	001110001011	o	011101011000	=	001101000111
P	101011000011	p	101011010100	(	110000110011
Q	011110010001	q	101000011101	)	010100111001
R	100100011101	r	101101010100	.	001110101001
S	101111100000	s	010101010011	,	101001110010
T	001111011000	t	011001101001	:	010011000111
U	101100010101	u	111000010101	;	110010101010
V	110100011010	v	111101001000	?	001101100011
W	111110000100	w	110111010000	!	110010110100
X	111011000001	x	001101001011	”	100110110100
Y	101101001001	y	000111001011	’	101010011010
Z	100001101101	z	101110110000	*space*	000110010111

Table 1: A sample encoding scheme (randomly generated). It is used to convert a plain text character to a bitstring consisting of equal numbers of “0” and “1”.

B Expectation Values of Second Order Paulis

In the following, we consider situations where the quantum circuit can be executed on either noise-free hardware (a quantum simulator) or a noisy quantum processing unit (QPU).

B.1 Noiseless simulation

Figure 10 displays a cumulative distribution function (CDF) of the absolute values of second order Pauli expectation values:

$$F(X) = \text{Probability}\left(|\left\langle P_k P_l \right\rangle| \leq X\right).$$

The quantum circuit structure used to build the CDF corresponds to the one given by Figure 3 with random rotation angles and random choices of measurement bases. In total, we used 5,000 random configurations (rotation angles and Pauli pairs) to build the CDF. Each expectation value was calculated with 1,000,000 quantum circuit runs, that is bitstring samples. The unique configurations were created by five different allocations of H and $HS^\dagger$ gates in the final layer of the quantum circuit and 100 randomly generated configurations of rotation angles for each allocation of H and $HS^\dagger$ gates. Taking into account that the 5-qubit quantum circuit can support the calculation of expectation values for 10 unique Pauli pairs, the total number of calculated second order Pauli expectation values is $5 \times 10 \times 100 = 5,000$.

We also calculated the standard deviations of expectation values for 50 random Pauli pairs and circuit configurations. The average standard deviation has been estimated to be 0.00096 for 1,000,000 quantum circuit runs. Applying the 6-sigma rule (standard confidence bands widely used in experimental disciplines to claim a scientific discovery) we get an estimate for the value of ϵ: 0.0058.

The right chart in Figure 10 shows that 12.1% of second order Pauli expectation values can be found in the interval $[-0.0058, 0.0058]$. This is the proportion of Pauli pairs that must be discarded: if the expectation value of a second order Pauli is close to zero we cannot be certain that its sign is correct and that we will observe the same sign when this expectation value is calculated next time. Parameter ϵ provides conservative confidence bands.

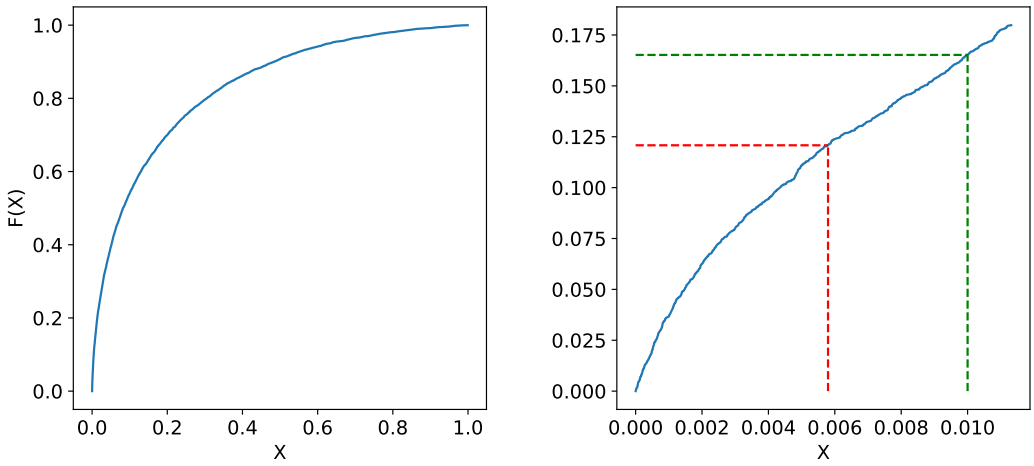

Figure 10: Cumulative distribution function for second order Pauli expectation magnitudes across 5,000 randomly generated configurations; 100,000 quantum circuit runs; noise-free quantum simulator. The 6-sigma threshold is $\epsilon = 0.0058$.

