Deniable Encryption in a Quantum World

Andrea Coladangelo
UC Berkeley & Simons Institute for the Theory of Computing
Berkeley, California, USA

Shafi Goldwasser
UC Berkeley & Simons Institute for the Theory of Computing
Berkeley, California, USA

Umesh Vazirani
UC Berkeley & Simons Institute for the Theory of Computing
Berkeley, California, USA

ABSTRACT

(Sender-)Deniable encryption provides a very strong privacy guarantee: a sender who is coerced by an attacker into "opening" their ciphertext after-the-fact is able to generate "fake" local random choices that are consistent with any plaintext of their choice. The only known *fully-efficient* constructions of public-key deniable encryption rely on indistinguishability obfuscation (iO) (which currently can only be based on sub-exponential hardness assumptions).

In this work, we study (sender-)deniable encryption in a setting where the encryption procedure is a quantum algorithm, but the ciphertext is classical. First, we propose a quantum analog of the classical definition in this setting. We give a fully efficient construction satisfying this definition, assuming the quantum hardness of the Learning with Errors (LWE) problem.

Second, we show that quantum computation unlocks a fundamentally stronger form of deniable encryption, which we call *perfect unexplainability*. The primitive at the heart of unexplainability is a quantum computation for which there is provably no efficient way, such as exhibiting the "history of the computation," to establish that the output was indeed the result of the computation. We give a construction which is secure in the *random oracle model*, assuming the quantum hardness of LWE. Crucially, this notion implies a form of protection against coercion "before-the-fact", a property that is impossible to achieve classically.

CCS CONCEPTS

• **Theory of computation** → *Quantum complexity theory*; *Cryptographic protocols.*

KEYWORDS

quantum cryptography, deniable encryption

ACM Reference Format:
Andrea Coladangelo, Shafi Goldwasser, and Umesh Vazirani. 2022. Deniable Encryption in a Quantum World. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520019

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06…$15.00
https://doi.org/10.1145/3519935.3520019

1 INTRODUCTION

This work is motivated by the following overarching question: do *local* quantum computations alone provide an advantage in cryptography? In other words, is there any quantum advantage in the setting where honest parties can perform *local* quantum computations but are restricted to sending and storing *classical* information?

While many examples of quantum advantage are known which leverage quantum communication (or shared entanglement), e.g. key distribution [6, 16] and oblivious transfer [4, 14], or that rely on storing quantum information, e.g. quantum money [22], copy-protection [1], and various other unclonable primitives [5, 10, 13], quantum advantage that results purely from local quantum computations is essentially restricted to certifiable randomness [7].

Here, we study the notion of *deniable encryption* in the setting where honest parties have access to a quantum computer, but no quantum communication. Deniable encryption was introduced by Canetti et al. [11]. In a deniable encryption scheme, honest parties are able to generate a "fake" secret key (in the case of *receiver* deniability) and "fake" randomness (in the case of *sender* deniability) to claim that the public communication is consistent with any plaintext of their choice. This allows them to preserve the privacy of the true plaintext even if an adversary coerces them after-the-fact into disclosing their private information.

We restrict our attention to non-interactive public-key schemes, and we focus on *sender-deniable* encryption, namely the setting in which we only protect the sender against coercion by an attacker.

A bit more formally, a public-key encryption scheme is sender-deniable if there exists a "faking" algorithm that takes as input a pair of messages m_0, m_1, an encryption $c = \mathsf{Enc}(m_0, r)$, and the randomness r used in the encryption, and outputs some "fake" randomness r', which should look consistent with a genuine encryption of m_1. More precisely, the view of an attacker who receives m_1, c, and the fake randomness r', should be computationally indistinguishable from the view of an attacker who receives m_1, along with a genuine encryption of m_1, and the true randomness used.

In their original paper [11], Canetti et al. gave a construction of a deniable encryption scheme where the real and fake views are computationally indistinguishable up to inverse polynomial distinguishing advantage - we will call the distinguishing advantage the "faking probability". More generally, they show that the size of the ciphertext grows with the inverse of the faking probability.

