# Resource-Efficient Secure Delegation via Pauli-Based Computation

Daniel Lovsted

Thesis submitted to the University of Ottawa in partial fulfillment of the requirements for the degree of
Master of Science Mathematics and Statistics*

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa



*The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

# Abstract

The scarcity of quantum resources is a significant theme in quantum cryptography and computation. Within quantum cryptography, *secure delegated quantum computation* (SDQC) allows low-resource users to access quantum computation with privacy. From the study of models of quantum computation, *Pauli-based computation* (PBC) has emerged as a resource-efficient alternative to conventional models.

In this thesis, we merge these two strands of work, presenting the first protocol for SDQC in the PBC model. Our protocol allows a client with minimal quantum resources to execute a PBC on a secret classical input of their choice by interacting with a quantum server. We prove that the protocol gives the client perfect, information-theoretic security. Our protocol inherits the resource efficiencies of PBC, significantly reducing the size of the quantum register required by the server, implying that SDQC could be available sooner than anticipated in a world of scarce quantum resources.

# Acknowledgements

I would like to thank my supervisor, Anne Broadbent, for the exceptional support she has provided me throughout the preparation of this thesis. I am grateful for her guidance, generosity, and deeply perceptive insight on all things quantum and cryptographic.

I would like to thank my other collaborators on this investigation of PBC and cryptography: Selman Ipek, Joshua Nevin, and Filipa Peres. This project has been as fascinating and fun as I could have hoped for, and I look forward to continuing to work with you all.

I would also like to thank my thesis examiners, Jason Crann and Arthur Mehta, whose comments significantly improved the final version of this thesis.

I am grateful to have been taught and encouraged by faculty, staff, and friends in the Department of Mathematics and Statistics at the University of Ottawa, and especially in the QUASAR research group.

I would like to thank my parents, Maria and Finn, for supporting me always and inspiring me to continue learning, and my sister, Gwen, for starting my mathematical education at a young age.

I thank my cat, Licorice, for the comfort he has provided me during the writing of this thesis, and for his helpful comments on the proof of lemma 7.4.2.

Finally, I am immensely grateful to my partner, Valerie, for believing in me, giving excellent advice, listening to me talk about math, and being a source of light and joy in my life, through this entire process.

# Contents

**List of Figures** viii

**List of Tables** ix

**1 Introduction** 1

1.1 Context . . . 2

1.2 Summary of Contributions . . . 3

1.3 Outline . . . 5

**2 Preliminaries** 6

2.1 Mathematical Preliminaries . . . 6

2.2 Computational and Cryptographic Preliminaries . . . 7

2.3 Quantum Information . . . 8

2.4 Models of Quantum Computation . . . 10

2.5 Pauli Group . . . 11

2.6 Clifford Group . . . 12

**3 Secure Delegated Quantum Computation** 17

3.1 Setting . . . 18

3.1.1 Correctness and Security . . . 18

3.1.2 Resources . . . 19

3.1.3 Input-Hiding and Circuit-Hiding . . . . . 20
3.2 Historical Overview . . . . . 21
3.3 Resource Requirements of Protocols of Interest . . . . . 23
3.3.1 Universal Blind Quantum Computation . . . . . 23
3.3.2 Quantum Computing on Encrypted Data . . . . . 24
3.4 Related Topics . . . . . 25

**4 Pauli-Based Computation 26**
4.1 Background . . . . . 26
4.2 The PBC Model . . . . . 27
4.2.1 PBC Algorithm . . . . . 27
4.2.2 Proofs of Correctness . . . . . 31
4.2.3 Proofs of Efficiency . . . . . 35
4.2.4 Remarks on the Algorithm . . . . . 37
4.3 Three Ways of Looking at a PBC . . . . . 40
4.3.1 PBC as Interaction . . . . . 40
4.3.2 PBC as Measurement Sequence . . . . . 40
4.3.3 PBC as Binary Tree . . . . . 42
4.4 Advantages of PBC . . . . . 43
4.4.1 Qubit Virtualization . . . . . 44
4.5 Example . . . . . 48
4.5.1 PBC Algorithm . . . . . 48
4.5.2 Measurement Sequence and Tree . . . . . 51

**5 Protocol: Input-Concealed PBC 53**
5.1 Intuition . . . . . 53
5.2 Protocol Description . . . . . 55

5.3 Correctness . . . . . 58
5.4 Qubit Virtualization . . . . . 60
5.5 Example . . . . . 62

**6 Advantages of the Protocol** **68**
6.1 Resource Requirements of Input-Concealed PBC . . . . . 69
6.1.1 Client Resources . . . . . 69
6.1.2 Communication Resources . . . . . 70
6.1.3 Server Resources . . . . . 71
6.2 Discussion of Comparison . . . . . 71
6.3 Qubit Virtualization . . . . . 73
6.4 X-Invariant Circuits . . . . . 73

**7 Security** **77**
7.1 Intuition . . . . . 77
7.2 Security Definition . . . . . 78
7.3 Protocol **S** . . . . . 79
7.4 Security of **S** . . . . . 83
7.5 Security of Qubit Virtualization . . . . . 87

**8 Conclusion** **88**

**Bibliography** **96**

# List of Figures


2.1 The propagation of a Pauli past gates (a) H, (b) S, and (c) CNOT. . 14

2.2 A T gadget, used in magic state injection to replace T gates in a circuit. . . . . 16

4.1 A generic PBC tree. . . . . 43

4.2 The circuit $C$ to be executed in the example run of the PBC algorithm. 48

4.3 The circuit $C'$, the magic-injected form of $C$, produced in step 1 of the PBC example. . . . . 49

4.4 $C'$ after iteration 1 of step 3 of the PBC example. . . . . 49

4.5 The PBC tree of the circuit $C$ used in the PBC example. . . . . 52

5.1 The propagation of a Pauli past (a) a T gate and (b) a T gadget. . . 58

5.2 Correctness of the randomization step in the input-concealed PBC protocol. . . . . 59

5.3 Additional propagation rules for input-concealed qubit virtualization. 61

5.4 The circuit $C$ to be executed in the example of input-concealed PBC. 63

5.5 First propagation steps in the input-concealed PBC example. . . . . 64

5.6 First randomization step in the input-concealed PBC example. . . . 65

5.7 Continued propagation steps in the input-concealed PBC example. . 66

5.8 Second randomization step in the input-concealed PBC example. . . 66

5.9 Third and final propagation step in the input-concealed PBC example. 67

5.10 Alice's last-layer Pauli corrections in the input-concealed PBC example. . . . . 67

5.11 The processed circuit $C^*$ in the input-concealed PBC example. . . . 67

6.1 An X-invariant circuit with four T gates. . . . . 75

# List of Tables

4.1 Table representation of step 3(b), case determination, of the PBC algorithm (algorithm 1). . . . . 32

4.2 Table representation of step 3(c), case actions, of the PBC algorithm (algorithm 1). . . . . 32

6.1 Resource cost comparison of our protocol and existing schemes. . . . 72

# Chapter 1

# Introduction

This thesis connects two lines of work in the theory of quantum computation: secure delegated quantum computation (SDQC), a cryptographic task, and Pauli-based computation (PBC), a model of quantum computation. While coming from distinct areas in the field, these two topics share a common motivational starting point: the scarcity of quantum computational resources. From the perspective of SDQC, this scarcity implies that quantum computers, when they first appear at useful scale, will likely emerge concentrated in the hands of a few actors with the technical and financial capacity to construct them. For anyone else wishing to access quantum computation, then, computation will need to be *delegated*, that is, quantum programs and data will need to be sent to a remote site for execution. This raises the question of how these programs and data may be protected cryptographically while being correctly executable, which SDQC seeks to address. From the perspective of PBC, the scarcity of quantum resources means that models of computation which lead to resource savings are highly desirable. PBC, introduced in [BSS16], is one such model, offering a way of efficiently simulating quantum computation with a smaller quantum processor than would be required by other models. In this thesis, we present the first protocol for secure delegation in the PBC model, a protocol which preserves the resource efficiencies of PBC and thus offers advantages over existing SDQC protocols.

In this chapter, we give an informal introduction to the context of our work and a summary of our contributions. We end with an outline of the thesis.

## 1.1 Context

Secure delegated quantum computation (SDQC) addresses the following scenario. A low-resource client has in mind a quantum computation which she would like to execute on an input of her choosing. She lacks the quantum computational resources to do so locally, but a remote, high-resource server is willing to help her (for a price, no doubt). The client is willing to rent the server's computational help, but is unwilling to simply send a description of her computation and input to the server. Perhaps her computation is a new algorithm, representing valuable intellectual property, or perhaps her input contains sensitive data. Can the client use the server's help to perform this computation and get a correct outcome, without telling the server what the computation is?

Remarkably, the answer is yes. A rich SDQC literature has produced a range of protocols in which, after multiple rounds of client-server interaction, the client gets a correct outcome, and the server does not know what computation it has just performed. Many of these protocols achieve this result efficiently and even with perfect security, but they differ from one another in the resources they require of the client, of the communication between client and server, and of the server. Since SDQC is premised on the at least partial scarcity of quantum resources, the resource cost of a protocol is a critical consideration. Reducing the resources required for SDQC has been a significant theme in work on the topic.

To find novel SDQC resource savings, we look to research on quantum computation itself. The difficulty of constructing quantum hardware has motivated research into quantum resource-efficient models of quantum computation. One such model, Pauli-based computation (PBC), achieves quantum resource reductions by offloading certain steps and aspects of a quantum computation to a classical processor. It also allows for a range of other benefits, such as the intriguing possibility of virtualizing (simulating) qubits which exceed the hardware capacity of a quantum processor.

In its resource efficiency, natural separation of a computation into classical and quantum components, and other features, PBC seems well-suited for use in SDQC protocols. In this thesis, we seek to harness these affinities by producing a protocol for SDQC in the PBC model.

## 1.2 Summary of Contributions

The first main contribution of this thesis is a protocol for *input-concealed PBC* (algorithm 2), which offers a range of benefits over existing SDQC protocols:

- *Server resource reduction.* In the SDQC regime of input-hiding with classical input and output, our protocol achieves perfect security with a significantly lower server resource cost compared to existing SDQC methods, while matching or slightly improving on the optimal client and communication resource costs among existing protocols (section 6.2). Server resources have been relatively understudied in SDQC, compared to client and communication resources. It is easy to forget about server resources altogether in the SDQC picture, since the server is imagined as the resource-rich party. However, we argue that server resource reduction is the most pressing question of the three, in a sense. While client and communication resource reductions are undoubtedly well motivated, corresponding to wider client access to SDQC and reduced cost, these benefits apply only *in a world with quantum computation* of useful scale. We are not yet in such a world. Today, insufficient server resources are the main bottleneck to feasible SDQC. In contrast, typical SDQC client and communication resources, e.g., state preparation and transmission, *are* feasible in some form with current technology, and have even been deployed at scale in quantum key distribution (QKD) networks (see, e.g., [CZW+22]). These observations are borne out by recent SDQC experimental demonstrations ([DNM+24], [WSS+25]), where heavily limited server sizes (e.g., the server of [DNM+24] has just two qubits) and relatively inexpensive client and communication implementations influence simplifications to the SDQC protocols being demonstrated. Server resources in SDQC are therefore an issue of underappreciated importance, with reductions corresponding to a shorter timeline to feasible SDQC.

- *Qubit virtualization.* Just as PBC enables qubit virtualization, our SDQC protocol can be extended to secure qubit virtualization (section 5.4), allowing for a computation on $t + k$ qubits to be run on a register of just $t$ qubits, at a cost exponential in $k$ only, again with perfect security. No comparable technique exists in the SDQC landscape. In scenarios where the register size of a computation exceeds the size of the available quantum hardware, existing SDQC

protocols become intractable, with runtimes scaling in $t + k$ and with no use made of the $t$-qubit register.

- *Classical-client SDQC beyond Cliffords.* Existing SDQC protocols automatically "revert" to classical client resources on Clifford circuits only, a class which is classically simulable. Our protocol runs with a fully classical client on a class which we define and call *X-invariant circuits* (definition 6.4.1), of which Cliffords are a strict subset. X-invariant circuits can include arbitrarily many T gates and we are not aware of a means for simulating them classically. Our protocol therefore allows for classical-client SDQC with perfect security on a non-trivial extension of Clifford circuits, to our knowledge for the first time.

- *SDQC in PBC.* To the best of our knowledge, ours is the first protocol to achieve an SDQC task in the PBC model, or even to consider PBC cryptographically in any sense. We expect our work to lay the groundwork for future study along this line.

The second main contribution is the security proof of the input-concealed PBC protocol (theorem 7.4.4 and proof). Our approach to the security proof includes several innovative aspects:

- *Translators.* In our proof, we introduce the notion of a *translator*, a classical machine which converts between the messages of two parties, each performing one side of a *different* delegated protocol. We use the translator to lift a proof of security of one protocol, which in our case is introduced as a proof technique, to another.

- *Bypassing entanglement-based approaches.* Existing SDQC proofs, notably in [Bro15], use entanglement-based techniques. In our case, because of our classical-input and classical-output regime, we are able to give a somewhat simpler proof, bypassing entanglement-based arguments. We expect this to be of interest for future work on security in SDQC.

A secondary contribution of this thesis is an exposition chapter on PBC (chapter 4) with some novel aspects. These include a new presentation of the PBC algorithm (algorithm 1), a streamlined proof of correctness (section 4.2.3), a comparison

of three perspectives on the PBC model which organizes existing analytical tools (section 4.3), and a new worked example of the PBC algorithm (section 4.5). We expect these contributions to be pedagogically useful, given the low volume of accessible work on PBC. Finally, we note that the discussion of *X-invariant circuits* already mentioned may be of independent interest, beyond the SDQC context, to research on quantum algorithms and the classical simulation of quantum computation.

## 1.3 Outline

The thesis is structured as follows.

Chapter 1 gives a brief, high-level introduction to the major components of the thesis, including SDQC, PBC, our protocol, and its advantages.

Chapter 2 presents the preliminary notions from mathematics, computer science, cryptography, and the theory of quantum information and computation which are necessary for the subsequent chapters.

Chapters 3 and 4 are primarily background chapters. In them, we present the two strands of work, SDQC and PBC, respectively, to be integrated; some contributions in chapter 4 are indicated in that chapter.

Chapters 5 to 7 contain the main contributions of the thesis. Chapter 5 presents the novel protocol, proves its correctness, and demonstrates the qubit virtualization extension. In chapter 6, we compare our protocol to existing SDQC protocols to explain in detail the advantages gained from performing SDQC in the PBC framework. In chapter 7, we present our proof of security, which utilizes some novel techniques in comparison to previous SDQC security proofs.

We conclude in chapter 8 with some open questions and new directions stemming from the work in this thesis.

# Chapter 2

# Preliminaries

## 2.1 Mathematical Preliminaries

We use standard set notations for the natural numbers, real numbers, and complex numbers: $\mathbb{N}$, $\mathbb{R}$, and $\mathbb{C}$, respectively. For us, $\mathbb{N} = \{1, 2, ...\}$, and we use the shorthand $[k]$ for $k \in \mathbb{N}$ to represent the set $\{1, 2, ..., k\}$. We are often interested in the set $\{0, 1\}$, the elements of which are called *bits*, its $n$-fold Cartesian product, denoted $\{0, 1\}^n$, for $n \in \mathbb{N}$, and the union over all $n$ of these sets, denoted $\{0, 1\}^* = \bigcup_{n=0}^{\infty} \{0, 1\}^n$. An element $x \in \{0, 1\}^*$, is called a *(binary) string* with length denoted $|x| = k$, for that $k \in \mathbb{N}$ for which $x \in \{0, 1\}^k$, and the bit of $x$ at index $i$ is denoted $x_i$. To $\{0, 1\}$ we associate addition modulo 2, denoted $\oplus$, and multiplication modulo 2, denoted $\cdot$ or by concatenation, creating the field $\mathbb{F}_2$, and we will often use the notation $\{0, 1\}^n$ to refer to the field $\mathbb{F}_2^n$ with these operations applied bitwise. For $a \in \mathbb{F}_2$, we denote by $\overline{a}$ the complement or bitflip of $a$, i.e., $\overline{a} = 1 - a$.

We assume familiarity with the basics of linear algebra. In this thesis, we will work only in finite-dimensional vector spaces. In a vector space $V$, a linear map from $V$ to $V$ is called a *(linear) operator*. We denote the space of linear operators on a space $V$ by $\mathrm{L}(V)$. Every linear operator has a matrix representation in $\mathrm{Mat}_n(F)$, the ring of $n$ by $n$ matrices with entries from $F$, where $n$ is the dimension of $V$ and $F$ the underlying field; likewise every such matrix corresponds to a linear operator. These correspondences are unique if we fix a basis for $V$. In $\mathbb{C}^n$, we will always use the standard basis and express operators as matrices. Since we consider only finite-

dimensional spaces, the notion of a *Hilbert space* coincides with an *inner product space over* $\mathbb{C}$; we will prefer the term Hilbert space, as is usual in quantum information. We use calligraphic letters (e.g., $\mathcal{A}$, $\mathcal{B}$) to denote Hilbert spaces. We use the notation $U^\dagger$ for the adjoint of operator $U$. If $UU^\dagger = I$, where $I$ is the identity operator, $U$ is said to be *unitary*. The unitary operators in a space of dimension $n$ form a group, denoted $\mathcal{U}(n)$ and called the *unitary group (of order n)*.

Concepts from group theory also play an important role. We again assume a basic familiarity and highlight that, for a subgroup $H$ of a group $G$, the *normalizer* of $H$ in $G$, denoted $\mathrm{N}_G(H)$, is the subgroup of $G$ consisting of those elements $g \in G$ which preserve membership in $H$ by a conjugation mapping, i.e., for which, for any $h \in H$, there exists an $h' \in H$ such that $ghg^{-1} = h'$. This can be expressed in terms of left and right cosets as $\mathrm{N}_G(H) = \{g \in G : gH = Hg\}$.

From probability, we frequently refer to *(probability) distributions*. For us, these will always be over a finite nonempty set $\mathcal{X}$, so for our purposes a distribution is an association to each element $x \in \mathcal{X}$ a value $p_x \in [0, 1] \subseteq \mathbb{R}$ such that $\sum_{x \in \mathcal{X}}(p_x) = 1$. A *uniform distribution* over $\mathcal{X}$ is a probability distribution where $p_x = p_y$ for all $x, y \in \mathcal{X}$. We write $x \in_R \mathcal{X}$ and say that $x$ is selected *uniformly at random* from $\mathcal{X}$ when $x$ takes a value in $\mathcal{X}$ according to a uniform distribution on $\mathcal{X}$. We write the expectation of a random variable $X$ as $\mathbb{E}(X)$, its variance as $\mathrm{Var}(X)$, and we take variance to be defined as $\mathrm{Var}(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2$.

## 2.2 Computational and Cryptographic Preliminaries

We describe the complexity of algorithms in terms of their time and (occasionally) space requirements. We quickly review a few important elements here and refer to [Sip12] for a thorough explanation. For an algorithm $A$ which always halts, the *runtime* or *time complexity* of $A$ is described by a function $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of steps a Turing machine implementation of $A$ makes before halting over all inputs of size $n$. The *space complexity* of $A$ is similarly described by a function from $n$ to the maximum number, over all inputs of size $n$, of read/write cells required by a Turing machine implementation of $A$. We use *big-O* notation to describe the asymptotic behaviour of functions, especially of functions describing algorithmic complexity, as follows. For functions $f, g : \mathbb{N} \to \mathbb{N}$, $f \in \mathcal{O}(g)$ if there

exists a constant $c \in \mathbb{R}$ such that $c \cdot g(n) \geq f(n)$ for all sufficiently large $n$. We say that an algorithm requires (or "is") *polynomial time* if its runtime is described by a function $f(n)$ is such that $f(n) \in \mathcal{O}(n^k)$ for some $k \in \mathbb{N}$. For convenience, we frequently refer to polynomial time algorithms as *efficient*.

For our purposes in this thesis, "classical" computation includes randomness. That is, classical computation is what is performed by a probabilistic Turing machine, i.e. a deterministic Turing machine augmented by a read-tape of random values. The runtime function $f(n)$ of a probabilistic Turing machine is its worst-case runtime over all inputs of size $n$ *and* all possible random tapes.

From cryptography, we refer to two broad categories of security, *computational security* and *information-theoretic security*. For our purposes, the difference is that computational security holds against computationally-bounded adversaries (usually, adversaries limited to polynomial time computation), whereas information-theoretic security holds against all adversaries, including computationally unbounded ones. Many varieties of security exist within each category, but the broad distinction is often useful. We also refer to the *classical one-time pad*, a cryptosystem which works as follows. Suppose Alice wants to send a message to Bob over an insecure channel monitored by eavesdropper Eve. For a message $m \in \{0,1\}^n$, a key $k$ is sampled uniformly at random from the same set, $k \in_R \{0,1\}^n$ and shared with Bob but not with Eve. A ciphertext $c$ is generated as $c = m \oplus k$. The ciphertext can then be sent over the insecure channel with information-theoretic security, since the distribution on $c$ is independent of the distribution on $m$ (actual or as perceived by Eve); Bob can decrypt the message as $m = c \oplus k$. More details on this and other classical cryptographic concepts can be found in [KL21]. There is also a *quantum one-time pad*, which achieves a similar encryption on a quantum state (the message) by applying a sequence (the key) of independently randomly selected operators to each qubit; the details are not important for us, but we occasionally refer to its existence.

## 2.3 Quantum Information

We assume familiarity with the basics of quantum information and computation, as explained in [NC10]. For most of the thesis, we work only with pure states, using Dirac notation for quantum states, unitary operators, and projective measurements.

We use the following notation for some standard unitary operators:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix},\ S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix},\ CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix},\ T = \begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}. \quad (2.3.1)$$

Note the use of $S$ for the phase gate, rather than the more standard $P$. We prefer to keep $P$ free for now because of the abundance of Paulis that arise in Pauli-based computation. We frequently refer to the single-qubit Pauli operators:

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

As for specific state notations, we use $|+\rangle$ and $|-\rangle$ for the $+1$ and $-1$ eigenstates, respectively, of the Pauli $X$ operator and we use $|i\rangle$ and $|-i\rangle$ for the $+1$ and $-1$ eigenstates, respectively, of the Pauli $Y$ operator. Finally, we make abundant use of the following $T$-state, which falls into the class of *magic states*:

$$|T\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}. \quad (2.3.2)$$

In chapter 7, when we define and prove security of our protocol, we switch to the density operator formalism for quantum information. This is standard in security proofs in this area, e.g., in [Bro15], and is convenient since mixed states capture the probabilistically-chosen pure states sent and received in our protocol. Following notation in [Bro15], for finite-dimensional Hilbert spaces $\mathcal{A}$ and $\mathcal{B}$, we let $\mathrm{D}(\mathcal{A})$ be the set of density operators on $\mathcal{A}$ and $\mathrm{L}(\mathcal{A},\mathcal{B})$ the set of linear maps from $\mathcal{A}$ to $\mathcal{B}$, with $\mathrm{L}(\mathcal{A})$ denoting $\mathrm{L}(\mathcal{A},\mathcal{A})$ (as in section 2.1). Then we consider maps $\Phi : \mathrm{L}(\mathcal{A}) \to \mathrm{L}(\mathcal{B})$, called *super-operators*; when a super-operator $\Phi$ is completely positive and trace-preserving we call it a *(quantum) channel*.

To formalize the notion of distinguishability of two quantum channels $\Phi, \Psi : \mathrm{L}(\mathcal{A}) \to \mathrm{L}(\mathcal{B})$, we use the *diamond distance*, denoted $||\Phi - \Psi||_\diamond$, and defined as

$$||\Phi - \Psi||_\diamond = \max_{\rho \in \mathrm{D}(\mathcal{A}\otimes\mathcal{W})} ||(\Phi \otimes I_\mathcal{W})\rho - (\Psi \otimes I_\mathcal{W})\rho||_1$$

where $\mathcal{W}$ is a Hilbert space of the same dimension as $\mathcal{A}$ and $||\cdot||_1$ is the *trace norm*, defined for an operator $\rho$ as $||\rho||_1 = \mathrm{Tr}\sqrt{\rho^\dagger \rho}$.

## 2.4 Models of Quantum Computation

We deal with several quantum computational models in the thesis. Most familiar is the *circuit model*; we refer to [NC10] for a full discussion and just note some conventions. To avoid directional confusions, we describe element ordering in a circuit in terms of time (before, after, earlier, later, beginning, end, etc.), since left-right directions are reversed between equations and circuits. We use the convention of sans serif font (e.g., $\mathsf{H}$) to denote a gate which implements a unitary operation of the same label; we denote the unitary in serif font (e.g., $H$), which we use mainly in formulas and equations. So, for example, we might write, "$SZ = ZS$ so we can exchange the order of the $\mathsf{S}$ and $\mathsf{Z}$ gates...."

Throughout this thesis, circuits are assumed to use the gate set $\{\mathsf{H}, \mathsf{S}, \mathsf{CNOT}, \mathsf{T}\}$, called the *Clifford* + $\mathsf{T}$ gate set, with each gate corresponding to the operator of the same label in eq. (2.3.1). This gate set is *universal*, meaning it can approximate any unitary operation to arbitrary precision $\epsilon$ at a cost on the order of $\log^c(\frac{1}{\epsilon})$ for a small constant $c$, by the Solovay-Kitaev Theorem ([NC10]). With a universal gate set, qubit preparation in the $|0\rangle$ state, and computational basis measurements, the circuit model enables *universal quantum computation*. Therefore, in particular, the set of circuits $C$ executed on the all-zero state $|0^n\rangle$, with Clifford + $\mathsf{T}$ gate set and computational basis measurements, is sufficient for universal quantum computation.

Throughout this thesis, we draw circuit diagrams using the excellent Quantikz package, which we cite as [Kay18] according to the instructions in its documentation.

We also discuss models in the broad family called *measurement-based quantum computation* (MBQC), where the computation is driven by a sequence of measurements on an initial state, the outcomes of which determine future measurements. Likely the best-known MBQC model is *one-way quantum computation*, or 1WQC,

introduced in [RB01], where the computation consists of single-qubit measurements made on an initial entangled state.[1] Other MBQC models include teleportation-based ([Leu04]), state transfer ([Per05]), and, most importantly for us, Pauli-based computation (PBC) ([BSS16]).

## 2.5 Pauli Group

As the names suggest, the *Pauli group* is of central importance to PBC.

**Definition 2.5.1** (Pauli group)**.** The *n-qubit Pauli group* $\mathcal{P}_n$ is the set of $n$-qubit operators

$$\mathcal{P}_n = \{b \cdot P_1 \otimes ... \otimes P_n : b \in \{\pm 1, \pm i\}, P_i \in \{I, X, Y, Z\}\}.$$

These operators form a group under multiplication, specifically a subgroup of $\mathcal{U}(2^n)$. We also refer to the *(n-qubit) phaseless Pauli group*, denoted $\mathcal{P}_n^*$, where the phases in $\mathcal{P}_n$ are ignored; we can formally consider $\mathcal{P}_n^*$ as the quotient group $\mathcal{P}_n^* = \mathcal{P}_n/\mathcal{U}(1)$. The set of all *Hermitian* Pauli operators on $n$ qubits can be formed by restricting $b$ in definition 2.5.1 to $\{\pm 1\}$ or, equivalently, by adding a $\pm 1$ overall phase to $\mathcal{P}_n^*$; we notate this set as $\pm\mathcal{P}_n^*$, reflecting the latter idea.

