Quantum Homomorphic Encryption: A Survey

Ben Hamlin[1], Steven Libby[2], and Nate Launchbury[3]

[1]Portland State University hamlinb@pdx.edu
[2]Portland State University slibby@pdx.edu
[3]Portland State University njl2@pdx.edu

June, 2017

Abstract

Fully homomorphic encryption in the quantum setting is an enormously appealing cryptographic goal, not least because access to quantum computers will not likely be widespread in the next few decades. Being able to outsource quantum computations to an untrusted party while maintaining the privacy of the data would allow individuals to utilize the power of quantum algorithms securely. But achieving a fully homomorphic quantum encryption scheme is no small task. We explain the concept of homomorphic encryption in general and survey the current best efforts towards a fully homomorphic scheme in the quantum setting.

1 Introduction

Quantum computing is a promising new avenue for algorithm development. However, quantum computers will not be available to the general public for the foreseeable future. One potential solution allows users to access a quantum server to run their computations in a "quantum cloud", but this comes with a cost. To run a computation on a server, the user must be willing to share their data. This loss of privacy is unacceptable for many users.

A solution to the corresponding problem in the classical setting exists in the form of fully homomorphic encryption (FHE). The idea comes from a simple question: can we encrypt data and run arbitrary computations on it, without ever decrypting the data? Surprisingly, Rivest et al. showed that this was possible [RAD78]. If we want to perform computations in the quantum cloud, we would like the privacy provided by fully homomorphic encryption but the ability to pass around quantum data. This is the premise of quantum fully homomorphic encryption (QFHE).

In this work, we survey the state of the art in QFHE research: In Section 2, we briefly describe the state of the art in classical FHE. After familiarizing the reader with a few background concepts in Section 3, we describe a few of the quantum homomorphic primitives that occur in the literature in Section 4 and discuss how they fail to completely solve the QFHE problem. In Section 5, we describe the state of the art in quantum homomorphic encryption. Finally, in Section 6, we conclude and mention some topics in this area we feel bear further exploration.

2 Classical Fully Homomorphic Encryption

Classical fully homomorphic encryption is built on the concept of a homomorphism from the field of abstract algebra. The central idea is that a homomorphism is a structure preserving function. For example, the determinant from linear algebra is a homomorphism from the space of $N \times N$ matrices to real numbers. The structure that is preserved is multiplication on matrices. That is, for any two matrices A and B, $|AB| = |A| \cdot |B|$.

There are many examples of homomorphisms in various areas of math including exponentiation, $e^{x+y} = e^x e^y$, logarithms $\log(x+y) = \log(x)\log(y)$, and complex norm $\|xy\| = \|x\|\|y\|$. In general if G is a group with binary operation $*$ and H is a group with binary operation $\cdot$, then a function $f : G \to H$ is a homomorphism if for all $a, b \in G$ $f(a * b) = f(a) \cdot f(b)$. There are many interesting and useful results about homomorphisms in their own right, but in this paper we are concerned with the ability to construct a single homomorphism that maps operations on plain text into operations on encrypted text.

At first the idea of applying homomorphisms to encryption might seem like an odd connection to make, but this arose very naturally in the history of cryptography. Rivest at el. proposed the idea shortly after publishing their now famous RSA encryption scheme. The scheme itself was integral the development of homomorphic encryption because it encrypts data by translating the plain text into elements of a group. Specifically the group $\mathbb{Z}_n$ where n is the product of two large primes. The idea is that if $de = n$ then $de \equiv 1 \pmod{n}$, so for any message m, $(m^e)^d = m^{ed} = m^n \equiv m \pmod{n}$. There is a natural homomorphism here in that $(mn)^x = m^x n^x$. While this is an interesting idea, the textbook RSA algorithm presented here isn't secure. A more compelling example is the ElGamal Scheme.

2.1 Homomorphic properties of ElGamal encryption

ElGamal comes from the Diffie-Hellman key exchange protocol. For two parties, Alice and Bob, to send encrypted data Alice first picks a group G, a generator g and computes q, the order of g. Finally she choses a random x in $\{1 \ldots q-1\}$ and computes $h = g^x$, publishing $\langle G, g, q, h \rangle$ Here h is the public key, and x is the private key. Bob then choses a random y and encrypts the message m as $(g^y, m \cdot g^{xy})$. Since Alice knows x and g^y it's easy enough for here to compute $(g^y)^x = g^{xy}$, and from there, compute the inverse. For security Bob will have to chose a new y for every message he encrypts, otherwise an adversary could use multiple messages to learn g^{xy}.