In practice, we may want to add a safety margin to account for the "model risk" and set $\epsilon = 0.01$. In this case, the proportion of Pauli pairs that must be discarded as ineligible for encoding increases to 16.5% or, roughly, one in six.

We can only rely on the 6-sigma rule if the noise in the estimation of second order

Pauli expectation values is normally distributed, i.e., there are no heavy tails. Let $\overline{X}$ be the true expectation value of a Pauli pair and X be the estimate of the expectation value of the same Pauli pair obtained with L quantum circuit runs. Then the noise, ξ, can be quantified as the deviation from the true value:

$$\xi = X - \overline{X}.$$

If we repeat the calculation of X many times, we can use the sample mean $\hat{X}$ as an approximation of the true expectation value $\overline{X}$:

$$\xi = X - \hat{X}.$$

Figure 11 displays the distribution of ξ as the deviation from the sample mean. The distribution is bell-shaped without heavy tails. The vertical dashed red lines indicate 1 standard deviation bands that contain approximately 2/3 of the dataset population. This suggests that the application of the 6-sigma rule is a reasonable and workable approach.

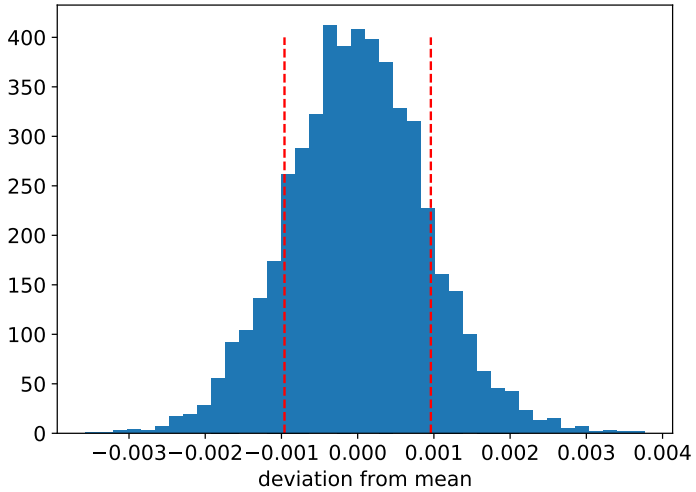

Figure 11: Histogram of the noise component in the estimation of second order Pauli expectation values. Vertical dashed red lines indicate 1 standard deviation bands.

Measurement noise depends on the number of shots performed on the quantum computer and this noise can be measured and quantified. To obtain good results, noise due to gate errors should be less than the measurement noise.

Executing 1,000,000 runs of the quantum circuit can be time consuming. We are talking about tens of seconds if the circuit is run on a superconducting qubit processor. This can be justified if we are dealing with a large amount of data that must be encrypted/decrypted. However, it may not be the most efficient solution for datasets of relatively small sizes. A faster protocol would trade an increase in speed (i.e., smaller number of runs) for a smaller number of eligible Pauli pairs (due to an increase in the noise). This may be a sensible trade-off for small datasets.

Figure 12 shows the dependence of ϵ (6 standard deviations) and the proportion of ineligible Pauli pairs as functions of the number of quantum circuit runs. A short message can be encrypted in milliseconds at the price of throwing out 2/3 of possible Pauli pairs, which should not be a problem for a short message and sufficiently wide quantum circuit.