We usually write an element $P \in \mathcal{P}_n$ without the tensor product notation, so for example $-iXYZY \in \mathcal{P}_4$. We use the notation $b \cdot W_k$ for $W \in \{X, Y, Z\}$ to mean the $n$-qubit Pauli (where $n$ will be specified or clear from context) with $W$ in the $k$-th place and $I$ elsewhere, and phase $b$, so for example, on $n$ qubits,

$$Z_k = \overbrace{I...I}^{k-1} Z \overbrace{I...I}^{n-k}.$$

We extend this notation to Paulis with up to two non-identity positions, so for example in $\mathcal{P}_4$ we write $-X_2Y_4 = -IXIY$. An element of $\mathcal{P}_n^*$ can be represented as a vector in $\mathbb{F}_2^{2n}$, with 2 bits representing the Pauli at each of the $n$ places, and a suitable representation means that vector addition corresponds to multiplication of the corresponding Paulis; we only need to use this vector representation occasionally. We

[1] The 1WQC model itself is sometimes called "MBQC"; we resist this trend and use MBQC for a broad family of models, of which 1WQC is one member.

sometimes refer to elements of $\mathcal{P}_n^*$ as *Pauli strings*, understanding them as $n$ character strings drawn from the alphabet $\{I, X, Y, Z\}$.

Every pair of $n$-qubit Pauli operators $P_1, P_2 \in \mathcal{P}_n$ either commutes or anti-commutes, as can be seen by observing that all pairs of single-qubit Pauli operators have this property. If $P_1$ and $P_2$ commute, we write $[P_1, P_2] = 0$, and if they anti-commute, we write $\{P_1, P_2\} = 0$.

*Non-destructive Pauli measurements* are an important concept in Pauli-based computation. Pauli operators in $\pm\mathcal{P}_n^*$ are Hermitian as well as unitary, so they encode observables, and we can talk about *Pauli measurement* of an $n$-qubit Pauli $P$ on an $n$-qubit state $|\psi\rangle$ as the projective measurement encoded by $P$, performed on $|\psi\rangle$. Every $n$-qubit, non-identity Hermitian Pauli operator has exactly two eigenvalues, $+1$ and $-1$, each associated to an eigenspace of dimension $2^{n-1}$. The measurement is therefore a probabilistic projection of $|\psi\rangle$ into one of two subspaces, and produces a single-bit measurement outcome $m \in \{0, 1\}$, where $(-1)^m$ is the eigenvalue of the space projected into. By *non-destructive*, we mean that we consider the projected state to still live in $2^n$ dimensions. In the circuit picture, the non-destructive aspect means that all wires continue through a Pauli measurement, that is, if $n$ wires enter the measurement then $n$ wires leave it.

## 2.6 Clifford Group

Another important group for us is the *Clifford group*, defined as follows.

**Definition 2.6.1** (Clifford group)**.** The *$n$-qubit Clifford group*, denoted $\mathcal{C}_n$, is the normalizer of $\mathcal{P}_n$ in $\mathcal{U}(2^n)$ with global phase ignored, i.e. $\mathcal{C}_n = \mathrm{N}_{\mathcal{U}(2^n)}(\mathcal{P}_n)/\mathcal{U}(1)$.

With global phase ignored, the Clifford group is finite. Its size is given by

$$|\mathcal{C}_n| = \prod_{j=1}^{n} 2(4^j - 1)4^j$$

as stated and explained by a counting argument on the Pauli group in [Ozo08]. The size of the Clifford group thus increases rapidly with $n$; for instance, $|\mathcal{C}_1| = 24$, $|\mathcal{C}_2| = 11,520$, and $|\mathcal{C}_3| = 92,897,280$.

The group of $n$-qubit Clifford unitaries for $n > 1$ is generated by the set of one- and two-qubit operators $\{H, S, CNOT\}$ acting on any qubits, taken up to a global phase.[2] This means that any Clifford unitary can be represented as a circuit on $n$ qubits with gates drawn from the set $\{\mathsf{H}, \mathsf{S}, \mathsf{CNOT}\}$ (again ignoring global phase). We call circuits of this kind *Clifford circuits*, the gates of that set *Clifford gates*, and we denote the class of Clifford circuits as $\mathsf{CLIF}$. Certain circuits not in $\mathsf{CLIF}$ may also represent Clifford unitaries, for example the single-qubit circuit with two $\mathsf{T}$ gates on a single wire, since $T^2 = S$.

Because $\mathcal{C}_n$ is the normalizer of $\mathcal{P}_n$, for any $C \in \mathcal{C}_n$ and $P \in \mathcal{P}_n$, we have $CPC^\dagger = P'$ for some $P' \in \mathcal{P}_n$, which implies that $CP = P'C$. We are often interested in exchanging the order of a Pauli operator and a Clifford operator, leaving the Clifford unchanged and yielding a new Pauli, in the way suggested by this formula. In the circuit context we call this *propagating* a Pauli gate past a Clifford gate. Given a Clifford circuit, we can perform this propagation efficiently (in the size of the circuit) by using propagation rules for each individual Clifford gate. We give explicit propagation rules for propagating Paulis through Cliffords in fig. 2.1.

It is important for our purposes that these propagations apply to Paulis whether they are understood as gates or as measurements (for Hermitian Paulis). In particular, if $CP = P'C$ as above, then applying $C$ and *measuring* $P'$ has the same effect, in outcome statistics and post-measurement state, as does measuring $P$ and applying operator $C$ after. One way to see this is by expanding $P$ into its spectral decomposition as $P = \Pi_0 - \Pi_1$ for orthogonal projectors $\Pi_j$ for $j \in \{0, 1\}$ (as remarked above, every Pauli has exactly the eigenvalues $\pm 1$). Then we get that

$$P' = C\Pi_0 C^\dagger - C\Pi_1 C^\dagger \tag{2.6.1}$$

and verifying that the terms $C\Pi_j C^\dagger$ are orthogonal projectors (given that the $\Pi_j$ are) shows that eq. (2.6.1) is the spectral decomposition of the Pauli $P'$. From here, we can see that measurement of observable $P'$ on $C|\psi\rangle$ for an arbitrary state $|\psi\rangle$ produces outcomes $(-1)^j$ with probabilities

[2] That is, the group generated in this way is such that $\langle H, S, CNOT\rangle / \mathcal{U}(1) = \mathcal{C}_n$. The gate generation method produces global phases $e^{ik\pi/4}$ for $k \in [8]$ and so "over-generates" $\mathcal{C}_n$ by a factor of 8, until global phase is dropped from the generated group.

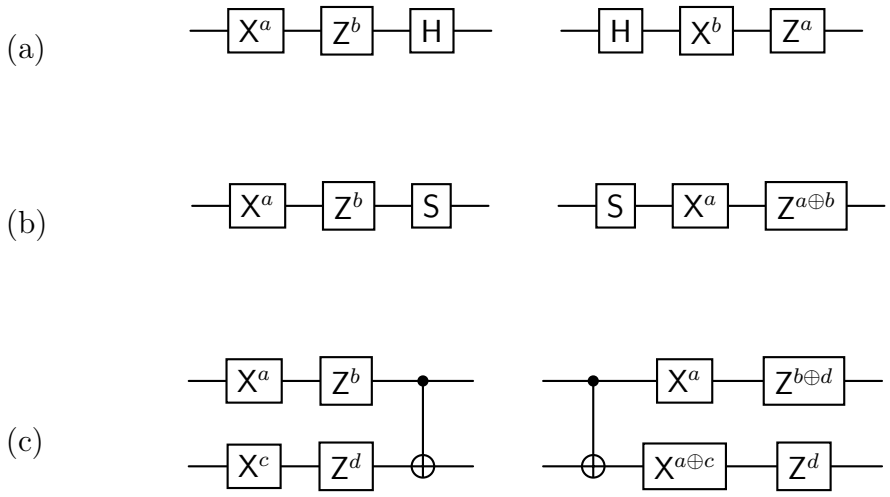



Figure 2.1: The propagation of a Pauli past gates (a) H, (b) S, and (c) CNOT. In each case, the left-hand side is equivalent to the right-hand side, meaning left-hand circuit element can always be replaced by the one on the right, and vice versa. Here, $a, b, c, d \in \{0, 1\}$, and $X^a Z^b$ is the form of a generalized Pauli in this context since $XZ = Y$ up to global phase, which we ignore in this context. Propagation rules in the reverse direction (i.e., with Paulis moving earlier in the circuit) are also useful and can be found by reverse engineering the Boolean formulas on the right-hand side.

$$p_j = (\langle\psi| C^\dagger) C \Pi_j C^\dagger (C |\psi\rangle) = \langle\psi| \Pi_j |\psi\rangle$$

which are identical to the outcome statistics of measuring the observable $P$ on state $|\psi\rangle$. Also, the post-measurement state from measuring $P'$ on $C|\psi\rangle$ with outcome $(-1)^j$ is

$$\frac{(C\Pi_j C^\dagger) C |\psi\rangle}{\sqrt{p_j}} = \frac{C\Pi_j |\psi\rangle}{\sqrt{p_j}}$$

which is identical to the post-measurement state produced by measuring $P$ on $|\psi\rangle$ with outcome $(-1)^j$ and then applying the Clifford operation $C$.

We note that for any two pairs of distinct, non-identity $n$-qubit Hermitian Paulis $P_1, P_2, Q_1, Q_2 \in \pm\mathcal{P}_n^*$, there exists a Clifford unitary $C \in \mathcal{C}_n$ such that $CP_1C^\dagger = Q_1$ and $CP_2C^\dagger = Q_2$ *if and only if* the commutation relation of $P_1$ and $P_2$ is the same as that of $Q_1$ and $Q_2$, i.e. either $[P_1, P_2] = [Q_1, Q_2] = 0$ or $\{P_1, P_2\} = \{Q_1, Q_2\} = 0$.

One proof of this useful fact is given in [Wri24].

Clifford circuits have great relevance in the context of classical simulation of quantum computation. We first define *weak (classical) simulation*, paraphrasing from [JV14] where the notion was introduced.

**Definition 2.6.2** (Weak simulation)**.** A classical procedure is said to *weakly simulate* a quantum computation $C|\psi\rangle$ if, given classical descriptions of circuit $C$ and input state $|\psi\rangle$, it produces an output sampled from the exact output distribution of $C|\psi\rangle$.

Weak simulation is contrasted in [JV14] with *strong simulation*, which describes a classical procedure that explicitly produces the probability values for some given possible outputs of $C|\psi\rangle$. We note that strong simulation is a very strong notion; as observed in [Per24], it is unlikely that quantum computers can strongly simulate themselves. Weak simulation captures the output that an actual quantum computer would produce and is usually sufficient.

We say an algorithm, classical or quantum, is *sampling-universal* if it weakly simulates universal quantum computation, for example by weakly simulating $C|0^n\rangle$ for any circuit $C$.

Then, returning to Clifford circuits, the following theorem of [Got98] (paraphrased here) is significant for simulation.

**Theorem 2.6.3** (Gottesman-Knill Theorem (paraphrased))**.** There exists an efficient classical algorithm which weakly simulates $C|0^n\rangle$ given any Clifford circuit $C$ on $n$ qubits with final computational basis readout measurements.

Clifford gates are also relevant to *fault-tolerant quantum computing*, which uses quantum error-correcting codes to preserve the correctness of a computation under noisy conditions. For a full discussion of fault-tolerance, we again refer to [NC10]. For our purposes, it suffices to note that Clifford gates can be implemented conveniently on a useful class of quantum error-correcting codes called stabilizer codes. The $\mathsf{T}$ gate cannot, however, which motivates the technique of *magic state injection*, important for this thesis. Magic state injection depends on the circuit identity shown in fig. 2.2. For an $n$-qubit circuit $C$, the magic state injected form of $C$ replaces all $\mathsf{T}$ gates in $C$ with the $\mathsf{T}$ gadget shown in fig. 2.2, producing a circuit $C'$ on $n+t$ qubits, some of which are initialized to the $|T\rangle$ state (defined in eq. (2.3.2)).

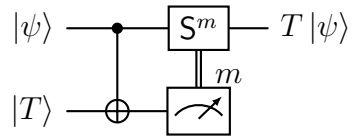



Figure 2.2: A $\mathsf{T}$ gadget, used in magic state injection to replace $\mathsf{T}$ gates in a circuit. The overall effect of the gadget is to apply a $T$ operation to an arbitrary single-qubit input state $|\psi\rangle$.

Note that the circuit $C'$ resulting from magic state injection on any circuit $C$ is composed entirely of Clifford gates, which is desirable for fault-tolerance. However, some of the Clifford gates in $C'$ are classically controlled by measurements. We call this kind of circuit an *adaptive Clifford circuit*. More generally, we use *adaptivity* to refer to this kind of internal classical control (sometimes also called *feedforward*). During the execution of an adaptive circuit, once a controlling measurement is made, we say that the adaptivity dependent on that measurement has been *fixed*, as the dependent gate is now either present or not, according to the outcome.

# Chapter 3

# Secure Delegated Quantum Computation

This chapter gives a background presentation of *secure delegated quantum computation* (SDQC), the cryptographic task for which we later give a novel protocol. We begin with a general description of the cryptographic setting of SDQC (section 3.1). We then provide a brief historical overview of the topic (section 3.2). Next, we highlight two especially relevant SDQC protocols and enumerate their resource requirements (section 3.3), since they will be the main points of comparison when we discuss the advantages of our novel protocol in chapter 6. As we cannot cover the large and varied SDQC literature comprehensively, we end by briefly mentioning some other significant topics closely connected to SDQC (section 3.4) and refer the reader to [Fit17] for a more complete survey.

Before beginning, some comments on terminology are called for. First, "secure delegated quantum computation" (or "SDQC") is used with somewhat differing scopes in the literature. The term has been applied both more narrowly (e.g., in [KLMO26], where only protocols with verification are considered "SDQC") and more broadly (e.g., in [BCC+20], where protocols which do not achieve information-theoretic security are included under the label) than it will be here. Second, other terms have been used to refer to what we here call "SDQC," including, most notably, "blind quantum computing" (or "BQC"). Despite the familiarity of "BQC," we opt for "SDQC" in this thesis. The reason is that "BQC," too, is used inconsistently in the literature; while it sometimes covers the ground we intend, it also seems to have a frequent narrower

meaning. "BQC" is closely associated with the protocol of [BFK09] and sometimes (e.g., in parts of [Fit17]) the term refers to protocols which share particular setting features with [BFK09], such as presentation in the 1WQC model (see section 2.4) and security for the entire computation. Since we mean to be more general (indeed, we are especially interested in protocols presented in other computational models and which hide the input to a computation only), we use "SDQC."

## 3.1 Setting

*Secure delegated quantum computation* (SDQC) refers to a cryptographic setting and task involving two parties, the client (or Alice) and the server (or Bob).[1] The client has in mind a quantum computation, usually identified with a circuit $C$, and an input state $|\psi\rangle$. The client wishes to compute $C|\psi\rangle$, but does not have sufficient resources to compute it locally. So, the client *delegates* the computation to a high-quantum-resource server. Multiple rounds of quantum and/or classical interaction between client and server are allowed.

### 3.1.1 Correctness and Security

The goals of an SDQC protocol are *correctness* and *security*. For correctness, the client should receive her correct answer $C|\psi\rangle$ by the end of the protocol. The correctness requirement only covers cases where the server follows the protocol honestly. That is, it is not required that the client can recover a correct output when the server deviates from the protocol, or even detect that this has happened (rather, this detection requirement is called *verification* — see section 3.4). For *security*, the server is considered to be the cryptographic adversary. Security requires that the server should not learn some aspect of the computation, typically either $C$ or $|\psi\rangle$ or both. Security *is* required to hold against deviating servers. The level of security typically sought is *information-theoretic*, i.e., security that does not depend on computational assumptions. Some minimal information about the computation, for example an up-

[1] In this chapter, we use the client/server language, as is more standard in SDQC literature, but in later chapters, when PBC is involved, we will freely switch between client/server and Alice/Bob naming. Alice/Bob is not uncommon in the SDQC literature, either, for instance in [MF13].

per bound on its space or time complexity, is inevitably gained by the server, but this is allowed and accounted for in the security definition.

Formal security definitions in SDQC vary between precise settings, but they typically involve a *simulator*[2], a machine which does not interact with the client and computes some output. Security requires that, for any server deviating from the protocol, even computationally unbounded ones, there exists a simulator that can produce an output distribution that is very close to the server's, for some reasonable measure of closeness. Proofs of security in SDQC usually give a construction of such a simulator. The simulator may be given access to aspects of the computation that the protocol does not intend to hide. For example, when the security goal is input-hiding, the simulator will be given the circuit description as input, just as the server is assumed to be given it in the protocol. Often, for full generality, we allow that the server may share prior entanglement with the client, and then we aim to show that the server cannot *increase* its correlation with the client via the interaction. This is modeled by allowing the simulator also to access the server's subsystem of some general input state, of which the client holds the other subsystem. A formal SDQC security definition, in the context of our novel protocol, is presented in chapter 7.

### 3.1.2 Resources

Resource reduction is a major theme of the SDQC literature. In fact, efficient SDQC with full, unconditional security has technically been possible since the very earliest SDQC protocol (of [Chi05]), but at a very high resource cost (see section 3.2 for more discussion). All subsequent work on SDQC can therefore be understood as introducing new methods to achieve better resource performance. The resource requirements of an SDQC protocol can be broadly divided into client resources, communication resources, and server resources, which we consider in turn. Client resources vary from instance to instance and are rarely fully classical.[3] However, the client's resources

[2]This notion of a simulator, common in cryptography, is unconnected to classical simulation of quantum computation, as discussed for example in section 2.6. Perhaps confusingly, due to the inclusion of cryptography *and* PBC, both senses of "simulation" appear in this thesis and should be kept separate.

[3]At the security level and setting we are interested in, *all* known protocols require some quantum resources for the client. For some relaxed security levels and in some alternative settings, classical client SDQC has been achieved — see sections 3.2 and 3.4.

must be insufficient for universal quantum computation, or else the delegation premise is defeated. Typical examples of limited quantum resources possessed by the client include the ability to perform measurements or operations from a limited set (e.g., single-qubit Pauli measurements or gates) and the ability to prepare and send qubits chosen randomly from a small finite set. Communication resources encompass the number of rounds of interaction, the kind (classical or quantum) of interaction, and the volume (e.g., number of bits or qubits) of messages sent and received. Server resources in SDQC include quantum memory (i.e., the size of the quantum register required) and type and number of operations and measurements. The server's resources should be greater than, or at least distinct from, the client's, to motivate the delegation premise.

Of these three broad categories, client resources have been the main focus of most work on resource reduction. There is a good reason for this: since the initial motivation for SDQC is to allow low-resource parties to access quantum computation, reducing client resources corresponds to increasing this access even more widely. Reduction of communication resources, especially of quantum communication, has also been an active area of work. Server resource reduction remains comparatively understudied.

### 3.1.3 Input-Hiding and Circuit-Hiding

A major distinction within the SDQC landscape, which has special importance for this thesis, is between protocols where the security goal is to keep the input secret and those which aim to conceal the computation. We call these categories *input-hiding* and *circuit-hiding* respectively.

It is important to appreciate that the distinction is at least somewhat artificial. An input-hiding protocol may be "lifted" to circuit-hiding in a few ways. For one, given an input-hiding protocol, the client can derive a circuit-hiding protocol as follows. Let $U$ be a *universal circuit* which accepts as input a pair $(C, |\psi\rangle)$, where $C$ is a classical description of a quantum circuit and $|\psi\rangle$ is a quantum state, and runs $C$ on $|\psi\rangle$ so that $U((C, |\psi\rangle) = C|\psi\rangle$. Such a circuit $U$ is clearly constructable. Then, declare the circuit in the input-hiding scheme to be $U$ and execute the protocol to hide the input $(C, |\psi\rangle)$ to $U$. The protocol will correctly return $C|\psi\rangle$ to the client,

and its security implies that $C$ cannot be learned by the server. We have thus derived a circuit-hiding scheme. This use of a universal circuit construction is discussed in [ABE10], for instance. As another method, suggested in [Chi05], an input-hiding protocol may be iterated gate by gate over a fixed, repeating sequence of single-gate computations, for example $(\mathsf{H}, \mathsf{S}, \mathsf{CNOT}, \mathsf{T}, \mathsf{H}...)$, with unwanted gates applied secretly to auxiliary qubits which are then ignored by the client.

Despite this, we take the distinction between input-hiding and circuit-hiding seriously. First, such a lifting from input-hiding to circuit-hiding comes at a huge resource cost. For example, in the universal circuit construction, $U$ includes many more non-Clifford gates than $C$, and the gate iteration method incurs enormous communication overhead. As noted, resource-efficiency is all-important when comparing SDQC schemes. Second, it is typically very easy to tell from a protocol's design whether it primarily performs input-hiding or circuit-hiding. We will sometimes use the terms *intrinsically input-hiding* and *intrinsically circuit-hiding* to describe a protocol whose construction naturally achieves one or the other type of security, while acknowledging that it may in theory be converted to the other functionality through some more elaborate construction.

## 3.2 Historical Overview

In this section we provide a brief history of SDQC.

There are classical precursors to the task which cover many of the same themes. [Fei85] considered the problem of hiding the input $x$ to a delegated classical computation $f$ which is known to both parties, which the author calls *computing with encrypted data*. A subsequent work, [AFK87], showed a complexity-theoretic no-go result, namely, that a solution for NP-hard problems implies a collapse of the polynomial hierarchy, for a generalized input-hiding task. Interestingly, the phrase "blind quantum computation" was first used in [AS06], which proposed a partial solution to this *classical* problem using quantum resources.

The first secure delegated scheme for quantum computation is typically (see [Fit17]) credited to [Chi05], which is a scheme for (quantum) input-hiding. In the protocol, the client applies a quantum one-time pad to the input state, sends it to the

server for the execution of a specific gate (known to both parties), who then sends it back. The client decrypts and refreshes the one-time pad for the next gate. As a result, the protocol requires one round (i.e., a message in each direction) of quantum communication per Clifford gate and two rounds in the case of a T gate. A related early scheme is in [ABE10], which also performs input-hiding with a number of rounds of quantum communication proportional to the number of gates, but which requires only a constant register size of the client (whereas [Chi05] requires a register at least as large as the size of the input).

The protocol of [BFK09], called *universal blind quantum computation* (UBQC), marks a significant development. First, UBQC requires only a single instance of one-way quantum communication (from client to server), plus one additional round of two-way quantum communication if the input and output are quantum; all interaction during the protocol is classical. Also, UBQC is distinctive for being presented in the one-way model of quantum computation. Perhaps most significantly, UBQC is the first SDQC protocol to be intrinsically circuit-hiding. A complementary approach, where the client measures and the direction of one-way communication is reversed, was subsequently proposed in [MF13].

Of particular importance for our project is the protocol of [FBS+14] (overview and experimental demonstration) and [Bro15] (theoretical materials), called *quantum computing on encrypted data* (QCED). This work returned to the input-hiding task of early SDQC schemes, accomplishing it now with the same quantum communication resources as the UBQC protocol (i.e., one instance of one-way, plus one two-way round for a quantum input/output). Compared with UBQC, this protocol reduces the client resources by requiring four-way, rather than eight-way, state preparation; the factor of two reduction is directly related to the weaker task of input-hiding versus circuit-hiding (as observed in [Bro15]).

Some work has addressed the question of resource reduction more directly. The question of whether SDQC can be accomplished by a fully classical client has received particular attention, to the extent that [BS16] includes this question as a significant open question in quantum cryptography. A number of complexity theoretic no-go results ([ACGK19], [MK19], [MNTT19]) indicate that a fully classical client is unlikely. Other works aim to optimize or prove bounds on quantum communication resource requirements, especially the number of qubits transmitted. Examples include

[GMMR13], [MPF13], and, recently, [DK25], which uses entropy-bounding techniques to examine optimal quantum communication requirements in specific SDQC settings.

## 3.3 Resource Requirements of Protocols of Interest

In this section, we enumerate the resource requirements of the UBQC ([BFK09]) and QCED ([FBS+14], [Bro15]) protocols, to facilitate a comparison with our novel protocol later in chapter 6. We focus on these two since they are the most closely comparable protocols to our own, and since we are not aware of any others which achieve a comparable task with better resource costs. The resources required by the protocols is summarized later, when we do the comparison, in table 6.1; it may be helpful to refer to that table while reading this section. We refer the reader to the original papers for explanations of how the protocols work, since these are not needed for our purposes.

### 3.3.1 Universal Blind Quantum Computation

We examine the resource requirements for the UBQC protocol when executed on a computation expressed as a circuit $C$ with $n$ data qubits, $t$ $\mathsf{T}$ gates, and depth $d$.[4]

We begin with client resources. First, the client is required to prepare single-qubit states selected at random from a set of eight spaced equally in the equatorial plane of the Bloch sphere:

$$\{\frac{1}{\sqrt{2}}\big(\,|0\rangle + e^{i\theta}\,|1\rangle\,\big) : \theta \in \{0, \frac{\pi}{4}, \frac{2\pi}{4}, ..., \frac{7\pi}{4}\}\},$$

which we call eight-way state preparation. The client must prepare $\mathcal{O}(nd)$ such states. (The exact number depends on the tiling pattern of the brickwork state, which depends on the arrangement of gates in the circuit, so we give an asymptotic description. The constant hidden by the $\mathcal{O}$ notation is not more than 10.) As for communication resources, each of these $\mathcal{O}(nd)$ qubits is sent to the server, in a single instance of one-way quantum communication. Subsequent communication is two-way classical

[4]The UBQC protocol can also be executed on a computation given in the 1WQC model. We focus on circuits to make a direct comparison possible with QCED and our protocol.

communication of a small constant number of bits per round. As for server resources, the server is required to store all $\mathcal{O}(nd)$ states and perform entangling operations on them. Then, the server performs single-qubit measurements in any of eight bases corresponding to the angles of preparation of the client's states, notably including magic angles.

The resources above are sufficient to hide the computation and an input which has a known circuit construction (with the $\mathcal{O}(nd)$ term increasing with the size of the circuit construction), while receiving a classical output (say, a sample from some readout measurements). We will be most interested in classical input states, which increase the $\mathcal{O}(nd)$ term by at most $n$, which is absorbed by the asymptotic description. The protocol may be extended (Protocols 2 and 3 of [BFK09]) to incorporate arbitrary quantum inputs and outputs, at the cost of a round of two-way quantum communication and the client capability of performing Pauli $X$ and $Z$ operations, but this does not concern us here.