ElGamal shares a homomorphic property with RSA: If Bob encrypts any two messages m_1 and m_2 the ciphertexts will be of the form $(g^{y_1}, m_1 \cdot g^{xy_1})$ and $(g^{y_2}, m_2 \cdot g^{xy_2})$. We can multiply the messages together to get $(g^{y_1} g^{y_2}, m_1 \cdot m_2 \cdot g^{xy_1} g^{xy_2}) = (g^{y_1+y_2}, m_1 \cdot m_2 \cdot g^{x(y_1+y_2)})$.

While we can multiply arbitrary group elements in the ElGamal scheme, that doesn't allow us to compute on encrypted data like we wanted. In order to allow more arbitrary computations we need to develop some theory. An encryption scheme is a tuple of $(n, Gen, Enc_{pk}, Dec_{sk})$. Here n is the security parameter, Gen is the key generation function which produces pk and sk, and $Dec_{sk}(Enc_{pk}(m)) = m$ for all messages m. An encryption scheme is homomorphic if there is an evaluation function $Eval_{evk}$ that can compute any function on encrypted data. That is, $Dec_{sk}(Eval_{evk}(f, (Enc_{pk}(m)))) = f(m)$. In practice this is usually weakened to

$$P[Dec_{sk}(Eval_{evk}(f, (Enc_{pk}(m)))) \neq f(m)] \leq \eta(n)$$

for some negligible function η.

2.2 Compactness

This definition leads to a very simple implementation of homomorphic encryption. Take any secure encryption scheme $(Gen, Enc_{pk}, Dec_{sk})$. Then we can define a new scheme by appending any function we want to compute to the end of the current encrypted text. This scheme, defined in [BJ15] is called the trivial homomorphic encryption scheme. It can evaluate any function, because it just applies that function to to plain text, and it's clearly correct. While it satisfies our definition, it violates the spirit of having a third party do the computation. To prevent this we also require that the encryption scheme satisfy a compactness property.

Roughly speaking the compactness property states that decrypting a ciphertext should not depend on any functions applied. To formalize this we need a concrete representation for functions, and a description for function complexity. We'll define a function as a circuit consisting of *and*, *or* and *not* gates. An elementary theorem from the theory of computation shows that all Turing computable functions are efficiently computable by circuits. This gives us confidence in the utility of this model. We then define the depth of a circuit to be the longest path from any input to any output of the circuit. There have been several definitions of compactness that have been shown to be equivalent, but one of the simplest is the following.

Definition 1. A homomorphic encryption scheme is compact if for all messages m and for all circuits c $|Eval_{evk}(Enc_{pk}(m))| \leq p(n)$ for some polynomial p. This has the side effect that all encrypted messages must be below a fixed length.

2.3 Fully Homomorphic Encryption

At this point we've seen two different implementations of homomorphic encryption, with neither one really satisfying what we were hoping for. The ElGammal scheme was secure and compact, but was only homomorphic over multiplication of group elements. This is *$\mathcal{F}$-homomorphic* encryption, where $\mathcal{F}$ is multiplication in the group. The trivial scheme, on the other hand was homomorphic over every possible function, but was not compact. This is *somewhat homomorphic*. Another possibility is *leveled homomorphic* where a function is homomorphic for all circuits of a constant depth. This was first achieved by Gentry et al. in 2008 [GPV08]. Finally if an encryption scheme is homomorphic over every possible function, and is compact it is *fully homomorphic*. Building on his previous work, Gentry showed how to construct a fully homomorphic encryption scheme in 2009 [Gen09].

Gentry's 2009 work is based on the idea of bootstrapable encryption. The idea is to construct a somewhat homomorphic encryption scheme that can evaluate it's own decryption algorithm. At first this would seem counter productive because the entire point of a homomorphic encryption scheme is to evaluate functions on encrypted data. However, if we encrypt the data twice, then we can use our homomorphic evaluation to remove the inner encryption. Now if we can evaluate a single *nand* gate along with the decryption algorithm, then we can construct any computable function. This idea of doubly encryption a message in order to remove the inner encryption is called *recypryption*. There are a few things to note about this algorithm. The first is that to evaluate the decryption algorithm we need to know the secret key. Normally this would break security, however since we encrypted the message twice, we can just encrypt the secret key, and then when we homomorphicly evaluate *Dec* we will have the secret key. The second important point is that every circuit can be decomposed into nand gates, and evaluated stepwise. The details of construction a scheme are based in the theory of ideal lattice problems, and are outside the scope of this paper.