In a breakthrough work [20], Sahai and Waters gave the first construction with negligible faking probability and compact (i.e polynomial-size) ciphertexts under the assumption that secure indistinguishability obfuscation (iO) exists. Recently, a breakthrough of Jain, Lin and Sahai [18], showed how to construct iO from a few concrete computational hardness assumptions. However, the latter assumptions require *sub-exponential* hardness of the underlying

problems. Moreover, the constructions from [18] are not quantum-secure due to their use of bilinear maps. Other constructions that are based on "LWE-like" assumptions [8, 15, 17, 21] also require sub-exponential hardness for a fundamental reason: they rely on compilers from weaker objects (like functional encryption) to iO, which seem to inherently introduce sub-exponential hardness. Finally, a recent work [3] achieves deniable encryption with compact ciphertexts and negligible faking probability based on polynomial-time hardness of LWE. However, the running time of the encryption algorithm is non-polynomial, growing with the inverse of the faking probability. In sum, no constructions of deniable encryption from polynomial-time hardness assumptions achieve negligible faking probability and are *fully efficient* - namely, achieve compact ciphertexts and polynomial-time encryption and decryption. Thus, the following is an outstanding open question:

Can fully efficient deniable encryption be based on polynomial-time hardness assumptions?

In this work, we provide an affirmative answer to this question in the setting where encryption is a *quantum* algorithm, but the ciphertext is *classical*. For simplicity, we will refer to this as the *quantum setting*. Our constructions illustrate that quantum computation provides a fundamentally new kind of advantage for deniability: while classical deniability can only handle coercion *after-the-fact* (i.e. the attacker approaches the sender *after* she has sent her ciphertext), quantum computation can also protect against coercion *before-the-fact* (i.e. the attacker approaches the sender *before* she sends her ciphertext).

1.1 Our Contributions

We propose two notions of deniability in the quantum setting, namely *quantum deniability* and *unexplainability*, and we provide a construction for each. In this section, we give an overview of the two notions and the corresponding constructions. Our second construction satisfies a strong form of unexplainability that we call *perfect* unexplainability. The latter notion implies a form of protection against coercion before-the-fact.

Quantum Deniability. Recall that the classical definition of deniability is centered around the notion of *input randomness*. Unfortunately, the latter is not well-defined for a quantum algorithm, since randomness can be intrinsically the result of a measurement, and depends on the basis in which the measurement is carried out. Instead, the correct analog of randomness is the *unmeasured quantum state* of the encryption algorithm.

Now, the strongest definition of quantum deniability that one could imagine would require that revealing the final quantum state of the encryption algorithm (right before measurement) to an attacker *does not* compromise the secrecy of the plaintext. However, this definition is clearly impossible to achieve, since one can rewind the computation to recover the input state, and hence the plaintext. Instead, in the case where the plaintext is a single bit, our definition of quantum deniability allows the sender to measure a *single qubit* of the final quantum state of the encryption algorithm (corresponding to one bit of the ciphertext), and requires that revealing *all the other qubits* to the attacker *does not* compromise the secrecy of the plaintext. One might expect that such a definition is still impossible to achieve. However, remarkably, there is a construction that achieves it.

The setting is somewhat subtle, so let us be a little more precise. The encryption of a single bit b can be thought of as a pair (z, aux), where $z \in \{0,1\}$ and $\mathsf{aux} \in \{0,1\}^n$ for some n (recall that the ciphertext is entirely classical). The recipient (who has the secret key) can efficiently recover b from z using aux[1], while an adversary who only sees (z, aux) has negligible advantage in guessing b. For deniability, we require that if the sender's quantum encryption algorithm were to only measure the qubit register corresponding to z, then the remaining final quantum state of the encryption algorithm (which includes aux in superposition) still does not reveal any non-negligible information about b to the adversary. Note that, in practice, encryption requires the honest sender to measure *both* z and aux. The definition of quantum deniability is designed to show that *even if* the sender were to preserve a coherent state over as many qubits as possible – i.e. all but the single qubit corresponding to z – this still does not reveal any information about the plaintext b to the attacker.

Theorem 1. *There exists a deniable encryption scheme (in the sense of the above definition), assuming the quantum hardness of LWE.*

The scheme that makes the theorem true is simple, and is inspired by the use of trapdoor claw-free functions in [7, 19].