### 3.3.2 Quantum Computing on Encrypted Data

On the same circuit $C$, with $n$ input qubits, $t$ $\mathsf{T}$ gates, and depth $d$, the QCED scheme offers to hide an input to the circuit at the following resource costs. For the client, the state preparation requirement is similar but now only four-way, i.e., from the following set of four states:

$$\{\frac{1}{\sqrt{2}}\big(\,|0\rangle + e^{i\theta}\,|1\rangle\,\big) : \theta \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}\}.$$

The client prepares and sends exactly $t$ such qubits to the server in a single instance of one-way quantum communication. $t$ rounds of single-bit, two-way classical communication follow. The server needs a register of size $n+t$: $n$ to store the input state, and $t$ to store the qubits sent by the client. (Alternatively, a register of size $n+1$ suffices, but at the cost of a significant increase in quantum communication, if the client's states are sent as $t$ separate single-qubit, one-way instances of quantum communication, occurring online during the protocol execution.) The server also requires the ability to perform universal quantum computation in the circuit model (specifically, gates from the set $\{\mathsf{H}, \mathsf{S}, \mathsf{CNOT}, \mathsf{T}\}$ and single-qubit computational basis measurements).

## 3.4 Related Topics

We conclude by briefly surveying a few topics which are closely linked to SDQC.

*Quantum homomorphic encryption* (QHE), like SDQC, involves the delegation of a computation from a low-resource client to a quantum server. For our purposes, the key differences are that QHE does not allow interaction, and attains security only with respect to computational assumptions, as opposed to the information-theoretic security typically sought in SDQC. QHE was defined and first investigated in [BJ15], where a partial QHE scheme was introduced; this was later extended in [Mah18] to a scheme for full QHE with a classical client. *Verification* is the feature of a delegated computation whereby the client can detect an incorrect output given by a deviating server. Verification is often layered on top of an SDQC protocol, and they naturally co-occur to such an extent that it has been speculated whether verification and SDQC are intrinsically linked tasks ([Fit17]). SDQC scenarios with *multiple servers* have been considered since [BFK09], where it was shown that classical-client SDQC is possible if the servers are prevented from communicating (though they may share entanglement). Finally, we mention that the *experimental demonstration* of various SDQC protocols has been an active line of work, for example in [BKB+12], [BFKW13], and [FBS+14], and more recently in [DNM+24] and [WSS+25].

# Chapter 4

# Pauli-Based Computation

In this chapter, we present Pauli-based computation (PBC), the model of quantum computation introduced in [BSS16] that is the focus of the remainder of the thesis. After some background (section 4.1), we describe PBC model algorithmically and prove its correctness, efficiency, and universality (section 4.2). Next, we present three equivalent formulations of PBC which provide useful ways of understanding the model (section 4.3). We then describe some advantages of PBC, with particular attention to qubit virtualization, also known as hybrid computation, a technique that PBC enables (section 4.4). We conclude with an example of the PBC algorithm executed on a small circuit, to aid in understanding the model (section 4.5).

This chapter is primarily a literature review. Exceptions are certain novel aspects of the presentation of PBC; those contributions are indicated with footnotes.

## 4.1 Background

PBC is a model of quantum computation introduced in [BSS16]. PBC uses efficient classical processing to reduce the quantum computational work required to perform a computation, compared to other models. PBC is *sampling-universal*, as defined in section 2.6; it samples from the output distribution of (or weakly simulates) universal quantum computation. We will prove this fact in section 4.2 by giving a derivation from an arbitrary Clifford + $\mathsf{T}$ circuit $C$ to a PBC which samples from the same distribution as $C\left|0^n\right\rangle$.

PBC is a relatively understudied model of quantum computation (see [Per24]). Important theoretical works on the model include the papers [PG23] and [Per23], which find optimized implementations of PBC and consider PBC on *qudits*, or higher-dimensional quantum systems, respectively, and the Ph.D. thesis [Per24], which encompasses the two previously-mentioned papers and which we cite frequently in this section because of its excellent exposition of PBC. Further implementation optimizations were reported in [PG25]. The recent preprint [HCWY25] develops quantum error correction techniques for use with PBC, highlighting the relevance of PBC to fault-tolerant quantum computation.

## 4.2 The PBC Model

In this section, we present the PBC model as an algorithm (algorithm 1) which takes an arbitrary $n$-qubit circuit $C$ as input and returns a sample from the output distribution of $C|0^n\rangle$. The algorithm proceeds as an interaction between a classical party, Alice, and a quantum party, Bob, who work together to produce the required sample.

### 4.2.1 PBC Algorithm

The PBC algorithm is presented in algorithm 1.[1] The algorithm takes as input an $n$-qubit circuit $C$ and proceeds in four steps. Steps 1 and 2 are classical, comprising pre-processing of $C$ and an initialization step, which Alice does locally. Step 3, the core step of the algorithm, is a for-loop that runs over each measurement in the processed circuit. For each measurement, Alice determines which of three relevant cases it falls into, by a comparison with previous measurements, then determines its outcome, sometimes classically and sometimes by querying Bob who uses his quantum resources; interestingly, Alice requires quantum help from Bob in only one of the three cases. In step 4, Alice takes a subset of the measurement outcomes from step 3 and returns them; this output is a sample from the output distribution of $C|0^n\rangle$.

We now describe the PBC algorithm (algorithm 1) in greater detail. Alice starts

[1] Our presentation of the algorithm, in terms of the structure and organization of steps, is novel. The content is due to [BSS16], with many details filled in previously in [Per24].

**Algorithm 1** PBC Algorithm

Initially, Alice holds circuit $C$ with an $n$-qubit data register, $t$ $\mathsf{T}$ gates and $w \leq n$ readout measurements. Bob holds a magic register of $|T\rangle^{\otimes t}$.

1. Alice performs magic state injection on $C$ to produce circuit $C'$ with $t$ auxiliary magic qubits, $t$ internal gadget measurements, and $w$ readout measurements.
2. Alice creates list IC of independent and commuting Pauli operators on $n+t$ qubits and initializes it to IC $= \{Z_1, Z_2, ..., Z_n\}$.
3. For each measurement $i$ in $C'$, for $1 \leq i \leq t+w$:
   (a) Alice propagates it to the beginning of $C'$ resulting in Pauli $P_i$.
   (b) Alice compares $P_i$ to the elements of IC, and determines which of these three cases it falls into:
      (AC) $P_i$ anti-commutes with some element of IC.
      (DC) $P_i$ commutes with all elements of IC and depends on a subset of them.
      (IC) $P_i$ commutes with all elements of IC and is independent of them.
   (c) Alice determines the measurement outcome $m_i \in \{0, 1\}$ and performs some updates, according to the case, as follows:
      (AC) Alice draws $m_i$ uniformly at random, updates $C'$ by appending a unitary $V_i^{(m_i)}$ to the beginning, and does not update IC.
      (DC) Alice finds $m_i$ deterministically from past outcomes and updates neither $C'$ nor IC.
      (IC) Alice sends a message to Bob, requesting a quantum measurement on his register. Bob replies with the outcome $m_i$. Alice does not update $C'$ and adds $P_i$ to IC.
   (d) Alice updates $C'$ to fix the adaptivity of the $i$-th gadget according to $m_i$.
4. Alice takes the measurement outcomes $m_i$ for $t+1 \leq i \leq t+w$, i.e. those corresponding the readout measurements, to be the required sample from the output distribution of $C|0\rangle$.

with a classical description of a Clifford + $\mathsf{T}$ circuit, $C$, with $n$ input or data qubits, $t$ $\mathsf{T}$ gates, and $w \leq n$ final computational basis measurements which we call the *readout* measurements. Bob starts with a quantum register initialized to $|T\rangle^{\otimes t}$. Alice wishes to receive a sample from the output distribution (i.e., from the probability distribution on the outcomes of the readout measurements) of $C|0^n\rangle$. In the PBC algorithm, the

input state to $C$ is always the all-zero state $|0^n\rangle$; we make a small generalization about this at the end of this section.

In step 1, Alice performs magic state injection on $C$, producing a new circuit $C'$ on $n+t$ qubits with $t$ internal computational basis measurements and $w$ readout measurements (see section 2.6 for an explanation of magic state injection). To be more precise, Alice is here operating on a classical description of $C$ to produce a classical description of $C'$ (so, for instance, she does not actually initialize the magic states of $C'$). $C'$ is an adaptive Clifford circuit.

In step 2, Alice creates her list of quantumly measured Paulis which she will update through step 3. The list is called IC for *independent and commuting* since, as will be seen, all quantumly measured Paulis will have these properties. Formally, IC is a set of $(n+t)$-qubit Pauli operators. Alice initializes IC to include Paulis $Z_1, Z_2, ..., Z_n$ (and no others), which we note is the stabilizer of the input state $|0^n\rangle$.

In step 3, Alice indexes the measurements of $C'$ in order from earliest to latest with indices 1 to $t+w$. (For measurements occurring at the same time step, any ordering can be used; we use the convention of ordering them from bottom to top in the circuit, or equivalently from highest wire index to lowest wire index.) For each measurement $i$, for $1 \leq i \leq t+w$:

In step 3(a), Alice considers the $i$-th measurement to be a measurement of Pauli operator $Z_k$ (where $k \in [n+t]$ is the index of the wire the measurement is on), since it is a computational basis measurement. She propagates $Z_k$ to the beginning of $C'$, yielding a Pauli operator $P_i$. This is a straightforward conjugation through a *non-adaptive* Clifford circuit, since the algorithm will fix (i.e., remove by deciding an outcome for the controlling measurement) adaptivity progressively so that, when the $i$-th measurement is considered, all earlier adaptivity in $C'$ has been fixed.

In step 3(b), Alice compares the propagated Pauli $P_i$ to the current elements of IC (note that IC is updated at some steps of the algorithm). She decides which of three relevant cases holds. The cases are:

(AC) *Anti-commuting (AC) case:* Here, $\{P_i, Q_j\} = 0$ for some $Q_j \in$ IC.

(DC) *Dependent and commuting (DC) case:* In this case, $[P_i, Q_j] = 0$ for all $Q_j \in$ IC *and* $P_i$ is multiplicatively dependent on operators in IC (that is, $P_i = \prod_{Q_j \in S} Q_j$ for some $S \subseteq$ IC).

(IC) *Independent and commuting (IC) case:* Here, $[P_i, Q_j] = 0$ for all $Q_j \in \text{IC}$ *and* $P_i$ is multiplicatively independent of IC (i.e., there does not exist any $S \subseteq \text{IC}$ such that $P_i = \prod_{Q_j \in S} Q_j$).

Recalling that any two Paulis either commute or anti-commute, it is clear that these cases are exhaustive.

In step 3(c), depending on the case determined in step 3(b), Alice determines $m_i$ and makes updates to the circuit and IC, as follows:

(AC) Alice draws the measurement outcome $m_i$ uniformly at random from $\{0, 1\}$. She updates the circuit $C'$ by appending the Clifford unitary gate

$$V_i^{(m_i)} = \big((-1)^{m_i} P_i + (-1)^{m_j} Q_j\big)/\sqrt{2}$$

to the beginning of the circuit, where $Q_j$ is the Pauli in IC, found in step 3(b), with which $P_i$ anti-commutes and $m_j$ is its measurement outcome.[2] Alice does not update IC.

(DC) Alice computes $m_i$ deterministically as

$$m_i = \bigoplus_{Qj \in S} m_j,$$

where $S$ is the subset of IC, found in step 3(b), on which $P_i$ depends, and $m_j$ is the measurement outcome of the $S$ element $Q_j$.[3] Alice updates neither $C'$ nor IC.

(IC) In this case only, the measurement must be handled quantumly. Alice sends a classical message to Bob consisting of $P_i$ truncated to its final $t$ qubits and phase (i.e., with the first $n$ places of the Pauli string deleted). Bob measures this $t$-qubit Pauli operator non-destructively on his $t$-qubit register, causing the state

[2]Note that, in general, $Q_j$ is not unique; $P_i$ may anti-commute with multiple IC Paulis. Any such $Q_j$ may be chosen here and correctness will hold, as can be seen from the fact that the proof of proposition 4.2.3 does not depend on the choice of $Q_j$.

[3]In this case, $S$ *is* unique, since the Paulis in IC are mutually independent. That is, suppose subsets $S = \{S_1, ..., S_k\}, R = \{R_1, ..., R_\ell\} \subseteq \text{IC}$ are such that $S_1 \cdots S_k = R_1 \cdots R_\ell = P_i$. If the subsets are distinct, then one set contains an element not in the other; say without loss of generality that this set is $R$ and the element is $R_\ell$. Then $R_{\ell-1} \cdots R_1 S_1 \cdots S_k = R_\ell$, contradicting the independence of IC. So $S = R$.

of his register to evolve and producing a classical outcome $m_i \in \{0, 1\}$. Bob sends $m_i$ to Alice. Alice does not update $C'$ and adds $P_i$ (the full $(n+t)$-qubit Pauli) to IC.

Note that in step 3(b), cases AC and DC, if needed to compute $V_i^{(m_i)}$ or $m_i$, respectively, Alice takes the measurement outcomes of the initial $n$ IC-elements, $Z_1, ...Z_n$, all to be 0.

In step 3(d), regardless of which of the three cases was just entered and performed in steps 3(b) and 3(c), Alice updates $C'$ to fix the adaptivity at the $i$-th $\mathsf{T}$ gadget according to the value of $m_i$ just determined. So, if $m_i = 1$, Alice adds the $\mathsf{S}$ gate controlled by the measurement to the circuit, and she removes it if $m_i = 0$.

Steps 3(a) through 3(d) are iterated $t + w$ times, once for each measurement in $C'$, before moving on to step 4.

In step 4, Alice takes $m_{t+1}$ to $m_{t+w}$ (i.e., those outcomes generated by the final $w$ iterations of step 3) to be the outcomes of the corresponding readout measurements. These outcomes are sampled from the output distribution of $C|0^n\rangle$.

This concludes the detailed description of the PBC algorithm. Tables 4.1 and 4.2 depict steps 3(b) and 3(c), respectively, the most involved parts of the procedure, in a graphical way. Also, in section 4.5, we work through an example of the PBC algorithm executed on a small circuit. The reader may wish to consult some of these materials before proceeding.

### 4.2.2 Proofs of Correctness

We now prove the correctness of algorithm 1. We first state the correctness as a theorem:

**Theorem 4.2.1** (Correctness of PBC)**.** The output of the PBC algorithm run on circuit $C$ is distributed identically to the output of $C|0^n\rangle$.

The remainder of this section proves this theorem.[4] We specifically aim to show that, at an iteration $i$ of step 3 of algorithm 1, the procedure of each PBC case (AC,

[4]The structure of our proof of correctness is novel, and we fill in some details left implicit in earlier work. The original proof of correctness of PBC is in [BSS16].

Table 4.1: Table representation of step 3(b), case determination, of the PBC algorithm (algorithm 1). A case (AC, DC, or IC) is selected for the next step according to the outcomes of two checks, dependence and commutativity.

| | | Commutativity check | |
|---|---|---|---|
| | | Anti-commutes ($\exists Q_j \in$ IC s.t. $\{P_i, Q_j\} = 0$) | Commutes ($\forall Q_j \in$ IC : $[P_i, Q_j] = 0$) |
| Dependence check | Dependent ($\exists S \subseteq$ IC s.t. $P_i = \prod_{Q \in S} Q$) | AC | DC |
| | Independent ($\forall S \subseteq$ IC : $P_i \neq \prod_{Q \in S} Q$) | AC | IC |

Table 4.2: Table representation of step 3(c), case actions, of the PBC algorithm (algorithm 1).

| Case | $m_i$ determined as... | $C'$ update | IC update |
|---|---|---|---|
| AC | $m_i \in_R \{0, 1\}$ | $V_i^{(m_i)} = \frac{1}{\sqrt{2}}\left((-1)^{m_i} P_i + (-1)^{m_j} Q_j\right)$ appended to beginning | n/a |
| DC | $m_i = \bigoplus_{Q_j \in S} m_j$ | n/a | n/a |
| IC | $m_i$ = outcome of quantum measurement of final $t$ qubits of $P_i$ | n/a | Add $P_i$ to IC |

DC, and IC) has an effect identical to the effect of quantumly measuring the full $(n+t)$-qubit Pauli $P_i$ determined at step 3(a). By *effect*, we mean the measurement outcome produced *and* the resulting state of the overall system. Formally, at the beginning of iteration $i$, we say the circuit and initial state are in the form $C'_i(|0^n\rangle \otimes |\psi_i\rangle)$, where $|\psi_i\rangle$ is the $t$-qubit state on Bob's register at the beginning of iteration $i$, and where we now index the circuit $C'$ by the iteration number $i$ to better track its changes (since it may be updated). Then iteration $i$ of step 3 of the PBC procedure has the effect of producing a measurement outcome $m_i$ and leaving the circuit and initial state in the form $C'_{i+1}(|0^n\rangle \otimes |\psi_{i+1}\rangle)$. And, measuring the full $(n+t)$-qubit Pauli $P_i$ on the state $|0^n\rangle \otimes |\psi_i\rangle$ would produce a measurement outcome $\widetilde{m_i}$, leave a post-measurement state of $|\widetilde{\phi_i}\rangle$ on $(n+t)$ qubits, and, once adaptivity is fixed according to the measurement outcome, produce an updated circuit of $\widetilde{C'_i}$. We will show, for each case AC, DC,

and IC, that:

(i) $m_i$ is distributed identically to $\widetilde{m_i}$, and

(ii) $C'_{i+1}(|0^n\rangle \otimes |\psi_{i+1}\rangle) = \widetilde{C'_i}(|\widetilde{\phi_i}\rangle)$ whenever $m_i = \widetilde{m_i}$,

where (ii) is an equality of quantum states.

Showing (i) and (ii) for a single iteration $i$ suffices to show the correctness of PBC, by the following argument. First, consider a hypothetical algorithm like PBC but where the full $(n+t)$-qubit Pauli $P_i$ is measured quantumly at every step. This hypothetical algorithm correctly produces a sample from the output distribution of $C'|0^n\rangle$, which follows straightforwardly from the correctness of magic state injection and the propagation of Pauli measurements. (This hypothetical procedure is very similar to a regular execution of the circuit $C'$, except with computational basis measurements propagated earlier and measured as $(n+t)$-qubit Paulis.) Second, showing that (i) and (ii) hold of a single iteration $i$ of step 3 of PBC inductively shows that PBC samples *every* measurement outcome $m_i$ from the same distribution as in the hypothetical procedure, since (as a base case to the inductive arugment) the starting circuit and state $C'|0^n\rangle$ are identical in the hypothetical procedure and in PBC. Chaining these two facts proves the correctness of PBC.

We will show that (i) and (ii) hold for each case AC, DC, and IC separately, in proposition 4.2.3, proposition 4.2.4, and proposition 4.2.5 respectively. First, however, we establish the following fact:

**Proposition 4.2.2.** At the start of iteration $i$ of step 3 of PBC, let $\mathrm{IC}_i$ denote the IC list (indexed now to track updates) and let $|0^n\rangle \otimes |\psi_i\rangle$ denote the $(n+t)$-qubit state at the beginning of $C'_i$ as above. Then $|0^n\rangle \otimes |\psi_i\rangle$ is in an intersection of eigenspaces of the Pauli operators in $\mathrm{IC}_i$.

**Proof.** Initially, at $i = 1$, IC has elements $\{Z_1, ..., Z_n\}$. Since $\{Z_1, ..., Z_n\}$ is the stabilizer of $|0^n\rangle$ and these operators act as identity on the final $t$ qubits, $|0^n\rangle \otimes |\psi_1\rangle$ is in the intersection of the eigenspaces of the IC operators.

Suppose the claim holds for $i = k$. Then at $i = k+1$, if iteration $k$ was AC or DC case, no change has been made to Bob's register, so $|\psi_k\rangle = |\psi_{k+1}\rangle$, and the IC list is also unchanged, so the claim holds. If iteration $k$ was IC case, then $\mathrm{IC}_{k+1} =$

$\text{IC}_k \cup \{P_k\}$. Also, $P_k$ has $Z$ and $I$ values only on its first $n$ qubits (since it commutes with $\{Z_1, ..., Z_n\}$), so $|0^n\rangle$ is an eigenstate of the first $n$ qubits of $P_k$. And, $|\psi_{k+1}\rangle$ is the result of projecting $|\psi_k\rangle$ into an eigenspace of the last $t$ qubits of $P_k$, meaning $|\psi_{k+1}\rangle$ is an eigenstate of these last $t$ qubits. So the full $(n+t)$-qubit state $|0^n\rangle \otimes |\psi_{k+1}\rangle$ is in an eigenspace of the full operator $P_k$. Also, $|0^n\rangle \otimes |\psi_{k+1}\rangle$ is in the intersection of eigenspaces of operators in $\text{IC}_k$, since all IC operators commute. So the claim holds for $i = k + 1$, and inductively, the claim holds for all iterations $1 \leq i \leq t + w$. ∎

Now we proceed with the correctness of each case:

**Proposition 4.2.3** (AC correctness)**.** (i) and (ii) hold when iteration $i$ of step 3 of regular PBC falls into the AC case.

**Proof.** For (i), in the AC case, $P_i$ anti-commutes with some operator $Q_j$ in IC. So, by a property stated in section 2.6, observing that the Paulis $X_1$ and $Z_1$ anti-commute, there exists an $(n + t)$-qubit Clifford operator $C \in \mathcal{C}_{n+t}$ such that $CP_iC^\dagger = X_1$ and $CQ_jC^\dagger = Z_1$. Also, by proposition 4.2.2, $|0^n\rangle \otimes |\psi_i\rangle$ is in an eigenspace of $Q_j$. Measuring $P_i$ on $|0^n\rangle \otimes |\psi_i\rangle$ is therefore unitarily equivalent to measuring $X_1$ on an eigenstate of $Z_1$, which has outcomes distributed uniformly in $\{0, 1\}$. Therefore the uniformly random draw of $m_i$ in the AC case of PBC produces an outcome distributed identically to $\widetilde{m_i}$.

For (ii), the measurement of $P_i$ on $|0^n\rangle \otimes |\psi_i\rangle$ with outcome $\widetilde{m_i}$ would produce a post-measurement state of $|\widetilde{\phi_i}\rangle$ which is exactly $V_i^{(m_i)}(|0\rangle \otimes |\psi_i\rangle)$ for $V_i^{(m_i)}$ as described in the PBC algorithm. Letting $|\psi_{i+1}\rangle = |\psi_i\rangle$ and appending the unitary $V_i^{(m_i)}$ to the beginning of $C'_i$, as is done in the AC case, therefore results in a circuit $C'_{i+1}$ which has the required property. ∎

**Proposition 4.2.4** (DC correctness)**.** (i) and (ii) hold for the DC case.

**Proof.** In the DC case, $P_i = \prod_{Q_j \in S} Q_j$ for a subset $S$ of $\text{IC}_i$. By proposition 4.2.2, $|0^n\rangle \otimes |\psi_i\rangle$ is already in the intersection of the eigenspaces of the elements of $S$, so the state is already in an eigenspace of $P_i$. The eigenvalue associated to this eigenspace of $P_i$ is the product of the eigenvalues of the eigenspaces of the $Q_j$ projections, namely $\prod_{Q_j \in S}(-1)^{m_j} = (-1)^{\bigoplus_{Q_j \in S} m_j}$, showing (i).

For (ii), we simply note that the measurement leaves the state unchanged, since the projection has in effect already been performed, meaning that $|\widetilde{\phi_i}\rangle = |0^n\rangle \otimes |\psi_i\rangle$.

Leaving the quantum state so $|\psi_{i+1}\rangle = |\psi_i\rangle$ and making no circuit update aside from fixing adaptivity so $C'_{i+1} = \widetilde{C'_i}$, as is done in the DC case, gives the required property. ∎

**Proposition 4.2.5** (IC correctness)**.** (i) and (ii) hold for the IC case.

**Proof.** In the IC case, $P_i$ commutes with all operators in $\mathrm{IC}_i$, and in particular commutes with $\{Z_1, ..., Z_n\}$, meaning that $P_i$ has only $Z$ and $I$ values on its first $n$ qubits. This means the measurement of $P_i$ on the first $n$ qubits produces an outcome of 0 and has no effect on the state of these qubits. We therefore have that the measurement outcome of the final $t$ qubits of $P_i$ is equal to the measurement outcome of all $(n+t)$ qubits, i.e. $m_i = \widetilde{m_i}$, showing (i).

Also, for the same reason, the post-measurement state resulting from measuring the final $t$ qubits of $P_i$ and leaving the first $n$ qubits unchanged is the same as the post-measurement state resulting from measuring all $(n+t)$ qubits, i.e. $|0^n\rangle \otimes |\psi_{i+1}\rangle = |\widetilde{\phi_i}\rangle$. Making no change to $C'_i$ aside from fixing adaptivity so that $C'_{i+1} = \widetilde{C'_i}$, as is done in the IC case, gives (ii). ∎

This concludes our proof of theorem 4.2.1.

### 4.2.3 Proofs of Efficiency

In this section, we prove that each of Alice's local computations in algorithm 1 can be done efficiently in the size of the circuit $C$.

Steps 1, 2, and 4 are clearly efficient. Moreover, step 3(c) can be done efficiently, as can be seen for instance by inspecting table 4.2, and step 3(d) requires only constant time. We therefore focus on the efficiency of steps 3(a) and 3(b).

Step 3(a) requires Alice to propagate the $i$-th Pauli measurement through part or all of $C'$ (at the $i$-th iteration). Step 3(d) of iterations $j < i$ ensures that all adaptivity in $C'$ earlier than the $i$-th measurement has been fixed. So, the Pauli propagates through a non-adaptive circuit with gates $\{\mathsf{H}, \mathsf{S}, \mathsf{CNOT}\}$, which is efficient by the rules given in section 2.6, and finally through some set of unitaries $V_j$ (with $j < i$). It remains to show that the propagation through these $V$-unitaries is also efficient. In particular, with notation simplified somewhat compared to above, we

show this by giving an explicit, efficiently executable rule for propagating Pauli $R$ through a unitary of the form $V = (-1)^p P + (-1)^q Q/\sqrt{2}$, where $P$ and $Q$ are anti-commuting Paulis with measurement outcomes $p, q \in \{0, 1\}$, respectively, and $R$, $P$, and $Q$ are all on the same number of qubits. The efficiency of this rule, along with the observations above, implies the efficiency of step 3(a).

**Proposition 4.2.6.** The propagation of an arbitrary Pauli $R$ through an arbitrary unitary $V = (-1)^p P + (-1)^q Q/\sqrt{2}$ is efficiently classically computable.

**Proof.** Our proof closely follows that of [Per24], Appendix A.

First, we show that $V$ is indeed unitary. Since $V$ is a sum of Pauli operators $P$ and $Q$, which are Hermitian, we get that $V = V^\dagger$. So, we need to show that $V^2 = I$. Computing the product explicitly gives

$$V^2 = \frac{1}{2}\big(P^2 + Q^2 + (-1)^{p+q} PQ + (-1)^{p+q} QP\big) = \frac{1}{2}\big(P^2 + Q^2\big) = I$$

where the second equality uses that $P$ and $Q$ anti-commute, and the third uses the fact that Paulis square to identity.