Gentry $Gen' = Gen$
$Enc'_{pk} = Enc_{pk} \circ Enc_{pk}$

$Eval_{evk}(f)' = Eval_k \circ \ldots \circ Eval_0$
where $(N_1 \ldots N_k) = NANDS(f)$
$Eval_i = Enc_{pk} \circ Eval_{evk}(N_i \circ Dec_{sk})$

$Dec'_{sk} = Dec_{sk} \circ Dec_{sk}$

Figure 1: **Gentry's Fully Homomorphic Encryption:**

3 Preliminaries

Central to the discussion of homomorphic encryption in the quantum setting is the gate set called the Clifford group:

Definition 2. The *Clifford group* is the set of circuits generated by the following unitary gates:

$$\mathcal{C} = \{X, Z, P, H, CNOT\}$$

where,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Definition 3. The T-gate, also known as the phase-shift gate R_θ with $\theta = \pi/4$, is the following unitary:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

Lemma 4. $\mathcal{C} \cup \{T\}$ *forms a universal gate set.*

This relies on the fact that $T \notin \mathcal{C}$ and then follows directly from Theorem 6.5 of Nebe et al. [NRS01].

Another important concept is what we choose, in this paper to call a "pad".

Definition 5. A *pad* is a scheme that encrypts a quantum state ρ by applying a sequence of quantum gates $\mathcal{P} = \langle U_1, \ldots, U_n \rangle$ based on a sequence of uniformly random key bits $\langle k_1, \ldots, k_n \rangle$:

$$Enc_{k_1,\ldots,k_n}(\rho) = U_1^{k_1} \cdots U_n^{k_n} \rho U_1^{\dagger(k_1)} \cdots U_n^{\dagger(k_n)}$$

The decryption algorithm varies somewhat for different pad schemes, but inevitably, it

1. requres the same key bits, $\langle k_1, \ldots, k_n \rangle$, and
2. involves applying a pad with a possibly different sequence of gates.

To anyone who doesn't know the key, encrypting a quantum state ρ with a pad can be viewed as a function ϵ, which applies one of 2^n gates to ρ with equal probability:

$$\epsilon(\rho) = \sum_{k_1,\ldots,k_n \in \{0,1\}} \frac{1}{2^n} U_1^{k_1} \cdots U_n^{k_n} \rho U_1^{\dagger(k_1)} \cdots U_n^{\dagger(k_n)}$$

This leads us to the security definition commonly used in the literature:

Definition 6. We call a scheme *information-theoretically secure* (*IT-secure*) if, to any party who doesn't know the key, encrypting yields

$$\epsilon(\rho) = \frac{1}{2}\mathbb{1}$$

4 Quantum Homomorphic Primitives

Several authors have devoted themselves to the task of deriving encryption primitives that

1. are information-theoretically secure, and
2. allow some set of quantum gates to be homomorphically applied to the encrypted plaintext.

Liang [Lia14] and Tan et al. [TKO$^+$16] are two prominent examples.

In the literature we reviewed, these primitives universally took the form of pads, as defined in Section 3. In this section, we discuss the general form these schemes take and describe one of them, the *Clifford scheme* (CL), upon which much of the state of the art is based. In the remainder of this paper we also use the term *quantum one-time pad* (QOTP) to refer to the pad used in the CL scheme.

4.1 Pads in General

The earliest research on quantum pads was performed by Ambainis et al. [AMTdW00]. The question they asked was information-theoretic: Is it possible to encrypt a quantum state with IT security using some number of classical bits, and if so what is the minimum number of classical bits needed? This question mirrors the one Claude Shannon answered in 1945 with respect to classical information [Sha45].