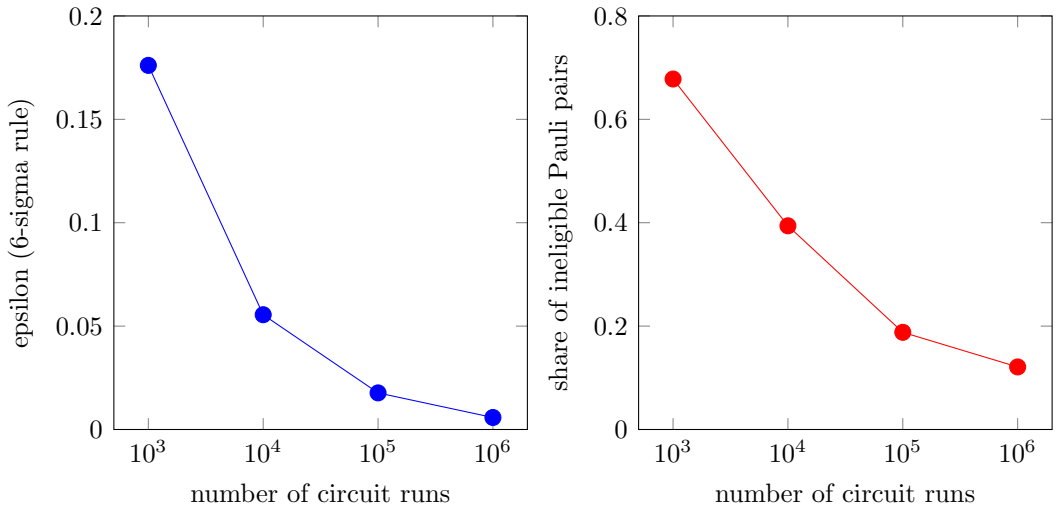

Figure 12: Dependence of encryption algorithm parameters on the number of quantum circuit runs. The left plot shows how ϵ (6 standard deviations) depends on the number of quantum computer runs. The right plot displays how the share of ineligible Pauli pairs depends on the number of quantum computer runs.

B.2 Noisy hardware

Although the proposed algorithm has been specified with fault-tolerant quantum computers in mind, it would be instructive to run it on existing noisy quantum hardware to quantify the impact of noise present in the current generation of QPUs. Would it be possible, in principle, to execute the proposed symmetric encryption protocol on NISQ computers?

To answer this question we have used Ankaa-2, the latest 4th generation QPU from the Rigetti Computing suite of superconducting qubit processors [55]. In order to make results comparable to those obtained on the noise-free quantum simulator, we used the same 5-qubit circuit shown in Figure 3. By changing the allocation of H and $HS^\dagger$ gates in the final layer we changed the measurement bases and calculated expectation values for all possible second order Paulis. With $N = 5$ the number of all possible Pauli pairs is $9 \times N(N-1)/2 = 90$. The number of possible random configurations of rotation angles was set at 56. This gave us $90 \times 56 = 5,040$ expectation values of second order Paulis, all calculated with 100,000 runs of the quantum circuit.

Figure 13 displays the corresponding CDF of the absolute values of second order Pauli expectation values. To make it even more comparable to the results obtained on the noise-free quantum simulator, we repeated the procedure described

in Section B.1 but with 100,000 quantum circuit runs – these results are shown in Figure 14. The vertical dashed red lines cross the X axis at $X = \epsilon$, where ϵ is 6 standard deviations of the second order Pauli expectation values from the corresponding mean values. The horizontal dashed lines indicate the probability that the second order Pauli expectation value would be smaller than ϵ, i.e., that it cannot be used for encoding purposes.

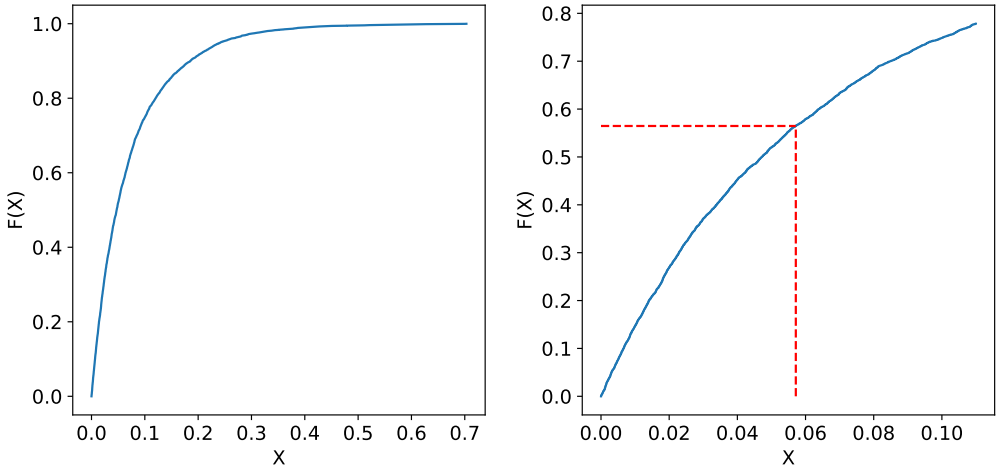

Figure 13: Cumulative distribution function for second order Pauli expectation magnitudes across 5,040 randomly generated configurations; 100,000 quantum circuit runs; noisy QPU (Ankaa-2). The 6-sigma threshold is $\epsilon = 0.057$.