Now we consider what happens when an arbitrary Pauli $R$ is to be propagated past $V$. The Pauli we seek to compute efficiently, given $R$ and $V$, is $VRV^\dagger$, which is equal to $VRV$ by the above. There are four cases, depending on whether $R$ commutes with both $P$ and $Q$, one of them, or neither:

1. $[P, R] = [Q, R] = 0$: Then $R$ commutes with $V$, so $VRV = VVR = IR = R$.

2. $[P, R] = \{Q, R\} = 0$: Then we have

$$\begin{aligned} VRV &= \frac{1}{2}\big((-1)^p P + (-1)^q Q\big) R \big((-1)^p P + (-1)^q Q\big) \\ &= \frac{1}{2}\big((-1)^p P + (-1)^q Q\big)\big((-1)^p P - (-1)^q Q\big) R \\ &= \frac{1}{2}\big(P^2 - Q^2 - (-1)^{p+q} PQ + (-1)^{p+q} QP\big) R \\ &= \frac{1}{2}\big(-2(-1)^{p+q} PQ\big) R \\ &= (-1)^{\overline{p+q}} PQR. \end{aligned}$$

3. $\{P, R\} = [Q, R] = 0$: Since $P$ and $Q$ are arbitrary anti-commuting Paulis, this case is the same as the previous by interchanging $P$ and $Q$.

4. $\{P, R\} = \{Q, R\} = 0$: Then $R$ anti-commutes with $V$, so $VRV = -VVR = -IR = -R$.

It takes time linear in the length of the Paulis to determine the case and again at most linear time to compute the propagated Pauli. Therefore this propagation is efficient. ∎

For step 3(b), checking commuting relations of pairs of Paulis is clearly efficient, and we observe also that checking the dependence or independence of a Pauli $P_i$ on any subset of IC can also be computed efficiently. This is because, thanks to the vector representation of Paulis in $\mathbb{F}_2^{2n}$ mentioned in section 2.5, determining the dependence relations of a set of Paulis is equivalent to determining linear dependencies in a set of vectors over $\mathbb{F}_2$, which can be done efficiently, for example with Gaussian elimination over $\mathbb{F}_2$.

### 4.2.4 Remarks on the Algorithm

In this section, we collect a few miscellaneous remarks on the PBC algorithm.

#### 4.2.4.1 PBC and Simulation

The correctness of PBC (theorem 4.2.1) for an arbitrary circuit $C$ and the universality of the quantum circuit model (see section 2.4) together imply the following theorem.

**Theorem 4.2.7.** The PBC model is *sampling universal*, that is, it weakly simulates universal quantum computation.

We observe that there are at most $t$ quantum measurements performed in a PBC run, since at most $n+t$ Paulis on $n+t$ qubits can be independent and commuting, and IC is initialized with $n$ elements, leaving room for at most $t$ more. This observation, along with the efficiency proofs of section 4.2.3 and the sampling-universality, give the following theorem.

**Theorem 4.2.8** (Extended Gottesman-Knill Theorem)**.** Any adaptive Clifford circuit $\mathcal{A}$ on $n+t$ qubits with input state $|\phi\rangle\otimes|\psi\rangle$, where $|\phi\rangle$ is an arbitrary $n$-qubit stabilizer state and $|\psi\rangle$ is any arbitrary $t$-qubit quantum state, can be weakly simulated by a $t$-qubit adaptive Clifford circuit $\mathcal{A}^*$ with input $|\psi\rangle$, assisted by classical processing efficient in the size of $\mathcal{A}$, where $\mathcal{A}^*$ has at most $t$ measurements.

This theorem was first stated under the name Extended Gottesman-Knill Theorem in [YJS19], but the proof is exactly the derivation of PBC from an arbitrary circuit given in [BSS16] and presented above. In the theorem, $\mathcal{A}$ corresponds to what we have been calling $C'$, and $\mathcal{A}^*$ is the associated PBC, executed as a circuit. This is an extension of the Gottesman-Knill Theorem in the sense that when $t = 0$ we recover the original Gottesman-Knill Theorem as a special case.

#### 4.2.4.2 PBC Inputs

So far we have shown how to simulate the computation $C|0^n\rangle$ using PBC for any circuit $C$. Using PBC to simulate a circuit $C$ executed on a different input is an important topic for our purposes in the thesis. As suggested in the Extended Gottesman-Knill Theorem, PBC can simulate a computation $C|\phi\rangle$ for any circuit $C$ and any stabilizer state $|\phi\rangle$. There are at least two convenient ways to extend algorithm 1 to sample from $C|\phi\rangle$ with PBC. First, if IC is initialized to the stabilizer of $|\phi\rangle$, rather than to the stabilizer of $|0^n\rangle$, $\{Z_1, ..., Z_n\}$, the rest of the procedure can proceed exactly as usual. Second, the circuit preparing $|\phi\rangle$ from $|0^n\rangle$, $C_{\text{prep}}$, can be appended to the beginning of $C$ and the procedure can be run on the compound circuit $CC_{\text{prep}}$. Since $|\phi\rangle$ is a stabilizer state, $C_{\text{prep}}$ is Clifford, and this introduces only a polynomial-time classical overhead.

We are not aware of a general method to accommodate arbitrary quantum inputs in PBC, i.e., to simulate $C|\psi\rangle$ for arbitrary $|\psi\rangle$ via PBC. If a circuit preparation of $|\psi\rangle$ is known, it may be appended to $C$ as in the stabilizer case, at the cost of additional $\mathsf{T}$ gates in the case of a non-stabilizer $|\psi\rangle$.

#### 4.2.4.3 Post-selection

The *post-selection method* is an important technique for simplifying the implementation and analysis of PBC. In algorithm 1, Alice flips a coin to determine the measurement outcome corresponding to an anti-commuting Pauli $P_i$. However, in the case of anti-commuting gadget measurements, all that we require for correctness is that the state is evolved equivalently to the measurement of $P_i$. For any anti-commuting gadget measurement $P_i$, therefore, we may *post-select* the outcome $m_i$ to be 0 (that is, deterministically decide that $m_i = 0$, rather than flipping a coin) so long as we append the correct unitary as required by the AC case, i.e. $V_i^{(0)}$. From the perspective of implementation, post-selection gives several benefits, including reduced classical randomness requirements and faster compilation. From the perspective of analysis of PBC, using post-selection means that anti-commuting gadget measurements are deterministic; this will be important when we consider the PBC tree in section 4.3. We emphasize that post-selection can only be used with anti-commuting *gadget* measurements; anti-commuting readout measurements must be determined randomly, since the statistics from these measurements form part of the output sample.

#### 4.2.4.4 Sign Suppression

*Sign suppression* is another technique for simplifying the PBC algorithm, which works as follows. Sometimes, the Pauli $P_i$ under consideration at iteration $i$ of step 3 will have a phase of $-1$. (These cases arise from the $V$-unitary propagation rules given in proposition 4.2.6.) In these cases, Alice may request the $+1$ phase version of $P_i$ instead, and invert Bob's measurement outcome. Formally, if $P_i \in \mathcal{P}_n$ is such that $P_i = -W$ for $W \in \mathcal{P}_n^*$ (the phaseless Pauli group), and Alice requests that Bob measure $W$ instead of $P_i$, she may do so, taking $m_i = \overline{m_W}$ for Bob's reported measurement outcome $m_W$. This is equivalent because the eigenspaces of $P_i$ and $W$ are identical but with the $+1$ and $-1$ associated eignevalues exchanged. Assuming that sign suppression is used can be useful for analyzing PBC.

## 4.3 Three Ways of Looking at a PBC

So far, we have presented PBC as an interactive algorithm between classical Alice and quantum Bob. In this section, we consider two more ways of looking at the PBC model which complement the interactive description.[5]

### 4.3.1 PBC as Interaction

The interactive presentation of PBC, already presented in algorithm 1, is useful for a few reasons. First, it emphasizes the main theme and motivating idea of PBC, i.e., the offloading of resources from quantum to classical processing. As much as possible, in the interactive algorithm, computation is handled on Alice's end, with queries to Bob being made only when necessary. Second, the interactive presentation allows for a detailed, step-by-step account of the algorithm which lends itself well to proofs of correctness, as in section 4.2.3. A third useful aspect of the interactive perspective is that, in later chapters, we will consider PBC cryptographically, and then it will be convenient to have already introduced PBC as an interaction between "Alice" and "Bob" parties. It is interesting to note, however, that the interactive perspective, and even the Alice/Bob naming, was already used in [Per24], prior to any cryptographic consideration of PBC. This demonstrates that interaction is a conceptually natural way to look at PBC and motivates our cryptographic angle in this thesis.

All this said, however, PBC is not intrinsically interactive, except in the fact that two distinct processors (one classical and one quantum) are involved and thus some level of "interaction" between them is required.

### 4.3.2 PBC as Measurement Sequence

It often makes sense to set aside the classical side of the interactive algorithm and identify a PBC with the sequence of quantum measurements performed by Bob, thus ignoring the exact process by which he (or Alice) determines which measurement to

[5]This comparison of three perspectives on PBC is novel, although each perspective is present in earlier work. The idea of an interactive description comes from [Per24]. The PBC tree is explicit in [BSS16], and the measurement sequence description is present in some form in both of those works.

make next. For instance, we may want to think of the two "sides" of the interactive procedure as occurring in the same machine, as we would with other models of quantum computation that involve elements of classical processing. Also, while the interactive perspective is useful for examining the procedure in detail, it is sometimes too intricately detailed to be useful for understanding the structure of the model.

We use the following notation. Recall that $P_i$ denotes the Pauli propagated to the beginning of the circuit at iteration $i$ of step 3(a) of the PBC algorithm, and that there are $t + w$ such Paulis. When a Pauli $P_i$ falls into the IC case, and is measured quantumly and added to the IC list, we relabel it as $Q_i$ if it comes originally from a gadget measurement (equivalently, if $i \leq t$) and as $\hat{Q}_i$ if it comes originally from a readout measurement (equivalently, if $t + 1 \leq i \leq t + w$). Note that we retain the original indexing from the $P_i$ Pauli. There are at most $t$ such Paulis. So, we generate a sequence of measurements

$$Q_{j_1} \prec Q_{j_2} \prec \cdots \prec Q_{j_\ell} \prec \hat{Q}_{k_1} \prec \hat{Q}_{k_2} \prec \cdots \prec \hat{Q}_{k_{\ell'}}$$

and we label their indices as $j_1, ...j_\ell \in [t]$ for those originating in gadget measurements and $k_1, ..., k_{\ell'} \in \{t+1, ..., t+w\}$ for those originating in readout measurements. Indices are assigned in order, so $j_a < j_b$ if and only if $a < b$ and likewise for the $k$-indices. So, there are $\ell$ gadget Paulis which are measured quantumly and $\ell'$ such readout Paulis, and we have that $\ell + \ell' \leq t$.

This is the *measurement sequence* perspective of a PBC. We identify the PBC with this sequence of Pauli measurements performed on an initial $|T\rangle^{\otimes t}$ state. The measurement sequence perspective emphasizes that PBC is a model belonging to the broad measurement-based quantum computation (MBQC) family. Although we often think of a PBC instance as being derived from and thus anchored to an initial circuit $C$, it is ultimately a model of computation consisting of a sequence of adaptively chosen measurements on a generic initial state, which are hallmarks of MBQC models. This perspective gives us greater flexibility to analyze PBC, for example by considering arbitrary sequences of Pauli measurements on $|T\rangle^{\otimes t}$ which may or may not be derivable from some circuit $C$.

### 4.3.3 PBC as Binary Tree

The *PBC tree* is a binary tree which collects all possible measurement sequences which may occur when a PBC is executed, for a given set of initial conditions ([BSS16]). For our purposes now, these initial conditions are just the circuit $C$, which we assume to be executed on $|0^n\rangle$. Each possible measurement sequence is a branch in the tree. Since which measurement is performed next may depend on which of two measurement outcomes (0 or 1) was just obtained, a PBC is associated to a binary tree structure of possible measurement sequences. As a convention, while it is often the case that the two "child" nodes stemming from a Pauli measurement are the same (i.e., identical as Paulis), we still represent them separately in the tree. Also, it is possible that a certain quantum measurement is deterministic, and one outcome is never obtained, but we still represent the never-obtained child in the tree. The tree turns out to be a very useful tool for analyzing and understanding PBC.

A generic PBC tree is given in fig. 4.1. We use the same notation for quantumly-measured Paulis as in the previous section. We add a superscript $(s)$ for $s \in \{0,1\}^*$ to each Pauli, where $s$ denotes the sequence of previous *quantum* measurement outcomes already determined, read from left to right. So, for example, a Pauli $Q_{j_d}$ for $d \in \mathbb{N}$ will have a superscript of $(s)$ for $s \in \{0,1\}^{d-1}$; moreover, for $1 \leq d \leq \ell$, the $d$-th level of the tree will have $2^{d-1}$ entries $Q_{j_d}$ each marked with a different superscript $s$.

Some important remarks are needed to justify the presentation in fig. 4.1. First, we assume that post-selection is used; this ensures that anti-commuting gadget measurements are deterministic and thus branching is caused only by the quantum measurements. Second, we assume that sign suppression is used, ensuring that branching is caused only by gadget measurements, as depicted in fig. 4.1.[6] With sign suppression, readout measurements do not cause branching because, once all gadget measurements are evaluated, the circuit adaptivity is fixed. This means that the only circuit updates depending on readout measurements are in the appended $V$-unitaries, which affect future Pauli measurements only up to a $\pm$ phase, since only an internal $\pm$ sign in the $V$ construction depends on the measurement outcome. Third, we assume that certain measurement indices in $C'$ *always* fall into the IC case, yielding quantum measurements, and the others *never* do. This is reflected in fig. 4.1 in that we index all Paulis at a given level of the tree with an identical index (e.g., the single value $j_2$ is attached

[6] Post-selection and sign suppression are discussed in section 4.2.4.

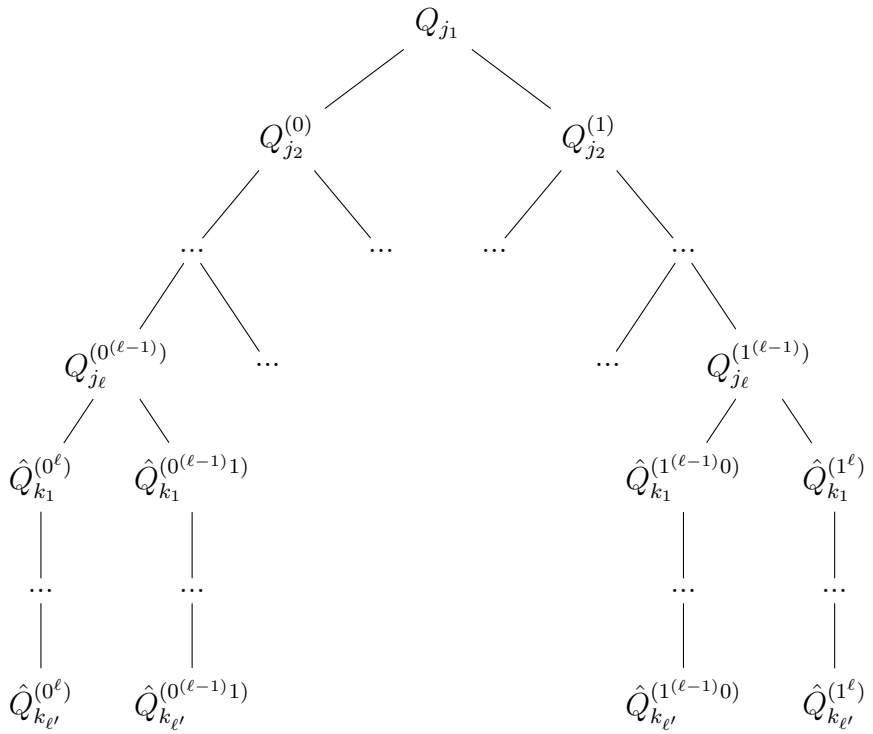



Figure 4.1: A generic PBC tree. Each $Q$ is a Pauli operator, with the superscript representing the sequence of previous measurement outcomes obtained to result in that Pauli being measured. The hat (e.g., $\hat{Q}$) indicates a readout measurement.

to both quantum measurements of the second level). In fact, this is always the case in PBC instances derived from circuits, so the assumption is safe to make, but we do not prove this here.

## 4.4 Advantages of PBC

PBC has several significant advantages that make it a promising practical solution for running quantum computation.

First, PBC offers several ways of reducing quantum computational resource requirements (or, equivalently, running larger computations with a fixed amount of resources) compared to other models. Most straightforwardly, for a circuit $C$ on $n$ qubits with $t$ $\mathsf{T}$ gates and $w$ readout measurements, PBC requires a quantum register of size $t$ rather than $n$. In regimes where $t < n$, this provides an immediate

advantage. Moreover, since T gates cannot be implemented fault tolerantly, many fault tolerant implementations of quantum computation begin with a magic state injection compilation step which immediately increases the size of the register needed to run $C$ from $n$ to $n+t$; PBC still requires just $t$ qubits for this scenario, since Pauli measurements can be implemented fault tolerantly. In fault tolerant settings, then, PBC provides a universal (i.e., not regime-specific) advantage by reducing the register size from $n+t$ to $t$. As qubit count is limited and fault tolerance widely desirable in practical quantum computing settings, these advantages are significant. Additionally, only $t$ measurements are required, rather than $w$ in a circuit-based implementation, which is advantageous when $t < w$.

The other significant advantage that PBC offers is that it enables a technique called *qubit virtualization*, sometimes also called *hybrid computation*[7]. We describe qubit virtualization next.

### 4.4.1 Qubit Virtualization

Qubit virtualization addresses the following scenario. Suppose we have a quantum register of size $t$. We have already shown that we can use standard PBC to run a circuit on $n$ qubits with $t$ T gates on this register, regardless of $n$. Now, however, what happens if the circuit we wish to run, even after PBC compilation, requires $t+k$ qubits for some integer $k > 0$ — can we execute the computation with the help of our $t$-qubit register at a cost that depends primarily on $k$, and not on $t+k$?

We emphasize that it is not clear how to do this, *a priori*, and the cost consequences are significant. Imagine, to borrow an example from [BSS16], that you needed to run a computation requiring 101 qubits, and had at your disposal a classical computer and a quantum computer with a 100-qubit register. It is not clear that you would be any better off with the the 100-qubit register than you would have been with no quantum resource at all! The one "extra" qubit cannot obviously be simulated classically, separately from the others, and then combined back into the computation,

[7] We avoid the term "hybrid computation" in this thesis, although it is standard in works like [PG23], because of the potential for confusion with *regular* PBC as presented in algorithm 1. PBC already has a central "hybrid" component, intuitively, in that computational work is accomplished by a combined effort of the classical and quantum processors. "Qubit virtualization" better emphasizes that this is a distinct technique which extends regular PBC.

so one would expect a simulation cost exponential in 101, i.e., exactly the cost of simulating the *entire* computation classically. This seems like an unfortunate waste of an almost-large-enough quantum resource!

Qubit virtualization gives a way, using PBC, to estimate the output of the $(t+k)$-qubit computation with error less than any desired constant, using a $t$-qubit register, at a cost that scales exponentially in $k$ only. (There is also a polynomial overhead that scales with both $t$ and $k$, but the key point is that the exponential factor depends on $k$ alone.) So, in the previous example, we can achieve with qubit virtualization a runtime scaling that is polynomial in 101 but exponential only in 1 (corresponding to the one extra qubit), i.e., a runtime proportional to $\text{poly}(t,k) \cdot 2^{\mathcal{O}(1)}$, a remarkable improvement. Note that we cannot reasonably expect to do *better* than exponential in $k$, or else we could set $t = 0$ to achieve a subexponential simulation of any quantum computation.

Informally, qubit virtualization works by computing on the $k$ extra qubits "virtually". First, we decompose the $k$ extra magic state qubits into a linear combination of exponentially many (in $k$) stabilizer states. Then, we run the PBC on each term of the decomposition, tensored with the remaining $t$ magic states; since each term of the decomposition is a stabilizer state, we may "absorb" that part of the state into the data register of the PBC and, for the same reason as in regular PBC, we never need to handle those $k$ qubits quantumly. Finally, since the sum of these PBC runs is an unbiased estimator for the PBC on $t + k$ qubits, we generate an estimate of the desired sample with exponential-in-$k$ many runs, multiplied by some polynomial factor in $t$ and $k$ required to collate the outcomes and reach a desired level of error.

Next, we present this procedure formally. We follow [BSS16] and [Per24] fairly closely but show a few more steps. Let $\mathcal{Q}(|T\rangle^{\otimes(k+t)})$ be the desired PBC on $k + t$ qubits, initialized to the $|T\rangle$ state as usual in PBC. For ease of presentation, we will restrict our consideration to computations onto a single bit, i.e., we assume that $\mathcal{Q}$ yields a single output bit $b \in \{0, 1\}$. (The argument generalizes straightforwardly to computations with $w$ readout measurements for arbitrary $w$, as we have been considering in the previous sections.) $b$ is a random variable; let $p_{\mathcal{Q}}$ be the probability that $b = 1$.

First, we observe that the magic state $|T\rangle$ decomposes as

$$|T\rangle = \alpha_1 |+\rangle + \alpha_2 |-\rangle + \alpha_3 |i\rangle \tag{4.4.1}$$

where

$$\alpha_1 = \frac{1}{2}, \alpha_2 = \frac{1-\sqrt{2}}{2}, \alpha_3 = \frac{1}{\sqrt{2}}, \tag{4.4.2}$$

so $|T\rangle^{\otimes k}$ can be written as the $k$-fold tensor product of this linear combination, which has $3^k$ terms, each of them a stabilizer state multiplied by a scalar coefficient. (There exist lower-term-count decompositions of $|T\rangle^{\otimes k}$ for higher values of $k$, so there is room for optimization here, but showing just this one suffices for our purposes.) So, that is, we can write

$$|T\rangle^{\otimes k} = \sum_{i=1}^{\chi} \beta_i |\phi_i\rangle$$

where $\beta_i$ are real numbers, $|\phi_i\rangle$ are $k$-qubit stabilizer states, and $\chi = 3^k$. Then

$$|T\rangle^{\otimes (k+t)} = \Big( \sum_{i=1}^{\chi} \beta_i |\phi_i\rangle \Big) \otimes |T\rangle^{\otimes t} = \sum_{i=1}^{\chi} \beta_i \big( |\phi_i\rangle \otimes |T\rangle^{\otimes t} \big)$$

and by the linearity of quantum computation we get

$$\mathcal{Q}(|T\rangle^{\otimes (k+t)}) = \sum_{i=1}^{\chi} \beta_i \mathcal{Q}\big( |\phi_i\rangle \otimes |T\rangle^{\otimes t} \big). \tag{4.4.3}$$

Then, by the PBC algorithm, for any computation on $k+t$ qubits where the input on the first $k$ qubits is a stabilizer state, we can find a PBC on $t$ qubits which has an identical output distribution. So, for each $1 \leq i \leq \chi$, we define $\mathcal{Q}_i(|T\rangle^{\otimes t})$ to be such a PBC on a $t$-qubit quantum register which has an identical distribution to $\mathcal{Q}(|\phi_i\rangle \otimes |T\rangle^{\otimes t})$. So it follows from eq. (4.4.3) that the computation

$$\sum_{i=1}^{\chi} \beta_i \mathcal{Q}_i\big( |T\rangle^{\otimes t} \big)$$

has an output identically distributed to that of $\mathcal{Q}(|T\rangle^{\otimes(k+t)})$. If we define random variables $b_i$ as the output bit of $\mathcal{Q}_i$ and let $p_{\mathcal{Q}_i}$ be the probability that $b_i = 1$, that is, $p_{\mathcal{Q}_i} = \mathbb{E}(b_i)$, then this implies that

$$p_{\mathcal{Q}} = \sum_{i=1}^{\chi} \beta_i p_{\mathcal{Q}_i}.$$

Next, define a random variable $\xi$ as

$$\xi = \sum_{i=1}^{\chi} \beta_i b_i$$

and observe that

$$\mathbb{E}(\xi) = \sum_{i=1}^{\chi} \beta_i \mathbb{E}(b_i) = \sum_{i=1}^{\chi} \beta_i p_{\mathcal{Q}_i} = p_{\mathcal{Q}}$$

which shows that $\xi$ is an unbiased estimator for $p_{\mathcal{Q}}$, so we may produce an estimate of the outcome of $\mathcal{Q}$ by sampling $3^k$ PBCs and computing a weighted sum of the results. The time required to run this procedure once then scales as $\text{poly}(t,k) \cdot 2^{\mathcal{O}(k)}$.

We incur some error in making this estimate, and may wonder how many repetitions are required to reach a constant error level $\epsilon$ with constant confidence parameter $c$. The number of samples $N$ that we need is given by Chebyshev's inequality as

$$N \geq \frac{\sigma^2}{\epsilon^2 c}$$

where $\sigma^2$ is the variance of $\xi$. To be more precise about the $\sigma^2$ term, we can further observe that

$$\sigma^2 = \text{Var}(\xi) = \text{Var}\Big(\sum_{i=1}^{\chi} \beta_i b_i\Big) = \sum_{i=1}^{\chi} \beta_i^2 \text{Var}(b_i) < \sum_{i=1}^{\chi} \beta_i^2$$

where the third equality holds because the $b_i$ are independent and the fourth because each $b_i$ has variance less than one. $\sigma^2 < 1$, so for constant error and confidence a number of repetitions which scales as a constant suffices. The qubit virtualization

technique with constant error and confidence therefore has an overall runtime of $\text{poly}(t,k) \cdot 2^{\mathcal{O}(k)}$, as advertised. We note that better sampling methods and error bounds than the ones presented here are given in [Per24].

## 4.5 Example

In this section, we give an example of PBC on a toy circuit $C$, shown in fig. 4.2.[8] $C$ has parameters $n = t = w = 2$.

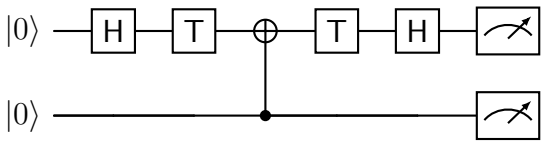



Figure 4.2: The circuit $C$ to be executed in the example run of the PBC algorithm.

### 4.5.1 PBC Algorithm

We begin by showing the PBC algorithm executed on $C$. As a reminder, since we use it heavily, the notation $P_k$ for $P \in \{X, Y, Z\}$ and $k \in \mathbb{N}$ means the multi-qubit Pauli (with the exact number of qubits depending on context) with $P$ on the $k$-th qubit and $I$ on all other qubits; and similarly, for example, $Z_1Y_4$, as an element of $\mathcal{P}_4$, is interpreted as $ZIIY$.