Ambainis et al. answer the first question in the affirmative, using the definion of IT security we provide in Section 3. For the second question, they arrive at the following theorem:

Theorem 7. *In order to encrypt a single qubit with IT security, absent any shared entanglement, two bits are both necessary and sufficient.*

Once again, this mirrors Shannon's result that one key bit is both necessary and sufficient to encrypt a bit of classical information with IT security. Ambainis et al. go on to propose the general framework for encrypting quantum states using pads, which we describe in Section 3.

Although Ambainis et al. did not set out to find a homomorphic scheme, pads frequently have this property. Trivially, if a gate V commutes with all of $U_1, \ldots, U_n$, then applying V to a ciphertext yields,

$$\begin{aligned} V\ Enc(\rho)\ V^\dagger &= VU_1 \cdots U_n \rho U_n^\dagger \cdots U_1^\dagger V \\ &= U_1 \cdots U_n V \rho V U_n^\dagger \cdots U_1^\dagger \end{aligned}$$

Hence this scheme is naturally at least $\{V\}$-homoporphic. The same holds if V anticommutes with all of $U_1, \ldots, U_n$, and n is even. A slightly less trivial possibility is that either the operation V

[Lia14] or the key $k_1, \dots, k_n$ [BJ15] must be modified to apply V to the encrypted plaintext, i.e. for some gate update function $g(V)$ and key-update functions $\{f_i^V(\cdot)\}$,

$$g(V)\ Enc_{k_1,\dots,k_n}(\rho) = Enc_{f_1(k_1),\dots,f_n(k_n)}(V\rho)$$

Since IT-secure pads require very long keys (two bits per qubit), they are frequently coupled with some classical public-key scheme $PubK$ in the following way:

0. Alice generates the public and private keys and (pk, sk) for $PubK$.
1. Alice distributes pk.
2. Bob encrypts a quantum message ρ as a pair $(\sigma, \kappa) = (Enc_{k_1,\dots,k_n}(\rho), Enc_{pk}^{PubK}(k_1, \dots, k_n))$
3. To decrypt, Alice first derives $k_1, \dots, k_n = Dec_{sk}^{PubK}(\kappa)$, then derives $\rho = Dec_{k_1,\dots,k_n}(\sigma)$.

If at least one of the key-update functions $\{f_i(\cdot)\}$ is not the identity, then $PubK$ must be classically fully homomorphic. Also, note that the scheme is only secure as $PubK$. Hence usually such schemes only acheive computational security.

Despite the work of several authors over nearly twenty years, no fully homomorphic pad scheme has been found (with the exception of Liang [Lia14], who requires that the evaluator be able to see the pad key). Hence much of the existing work on QFHE has been devoted to working around the inability of existing pad schemes to support universal gate sets. In the next section we describe one such $\mathcal{F}$-homomorphic scheme.

4.2 CL scheme

Several authors have proposed concrete pads, each with their own strengths and limitations. Nonetheless, the Clifford (CL) scheme from Broadbent and Jeffery [BJ15] is the most commonly used in the literature, so that is the one we describe here. It works as follows:

Definition 8. CL scheme

- $Enc_{k_1,k_2}^{CL}(\rho) = X^{k_1} Z^{k_2} \rho Z^{k_2} X^{k_1}$
- $Dec_{k_1,k_2}^{CL}(\rho) = X^{k_1} Z^{k_2} \rho Z^{k_2} X^{k_1}$

Note that X and Z anticommute, so this is correct.

The CL scheme is appealingly simple. In addition, it achieves IT security using the Ambainis-bound minimum two classical bits. It is easy to verify that

$$\begin{aligned} Enc(\rho) &= \frac{1}{4} \sum_{k_1,k_2 \in \{0,1\}} X^{k_1} Z^{k_2} \rho Z^{k_2} X^{k_1} \\ &= \frac{1}{4}(XZ\rho ZX + X\rho X + Z\rho Z + \rho) \\ &= \frac{1}{2}\mathbb{1} \end{aligned}$$

Furthermore, CL is $\mathcal{C}$-homomorphic (where $\mathcal{C}$ is the Clifford group), since

$$(\forall C \in \mathcal{C})\ C X_1^k Z_2^k = X^{f_1^C(k_1)} Z^{f_2^C(k_2)} C$$

Since the key update rules are not necessarily the identity, a classical FHE scheme must be used to encrypt k_1, k_2. To complete the definition of CL as a QHE scheme, we have

$$Eval^{CL}((k_1, k_2), C, \sigma) = ((f_1^C(k_1), f_2^C(k_2)), C\sigma C^\dagger)$$

This is compact, since the cipher text does not expand. Broadbent and Jeffery [BJ15] provide the necessary key-update functions for all $C \in \mathcal{C}$.