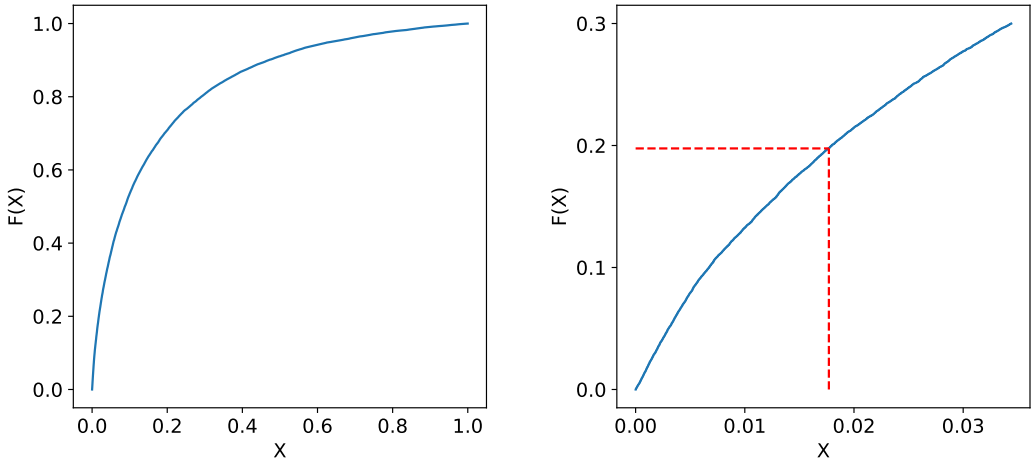

Figure 14: Cumulative distribution function for second order Pauli expectation magnitudes across 5,000 randomly generated configurations; 100,000 quantum circuit runs; noise-free quantum simulator. The 6-sigma threshold is $\epsilon = 0.018$.

We can see from Figures 13 and 14 that if we perform 100,000 quantum circuit runs, then the share of ineligible Pauli pairs is about 57% for the noisy QPU and

about 20% for the noise-free quantum simulator. This result should be seen as an indication that NISQ devices could be used for performing symmetric encryption as per Algorithm 1, especially taking into account the current pace of quantum hardware improvement.

Finally, we should mention that the quantities of positive and negative second order Pauli expectation values are almost perfectly balanced: In the Ankaa-2 experiment, the share of positive expectation values was 49.7% and the share of negative expectation values was 50.3%. The share of positive expectation values greater than ϵ was 21.6% and the share of negative expectation values less than $-\epsilon$ was 21.9%.

Figure 15 displays the CDF for the magnitude of all second order Pauli expectation values (blue curve), along with the CDF for the magnitude of only the positive second order Pauli expectation values (orange curve), and the CDF for the magnitude of only the negative second order Pauli expectation values (green curve). The distributions are all basically the same and therefore overlap each other.

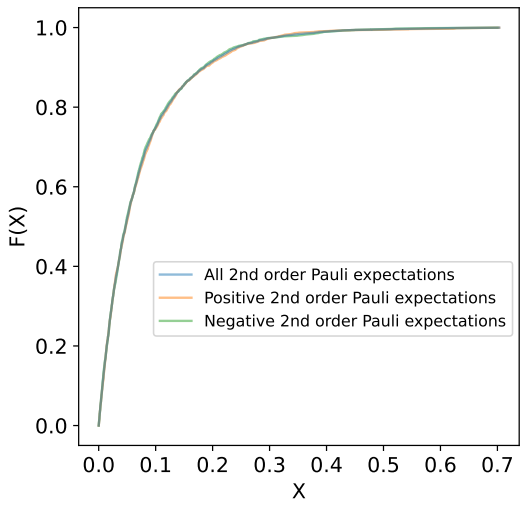

Figure 15: Cumulative distribution functions for second order Pauli expectation magnitudes across 5,040 randomly generated configurations; 100,000 quantum circuit runs; noisy QPU (Aanka-2). Comparison of all second order Pauli expectations (blue curve) with only positive Pauli expectations (orange curve) and only negative Pauli expectations (green curve).

Remark: The proposed symmetric encryption algorithm is based on the calculation of expectation values of second order Paulis. In principle, the same quantum circuit can be used for encrypting more bits using the expectation values of higher order Paulis: $\langle P_i P_j P_k \rangle$, $\langle P_i P_j P_k P_l \rangle$, ... However, any possible gain is likely to be immaterial since the expectation values of higher order Paulis may be smaller in magnitude and the measurement noise will be larger. Only a small fraction of the higher order Pauli expectations can be used for encryption if we keep the number of quantum circuit runs within a reasonable limit.