1. Step 1: Alice performs magic state injection on her classical description of $C$ to produce a description of $C'$. The circuit $C'$ after this step is shown in fig. 4.3.
2. Step 2: Alice initializes her IC list as: $\text{IC} = \{Z_1, Z_2\}$.
3. Step 3: We go through each iteration:
    3.1. Iteration 1: Alice propagates the first measurement, $Z_3$, to the beginning of the circuit as follows:

[8] This example was created newly for this thesis. [Per24] also works through a (different) toy example.

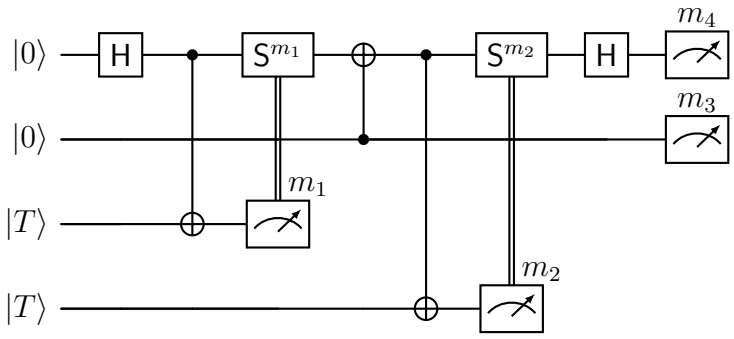



Figure 4.3: The circuit $C'$, the magic-injected form of $C$, produced in step 1 of the PBC example.

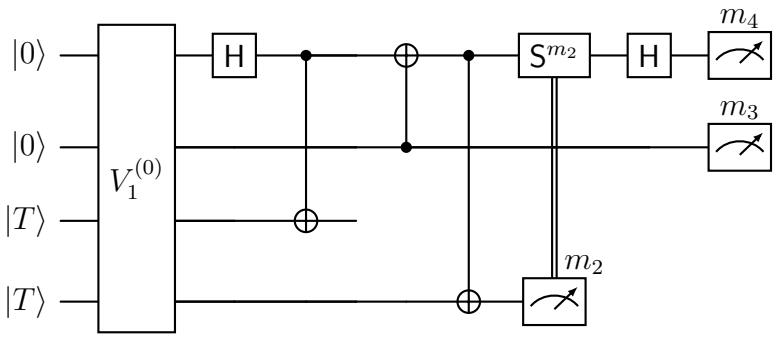



Figure 4.4: $C'$ after iteration 1 of step 3 of the PBC example. Note that $V_1^{(0)} = (Z_1 + X_1Z_3)/\sqrt{2}$.

$$Z_3 \xrightarrow{CNOT_{1,3}} Z_1Z_3 \xrightarrow{H_1} X_1Z_3 = P_1.$$

Since $X_1Z_3$ anticommutes with $Z_1 \in \text{IC}$, we are in the AC case. So, Alice post-selects $m_1 = 0$ and appends $V_1^{(0)}$ to the beginning of $C'$.[9] The resulting circuit is shown in fig. 4.4.

3.2. Iteration 2: The second measurement, $Z_4$ propagates back as follows:

$$Z_4 \xrightarrow{CNOT_{1,4}} Z_1Z_4 \xrightarrow{CNOT_{2,1}} ZZIZ \xrightarrow{H_1} XZIZ \xrightarrow{V_1^{(0)}} -ZZZZ = P_2$$

where the final propagation can be done efficiently by checking the com-

[9]Here we use the post-selection method, discussed in section 4.2.4, both to demonstrate it and to simplify presentation.

mutation relations between $Z_1$, $X_1Z_3$ (the $V_1$ constituents) and $XZIZ$: since $\{Z_1, XZIZ\} = [X_1Z_3, XZIZ] = 0$, the propagated Pauli is

$$-(Z_1)(X_1Z_3)(XZIZ) = -ZZZZ.$$

Then, $-ZZZZ$ commutes with all IC elements (i.e., with $Z_1$ and $Z_2$), and does not depend on them (clear from inspection, in as small an example as this). So, we are in the IC case, and Alice requests that Bob measure $-ZZ$ (the final two qubits of $P_i$) on the magic register. Bob replies with the measurement outcome, which is $m_2$. Alice adds $-ZZZZ$ to IC and fixes the next adaptive $S$ gate of $C'$ according to $m_2$.

Now the following steps depend on the outcome $m_2$. We will cover both cases in our analysis (but point out that an execution of PBC would follow one or the other case only). We work depth-first, completing all iterations under the first case $m_2 = 0$ before returning to $m_2 = 1$.

3.3. Iteration 3 (given $m_2 = 0$): Note that the third and fourth measurements are ordered arbitrarily. We take the one corresponding to $m_3$ and propagate it back as

$$Z_2 \xrightarrow{V_1^{(0)}} Z_2 = P_3$$

since $Z_2$ commutes with both constituent Paulis of $V_1^{(0)}$. This Pauli commutes with all IC elements and is multiplicatively dependent on a subset $S \subseteq$ IC of them (in particular, $Z_2$ is already an element of IC, so $S = \{Z_2\}$), so we are in the DC case. Alice therefore takes $m_3$ to be the XOR of measurement outcomes of the Paulis in $S$; $Z_2$ is an initial IC element and therefore has associated outcome 0, so $m_3 = 0$. Alice updates neither $C'$ nor IC. No adaptivity depends on $m_3$, so no final $C'$ update is needed either.

3.4. Iteration 4 (given $m_2 = 0$): The fourth measurement, $Z_1$, propagates as

$$Z_1 \xrightarrow{H_1} X_1 \xrightarrow{CNOT_{1,4}} X_1X_4 \xrightarrow{CNOT_{1,3}} XIXX \xrightarrow{H_1} ZIXX \xrightarrow{V_1^{(0)}} ZIXX = P_4$$

where the final propagation involves checking that

$$[Z_1, ZIXX] = [X_1Z_3, ZIXX] = 0.$$

Then $ZIXX$ commutes with each element of IC $= \{Z_1, Z_2, -ZZZZ\}$ and is independent of them. So this is an IC case: Bob measures $XX$ and sends his outcome $m_4$ to Alice, who adds $ZIXX$ to IC.

Now we consider iterations 3 and 4 under the other case, $m_2 = 1$.

3.3′ Iteration 3 (given $m_2 = 1$): The third measurement propagates as

$$Z_2 \xrightarrow{V_1^{(0)}} Z_2 = P_3$$

exactly as before. Again, this is DC, and $m_3 = 0$. Alice updates neither $C'$ nor IC and does not fix any adaptivity.

3.4′ Iteration 4 (given $m_2 = 1$): The fourth measurement, $Z_1$, propagates as

$$Z_1 \xrightarrow{H_1} X_1 \xrightarrow{S_1} Y_1 \xrightarrow{CNOT_{1,4}} Y_1X_4 \xrightarrow{CNOT_{2,1}} YZIX \xrightarrow{CNOT_{1,3}}$$
$$YZXX \xrightarrow{V_1^{(0)}} -IZYX = P_4$$

where the final propagation involves checking that

$$\{Z_1, YZXX\} = [X_1Z_3, YZXX] = 0.$$

Then $-IZYX$ commutes with each element of IC $= \{Z_1, Z_2, -ZZZZ\}$ and is independent of them. So this is again an IC case: Bob measures $-YX$ and sends his outcome $m_4$ to Alice, who adds $-IZYX$ to IC.

4. Alice's values for $m_3$ and $m_4$, found in iterations 3 and 4 are exactly her sample from the readout measurements.

### 4.5.2 Measurement Sequence and Tree

Finally, we consider this PBC from the other perspectives described in section 4.3. First, to represent the PBC as a measurement sequence, we consider just the quantum

measurements, i.e., $P_2$ and $P_4$. Following the measurement sequence notation, we label them as $P_2 = Q_2$ (so $j_1 = 2$) and $P_4 = \hat{Q}_4$ (so $k_1 = 4$). Then, if $m_2 = 0$, we see that $Q_2 = ZZ$ and $\hat{Q}_4 = XX$, so we get the sequence

$$ZZ \prec XX$$

and similarly if $m_2 = 1$ we get the sequence

$$ZZ \prec YX.$$

Finally, from these, we say that the PBC tree associated to $C$ is the one shown in fig. 4.5.

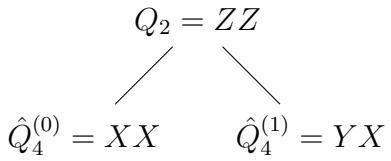



Figure 4.5: The PBC tree of the circuit $C$ used in the PBC example.

# Chapter 5

# Protocol: Input-Concealed PBC

In this chapter, we describe a protocol for *input-concealed PBC*. The goal is that a resource-limited Alice, with help from a quantum server Bob, receives a sample from the output distribution of a quantum circuit $C$ that is run on a computational basis state input $x$ of her choosing, while maintaining the secrecy of $x$ from Bob. We begin by providing some intuition behind the algorithm (section 5.1). We then present the algorithm (section 5.2), prove correctness through a series of circuit gadgets (section 5.3), and show how the algorithm can be extended to support the qubit virtualization technique with a secret input (section 5.4). We end by working through a small example (section 5.5) using the same circuit that we used to demonstrate the PBC algorithm in the previous chapter. The proof of security of the protocol is deferred to chapter 7, after we discuss some of its motivations and advantages in chapter 6.

This chapter is based on joint work with Anne Broadbent, Selman Ipek, Joshua Nevin, and Filipa Peres.

## 5.1 Intuition

In this section, we give some intuition behind the input-concealed PBC protocol, drawing especially on the PBC tree as an analytical tool. We start by considering the task of selecting a classical input $x \in \{0,1\}^n$ to the computation $C|x\rangle$.

First, we observe that the PBC tree associated to $C|0^n\rangle$ has the same vertex

labels (i.e., the same Pauli operators) as the PBC tree for $C|x\rangle$ for every classical input $x \in \{0,1\}^n$, in the same structure *except* possibly with pairs of sibling nodes (i.e., two child nodes of the same parent) exchanging places. (In the notation of section 4.3, this can be rephrased as, the trees are identical up to $(s)$ superscripts.) One way to see this is to consider $C|x\rangle$ as instantiated by the computation $C_x|0^n\rangle$, where $C_x$ is $C$ with a sequence of initial $\mathsf{X}$ gates on the data wires, encoding the input $x$. Comparing the tree of $C_x|0^n\rangle$ to that of $C|0^n\rangle$, the effect of these initial $\mathsf{X}$ gates is at most to exchange the $+/-$ sign of the Pauli $P_i$ found at each iteration of step 3(a) of the PBC algorithm, since all pairs of Paulis either commute or anti-commute. Moreover, as has been noted, the PBC algorithm works just the same if all quantum Pauli measurements are made in their $+$ form (i.e., as $Q$ for $Q \in \mathcal{P}_t^*$, even when $-Q$ is called for by the algorithm), so long as Alice flips the bit given by Bob's measurement result when this is the case. In the tree perspective, taking the bit complement of a measurement corresponds to exchanging the position (or, equivalently, $(s)$ superscript) of the two children of that node. Thus, we can consider that a circuit $C$ has a single, fixed PBC tree for all classical inputs $x$ (say, the PBC tree associated to $C|0^n\rangle$), and the choice of input $x$ to $C$ corresponds in PBC to a *redirection* down the branches of this tree, plus some final bit flip corrections on the readout measurements.

Thinking cryptographically, redirections through the PBC tree are seen by Bob, in that he receives the Pauli measurement requests corresponding to the vertices of the tree and thus can tell which branch the computation is proceeding down, in general. So, if Alice introduces an input $x$ to a PBC simply by redirecting Bob, Bob will learn $x$.

However, there is another technique Alice can use which *induces* a redirection through the PBC tree. If an $\mathsf{S}$ gate is performed on some qubits of the magic register $|T\rangle^{\otimes t}$, prior to the start of the interaction, then these must be corrected during the PBC by a set of corresponding redirections (compared to the branch directions that would be followed if no $\mathsf{S}$ gates had been performed), or the correct outcome will not be obtained.

We therefore use the following idea to achieve security of Alice's input $x$. Prior to the start of the interaction, Alice randomizes the magic state by performing $\mathsf{S}$ gates on randomly chosen qubits. She conceals the presence of these $\mathsf{S}$ gates with a further

set of random Z gates, then passes the register to Bob. In analogy to the classical one-time pad, the pattern of S gates serves as a random secret key. Alice's sequence of measurement requests during the PBC procedure, which describe to Bob a branch in the PBC tree of $C$, are a ciphertext, since they are a combination (in some sense an XOR, although we make this more precise later) of her *intended*, $x$-dependent redirection, which corresponds to her message or secret, and the redirection induced by the secret key. Since the key is random and secret, the direction Bob takes through the tree appears to him to be random, that is, he cannot tell which directions are due to Alice's desired input and which are due to the underlying S gates on his register. After a final classical correction, Alice can thus receive a sample from $C|x\rangle$ while maintaining perfect secrecy of $x$.

## 5.2 Protocol Description

Now we make this intuition precise. In our setting, Alice has a classical description $\delta \in \{0,1\}^*$ of a quantum circuit $C$. $C$ has $n$ input or data registers, $t$ T-gates (and other gates drawn from the set $\{\mathsf{H}, \mathsf{S}, \mathsf{CNOT}\}$), and $w \leq n$ readout measurements in the computational basis. She also has a classical input, $x \in \{0,1\}^n$, and she wishes to sample from the output distribution of $C|x\rangle$. We do not attempt to hide the circuit; we therefore assume that Bob is given $\delta$ ahead of time.

We achieve this task with a modification of the PBC procedure. After initial one-way quantum communication, Alice and Bob interact classically, and Bob makes at most $t$ Pauli measurements on a register of $t$ qubits. Our protocol is called *input-concealed PBC*. The main steps are presented at a high level in algorithm 2.

Next, we explain each of these steps in more detail:

1. Alice prepares $t$ randomly rotated magic states and sends them to Bob:
   (a) Alice draws bits $c_1, ..., c_t, d_1, ..., d_t \in_R \{0,1\}$ uniformly at random and prepares $|R_i\rangle = S^{d_i} Z^{c_i} |T\rangle$, $1 \leq i \leq t$.
   (b) Alice sends $\bigotimes_{i=1}^{t} |R_i\rangle$ to Bob.
2. Alice does classical pre-processing of the circuit and her input:

**Algorithm 2** Input-concealed PBC

Initially, Alice holds circuit $C$ with an $n$-qubit data register and $t$ $\mathsf{T}$ gates and a classical input $x \in \{0,1\}^n$.

1. Alice prepares $t$ randomly rotated magic states and sends them to Bob.
2. Alice does classical pre-processing of $C$, dependent on $x$, to produce $C^*$, an adaptive Clifford circuit.
3. Alice interacts classically with Bob to perform a PBC of $C^* |0^n\rangle$ where Bob's measurements are on the randomized magic states.
4. After the interaction concludes, Alice locally applies a final classical correction to the readout measurements.

(a) Alice performs magic state injection, as in regular PBC, producing $C'$.

(b) Alice represents her input as a sequence of $\mathsf{X}$ gates on the data wires of $C'$ at the beginning of the circuit (i.e., with $\mathsf{X}$ on the $j$-th wire if and only if $x_j = 1$, $1 \leq j \leq n$).

(c) Alice propagates these $\mathsf{X}$ gates later in the circuit, as follows:

- Alice tracks her entry on each wire as a general Pauli of the form $\mathsf{X}^a$, $\mathsf{Z}^b$. Initially, $b = 0$ on every wire; $a = x_j$ on the $j$-th wire.
- Between $\mathsf{T}$ gadgets, the circuit is Clifford, so Alice propagates Pauli gates $\mathsf{X}^a$, $\mathsf{Z}^b$ later yielding Pauli gates $\mathsf{X}^{a'}$, $\mathsf{Z}^{b'}$ by the update rules in section 2.6.
- When she reaches the $i$-th $\mathsf{T}$ gadget (indexed from earliest to latest in the circuit), Alice first propagates the Pauli gates $\mathsf{X}^a$, $\mathsf{Z}^b$ along its upper wire later past the gadget yielding gates $\mathsf{X}^a$, $\mathsf{Z}^{a\oplus b}$, and $\mathsf{S}^a$.
- Then, immediately after propagating past the $i$-th $\mathsf{T}$ gadget, Alice performs a *randomization step*: Alice rewrites the $i$-th $|T\rangle$ state as $S^{d_i} Z^{c_i} |T\rangle$ *and* adds correction gates $\mathsf{S}^{d_i}$, $\mathsf{Z}^{d_i \overline{m_i} \oplus c_i}$ to the upper wire of the $\mathsf{T}$ gadget, immediately after the $\mathsf{S}^{m_i}$ gate (and thus immediately before the propagation terms added just previously).
- Alice combines the gates gathered after the $\mathsf{S}^{m_i}$ gate. Once combined, these will include a single $\mathsf{S}$ gate with exponent $s_i$, where $s_i$ is a Boolean function of Alice's randomness, input, and previous mea-

surement outcomes. The remaining gates will be Pauli gates. The $\mathsf{S}^{s_i}$ gate is left behind by the gadget and the Pauli is propagated further toward the end of the circuit. We refer to this $\mathsf{S}^{s_i}$ gate left behind as a *cryptographic* $\mathsf{S}$ *gate*, particularly in chapters 6 and 7.

(d) When the Paulis have been propagated to just before the readout measurements, propagation halts. The Pauli gates on all $n$ input wires at this point can be represented as $\mathsf{X}^u$, $\mathsf{Z}^v$ for vectors $u, v \in \{0,1\}^n$. These are called the *last-layer Pauli corrections*. Alice records the exponents $u, v$ then removes the last-layer Pauli corrections from the circuit. The circuit formed in this way, after the removal of the final corrections, is called $C^*$.

3. Alice interacts classically with Bob to perform steps 3 and 4 of the PBC algorithm on the processed circuit $C^*$ (i.e., $C^*$ takes the place of $C'$ in step 3 of regular PBC). Bob makes his measurements on the randomized magic states $\bigotimes_{i=1}^{t} |R_i\rangle$ provided to him in step 1, and otherwise his actions are identical to regular PBC. Alice's procedure is identical to the regular PBC algorithm *except* for a small difference in step 3(d). Since the adaptivity of $C^*$ is more elaborate than in regular PBC, Alice must be sure after each iteration of step 3 to fix all occurrences of $m_i$ in the circuit with her newly-obtained outcome value. In $C^*$, these can appear in any later $s_j$ exponent ($s_j$ where $j > i$), rather than just in the $i$-th gadget measurement, as is the case in regular PBC.

4. After the interaction concludes, Alice has received $w$ outcome bits from step 4 of the regular PBC procedure, corresponding to the $w$ readout measurements of $C$. Now, she uses her last-layer Pauli correction vectors, $u$ and $v$, the places of which are the exponents of $\mathsf{X}$ and $\mathsf{Z}$ respectively, to determine a final classical correction to these $w$ bits. For a readout measurement outcome $m_p \in \{0,1\}$ coming from a measurement on data wire $j$:

   (a) If $u_j = 0$, Alice does nothing and considers $w$ to be the measurement outcome.

   (b) If $u_j = 1$, Alice flips the bit and considers $\overline{w}$ instead.

   (c) Once these changes are made, the corrected $w$ bits are Alice's desired sample from the output distribution of $C|x\rangle$.

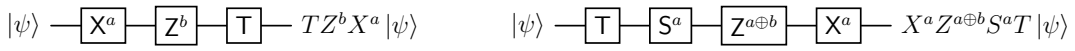
This concludes the detailed description of the input-concealed PBC protocol. An example on a small circuit is given in section 5.5. Next, in section 5.3, we prove the correctness of the protocol.

## 5.3 Correctness

The proof of correctness of algorithm 2 happens in two steps. First, we give a circuit-based argument that $C^* |0^n\rangle$, derived by Alice in step 2, along with the last-layer Pauli corrections, produces the same output as $C |x\rangle$. Second, we appeal to the correctness of the PBC algorithm, shown in chapter 4, to argue that the interaction at step 3 produces a correct sample from the output distribution of $C^* |0^n\rangle$ and therefore of $C |x\rangle$. Thus, in this proof, we do not need to engage directly with the PBC algorithm to show correctness, and can instead treat PBC as a black box whose correctness is established separately. We now state the correctness of the protocol as a theorem.

**Theorem 5.3.1.** The input-concealed PBC protocol of algorithm 2 produces a sample from the output distribution of $C |x\rangle$.

(a)




(b)

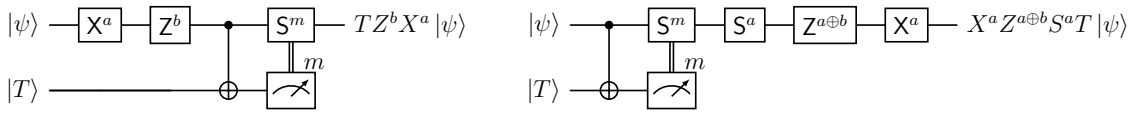



Figure 5.1: The propagation of a Pauli past (a) a T gate and (b) a T gadget. For each of (a) and (b), the left side circuit is equivalent to the right side circuit.

**Proof.** For the first, circuit-based part of the proof, we observe that the circuit formed by Alice in step 2(b), which we call $C_x$, is such that $C_x |0\rangle$ has an output

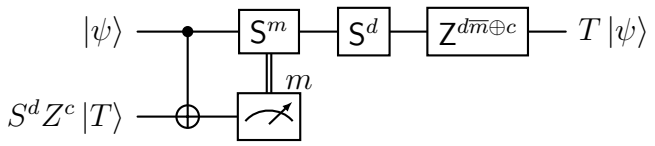



Figure 5.2: Correctness of the randomization step in the input-concealed PBC protocol. The effect of this gadget is exactly equivalent to a regular $\mathsf{T}$ gadget, for all states $|\psi\rangle$.

identical to $C|x\rangle$. This is clear since $X|0\rangle = |1\rangle$, so the $X$ gates Alice adds to form $C_x$ correctly produce her input $x$ from the all-zero input. Next, we argue that the procedure of step 2(c) preserves equivalence of this circuit. The correctness of Clifford propagations was established in section 2.6. The correctness of the propagation step of a Pauli past a $\mathsf{T}$ gadget, which is shown in fig. 5.1, holds since $TZ^bX^a|\psi\rangle = X^aZ^{a\oplus b}S^aT|\psi\rangle$ for all states $|\psi\rangle$. Finally, the randomization step holds because of the circuit identity shown in fig. 5.2. The figure shows that, in a $\mathsf{T}$ gadget, a rotation $S^dZ^c$, for $c, d \in \{0, 1\}$, applied to the $|T\rangle$ state is corrected by $\mathsf{S}^d$ and $\mathsf{Z}^{d\overline{m}\oplus c}$ gates added after the gadget so that the gadget plus corrections still has the effect of producing $T|\psi\rangle$ for any input state $|\psi\rangle$. Since each of the actions in step 2(c) preserves equivalence, we conclude that the circuit $C^*|0^n\rangle$ *along with* the last-layer Pauli corrections correctly produces the output of $C|x\rangle$. Finally, we observe that the last-layer Pauli corrections may be propagated past the readout measurements, inducing a bit flip of the outcome on wire $j$ exactly when the exponent of $X$ on wire $j$ is 1; thus, a sample from $C^*|0^n\rangle$ (i.e., *without* the last-layer Paulis), plus a final correction as in step 4 of algorithm 2, correctly produces a sample from the output distribution of $C|x\rangle$.

Second, we appeal to the correctness of the PBC algorithm to justify the correctness of step 3 of algorithm 2. We make the important note that, after the step of magic state injection, the particular auxiliary state provided in the magic register is not involved in the proof of the correctness of (regular) PBC. That is, a PBC where measurements are performed on a magic register initialized to *any* quantum state $|\psi\rangle$ on $t$ qubits has the same outcome as a circuit on $n+t$ qubits where the final $t$ qubits have $|\psi\rangle$ as input. So we can conclude, from the correctness of the circuit $C^*$ which has a magic register input of *rotated* magic states, that the PBC on this same rotated magic state register produces a correct sample from $C^*|0^n\rangle$, completing the proof. ∎

## 5.4 Qubit Virtualization

Our input-concealed PBC protocol can be extended to perform *input-concealed qubit virtualization*. Suppose Alice wishes to perform a computation $C$, with $n$ input wires and $w \leq n$ readout measurements as before, and now with $t + k$ $\mathsf{T}$ gates, on a secret classical input $x \in \{0,1\}^n$, but Bob has a quantum register of size only $t$ qubits. Then an extension of algorithm 2 allows Alice to receive a sample from $C|x\rangle$, while keeping $x$ hidden from Bob, at a runtime and communication cost that scales exponentially in $k$ only. We explain this procedure next.

Alice performs a decomposition of $|T\rangle^{\otimes k}$ into a linear combination of $k$-qubit stabilizer states $|\phi_i\rangle$ with real coefficients $\beta_i$, for $1 \leq i \leq \chi$, as described in section 4.4. Then, she prepares circuits $C_i$, each of which is derived from $C'$ (the magic state injected form of $C$) by replacing the input state to the first $k$ magic wires, which is $|T\rangle^{\otimes k}$ in $C'$, with the state $|\phi_i\rangle$. Then she considers $C_i$ as a circuit on $n + k$ input wires, initialized to $|0^n\rangle \otimes |\phi_i\rangle$, and $t$ magic wires (that is, she re-designates wires indexed $n + 1, ...n + k$ as input rather than magic wires). Then she runs algorithm 2 $\chi$ times, once for each circuit $C_i$, incorporating her secret input $x$ as in that algorithm. It is important that she selects new random bits, in step 1(a), at each iteration of algorithm 2 (this goes without saying, technically, since the randomness generation is part of the algorithm, but we emphasize it anyway since security would be seriously compromised if it were overlooked). The weighted sum (weighted by the $\beta_i$ terms) of the outcomes of algorithm 2 on each $C_i$ is an unbiased estimator for the outcome of $C|x\rangle$. She repeats this process polynomially many times (in $t + k$) to reduce her error to any constant level desired.

We need to specify how Alice performs algorithm 2 on the circuits $C_i$, since, in contrast to circuits we have considered so far, each $C_i$ has adaptivity that depends on measurements of *input* wires, specifically of the $k$ final input wires. This is because each of these wires was the bottom wire of a $\mathsf{T}$ gadget in $C'$, and the circuit components of these teleportation gadgets are exactly preserved in each $C_i$. (Since we no longer have $|T\rangle$ as the input state, we now call them *teleportation gadgets* to be more general.) We especially need to show that Alice can propagate the Paulis corresponding to her secret input $x$ (as required in step 2(c) of algorithm 2) past the teleportation gadgets in $C_i$ without generating a non-Pauli correction term.

We remark that the above point is crucial for security. If all $t + k$ gadgets in $C'$ received non-Pauli corrections during the input propagation steps, Alice would need to conceal these corrections from Bob with $t + k$ instances of randomness. But Bob's register has size $t$ only, so Alice cannot send him additional random qubits beyond the $t$ needed for the T gadgets. We should therefore be sure that *only* the $t$ T gadgets of $C'$ receive non-Pauli corrections, or else the protocol is not secure.