The CL scheme is tantalizingly close to a complete QFHE scheme. Unfortunately, as we state in Section 3, $\mathcal{C}$ is not a universal gate set. If we attempt to evaluate the non-Clifford gate T, for example, we get the following:

$$TX^{k_1}Z^{k_2} = X^{k_1}Z^{k_1 \oplus k_2}P^{k_1}T$$

The factor of P_{k_1} in the above cannot be represented as a QOTP ciphertext using any key-update rule, providing $k_1 \neq 0$. Hence decrypting $\sigma = Eval((k_1, k_2), T, Enc_{k_1,k_2}(\rho))$ yields $P^{k_1}\rho$. And since k_1 depends on the remainder of the evaluated circuit, this P factor is unpredictable. In consequence, most of the current QFHE literature focuses on correcting this P error. We describe a few schemes for doing so in the next section.

5 Leveled QFHE Schemes

In this section, we consider three quantum leveled homomorphic encryption schemes which represent the current best effort towards a fully homomorphic encryption scheme in the quantum setting. Each of these schemes has the ability to evaluate an arbitrary gate from a universal gate set but the circuits that can be evaluated are limited by the number of T-gates present. In all cases, the scheme uses the Clifford gate set with the addition of the T-gate for universality. Again, the challenge is to evaluate T-gates in such a way that the conditional phase error can be corrected without leaking the secret key.

5.1 EPR scheme

The EPR scheme of Broadbent and Jeffery [BJ15] leverages the use of entangled qubits during evaluation to delay phase error correction until decryption. At a high level, when Bob evaluates any gate from the Clifford group on the ciphertext $|c\rangle$ he can simply correct the error in the manner of the CL scheme from section [reference]. When he evaluates a T-gate, he uses an entangled qubit to create a new variable, v, which the correction of the phase error is contingent upon. However, measuring this variable accurately requires part of the secret key. Bob then simply creates v_i when he evaluates the i-th T-gate and sends both the result of the evaluations on $|c\rangle$ and this set of variables $\{v_i\}$ to Alice. Using the secret key, Alice is able to accurately measure each v_i and then correct the phase error herself before decrypting.

5.1.1 EPR - overview

The EPR scheme is based on the CL scheme from Section 4.2. Steps that appear only in EPR are marked with a + sign.

0. **Key Generation:**
    - Alice generates classical keys.
1. **Encryption:**
    - Alice applies quantum one-time pad to $|p\rangle$ to get $|c\rangle$.
    - Alice encrypts pad keys using a classical fully homomorphic encryption scheme.
2. **Evaluation:**
    - When Bob applies $G \in \mathcal{C}$ he computes f_G homomorphically on the pad keys.

+ When Bob applies T, he uses $|\Phi^+\rangle$ to entangle the conditional phase error with a new variable v_i.

3. **Decryption:**
   - Alice homomorphically decrypts the pad keys.
   + Alice performs measurements to recover each v_i and then corrects any phase errors that have occurred.
   - Alice uses these updated pad keys to decrypt and get $C|p\rangle$ where C denotes the circuit Bob evaluated on $|c\rangle$.

5.1.2 EPR - limitations

Since the work of phase error correction is done by Alice, the EPR scheme quickly loses the property of *compactness*. This is obvious since the amount of work Alice must do during decryption is not at all independent of the circuit used during evaluation. Indeed, if R is the number of T-gates evaluated on $|c\rangle$, then the complexity of decryption is R^2 [BJ15].

Recall that the trivial fully homomorphic encryption scheme has a decryption procedure which scales linearly with the number of gates being evaluated. Thus, the EPR scheme is preferable to the trivial scheme whenever the square-root of the number of T-gates is less than the total number of gates in the circuit.

5.2 AUX scheme

The second quantum homomorphic encryption scheme proposed by Broadbent and Jeffery attempts to solve the problem of correcting the conditional phase error without increasing the work done during decryption. The AUX scheme instead uses auxiliary states that are generated as part of the evaluation key which can be used during evaluation to correct phase error. These states encode part of the encryption key but must be combined to create a single useful state. Alice generates these auxiliary states during key generation and includes them as part of the evaluation key so that Bob has access to them. This allows all of the error correction to be done by Bob rather than by Alice at the time of decryption, which means that the AUX scheme satisfies the property of compactness for any circuit.