C Classical Bits Required to Represent a PQC

In Section 1, it was mentioned that N bits of a shared classical secret key are necessary and sufficient to transmit N bits of a message over a one-way classical channel in an information-theoretical secure manner. In our case, the role of the *shared secret* is played by a PQC. Due to its expressive power, the amount of information needed to fully specify the PQC can be significantly smaller than the amount of information that can be securely encoded using it. Let us see why.

Let N be the number of qubits and M be the number of layers of 1- and 2-qubit gates. Therefore, the PQC can be fully specified by providing a description for each of the $N \times M$ circuit nodes. This can be done in the following manner.

First, for a given circuit node, we need to specify whether it is a 1-qubit gate (including no gate, i.e., an identity gate I) or an element of a 2-qubit gate. Typically, there are only a few possible fixed 2-qubit gates used to create entanglement (e.g., CX, CZ, XY). For example, we can use the following 3-bit scheme to encode the gate type:

Bitstring	Meaning
000	1-qubit gate, identity, I
001	1-qubit gate, rotation around x axis, R_x
010	1-qubit gate, rotation around y axis, R_y
011	1-qubit gate, rotation around z axis, R_z
100	2-qubit gate, CX
101	2-qubit gate, CZ
110	2-qubit gate, $iSWAP$
111	2-qubit gate, $\sqrt{iSWAP}$

Second, for 1-qubit gates, we need to specify the rotation angle, which is a continuous variable on the interval $[-\pi, \pi]$. For that, we can use a standard 32-bit precision binary encoding.

Third, for 2-qubit gates, we need to specify whether the given circuit node is a *target* or a *control*. This is a binary indicator and one bit is sufficient to store this information. Additionally, we need to indicate the location of the second circuit node of the 2-qubit gate. Since it must be the same layer, we only have to indicate the qubit index, which is an integer number between 0 and $N - 1$. We need

$$\text{int}\left(\log_2(N)\right) + 1$$

bits to represent this number in a binary format. For example, it will be a 10-bit bitstring for $N = 1,000$ and a 17-bit bitstring for $N = 100,000$.

Therefore, in the worst case scenario, we would need at most

$$N \times M \times (3 + 32)$$

bits to fully specify the PQC with 32-bit precision for all rotation angles.

At the same time, there are 3^N possible configurations of the measurement bases. This large number is indicative of how much information we could encode using the PQC. However, in our algorithm we only use a tiny fraction of the expressive power of PQC available to us – we only use the expectation values of second order Paulis, $\langle P_i Q_j \rangle$, with

- $P = \{X, Y, Z\}$,
- $Q = \{X, Y, Z\}$,
- $i = \{1, \ldots, N-1\}$,
- $j = \{i+1, \ldots, N\}$.

The total number of possible combinations of second order Paulis is

$$\frac{N(N-1)}{2} \times 9.$$

We cannot use all possible combinations of second order Paulis due to the measurement noise as explained in Appendix B. For a reasonably large number of quantum circuit runs, we can expect to use about 80% of the total number of Pauli pairs.

The following table shows a comparison of the number of bits needed to specify the PQC versus the number of bits that can be encoded with the help of binarized expectation values of unique second order Paulis (positive $\Rightarrow$ 1; negative $\Rightarrow$ 0) for various values of N.

N	M	Maximum number of bits needed to specify the PQC, $35NM$	Maximum number of bits that can be encoded by the PQC, $4.5N(N-1)$	Expected number of bits that can be encoded by the PQC, $4.5N(N-1)\kappa$
1,000	63	2,205,000	4,495,500	3,596,400
2,500	100	8,750,000	28,113,750	22,491,000
5,000	141	24,675,000	112,477,500	89,982,000
10,000	200	70,000,000	449,955,000	359,964,000

Here, we set $\kappa = 80\%$ and assumed that the QPU has a square grid connectivity and, therefore, we would need

$$M = \text{int}\left(\sqrt{N} \times 2\right)$$

layers of 1- and 2-qubit gates to ensure that the most distant qubits can be connected. The maximum number of bits needed to specify the PQC grows with N as $N^{3/2}$ while the maximum number of bits that can be encoded by the PQC grows as N^2.