We show explicit Pauli propagation rules for teleportation gadgets involving the stabilizer terms in the decomposition of $|T\rangle$ given previously in eqs. (4.4.1) and (4.4.2), namely, the single-qubit stabilizer states $|+\rangle$, $|-\rangle$, and $|i\rangle$. These rules are shown as circuit identities in fig. 5.3. We note that, since the Paulis are propagated through *adaptive* Clifford circuits, we cannot use the straightforward Clifford propagation rules from section 2.6. Indeed, some of the propagated terms depend on the gadget measurement outcome $m$. Nevertheless, the propagation of an arbitrary Pauli through each of these gadgets produces Pauli terms only, as required.

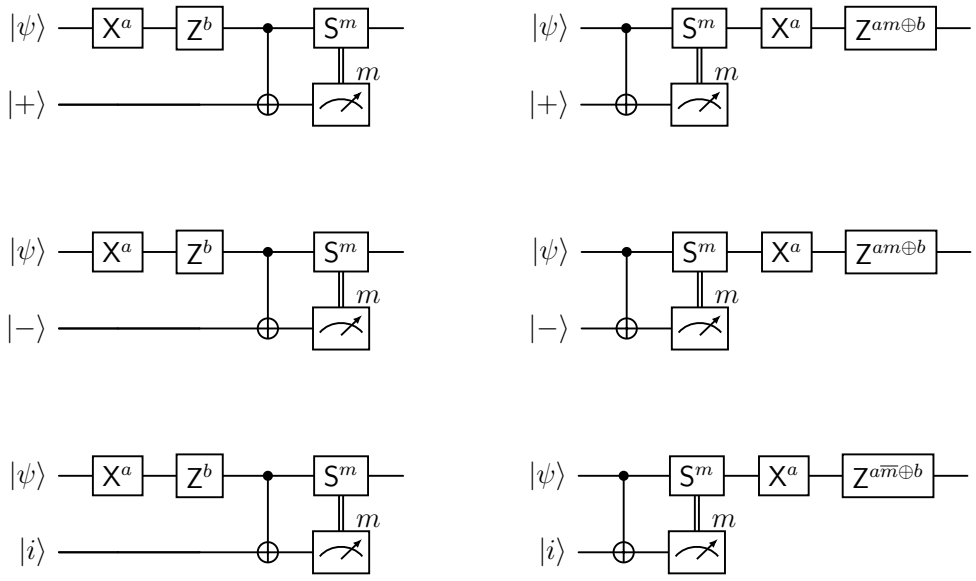



Figure 5.3: Additional propagation rules for input-concealed qubit virtualization.

These three rules suffice to compute the propagations required when we extend this decomposition to any $k$. As noted in section 4.4, to decompose $|T\rangle^{\otimes k}$ into stabilizer states, we may simply take the $k$-fold tensor product of the decomposition of $|T\rangle$ in eqs. (4.4.1) and (4.4.2). In the resulting decomposition of $|T\rangle^{\otimes k}$, each stabilizer

term $|\phi_i\rangle$ is a product of $k$ single-qubit states from the set $\{|+\rangle, |-\rangle, |i\rangle\}$. Propagation through each of the $k$ input-wire teleportation gadgets can therefore be handled by one of the three rules given in fig. 5.3. We have therefore shown enough to allow for input-concealed qubit virtualization to be performed in general, for any $k$, with runtime and communication scaling exponential in $k$ only.

We noted in section section 4.4 that other stabilizer decompositions of lower term count are possible, especially as $k$ increases. For any stabilizer decomposition, the rules for propagating a Pauli through its associated teleportation gadgets will produce only Pauli terms, and thus qubit virtualization is feasible for any stabilizer decomposition of $|T\rangle^{\otimes k}$. One way to see this is by viewing an adaptive Clifford circuit as a logical disjunction of possible (non-adaptive) Clifford circuits, corresponding to the possible internal measurement outcomes, so each of these possible Clifford circuits will have Pauli-to-Pauli propagation rules. The general rules may then be expressed in terms of the measurement outcomes. However, we note that lower-term-count stabilizer decompositions will in general achieve their simplicity by avoiding re-use of lower-dimension states as product terms in their $|\phi_i\rangle$ terms, meaning that the rule set for such a decomposition will grow in rough proportion to its level of term count optimization (below, say, the $3^k$ baseline that we show). For more on the rich topic of stabilizer decompositions of magic states, we refer the reader to [BBC+19].

## 5.5 Example

In this section, we give an example of the input-concealed PBC protocol. We demonstrate the protocol as executed on the same circuit $C$ as was used in section 4.5 of the previous chapter to demonstrate the regular PBC protocol.

The circuit diagrams in figs. 5.4 to 5.11 depict step 2 (pre-processing) of algorithm 2. In these figures, we colour-code certain gates and bits, for ease of understanding: blue is used for gates related to the propagation of the secret input, $x$, while orange and red denote elements related to Alice's randomness. To avoid being overly fine-grained in the colouring, we choose a single colour for any gate involving elements of the input and of randomness, according to its main function in the protocol.

The example proceeds as follows. Alice wishes to execute circuit $C$, shown in

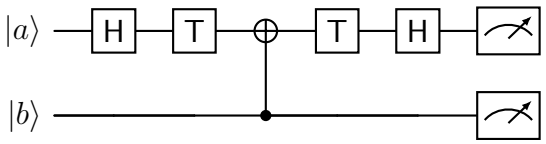



Figure 5.4: The circuit $C$ to be executed in our example run of input-concealed PBC. Alice's input is $x = ab \in \{0,1\}^2$, with $a, b \in \{0,1\}$ and juxtaposition denoting concatenation.

fig. 5.4, on some input $x \in \{0,1\}^2$. To keep the example general we do not make $x$ explicit but represent it as the concatenation $x = ab$ for $a, b \in \{0,1\}$. In step 1 (not shown in figures), Alice chooses $c_1, c_2, d_1, d_2 \in \{0,1\}$ uniformly at random, prepares a rotated magic register, and sends it to Bob.

Next, Alice performs step 2, the pre-processing of $C$ with respect to $x$. First, Alice takes the magic state injected form of $C|00\rangle$ and represents her input as a sequence of initial $\mathsf{X}$ gates, then propagates them to the right, up to and just past the first $\mathsf{T}$ gadget (fig. 5.5). She then performs a randomization step (fig. 5.6), leaving behind an $\mathsf{S}$ gate with exponent $d_1$ (so, using the notation from the protocol description, $s_1 = d_1$), and continues propagating the Pauli elements rightwards up to and just past the second $\mathsf{T}$ gadget (fig. 5.7). She performs another randomization step, leaving behind another $\mathsf{S}$ gate with exponent $s_2 = d_2 \oplus b$ (fig. 5.8), and does a final Pauli propagation (fig. 5.9). Finally for step 2, she records her last-layer Pauli corrections (fig. 5.10) and drops them from the circuit to get $C^*$ (fig. 5.11).

Then, not shown in figures, Alice does step 3, a PBC interaction with Bob of $C^*|00\rangle$ where Bob measures on the randomly rotated magic states supplied by Alice in step 1. Finally, Alice performs local final corrections as needed on the readout measurements that she found in step 3. In particular, Alice flips $m_3$ if and only if $b = 1$ and flips $m_4$ if and only if $f \oplus b \oplus e = 1$. The readout measurement values generated in this way are a sample from the distribution of $C|x\rangle$, as desired.

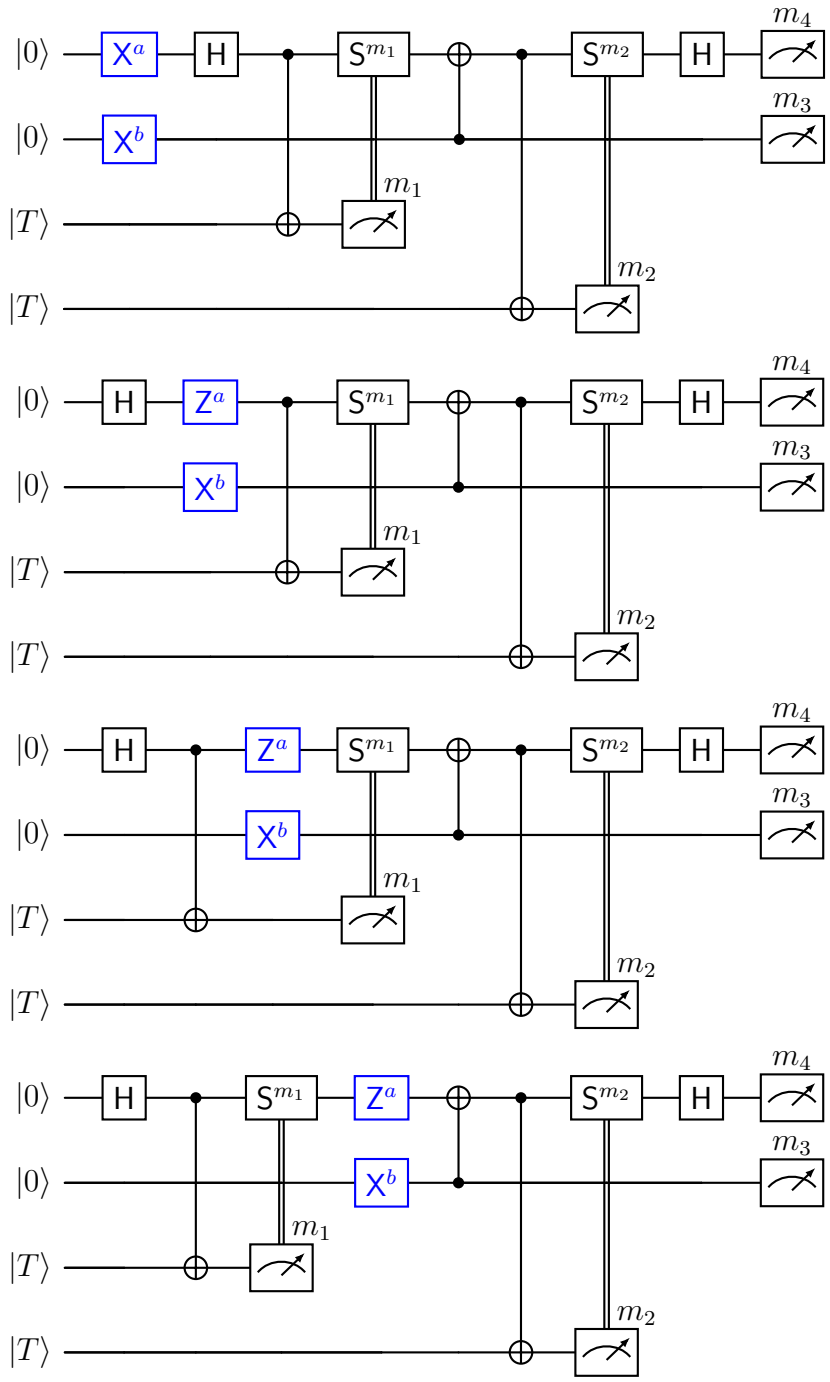
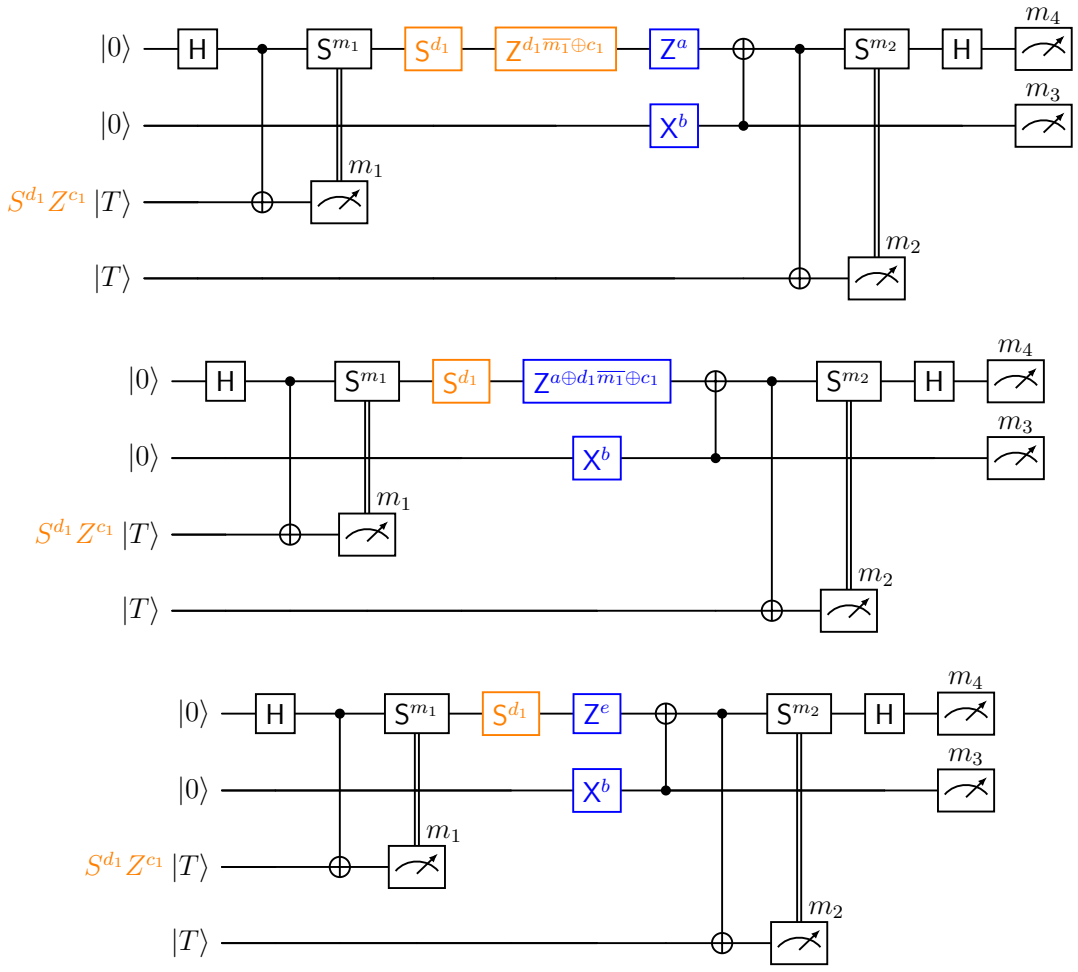
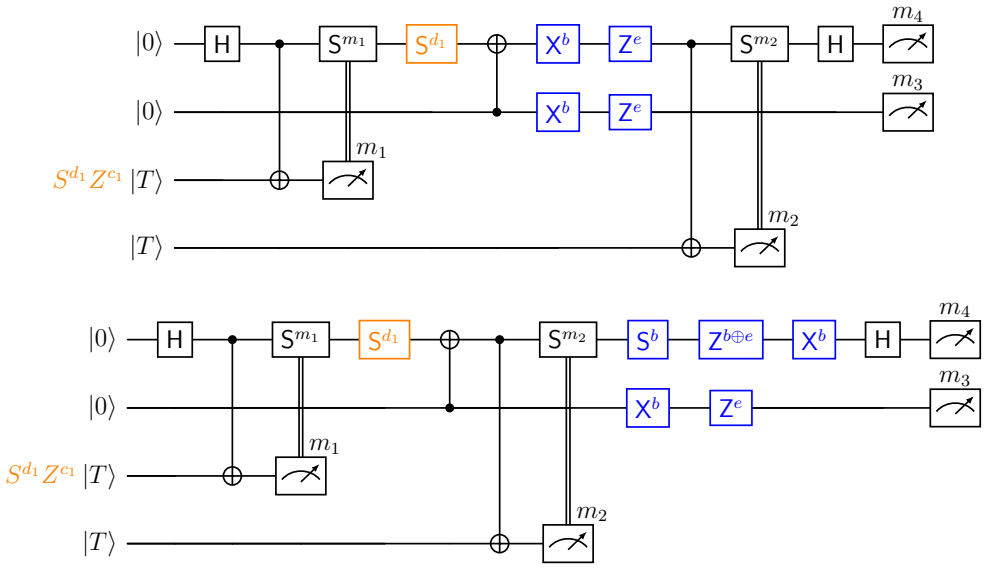



Figure 5.5: First propagation steps in the input-concealed PBC example. Alice represents her input $x = ab$ for $a, b \in \{0, 1\}$ as X gates on the data wires, then propagates them rightwards. We colour the Pauli gates coming from her input in blue.



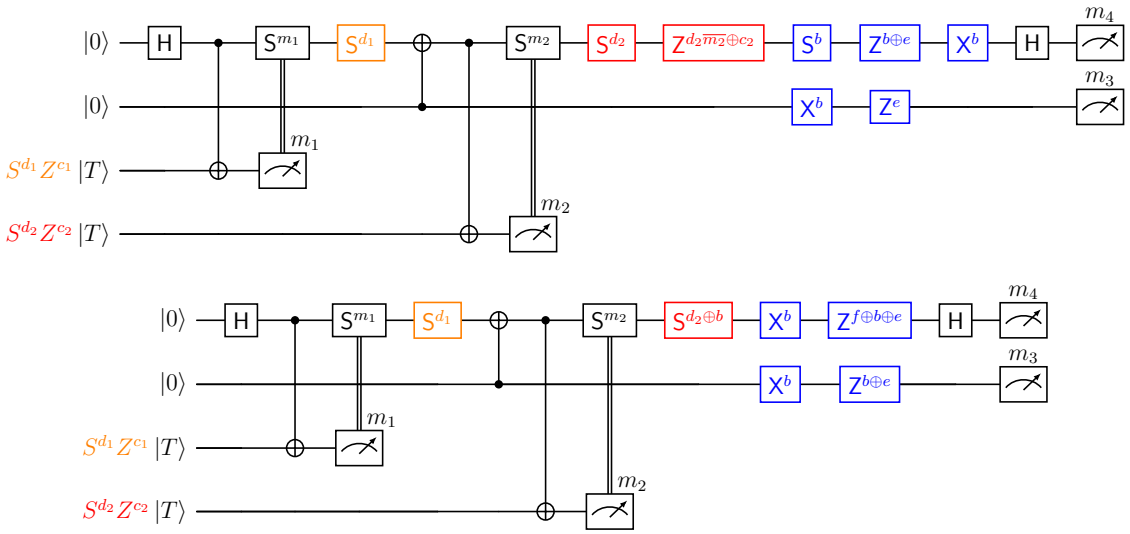

$$(e = a \oplus d_1\overline{m_1} \oplus c_1)$$

Figure 5.6: First randomization step in the input-concealed PBC example. We use orange to denote the magic state randomization operations and associated corrections and blue for Pauli gates coming from Alice's input.




$$(e = a \oplus d_1\overline{m_1} \oplus c_1)$$

Figure 5.7: Continued propagation steps in the input-concealed PBC example. Orange denotes the first randomization and correction and blue denotes Pauli gates coming from Alice's input.




$$(e = a \oplus d_1\overline{m_1} \oplus c_1)$$
$$(f = d_2 b \oplus d_2\overline{m_2} \oplus c_2)$$

Figure 5.8: Second randomization step in the input-concealed PBC example. Orange denotes the first randomization and correction, red denotes the second randomization and correction, and blue denotes Pauli gates coming from Alice's input.

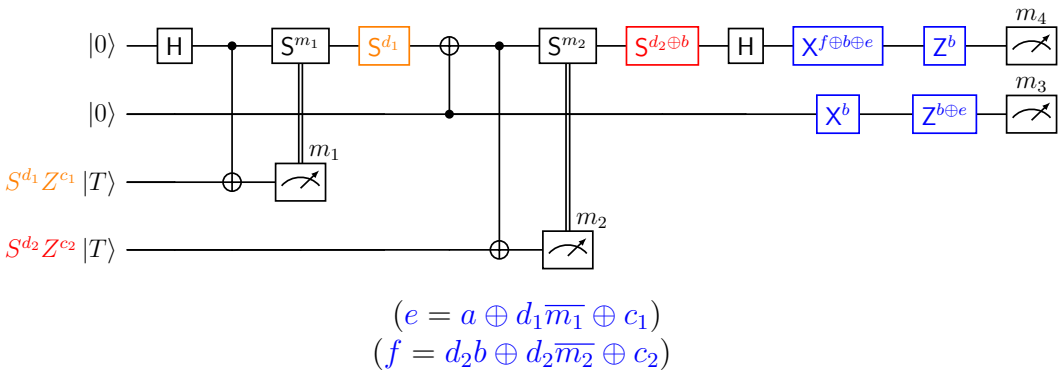



Figure 5.9: Third and final propagation step in the input-concealed PBC example. Orange denotes the first randomization and correction, red denotes the second randomization and correction, and blue denotes Pauli gates coming from Alice's input.

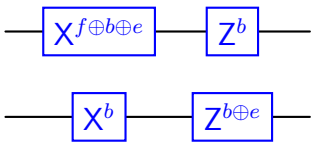



Figure 5.10: Alice's last-layer Pauli corrections in the input-concealed PBC example.

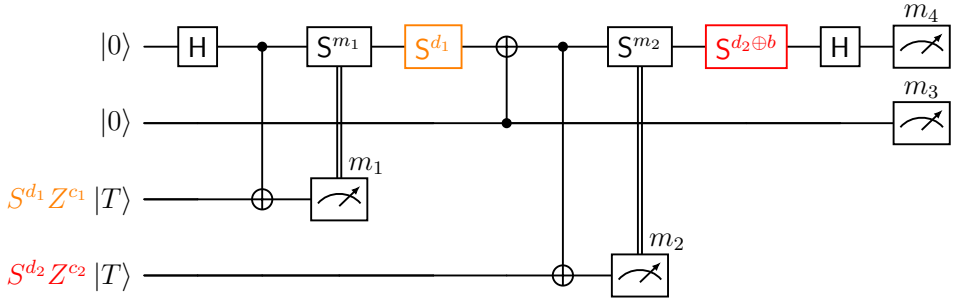



Figure 5.11: The processed circuit $C^*$ in the input-concealed PBC example. $C^*$ is the circuit that Alice uses for the PBC interaction in step 3 of algorithm 2.

# Chapter 6

# Advantages of the Protocol

In this chapter, we compare our protocol for input-concealed PBC to existing SDQC protocols to demonstrate its resource advantages. The schemes we will consider for comparison are the UBQC protocol of [BFK09] and the QCED protocol of [FBS+14] and [Bro15], the resource requirements of which are discussed in chapter 3. In section 6.1, we enumerate the resource needs of our protocol, dividing the discussion into client resources (section 6.1.1), communication resources (section 6.1.2), and server resources (section 6.1.3). We summarize the comparison in table 6.1 and include some discussion in section 6.2. Overall, we find that our protocol significantly reduces the server resources in certain regimes, while using comparable, or even slightly better client and communication resources compared to existing schemes. We then discuss two special cases, where our protocol offers additional advantages not directly comparable to existing SDQC protocols. In qubit virtualization (section 6.3), under specific server-side resource limitations, our protocol still runs at a reasonable time complexity where all other existing SDQC protocols become intractable. We then define *X-invariant circuits* (section 6.4), a special class of circuit for which our scheme becomes entirely classical on the client side. We compare this situation against other SDQC protocols, like QCED, which run with a classical client exactly for Clifford circuits, a strictly smaller class than X-invariant circuits. To our knowledge, this shows for the first time a method for classical-client SDQC with perfect security for a non-trivial extension beyond Clifford circuits.

## 6.1 Resource Requirements of Input-Concealed PBC

Here we discuss the resources required by our protocol, to facilitate a comparison with existing SDQC protocols. In particular, the presentation in algorithm 2 is not resource-optimal, since some steps are described with clarity rather than efficiency in mind. We therefore point out a few minor changes that can be made to optimize resource use of our protocol.

### 6.1.1 Client Resources

As for client resources, in our presentation of input-concealed PBC in algorithm 2, we say that Alice prepares states at random from the set

$$\{\frac{1}{\sqrt{2}}\big(\,|0\rangle + e^{i\theta}\,|1\rangle\,\big) : \theta \in \{\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}\}\}.$$

While this is a kind of four-way state preparation, similar to that of QCED, it is arguably less convenient, in that the states in this set are magic states.

This issue is preventable. We give three simple alternative preparations. First, we can have Alice prepare and send states from the same set as in QCED, i.e.,

$$\{\frac{1}{\sqrt{2}}\big(\,|0\rangle + e^{i\theta}\,|1\rangle\,\big) : \theta \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}\},$$

and have Bob perform a $\mathsf{T}$ gate on each one on receipt. Second, if applying these additional $\mathsf{T}$ gates online during the protocol is resource-intensive for Bob, he may prepare $|T\rangle$ states ahead of time (as is common in computation with magic states); then Alice can send states from the same set as in the first method and Bob can use a $\mathsf{T}$ gadget on each state to effect the necessary rotation. Third, and less plausibly since it incurs a quantum communication overhead, Bob could prepare magic states ahead of time, send them to Alice, who applies $\frac{\pi}{2}$ and $\pi$ rotations at random and sends them back. In table 6.1, we assume one of the first two methods is used. This change affects neither correctness nor security, but makes the client resources of our scheme somewhat lighter and exactly equivalent to those of the QCED scheme.

The number of single-qubit states to be prepared by the client in our protocol is

always exactly $t$.

As for client-side encryption and decryption costs, our scheme has no additional requirements. We do not even need to apply an $n$-bit classical one-time pad, since the input is never actually transmitted.

### 6.1.2 Communication Resources

Our protocol requires a single instance of one-way quantum communication of exactly $t$ single-qubit states.

In terms of classical communication, our protocol as stated in algorithm 2 requires one two-way round per measurement, for a total of $t' \leq t$ rounds. Also per the algorithm 2 procedure, each round requires Alice to describe a $t$-qubit Pauli measurement, requiring $2t+1$ bits, and requires Bob to reply with a single bit. However, these parameters can also be optimized. (Since classical communication is typically cheap, it may or may not be worthwhile making these optimizations in practice, but we offer them anyway.) First, as for the number of bits required from each of Alice's messages, we utilize the observation, discussed in section 5.1, that a circuit determines the vertices of the PBC tree for that circuit executed on all classical inputs $x$. Then, Alice needs only to convey one bit of information to Bob to request the next Pauli in the measurement sequence (corresponding, say, to a right or left branch in the binary tree), so long as Alice and Bob agree on a convention for ordering the vertices at each level, which may be dictated in the protocol. Thus we reduce the size of each of Alice's messages from $2t+1$ bits to exactly 1 bit. Second, as for the number of rounds of classical communication required, we use the observation, discussed in section 4.3, that branching in the tree is only caused by gadget measurements, and never by readout measurements. This means that Alice may cease sending messages after the $\ell$ quantum gadget measurements have been requested, and further classical communication may be one-way from Bob to Alice. This gives a total number of classical messages of $\ell$ from Alice to Bob and $\ell + \ell' = t' \leq t$ from Bob to Alice (where $\ell'$ is the number of quantum readout measurements). Usually, $\ell < t'$, so this also represents a savings.