5.2.1 AUX - overview

The AUX scheme is based on the CL scheme from section 4.2. Steps that appear only in AUX are marked with a + sign.

0. **Key Generation:**
   - Alice generates classical keys.
   + Alice generates auxiliary states as part of the evaluation key.
1. **Encryption:**
   - Alice applies quantum one-time pad to $|p\rangle$ to get $|c\rangle$.
   - Alice encrypts pad keys using a classical fully homomorphic encryption scheme.
2. **Evaluation:**
   - When Bob applies $G \in \mathcal{C}$ he computes f_G homomorphically on the pad keys.
   + When Bob applies T, he combines some of the auxiliary states in the evaluation key to correct the error.
3. **Decryption:**
   - Alice homomorphically decrypts the pad keys.

- Alice uses these updated pad keys to decrypt and get $C\,|p\rangle$ where C denotes the circuit Bob evaluated on $|c\rangle$.

5.2.2 AUX - limitations

The combining of auxiliary states for error correction also creates "cross terms", similar to how a polynomial multiplication like $(a+b)^2$ will create not just a^2+b^2 but also the $2ab$ "cross term". Unfortunately, these cross terms quickly become prohibitive in the AUX scheme since they add additional noise which must be corrected for future T-gate evaluations. This means that the size of the evaluation key grows extremely rapidly as the number of T-gate evaluations increases. In fact, the evaluation key size is a polynomial with degree exponential in the T-depth [BJ15]. This means that for AUX to be efficient, the number of T-gates evaluated needs to be constant. But, perhaps more importantly, since the evaluation key size is fixed during key generation, Bob has no flexibility in the number of T-gates he evaluates on $|c\rangle$ for a given security parameter.

Furthermore, since the auxiliary states are created based on the encryption key, the AUX scheme is necessarily symmetric-key, otherwise generating these states would be impossible.

5.3 TP scheme

A recent quantum homomorphic encryption scheme by Dulek, Schaffner, and Speelman [DSS16] utilizes teleportation through error correcting gadgets to allow evaluation of T-gates. The scheme, called TP, is also based on the CL scheme of Broadbent and Jeffery [BJ15]. After key generation, Alice creates an arbitrary number of gadgets designed to correct the phase error incurred by evaluation of a T-gate. She then sends these gadgets to Bob along with the ciphertext, $|c\rangle$. When Bob evaluates a T-gate, he teleports the ciphertext through one of the gadgets. There are several paths for the teleported qubit to take through each gadget and they are designed in such a way that if (and only if) $k_1 = 1$, the path will include a P^{-1} gate. This will correct the phase error exactly in the cases where it occurs. Of course, Alice must be the one to prepare these gadgets as they require knowledge of the secret keys of the quantum one-time pad.

5.3.1 TP - overview

The TP scheme is based on the CL scheme from section 4.2. Steps that appear only in TP are marked with a + sign.

0. **Key Generation:**
   - Alice generates classical keys
   - + Alice generates gadgets
1. **Encryption:**
   - Alice applies quantum one-time pad to $|p\rangle$ to get $|c\rangle$
   - Alice encrypts pad keys using a classical fully homomorphic encryption scheme
2. **Evaluation:**
   - When Bob applies $G \in \mathcal{C}$ he computes f_G homomorphically on the pad keys.
   - + When Bob applies T he teleports the ciphertext (after applying the T-gate) through one of the gadgets to correct the error.
3. **Decryption:**
   - Alice homomorphically decrypts the pad keys.
   - Alice uses these updated pad keys to decrypt and get $C\,|p\rangle$ where C denotes the circuit Bob evaluated on $|c\rangle$.

5.3.2 TP - limitations

Since Bob must use (and destroy) a gadget each time he applies a T-gate, the circuits that can be evaluated using the TP scheme are limited by the number of gadgets Alice is able to prepare. Additionally, this work cannot be offloaded to Bob since creation of the gadgets requires knowledge of the secret pad keys. This results in a scheme which can evaluate any circuit with polynomial size.