Note that the algorithm proposed in this paper also requires the mapping scheme of plain text symbol to bitstring encoding (see Appexdix A) to be exchanged securely

as part of the secret key. Table 1 contains $27 \times 3 = 81$ characters. Assuming a predetermined ordering, it requires a total of $81 \times 12 = 972$ bits to specify the mapping. This small overhead should be added to the number of bits needed to specify the PQC.

D Components of the Symmetric Encryption Algorithm

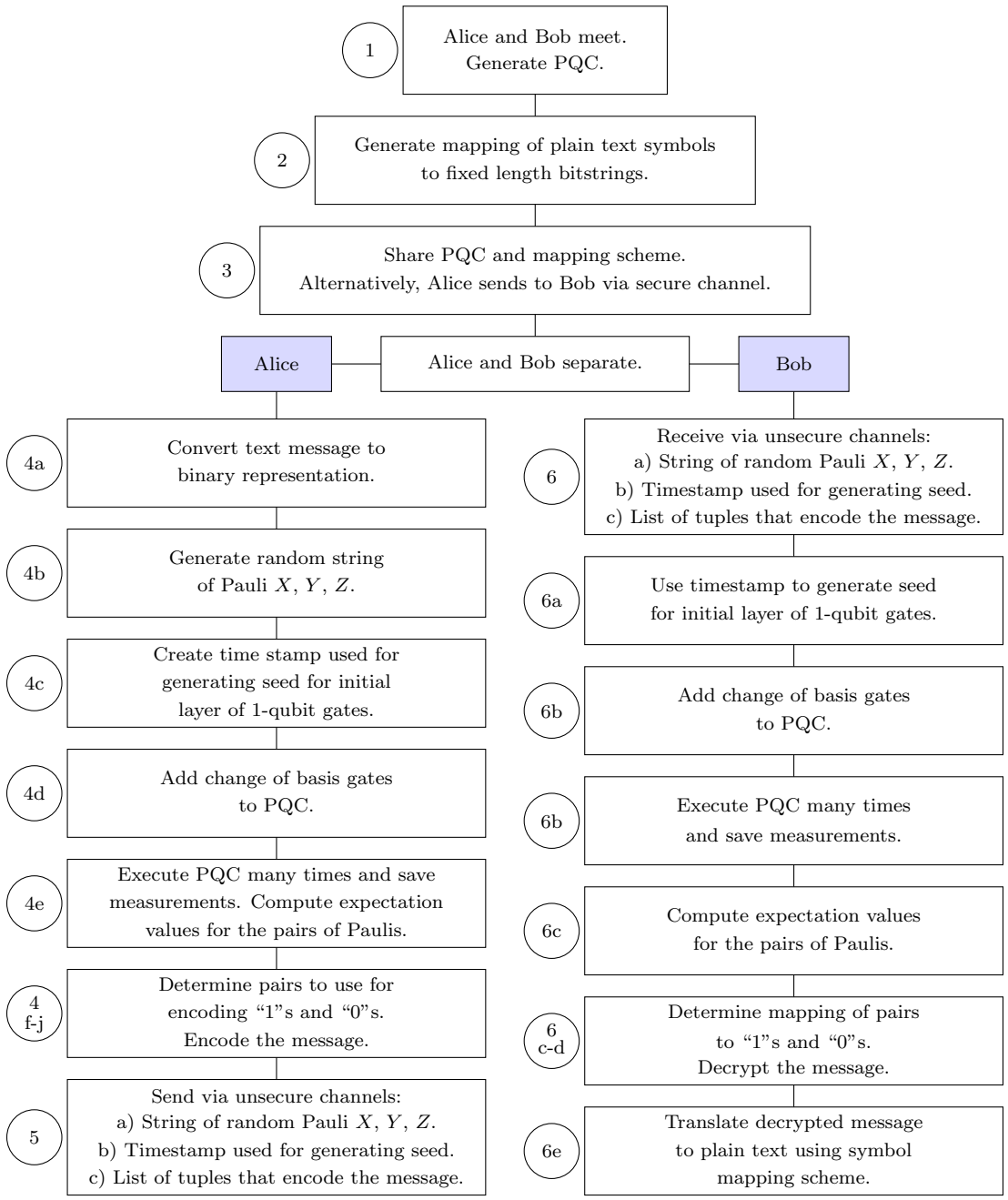

Figure 16: Components of the symmetric encryption algorithm. Detailed explanations of the blocks are provided in Section 5.