We have no additional classical communication required to transmit the encrypted classical input, since, again, the protocol does not involve sending it.

### 6.1.3 Server Resources

We achieve a server quantum register size of exactly $t$. This is the major resource advantage of our protocol. Moreover, since Pauli measurements can be performed fault-tolerantly, no register expansion is required by the server if fault-tolerance is demanded (see section 2.6).

## 6.2 Discussion of Comparison

We summarize the comparison of our protocol (with resource requirements just described) against the UBQC and QCED protocols (with resource requirements described in chapter 3) in table 6.1 and discuss the comparison in this section. The table describes the resource cost of each protocol executed on a circuit of $n$ input qubits, $t$ $\mathsf{T}$ gates, and depth $d$.

The comparison shows that our protocol is optimal or tied-for-optimal in all quantitatively-described resource parameters (that is, in all parameters except server capabilities, which are described qualitatively and which we discuss later in this section). We make a few observations that make this claim clear. First, with respect to the $\mathcal{O}(nd)$ term that appears repeatedly in the UBQC column, a depth-$d$ circuit on $n$ qubits has at most $nd$ total gates, and in particular at most $nd$ $\mathsf{T}$ gates, so $nd \leq t$, with equality in the implausible case that the circuit consists exclusively of $d$ layers of $n$ $\mathsf{T}$ gates each. Moreover, the $\mathcal{O}$ notation in this case hides a constant that is greater than 1, so the comparison between $\mathcal{O}(nd)$ and $t$ strictly favours $t$. Also, for comparisons involving $\ell$, $\ell'$, and $t$, we note that $\ell + \ell' \leq t$. Using these observations, it can be seen directly from the table that our protocol either matches (e.g., in client state preparation requirements) or improves on (e.g., in server register size) the best existing resource bound.

A small exception to this claim comes from the possibility, noted in section 3.3, that QCED may be alternatively run on a server register of $n + 1$ qubits, by sending quantum states from client to server one at a time. In this case, our protocol cannot be said to strictly improve on the QCED server register size requirement, since the relation of $n$ and $t$ depends on the parameters of the particular circuit. Still, in this case, a scaling of server resources in $t$ will often be preferable to $n$, just as this scaling

Table 6.1: Resource cost comparison of our protocol and existing schemes. We compare the protocols as run on an $n$-qubit, depth $d$ circuit with $t$ $\mathsf{T}$ gates. Input-concealed PBC is our novel protocol, described in algorithm 2. The UBQC and QCED protocols are assumed to be run on classical inputs and outputs, to level the comparison. $\ell$ and $\ell'$ are the number of quantum gadget and readout measurements in the PBC procedure, respectively.

| | | UBQC | QCED | Input-concealed PBC |
|---|---|---|---|---|
| Client | State preparation | 8-way | 4-way | 4-way |
| Communication | Quantum messages | 1 (client-server) | 1 (client-server) | 1 (client-server) |
| | Total qubits sent | $\mathcal{O}(nd)$ | $t$ | $t$ |
| | Classical messages | $\mathcal{O}(nd)$ each way | $2t$ ($t$ each way) | $\ell$ (client-server) and $\ell + \ell'$ (server-client) |
| | Total bits sent | $\mathcal{O}(nd)$ | $2t$ | $2\ell + \ell'$ |
| Server | Register size (qubits) | $\mathcal{O}(nd)$ | $n + t$ | $t$ |
| | Capabilities | Entangling gate (e.g., $\mathsf{CZ}$); single-qubit measurements at magic angles | Universal gate set; single-qubit, computational basis measurements | $\mathsf{T}$ gate *or* magic state preparation and gadget; multi-qubit, non-destructive Pauli measurements |

difference is a motivation behind regular PBC. Also, in this case, the number of quantum messages required by the QCED protocol increases significantly from 1 to $t$, and these messages must be sent online during the interaction, a major limitation.

The server capabilities are less directly comparable. Whether one set of capabilities is preferable to another will depend on the details of hardware and implementation, discussion of which is beyond the scope of this thesis. We note that the difference in server capabilities between the protocols closely replicates the differences in resource requirements between the computational model that each protocol is presented in: in UBQC, the server requires similar capabilities to the quantum requirements of 1WQC; in QCED, of the circuit model; and in input-concealed PBC, of PBC. Just as the quantum capability requirements of regular PBC are often seen

as advantageous over those of other computational models, likewise we expect the server capabilities of our protocol to be advantageous in many scenarios.

Finally, we note that these advantages apply only in the regime of input-hiding with classical input and output. While our protocol appears highly advantageous in this regime, it is not intrinsically circuit-hiding (as UBQC is) and cannot be adapted to accept quantum inputs and outputs (as UBQC and QCED can).

## 6.3 Qubit Virtualization

As observed in section 5.4, our protocol also allows for a version of the qubit virtualization technique with secret classical input. We are not aware of any existing SDQC protocol that supports a comparable technique, representing a categorical advantage of our protocol over previous work. In scenarios where the server's quantum register has size $t$ and the desired computation requires a $t + k$ qubit quantum memory, existing SDQC protocols simply cannot be run as designed. The only option in these scenarios for existing protocols seems to be to convert the entire computation to a classical simulation, and run this computation via the protocol, incurring a runtime and communication cost exponential in $t + k$ and rendering the $t$-qubit register irrelevant. With our protocol, an extension of $k$ qubits to a $t$-qubit register can be made at costs exponential in $k$ only, with the quantum resources still used to full advantage.

## 6.4 X-Invariant Circuits

Our protocol can also be run with a fully classical client on a strictly larger set of circuits than existing protocols. As observed in chapter 3, classical-client SDQC has not been achieved for general circuits, and there is some evidence suggesting that this is infeasible. However, some SDQC protocols "automatically" call for zero quantum resources from the client when the circuit $C$ in question comes from a restricted class. In QCED, for example, a client following the protocol will make no quantum operations and send/receive no quantum messages to/from the server in the case that $t = 0$, i.e., when the circuit $C$ is Clifford (assuming, as we do throughout this chapter, that only a classical input and output are required). By contrast, our protocol

can be extended with a simple classical conditional check to recognize a strictly larger set of circuits which do not require quantum resources of the client; we call this class *X-invariant circuits*, or XINV.

The simple modification of our protocol works as follows. Alice reverses the order of steps 1 and 2, so as to delay preparing and sending quantum states until after the classical pre-processing. Then, during the pre-processing (step 2 in algorithm 2, now performed first), Alice performs the propagation of inputs for a generic input $x = x_1...x_n \in \{0,1\}^n$ and generic values of classical random bits $c_1, ..., c_t, d_1, ..., d_t \in_R \{0,1\}$ rather than explicit values (as we do in the example of section 5.5). Then, after pre-processing, Alice inspects the exponents $s_i$ of the cryptographic S gates in $C^*$, $1 \leq i \leq t$, which are Boolean formulas which may in general involve random bits, measurement outcomes, and bits of the input $x$. Specifically, she performs an *invariance check* to check if, for every $s_i$ formula, that formula does not include any term that is a bit $x_j$ of $x$. If *yes*, Alice sends a classical message to Bob instructing him to proceed with a *regular* PBC of $C|0^n\rangle$ on an unrandomized magic state $|T\rangle^{\otimes t}$. If *no* (i.e., if $x_j$ appears in some $s_i$ for some $i \in [t], j \in [n]$), Alice proceeds with the rest of the input-concealed protocol as usual. These modifications and the invariance check can be performed efficiently.

In the case where the invariance check is *yes* and Alice instructs Bob to proceed classically, security is not compromised, since $C^*$ contains no input-dependent components, so all the random bits may be assumed by both parties to be 0 without any compromise of the secret. Correctness is also preserved, since Alice can (and must) still perform her last-layer Pauli corrections on the readout measurements, thus receiving a correct outcome exactly as she would have if, by chance, she had selected all the $c_i$ and $d_i$ to be 0. Moreover, just as regular PBC requires no quantum resources of Alice, so does this modified procedure: the client resources are completely classical.

We call a circuit which gives a *yes* result to the invariance check an *X-invariant circuit*, since its $C^*$ form does not vary under different choices of initial X gates:

**Definition 6.4.1.** A circuit $C$ is *X-invariant*, denoted $C \in$ XINV, if, for a generic classical input $x$ to $C$, every exponent formula $s_i$ in $C^*$ produced by step 2 of algorithm 2 includes no input-dependent term $x_j$ for $j \in [n]$.

We show in the next proposition that X-invariant circuits are a strictly larger

class than Clifford circuits.

**Proposition 6.4.2.** CLIF $\subsetneq$ XINV.

**Proof.** Recall that circuits in CLIF correspond exactly to circuits from our usual universal gate set with $t = 0$ T gates. To show the inclusion, we observe that for any circuit $C$ with $t = 0$, the pre-processing step of the input-concealed PBC protocol will include no randomization steps, and will thus leave a circuit $C^*$ with no cryptographic S gates. Hence the set of exponents $\{s_i\}$ of these gates is empty and they trivially pass the invariance check. So CLIF $\subseteq$ XINV.

To show that the inclusion is strict, we give an example circuit in fig. 6.1 that is in XINV\CLIF. That the circuit is not in CLIF can be seen from the fact that it includes T gates. That it is in XINV can be seen as follows. Any initial set of X gates, when propagated past the first row of H gates, will become Paulis with only a $Z$ term. They then propagate past T, CNOT, and S gates still as $Z$ terms only, and each randomization step $i$ will produce a cryptographic S gate with exponent exactly $d_i$, i.e. with no dependency on $x$. So the invariance check gives a *yes* outcome and the circuit is in XINV. ∎

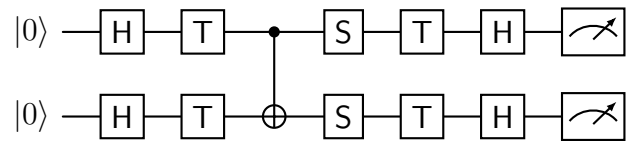



Figure 6.1: An X-invariant circuit with four T gates.

We remark that a much simpler circuit construction than the one in fig. 6.1 would suffice for the second part of the proof above. For example, a circuit on a single input wire consisting of a single H gate followed by a single T gate and final readout measurement is also in XINV\CLIF. We opt to work through a slightly more involved construction because the reasoning in the proof is instructive. First, the argument can be seen to be extendable to any circuit constructed in the following pattern: an initial layer of $n$ H gates, an arbitrarily large layer consisting of T, CNOT, and S gates, and an optional final layer of H gates. All such circuits will be in XINV, by the same reasoning as in the proof, and will not be in CLIF whenever $t > 0$. In turn, this shows that XINV includes circuits of substantially more complex structure than

CLIF (in particular, XINV includes circuits with arbitrarily many T gates), and all such circuits will be delegated with classical client resources by our protocol (augmented by the simple invariance check).

# Chapter 7

# Security

In this chapter, we prove the security of the input-concealed PBC protocol presented in chapter 5. We discuss some of the intuition behind the proof (section 7.1) and present the formal security definition we will use, following earlier work (section 7.2). Then, in sections 7.3 and 7.4, we prove security, eventually stated as theorem 7.4.4. A significant element in our proof is the introduction of another protocol, called **S** (algorithm 3), a modification of input-concealed PBC which is more convenient to analyze. We use a construction called a *translator* to show that the security of **S** implies the security of input-concealed PBC (lemma 7.3.1), then show the security of **S** (lemma 7.4.3). We end the chapter by sketching the proof of security for input-concealed qubit virtualization (section 7.5).

## 7.1 Intuition

At the start of chapter 5, we provided some intuition behind the input-concealed PBC protocol (algorithm 2). We ended by making an analogy with the classical one-time pad: Bob knows the circuit description $\delta$, so he knows the PBC tree of the computation, and sees himself being routed through the tree by Alice, down branches that he would not be sent along in a regular PBC on $C|0^n\rangle$. We compared this routing to the ciphertext, claiming (without proof) that it combined in an XOR-like way Alice's intentional, input-dependent branch-switching (the message), and some secret, random corrections induced by magic state rotations (the key). Our task in

this chapter is to formalize and prove these claims.

We will encounter some challenges in the proof, which we can again explain by a comparison to the classical one-time pad. First, in the first step of algorithm 2, Alice sends Bob, the adversary, the randomly rotated magic register, which (our analogy claims) contains the secret key! In the classical one-time pad setting, sending the key to the adversary would immediately compromise security. Second, setting this issue aside, in our protocol Bob inevitably learns certain bits of the key via the interaction pattern. Consider that, in the example of section 5.5, the first cryptographic $\mathsf{S}$ gate in the processed circuit $C^*$ (that is, the one depicted in orange in fig. 5.11) has an exponent, simply, of $d_1$, which is the first bit of the key. Since Bob can, in general, detect the presence or absence of a cryptographic $\mathsf{S}$ gate from measurement requests, it follows that he may learn $d_1$. So, in our protocol, Bob has at least two ways, *a priori*, to learn the key and thus to decrypt Alice's secret input.

We handle these challenges as follows. First, the key component of the magic register is not given to Bob in the plain; rather, it is itself encrypted via the random $Z$ rotations that are layered on top, and we show that this prevents Bob from learning it. Second, while Bob does indeed learn parts of the key as the procedure unfolds, he does so in an allowable order, so that at each step of the protocol, Alice's secret is concealed exclusively by parts of the key that Bob cannot yet have learned, ensuring that security is maintained.

## 7.2 Security Definition

We now formalize a security definition for a delegated computation protocol, closely following [Bro15].

Let a delegated computation protocol be specified by a pair $(A, B)$ representing an honest Alice and an honest Bob, both quantum. The protocol takes as input a classical string $\delta \in \{0,1\}^*$, describing the circuit to be executed, and a quantum state $\rho_{\text{in}} \in \mathrm{D}(\mathcal{A} \otimes \mathcal{B})$, where $A$ receives the register in $\mathcal{A}$ and $B$ receives those in $\mathcal{B}$. We will write $(A, B)_\delta$ for an execution of the protocol on classical input $\delta$.

For security, we consider pairs $(A, B')_\delta$ where $B'$ deviates arbitrarily from $B$,

while still following the interaction pattern prescribed by $\delta$.[1] For a given deviating Bob $B'$, we define a simulator, $\mathscr{S}[B']$, which induces a channel without interacting with Alice. Let $\mathcal{Z}$ be the output space of $B'$ and let $\Phi_\delta : \mathrm{L}(\mathcal{A} \otimes \mathcal{B}) \to \mathrm{L}(\mathcal{Z})$ be the channel induced by $B'$ via the interaction with $A$ according to $\delta$. The simulator $\mathscr{S}[B']$ has the same input and output dimensions as $B'$ and is given $\delta$, but does not interact with $A$ and acts on the reduced input state $\mathrm{tr}_{\mathcal{A}}(\rho_{\mathrm{in}})$. We label the channel induced by tracing out the client register of the input state and running $\mathscr{S}[B']$ on $\mathrm{tr}_{\mathcal{A}}(\rho_{\mathrm{in}})$ as $\Sigma_\delta : \mathrm{L}(\mathcal{A} \otimes \mathcal{B}) \to \mathrm{L}(\mathcal{Z})$.

**Definition 7.2.1** ($\epsilon$-privacy)**.** A protocol $(A, B)$ for a delegated quantum computation is $\epsilon$-private if, for every deviating Bob $B'$, there exists a simulator $\mathscr{S}[B']$, such that for every classical input $\delta$, $||\Phi_\delta - \Sigma_\delta||_\diamond \leq \epsilon$ where $\Phi_\delta$ and $\Sigma_\delta$ are as described above.

## 7.3 Protocol S

For convenience, through the remainder of this chapter we refer to the input-concealed PBC protocol (algorithm 2) as **P**. We aim to show that **P** is $\epsilon$-private for $\epsilon = 0$. To do this, we introduce another protocol, **S**, closely related to **P**, but for which it is more convenient to prove security. We show in this section that the security of **S** implies the security of **P** (lemma 7.3.1). Then, in the next section, we prove the security of **S** (lemma 7.4.3), thus completing our proof of the security of **P** (theorem 7.4.4).

To help the reader keep track of the letters **P** and **S**, which may seem arbitrary, we note that **P** is our previously-presented PBC protocol, whereas **S** is a scheme for showing security. Furthermore, as will be seen, the $s_i$ bits of $C^*$ play an important role in **S**.

**S** is presented in algorithm 3 at a high level. The main idea behind **S** is for Alice to reveal her processed circuit $C^*$ iteratively to Bob, by sending him at each step of their interaction the next exponent $s_i$ on the $i$-th cryptographic S gate. Then lemma 7.3.1 intuitively states that doing this reveals no more to Bob than is revealed by doing performing **P**. We describe **S** in more detail next.

[1] We could also consider security against adversaries $B'$ which do not follow the interaction pattern. This would require us to specify Alice's behaviour (e.g., aborting) in case of an unexpected interaction pattern, as well as what constitutes an unexpected pattern, which could be cumbersome.

**Algorithm 3** Protocol **S**. **P** refers to the input-concealed PBC protocol (algorithm 2). Initially, as in **P**, Alice holds circuit $C$ with an $n$-qubit data register and $t$ $\mathsf{T}$ gates and a classical input $x \in \{0,1\}^n$.

1. As in **P**, Alice prepares $t$ randomly rotated magic states and sends them to Bob.
2. As in **P**, Alice does classical pre-processing of $C$, dependent on $x$, to produce $C^*$, an adaptive Clifford circuit.
3. Differently from in **P**, Alice and Bob perform a modified PBC on $C^* |0^n\rangle$ as follows. For each measurement $i$ in $C^*$:
   (a) For $1 \leq i \leq t$, Alice and Bob each determine $P_i$ separately. If $P_i$ is AC or DC case, Alice and Bob each determine $m_i$ separately, using post-selection to ensure this is deterministic. If $P_i$ is IC case, Bob measures quantumly on the rotated magic register and sends $m_i$ to Alice. Then, after each $m_i$ is determined, Alice sends $s_i$ to Bob.
   (b) For $t+1 \leq i \leq t+w$, Alice and Bob again determine $P_i$ separately. Alice determines $m_i$ privately for AC and DC case $P_i$, using classical randomness for AC case. If $P_i$ is IC, Bob measures quantumly and sends $m_i$ to Alice.
4. As in **P**, after the interaction concludes, Alice locally applies a final classical correction to the readout measurements.

The initial setup and steps 1, 2, and 4 of **S** are identical to those in **P**. We therefore focus on the interactive step, step 3.

Step 3 of **S** proceeds as follows. As in step 3 of **P**, step 3 of **S** iterates through each measurement $i$ of $C^*|0^n\rangle$, indexed from earliest to latest, but the procedure for each iteration $i$ is different from in **P**.

In step 3(a) of **S**, Alice and Bob handle the gadget measurements, indexed $1 \leq i \leq t$, determining outcome $m_i$ for each one. First, both parties independently and locally perform the propagation required to find $P_i$, then assess it into AC, DC, or IC cases as in regular PBC. (Note that they can do this independently for $i = 1$, and will agree on $P_1$, since $C^*$ is identical to $C$ up to the first gadget measurement, and $C$ is known to both parties. We argue at the end of this paragraph that they can do this independently and agree on $P_i$ for each subsequent $i$.) Second, if $P_i$ is AC or DC, each party determines $m_i$ locally. They use the post-selection technique

(see section 4.2.4), so this happens deterministically in the AC case. The DC case is deterministic in regular PBC, so they follow the same procedure as there. Thus they arrive at the same $m_i$ in the AC and DC cases without needing to communicate. If $P_i$ is IC case, each party recognizes that they are in this case; Bob performs the quantum measurement on the rotated register that Alice sent him in step 1 and sends the outcome $m_i$ to Alice. So they agree on $m_i$ in every case. Third, at the end of each iteration $i$, regardless of case, Alice sends a one-bit message to Bob, $s_i \in \{0, 1\}$, which corresponds to the exponent on the cryptographic $\mathsf{S}$ gate at the $i$-th gadget, which she can now compute explicitly because $m_i$ has been determined. Sharing $s_i$ with Bob and observing that both parties agree on each $m_i$ ensures that they perform the $(i+1)$-th Pauli propagation identically and agree on $P_{i+1}$, since this Pauli propagates through $\mathsf{S}^{s_i}$ and $\mathsf{S}^{m_i}$. Thus they agree on $P_i$ for all $i$, $1 \leq i \leq t$.

We note that after the first $t$ measurements are handled, Alice has sent Bob exactly $t$ messages, each a single bit. Bob has sent Alice $\ell$ single-bit messages, $m_{j_1}, ..., m_{j_\ell}$, where $\ell$ is the number of quantum gadget measurements (this notation is the same as that of section 4.3).

In step 3(b), Alice and Bob handle the readout measurements, i.e., measurement $i$ for $t+1 \leq i \leq t+w$. Now, by an observation of section 4.2.4, branching in the PBC tree has ceased. Bob may determine independently for each remaining $P_i$ whether it is IC case, and measure each IC $P_i$ in its $+1$ phase, following the sign suppression technique. For each such IC $P_i$, he sends $m_i$ to Alice. Alice determines AC and DC case measurement outcomes locally, and receives the IC ones from Bob.

Over all iterations of step 3(b), then, Alice sends no messages to Bob, and Bob sends $\ell'$ single-bit messages, the quantum readout measurement outcomes $m_{k_1}, ..., m_{k_{\ell'}}$, to Alice.

This concludes our detailed description of **S**. We now state and prove that the security of **S** implies the security of **P**.

**Lemma 7.3.1.** If **S** is $\epsilon$-private with $\epsilon = 0$, then **P** is $\epsilon$-private with $\epsilon = 0$.

**Proof.** We construct an ***S*-*P*** *Translator*, an efficient classical machine denoted $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$, which interacts with an Alice-following-**S**, $A_{\mathbf{S}}$, on the left, and a Bob-following-**P**, $B_{\mathbf{P}}$, on the right. In this proof, we will denote the interaction of two machines $X$ and $Y$ as $X \rightleftharpoons Y$, so in the present case we have $A_{\mathbf{S}} \rightleftharpoons T_{\mathbf{S}\leftrightarrow\mathbf{P}} \rightleftharpoons B_{\mathbf{P}}$. The task of

$T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ is to simulate for $B_{\mathbf{P}}$ an interaction with $A_{\mathbf{P}}$, while $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ is actually interacting on the other side with $A_{\mathbf{S}}$, *and* to simulate for $A_{\mathbf{S}}$ an interaction with $B_{\mathbf{S}}$ while actually interacting with $B_{\mathbf{P}}$. In other words, for any $\delta$, $A_{\mathbf{S}} \rightleftharpoons T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ (considered as a compound system) should act as $A_{\mathbf{P}}$ does in an interaction with $B_{\mathbf{P}}$ according to $\delta$, and $T_{\mathbf{S}\leftrightarrow\mathbf{P}} \rightleftharpoons B_{\mathbf{P}}$ should act as $B_{\mathbf{S}}$ when interacting with $A_{\mathbf{S}}$ according to $\delta$.

We will construct $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ to translate step 3 of each protocol only. Since $\mathbf{P}$ and $\mathbf{S}$ are identical at each step except step 3, we will have $A_{\mathbf{S}}$ and $B_{\mathbf{P}}$ interact directly for the other steps, particularly step 1, and still achieve the desired effect.

We construct $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ as follows. Knowing $\delta$, $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ computes back-propagated Pauli measurements and receives $s_i$ messages from $A_{\mathbf{S}}$ until an IC Pauli $P_{j_1}$ appears, following the notation of section 4.3. (Possibly the first considered measurement is quantum, i.e., $j_1 = 1$, in which case no $s_i$ messages are sent/received at this first step.) Then $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ sends $P_{j_1}$ to $B_{\mathbf{P}}$ and receives $m_{j_1}$ in reply; $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ sends $m_{j_1}$ to $A_{\mathbf{S}}$. It continues to receive $s_i$ bits from $A_{\mathbf{S}}$ until the next Pauli needs to be measured quantumly, and so on. After it has received all of $s_1, ..., s_t$ from $A_{\mathbf{S}}$ by this procedure, it continues interacting with $B_{\mathbf{P}}$ to receive the results of the quantum readout measurements, $m_{k_1}, ..., m_{k_{\ell'}}$, sending these back to $A_{\mathbf{S}}$.

We say that $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ is a *perfect translator*, in the following sense. For any classical input $\delta$, $T_{\mathbf{S}\leftrightarrow\mathbf{P}} \rightleftharpoons B_{\mathbf{P}}$ performs exactly the operations, including interactive communications, as $B_{\mathbf{S}}$ does given $\delta$, and $A_{\mathbf{S}} \rightleftharpoons T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ performs exactly the operations and interactions as $A_{\mathbf{P}}$ given $\delta$. This can be verified by comparing the operations of $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ to those of the relevant party in $\mathbf{P}$ and $\mathbf{S}$.

For security, let $B'_{\mathbf{P}}$ be a deviating Bob in $\mathbf{P}$, and let the map induced by $B'_{\mathbf{P}}$ be denoted by $\Phi_\delta : \mathrm{L}(\mathcal{A}_{\mathbf{P}} \otimes \mathcal{B}_{\mathbf{P}}) \to \mathrm{L}(\mathcal{Z}_{\mathbf{P}})$. Then, by the perfect translator property, $T_{\mathbf{S}\leftrightarrow\mathbf{P}} \rightleftharpoons B'_{\mathbf{P}}$ is exactly some deviating Bob in $\mathbf{S}$, $B'_{\mathbf{S}}$, which induces a map $\Psi_\delta$. Also, the translator $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$ does not access, write to, nor become entangled with any quantum registers; it is entirely classical, and its input-output functionality is only to receive $\delta$ and then to interact classically with $A_{\mathbf{S}}$ and $B_{\mathbf{P}}$. We may therefore say further that $T_{\mathbf{S}\leftrightarrow\mathbf{P}} \rightleftharpoons B'_{\mathbf{P}}$ is exactly some $B'_{\mathbf{S}}$ whose map $\Psi_\delta$ is from and to the same quantum input and output spaces (respectively) as $B'_{\mathbf{P}}$, i.e., we have $\Psi_\delta : \mathrm{L}(\mathcal{A}_{\mathbf{P}} \otimes \mathcal{B}_{\mathbf{P}}) \to \mathrm{L}(\mathcal{Z}_{\mathbf{P}})$, and moreover

$$||\Phi_\delta - \Psi_\delta||_\diamond = 0 \tag{7.3.1}$$

since the channel induced by $B'_{\mathbf{S}}$ considered as $T_{\mathbf{S}\leftrightarrow\mathbf{P}} \rightleftharpoons B'_{\mathbf{P}}$ is induced by $B'_{\mathbf{P}}$ and not $T_{\mathbf{S}\leftrightarrow\mathbf{P}}$.