5.4 Overview

All three of the these quantum homomorphic encryption schemes (EPR, AUX, and TP) are based on the CL scheme from section 4.2. The differences in the schemes is how they attempt to correct the phase error which may be introduced by the evaluation of a T-gate.

EPR delays the error correction using entanglement at the cost of increasing the complexity of decryption. This of course will quickly violate the property of compactness as the number of T-gates evaluated increases.

The AUX scheme instead uses auxiliary states to encode portions of phase error correction based on the secret pad keys. These auxiliary states must be combined when a T-gate is applied to be useful in correcting any error that is introduced. Unfortunately, this combining process is prohibitively expensive and the scheme becomes unrealistic as the number of T-gates increases. Moreover, since the auxiliary states are created during key generation, the number of T-gates Bob can apply is fixed.

At first glance, the TP scheme seems very similar to the AUX scheme: an error correcting resource is created during key generation, Bob consumes that resource during evaluation of T-gates, and the number of T-gates Bob can apply is based on this resource. However, the TP scheme is an improvement over AUX in a number of ways. First, the number of gadgets Alice can create is not based on the evaluation key size and so can be decoupled from the security parameter. Second, these gadgets can be created at any time so if further communication between Alice and Bob is allowed [1], Alice can create more gadgets for Bob to use. Finally, since the cross terms that appear as by-products of the combining procedure in AUX exponentially influence the polynomial complexity of the AUX scheme, TP is far more efficient for a comparable number of T-gate evaluations.

All of these schemes are leveled homomorphic encryption schemes since they allow the evaluation of an arbitrary quantum gate but not an arbitrary quantum circuit. If R is the number of T-gates in a circuit, then EPR is not compact since decryption complexity scales with R^2. AUX can only efficiently evaluate circuits where R is constant and TP can only efficiently evaluate circuits where R is polynomially bounded. A fully homomorphic encryption scheme would be able to evaluate an arbitrary quantum circuit, not only circuits contingent on some bounding of R.

6 Conclusion and Future Work

In this paper, we surveyed several QHE schemes. We presented the CL scheme, a $\mathcal{C}$-homomorphic scheme. Since the CL scheme cannot handle non-Clifford gates without causing miltiplicative P-gate error on the processed plaintext, many authors have attempted to use error correction to extend the CL scheme, with significant success. Although none have yet achieved a true fully

[1] usually, more than one round of communication between Alice and Bob is disallowed by homomorphic encryption schemes, but there are real-world examples where continued communication is natural (including the motivating example of a cloud computing service).

homomorphic scheme, they are a large step toward achieving one. As Broadbent and Jeffery [BJ15] mention,

> "Historically, such results appeared as precursors to the breakthrough result establishing classical fully homomorphic encryption"

We began this project by asking the question, "Is there some pad, defined as in 3 that supports homomorphically applying a universal gate set?" Despite trying a number of possibilities for the sequence of gates $\mathcal{P}$, we did not succeed in finding one.

On the other hand, a number of other authors [Lia14, TKO$^+$16, BJ15] have attempted a similar task, and although none were able to derive such a pad, we also found no proof in the literature that such a pad could not exist. With this in mind, we believe that searching for such a pad could still advance the state of the art of QFHE.

References

[AMTdW00] Andris Ambainis, Michele Mosca, Alain Tapp, and Ronald de Wolf. Private quantum channels. 16th Annual Symposium on Foundations of Computer Science, January 2000.

[BJ15] Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low t-gate complexity, 2015.

[DSS16] Yfke Dulek, Christian Schaffner, and Florian Speelman. Quantum homomorphic encryption for polynomially-sized circuits. https://arxiv.org/pdf/1603.09717, June 2016.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, 2008.

[Lia14] Min Liang. Symmetric quantum fully homomorphic encryption with perfect security. https://arxiv.org/abs/1304.5087, August 2014.

[NRS01] Gabriele Nebe, Eric M Rains, and Neil JA Sloane. The invariants of the clifford groups, 2001.

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[Sha45] Claude Shannon. A mathematical theory of cryptography, 1945.

[TKO$^+$16] Si-Hui Tan, Joshua A. Kettlewell, Yingkai Ouyang, Lin Chen, and Joseph F. Fitzsimons. A quantum approach to homomorphic encryption. http://dx.doi.org/10.1038/srep33467, September 2016.