Then, by the assumption that $\mathbf{S}$ is secure, there exists a simulator $\mathscr{S}[B'_{\mathbf{S}}]$ inducing the channel $\Sigma_\delta : \mathrm{L}(\mathcal{A}_{\mathbf{P}} \otimes \mathcal{B}_{\mathbf{P}}) \to \mathrm{L}(\mathcal{Z}_{\mathbf{P}})$ (where these input and output spaces follow from the preceding observation) such that

$$||\Psi_\delta - \Sigma_\delta||_\diamond = 0. \tag{7.3.2}$$

From eq. (7.3.1) and eq. (7.3.2), we get that

$$||\Phi_\delta - \Sigma_\delta||_\diamond = 0 \tag{7.3.3}$$

and $\mathscr{S}[B'_{\mathbf{S}}]$ is a simulator for $B'_{\mathbf{P}}$ as well. Therefore $\mathbf{P}$ is $\epsilon$-private with $\epsilon = 0$. ∎

## 7.4 Security of S

Next, we analyze protocol $\mathbf{S}$ with the aim of showing it is secure.

**Lemma 7.4.1.** Alice's classical transmissions in $\mathbf{S}$ have the form $s_i = d_i \oplus f_i$, where $f_i$ represents (as a shorthand notation) the outcome of a Boolean function $f_i : \{0,1\}^* \to \{0,1\}$ with arguments

$$f_i(x, d_1, ..., d_{i-1}, c_1, ..., c_{i-1}, m_1, ...m_{i-1})$$

for $1 < i \leq t$, and with $f_1 = f_1(x)$. Moreover, each function $f_i$ is determined by $\delta$ only.

**Proof.** It follows from the detailed discussion of protocol $\mathbf{P}$ in section 5.2 that $s_i = d_i \oplus a$, where $a$ represents the exponent of $X$ in the Pauli $X^a Z^b$ which appeared immediately before the gadget and was propagated through it; we identify $a$ with $f_i$. The arguments of $f_i$ are correctly identified since $a$ is the result of a previous Pauli propagation; considering each Pauli propagation rule shows that the outcomes depend at most on $x$ and some $d$, $c$, $m$ values of index less than the gadget in question ($a$ is from the Pauli $X^a Z^b$ which appears *before* the gadget). Finally, it is clear from the propagation procedure in protocol $\mathbf{P}$ that each function $f_i$ is determined by $\delta$ only,

since, given $\delta$, that procedure can be run on generic $x$, $d$, $c$, and $m$ values (for instance as we do in the invariance check described in section 6.4) to generate a description of each $f_i$ sequentially. ∎

**Lemma 7.4.2.** Consider a run of protocol **S** on an arbitrary quantum circuit $C$ with $t$ T gates. At every round $k$ of the interaction at which Alice sends a message, for $0 \leq k \leq t$, Bob's view of Alice's total transmission up to and including round $k$ is represented by the density matrix $\rho_k = \frac{1}{2^{t+k}} \cdot I_{2^{t+k}}$, i.e., the $(t+k)$-qubit maximally mixed state.

**Proof.** We proceed by induction on $k$. We say Alice sends the randomly rotated magic states (step 1 of **S** as described in algorithm 3) at round $k = 0$, and at round $i$ for $1 \leq i \leq t$ sends the classical message $s_i \in \{0, 1\}$. Throughout the proof, we use the notation $\tau_\theta = \frac{1}{2}(|0\rangle + e^{i\theta}|1\rangle)(\langle 0| + e^{-i\theta}\langle 1|)$, for $\theta \in \mathbb{R}$.

Base case ($k = 0$):

At round 0, Alice sends $\rho_0 = \bigotimes_{i=1}^{t} |R_i\rangle\langle R_i|$ to Bob, where $|R_i\rangle = S^{d_i} Z^{c_i} |T\rangle$. Since $c_i$ and $d_i$ are chosen independently uniformly at random from $\{0, 1\}$, Bob's view of $\rho_0$ is represented by

$$\begin{aligned}\rho_0 &= (\frac{1}{4}\tau_{\pi/4} + \frac{1}{4}\tau_{3\pi/4} + \frac{1}{4}\tau_{5\pi/4} + \frac{1}{4}\tau_{7\pi/4})^{\otimes t} \\ &= (\frac{1}{2}I_2)^{\otimes t} \\ &= \frac{1}{2^t} I_{2t}\end{aligned}$$

as required.

Induction step:

We will regard the transmission of a classical bit $a \in \{0, 1\}$ as the transmission of the corresponding computational basis state $|a\rangle\langle a|$. So, at step $k+1$, Alice sends $|s_{k+1}\rangle\langle s_{k+1}|$, and by lemma 7.4.1, this can be written as

$$|s_{k+1}\rangle\langle s_{k+1}| = |d_{k+1} \oplus f_{k+1}\rangle\langle d_{k+1} \oplus f_{k+1}|.$$

We assume the claim holds for round $k$, and show it holds also for the total transmission at round $k+1$.

Note that it is not sufficient at this point to argue that, since $\rho_k$ is maximally mixed by the inductive hypothesis; and, since $d_{k+1}$ is uniformly random from Bob's view, the new classical transmission is therefore equal to $\frac{1}{2} \cdot I_2$; then, the new total transmission $\rho_{k+1}$ is a product of maximally mixed states and is thus maximally mixed. This is insufficient because, in the new transmission, $d_{k+1}$ and $f_{k+1}$ are, at least *a priori*, correlated to previous transmissions. In particular, $d_{k+1}$ appears in the expression of $|R_{k+1}\rangle\langle R_{k+1}|$ (and nowhere else, since all previous classical transmissions depend on $d_i$ values of lower index, by lemma 7.4.1); $f_{k+1}$ depends (in general) on $x$ and on $d_i, c_i, m_i$ values of index $i \leq k$ (again using lemma 7.4.1).

We analyze instead the total transmission at round $k+1$ as follows, isolating $|R_{k+1}\rangle\langle R_{k+1}|$ and the new classical transmission at first:

$$\begin{aligned}
\rho_{k+1} = &\left(|R_1\rangle\langle R_1| \otimes ... \otimes |R_k\rangle\langle R_k|\right) \otimes |R_{k+1}\rangle\langle R_{k+1}| \otimes \\
&\left(|R_{k+2}\rangle\langle R_{k+2}| \otimes ... \otimes |R_t\rangle\langle R_t|\right) \otimes \\
&|s_1\rangle\langle s_1| \otimes ... \otimes |s_k\rangle\langle s_k| \otimes |d_{k+1} \oplus f_{k+1}\rangle\langle d_{k+1} \oplus f_{k+1}| \\
&= \left(|R_1\rangle\langle R_1| \otimes ... \otimes |R_k\rangle\langle R_k|\right) \otimes S^{d_{k+1}} Z^{c_{k+1}} |T\rangle\langle T| (S^{d_{k+1}} Z^{c_{k+1}})^\dagger \otimes \\
&\left(|R_{k+2}\rangle\langle R_{k+2}| \otimes ... \otimes |R_t\rangle\langle R_t|\right) \otimes \\
&|s_1\rangle\langle s_1| \otimes ... \otimes |s_k\rangle\langle s_k| \otimes |d_{k+1} \oplus f_{k+1}\rangle\langle d_{k+1} \oplus f_{k+1}| .
\end{aligned}$$

$c_{k+1}$ and $d_{k+1}$ appear only in the subsystems indexed $k+1$ and $t+k+1$, i.e., those holding $|R_{k+1}\rangle\langle R_{k+1}|$ and $|d_{k+1} \oplus f_{k+1}\rangle\langle d_{k+1} \oplus f_{k+1}|$, respectively; nothing else in $\rho_{k+1}$ depends on them. Moreover, $f_{k+1}$ does not depend on either $c_{k+1}$ or $d_{k+1}$. (These observations again use lemma 7.4.1.) We will isolate subsystems $k+1$ and $t+k+1$ (we can imagine we trace out all others, although we will consider them again later), denoting this state as $\sigma_{k+1,t+k+1}$, and divide Bob's view of $\sigma_{k+1,t+k+1}$ into four cases, according to the values of $c_{k+1}$ and $d_{k+1}$:

1. If $d_{k+1} = 0$, $c_{k+1} = 0$, then $\sigma_{k+1,t+k+1} = \tau_{\pi/4} \otimes |f_{k+1}\rangle\langle f_{k+1}|$.
2. If $d_{k+1} = 0$, $c_{k+1} = 1$, then $\sigma_{k+1,t+k+1} = \tau_{5\pi/4} \otimes |f_{k+1}\rangle\langle f_{k+1}|$.

3. If $d_{k+1} = 1$, $c_{k+1} = 0$, then $\sigma_{k+1,t+k+1} = \tau_{3\pi/4} \otimes |\overline{f_{k+1}}\rangle \langle\overline{f_{k+1}}|$.

4. If $d_{k+1} = 1$, $c_{k+1} = 1$, then $\sigma_{k+1,t+k+1} = \tau_{7\pi/4} \otimes |\overline{f_{k+1}}\rangle \langle\overline{f_{k+1}}|$.

Since $c_{k+1}$ and $d_{k+1}$ are chosen uniformly at random by Alice, we can represent Bob's view of this state as an equal-probability mixture of these four cases. Bob's view of $\sigma_{k+1,t+k+1}$ is therefore represented by the density matrix

$$\begin{aligned}
\sigma_{k+1,t+k+1} = \frac{1}{4}\big(&\tau_{\pi/4} \otimes |f_{k+1}\rangle \langle f_{k+1}| + \tau_{5\pi/4} \otimes |f_{k+1}\rangle \langle f_{k+1}| + \\
&\tau_{3\pi/4} \otimes |\overline{f_{k+1}}\rangle \langle\overline{f_{k+1}}| + \tau_{7\pi/4} \otimes |\overline{f_{k+1}}\rangle \langle\overline{f_{k+1}}| \big) \\
&= \frac{1}{4}\Big( |f_{k+1}\rangle \langle f_{k+1}| \otimes \big(\tau_{\pi/4} + \tau_{5\pi/4}\big) + |\overline{f_{k+1}}\rangle \langle\overline{f_{k+1}}| \otimes \big(\tau_{3\pi/4} + \tau_{37pi/4}\big)\Big) \\
&= \frac{1}{4}\Big( |f_{k+1}\rangle \langle f_{k+1}| \otimes I_2 + |\overline{f_{k+1}}\rangle \langle\overline{f_{k+1}}| \otimes I_2\Big) \\
&= \frac{1}{4}\big(I_2 \otimes I_2\big) \\
&= \frac{1}{4} I_4.
\end{aligned}$$

Crucially, this holds independently of the distribution (actual or in Bob's view) on $f_{k+1}$. Since, as remarked before, $c_{k+1}$ and $d_{k+1}$ appear nowhere else in $\rho_{k+1}$, we may now use the inductive hypothesis to get that this same density matrix holds for each $\sigma_{i,t+i}$ for $1 \leq i \leq k$, and therefore Bob's view of Alice's total transmission at round $k+1$ is

$$\begin{aligned}
\rho_{k+1} &= (\frac{1}{4} I_4)^{k+1} \otimes \frac{1}{2^{t-(k+1)}} I_{2^{t-(k+1)}} \\
&= \frac{1}{2^{t-k-1+2k+2}} \cdot I_{2^{t-k-1+2k+2}} \\
&= \frac{1}{2^{t+k+1}} \cdot I_{2^{t+k+1}}
\end{aligned}$$

as required.

By induction, the claim holds for all $k$, $1 \leq k \leq t$. Since the circuit $C$ and parameter $t$ were arbitrary, the claim holds for all runs of **S** on all quantum circuits. ∎

**Lemma 7.4.3.** $\mathbf{S}$ is $\epsilon$-private with $\epsilon = 0$.

**Proof.** Let $B'_{\mathbf{S}}$ be a deviating Bob in $\mathbf{S}$, inducing a channel $\Phi_\delta : \mathrm{L}(\mathcal{A}_{\mathbf{S}} \otimes \mathcal{B}_{\mathbf{S}}) \to \mathrm{L}(\mathcal{Z}_{\mathbf{S}})$. For any $\delta$ describing $C$ with $t$ magic states, there exists a simulator $\mathscr{S}[B'_{\mathbf{S}}]$ which, at each round $k$ for $0 \leq k \leq t$, generates the state $\rho_k = \frac{1}{2^{t+k}} \cdot I_{2^{t+k}}$, then runs the round-$k$ operation of $B'_{\mathbf{S}}$ on $\rho_k$. The channel $\Sigma_\delta : \mathrm{L}(\mathcal{A}_{\mathbf{S}} \otimes \mathcal{B}_{\mathbf{S}}) \to \mathrm{L}(\mathcal{Z}_{\mathbf{S}})$ induced by this simulator is identical to the channel induced by the deviating server, $\Phi_\delta$, by lemma 7.4.2. So $||\Phi_\delta - \Sigma_\delta||_\diamond = 0$. Therefore $\mathbf{S}$ is $\epsilon$-private with $\epsilon = 0$. ∎

**Theorem 7.4.4.** $\mathbf{P}$ is $\epsilon$-private with $\epsilon = 0$.

**Proof.** Follows from lemma 7.3.1 and lemma 7.4.3. ∎

## 7.5 Security of Qubit Virtualization

We now sketch a proof of security of the input-concealed qubit virtualization procedure, described in section 5.4. We note that expanding this sketch into a full proof would proceed straightforwardly, following the proofs of sections 7.3 and 7.4. The procedure consists of a sequence of runs of $\mathbf{P}$ on circuits $C_i$, for $1 \leq i \leq \chi$, each executed on the same secret input $x$. As noted in section 5.4, the differences in circuit structure between the circuits $C_i$ and circuits we typically consider in input-concealed PBC are inconsequential for security, since the initial $k$ teleportation gadgets admit Pauli-to-Pauli propagations. Therefore $\mathbf{P}$ run on each $C_i$ is $\epsilon$-private with $\epsilon = 0$.

The security of the entire procedure follows closely. Since $\mathbf{P}$ runs on each $C_i$ with $\epsilon$-privacy for $\epsilon = 0$, for any deviating Bob $B_i$ inducing a channel $\Phi_i$ on an execution of $C_i$, there exists a simulator $\mathscr{S}_i[B_i]$ inducing a channel $\Sigma_i$ such that $||\Phi_i - \Sigma_i||_\diamond = 0$. Then any deviating Bob on the entire procedure induces a channel which is a concatenation of $\Phi_i$ channels, for some possibly-deviating Bob $B_i$ for each $i$, which may be simulated by concatenating the channels $\Sigma_i$ for each $i$. This implies the $\epsilon$-privacy of input-concealed qubit virtualization, again with $\epsilon = 0$.

# Chapter 8

# Conclusion

In this thesis, we have presented a novel SDQC protocol (algorithm 2) in the PBC model and proved its security (theorem 7.4.4 and proof). Our protocol offers a significant reduction in server resource requirements compared to existing protocols, showing a path to SDQC on feasible, near-term quantum hardware. We achieve this while matching or slightly improving on the best-known client and communication resources. In all, our work pushes the quantum resource requirements of SDQC close to an apparent minimum; short of finding a classical-client method, it is unclear where further SDQC resource efficiencies could come from.

Nevertheless, the work in this thesis suggests many interesting open questions. We end by listing a few of them:

- **Circuit-hiding in PBC.** A natural next direction involves upgrading the security aim of our protocol from input-hiding to circuit-hiding in an intrinsic way. We believe this to be possible, although a finer understanding of the structure of PBC may be necessary to achieve it.

- **Verification.** As discussed in section 3.4, a correctness-verification functionality is often layered onto SDQC protocols, making this another natural next step. Verification is usually accomplished by adding "test" qubits to an SDQC protocol, which the server cannot distinguish from the others, but for which the client can detect incorrect computation. Adding test qubits to our input-concealed PBC protocol via an expanded magic register seems like a promising approach.

- **Quantum input hiding.** Looking further afield, hiding a *quantum* input state $|\psi\rangle$ to a computation via PBC would be an interesting extension. This would require an understanding of how regular PBC can be adapted to accept quantum inputs, which would be of independent interest.

- **X-invariant circuits.** A deeper understanding of the class $\mathsf{XINV}$ (definition 6.4.1), and its relations to better-studied classes of circuits, is highly interesting. For instance, $\mathsf{XINV}$ closely resembles $\mathsf{IQP}$, a class of circuits introduced in [SB09] and shown to have (likely) quantum advantage in [BJS11], but we cannot quite say $\mathsf{IQP} \subseteq \mathsf{XINV}$ because of issues relating to gate set. More generally, what circuit classes admit classical-client SDQC beyond those which are known to be classically simulable?

# Bibliography


[ABE10] D. Aharonov, M. Ben-Or, and E. Eban. Interactive proofs for quantum computations. In *Innovations in Computer Science—ICS 2010*, pages 453–469, 2010.
Online: https://conferences.iiis.tsingua.edu.cn/ICS2010/content/papers/35.html.

[ACGK19] S. Aaronson, A. Cojocaru, A. Gheorghiu, and E. Kashefi. Complexity-theoretic limitations on blind delegated quantum computation. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, pages 6:1–6:13, 2019.
DOI: 10.4230/LIPIcs.ICALP.2019.6.

[AFK87] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. In *STOC '87: Proceedings of the ninetheenth ACM symposium on Theory of computing*, pages 195–203, 1987.
DOI: 10.1145/28395.28417.

[AS06] P. Arrighi and L. Salvail. Blind quantum computation. *International Journal of Quantum Information*, 4(5): 883–898, 2006.
DOI: 10.1142/S0219749906002171.

[BBC+19] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard. Simulation of quantum circuits by low-rank stabilizer decompositions. *Quantum*, 3: 181, 2019.
DOI: 10.22331/q-2019-09-02-181.

[BCC+20] C. Badertscher, A. Cojocaru, L. Colisson, E. Kashefi, D. Leichtle, A. Mantri, and P. Wallden. Security limitations of classical-client dele-

gated quantum computing. In *Advances in Cryptology — ASIACRYPT 2020*, volume 2, pages 667–696, 2020.
DOI: 10.1007/978-3-030-64834-3_23.

[BFK09] A. Broadbent, J. Fitzsimons, and E. Kashefi. Universal blind quantum computation. In *50th Annual Symposium on Foundations of Computer Science (FOCS 2009)*, pages 517–526, 2009.
DOI: 10.1109/FOCS.2009.36.

[BFKW13] S. Barz, J. F. Fitzsimons, E. Kashefi, and P. Walther. Experimental verification of quantum computation. *Nature Physics*, 9(11): 727–731, 2013.
DOI: 10.1038/nphys2763.

[BJ15] A. Broadbent and S. Jeffery. Quantum homomorphic encryption for circuits of low T-gate complexity. In *Advances in Cryptology — CRYPTO 2015*, volume 2, pages 609–629, 2015.
DOI: 10.1007/978-3-662-48000-7_30.

[BJS11] M. J. Bremner, R. Jozsa, and D. J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 467(2126): 459–472, 2011.
DOI: 10.1098/rspa.2010.0301.

[BKB+12] S. Barz, E. Kashefi, A. Broadbent, J. F. Fitzsimons, A. Zeilinger, and P. Walther. Demonstration of blind quantum computing. *Science*, 335(6066): 303–308, 2012.
DOI: 10.1126/science.1214707.

[Bro15] A. Broadbent. Delegating private quantum computations. *Canadian Journal of Physics*, 93(9): 941–946, 2015.
DOI: 10.1139/cjp-2015-0030.

[BS16] A. Broadbent and C. Schaffner. Quantum cryptography beyond quantum key distribution. *Designs, Codes and Cryptography*, 78(1): 351–382, 2016.
DOI: 10.1007/s10623-015-0157-4.

[BSS16] S. Bravyi, G. Smith, and J. A. Smolin. Trading classical and quantum computational resources. *Physical Review X*, 6(2): 021043, 2016.
DOI: 10.1103/PhysRevX.6.021043.

[Chi05] A. M. Childs. Secure assisted quantum computation. *Quantum Information & Computation*, 5(6): 456–466, 2005.
DOI: 10.26421/QIC5.6-4.

[CZW+22] Y. Cao, Y. Zhao, Q. Wang, J. Zhang, S. X. Ng, and L. Hanzo. The evolution of quantum key distribution networks: On the road to the Qinternet. *IEEE Communications Surveys & Tutorials*, 24(2): 839–894, 2022.
DOI: 10.1109/COMST.2022.3144219.

[DK25] E. Davies and A. Kay. Communication-optimal blind quantum protocols. E-print arXiv:2510.07112 [quant-ph], 2025.
arXiv: 2510.07112.

[DNM+24] P. Drmota, D. P. Nadlinger, D. Main, B. C. Nichol, E. M. Ainley, D. Leichtle, A. Mantri, E. Kashefi, R. Srinivas, G. Araneda, C. J. Ballance, and D. M. Lucas. Verifiable blind quantum computing with trapped ions and single photons. *Physical Review Letters*, 132(15): 150604, 2024.
DOI: 10.1103/PhysRevLett.132.150604.

[FBS+14] K. A. G. Fisher, A. Broadbent, L. K. Shalm, Z. Yan, J. Lavoie, R. Prevedel, T. Jennewein, and K. J. Resch. Quantum computing on encrypted data. *Nature Communications*, 5: 3074, 2014.
DOI: 10.1038/ncomms4074.

[Fei85] J. Feigenbaum. Encrypting problem instances: Or ... can you take advantage of someone without having to trust him? In *Advances in Cryptology — CRYPTO '85*, pages 477–488, 1985.
DOI: 10.5555/18262.25420.

[Fit17] J. Fitzsimons. Private quantum computation: an introduction to blind quantum computing and related protocols. *npj Quantum Information*, 3: 23, 2017.
DOI: 10.1038/s41534-017-0025-3.

[GMMR13] V. Giovannetti, L. Maccone, T. Morimae, and T. G. Rudolph. Efficient universal blind quantum computation. *Physical Review Letters*, 111(23): 230501, 2013.
DOI: 10.1103/PhysRevLett.111.230501.

[Got98] D. Gottesman. The Heisenberg representation of quantum computers. In *22nd International Colloquium on Group Theoretical Methods in Physics—GROUP 22*, pages 32–43, 1998.

[HCWY25] Z. He, A. Cowtan, D. J. Williamson, and T. J. Yoder. Extractors: QLDPC architectures for efficient Pauli-based computation. E-print arXiv:2503.10390 [quant-ph], 2025.
arXiv: 2503.10390.

[JV14] R. Jozsa and M. Van den Nest. Classical simulation complexity of extended clifford circuits. *Quantum Information & Computation*, 14(7&8): 633–648, 2014.
DOI: 10.5555/2638682.2638689.

[Kay18] A. Kay. Tutorial on the Quantikz package. E-print arXiv:1809.03842 [quant-ph], 2018.
arXiv: 1809.03842.

[KL21] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. CRC Press, 3rd edition, 2021.

[KLMO26] E. Kashefi, D. Leichtle, L. Music, and H. Ollivier. Verification of quantum computations without trusted preparations or measurements. *Advanced Quantum Technologies*, 9(5), 2026.
DOI: doi.org/10.1002/qute.202501018.

[Leu04] D. W. Leung. Quantum computation by measurements. *International Journal of Quantum Information*, 2(1): 33–43, 2004.
DOI: 10.1142/S0219749904000055.

[Mah18] U. Mahadev. Classical homomorphic encryption for quantum circuits. In *2018 59th Annual IEEE Symposium on Foundations of Computer Science*

*(FOCS 2018)*, pages 332–338, 2018.
DOI: 10.1109/FOCS.2018.00039.

[MF13] T. Morimae and K. Fujii. Blind quantum computation protocol in which alice only makes measurements. *Physical Review A*, 87(5): 050301, 2013.
DOI: 10.1103/PhysRevA.87.050301.

[MK19] T. Morimae and T. Koshiba. Impossibility of perfectly-secure one-round delegated quantum computing for classical client. *Quantum Information & Computation*, 19(3–4): 214–221, 2019.
DOI: 10.5555/3370245.3370247.

[MNTT19] T. Morimae, H. Nishimura, Y. Takeuchi, and S. Tani. Impossibility of blind quantum sampling for classical client. *Quantum Information & Computation*, 19(9–10): 793–806, 2019.
DOI: 10.5555/3370263.3370266.

[MPF13] A. Mantri, C. A. Pérez-Delgado, and J. F. Fitzsimons. Optimal blind quantum computation. *Physical Review Letters*, 111(23): 230502, 2013.
DOI: 10.1103/PhysRevLett.111.230502.

[NC10] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition, 2010.
DOI: 10.1017/CBO9780511976667.

[Ozo08] M. Ozols. Clifford group, 2008.
Online: http://home.lu.lv/~sd20008/papers/essays/Clifford%20group%20[paper].pdf.

[Per05] S. Perdrix. State transfer instead of teleportation in measurement-based quantum computation. *International Journal of Quantum Information*, 3(1): 219–223, 2005.
DOI: 10.1142/S0219749905000785.

[Per23] F. C. R. Peres. Pauli-based model of quantum computation with higher-dimensional systems. *Physical Review A*, 108(3): 032606, 2023.
DOI: 10.1103/PhysRevA.108.032606.

[Per24] F. C. R. Peres. *Optimizing Models of Hybrid Quantum/Classical Computation*. PhD thesis, University of Porto, 2024.
Online: https://repositorio-aberto.up.pt/bitstream/10216/161857/2/690986.pdf.

[PG23] F. C. R. Peres and E. F. Galvão. Quantum circuit compilation and hybrid computation using Pauli-based computation. *Quantum*, 7: 1126, 2023.
DOI: 10.22331/q-2023-10-03-1126.

[PG25] F. C. R. Peres and E. F. Galvão. Reducing depth and measurement weights in Pauli-based computation. *Physical Review A*, 112(6): 062604, 2025.
DOI: 10.1103/d3x5-cgky.

[RB01] R. Raussendorf and H. J. Briegel. A one-way quantum computer. *Physical Review Letters*, 86(22): 5188, 2001.
DOI: 10.1103/PhysRevLett.86.5188.

[SB09] D. Shepherd and M. J. Bremner. Temporally unstructured quantum computation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 465(2105): 1413–1439, 2009.
DOI: 10.1098/rspa.2008.0443.

[Sip12] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition, 2012.

[Wri24] J. Wright. Quantum coding theory, lecture 23: The Clifford group, 2024.
Online: https://people.eecs.berkeley.edu/~jswright/quantumcodingtheory24/scribe%20notes/lecture23.pdf.

[WSS+25] Y.-C. Wei, P.-J. Stas, A. Suleymanzade, G. Baranes, F. Machado, Y. Q. Huan, C. M. Knaut, S. W. Ding, M. Merz, E. N. Knall, U. Yazlar, M. Sirotin, I. W. Wang, B. Machielse, S. F. Yelin, J. Borregaard, H. Park, M. Lončar, and M. D. Lukin. Universal distributed blind quantum computing with solid-state qubits. *Science*, 388(6746): 509–513, 2025.
DOI: 10.1126/science.adu6894.

[YJS19] M. Yoganathan, R. Jozsa, and S. Strelchuk. Quantum advantage of unitary Clifford circuits with magic state inputs. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 475(2225): 20180427, 2019.
DOI: 10.1098/rspa.2018.0427.