In essence, in our encryption scheme, the public key is a choice of trapdoor claw-free function pair $(f_{k,0}, f_{k,1})$. The encryption of a single bit b is a pair (z, aux) where $\mathsf{aux} = (d, y)$ such that $z = b \oplus d \cdot (x_0 \oplus x_1)$, where x_0, x_1 are the pre-images of y. In other words, the bit $d \cdot (x_0 \oplus x_1)$ is used as a *one-time pad*. CPA security then follows straightforwardly from the fact that $d \cdot (x_0 \oplus x_1)$ is a hardcore bit, i.e. that it is computationally hard to guess $d \cdot (x_0 \oplus x_1)$ for uniformly random d and y.

To prove deniability, we establish that the final quantum state of all registers conditioned on the classical outcome z still does not leak any information about b.

The intuition for why quantum information is well-suited for deniable encryption is the following: the honest execution of the quantum encryption algorithm does not have any input randomness, instead the algorithm must be precisely such that all registers delicately interfere in the right way to result in a valid equation (where the randomness in the equation comes from the final measurement), leaving no other trace of the plaintext in the leftover workspace.

We point out that the scheme that makes Theorem 1 true can be instantiated with any 2-to-1 (noisy) trapdoor claw-free function pair, satisfying the *injective invariance* property from [19], which in turn can be constructed from the hardness of LWE.

Remark: One might wonder whether it is possible to formulate a definition of quantum deniability more closely aligned with the classical definition. There are two basic obstacles to this:

- Not only can quantum algorithms sample randomness by making measurements, but they can also "cover their tracks"

[1] In the scheme that we propose, z is special in the sense that toggling it (and leaving *aux* unchanged) toggles the plaintext, while *aux* plays the role of auxillary information that specifies the mapping between z and the plaintext.

by repeatedly performing measurements in an incompatible basis. As an example, consider a possibly more natural definition of deniability in which the attacker is only revealed the remaining quantum state after *both* z and aux are measured (i.e. whatever remains after the quantum encryption algorithm has produced the entire classical ciphertext). Unfortunately, such a definition would be trivial to satisfy by slightly modifying any classical encryption algorithm, and running it on a quantum computer: one can simply prescribe that the honest quantum encryption algorithm runs the classical encryption algorithm, using a register to sample the randomness, then measures this register in an incompatible basis, and finally appends the classical outcome of this measurement to the ciphertext as part of aux (decryption then ignores this part of the ciphertext). What this has accomplished is that the leftover quantum state is now essentially garbage, and trivially does not reveal anything to the attacker.

- The notion of a *transcript* of a computation is not well-defined for a quantum algorithm, in the sense that observing the computation at any step in general disturbs the computation.

It is somewhat fortunate that a stronger definition of quantum deniability, which does not appeal to the inner workings of the encryption algorithm, is achieved by a concrete scheme.

Unexplainable encryption. The second notion that we propose, *unexplainability*, takes a different viewpoint on the concept of input randomness: in the classical setting, input randomness is simply a proxy for a "proof" that the ciphertext is a valid encryption of a certain plaintext. The notion of unexplainability formalizes this notion of a proof, and achieves a single definition that provides a natural common view of deniability in the classical and quantum setting.

In an unexplainable encryption scheme, it is simply impossible, except with negligible probability, for an efficient sender to "prove" after-the-fact that they encrypted a particular plaintext (thus an attacker has simply no reason to bother coercing a sender into opening their ciphertext in the first place). The crux in formalizing this definition is to formalize what it means for a sender to "prove" that they encrypted a particular plaintext. We argue that the appropriate notion of a proof is akin to that of an *argument*.

Definition 1 (Explainability (informal)). *A public-key encryption scheme is* explainable *if there exists an efficient verification procedure* Verify*, taking as input a tuple of public key, ciphertext, message, and alleged proof* (pk, c, m, w)*, such that* $\mathsf{Verify}(\mathsf{pk}, c, m, w) = 0$ *if the triple* (pk, c, m) *is inconsistent. Moreover,* Verify *should satisfy the following:*

- Completeness*: there exists an efficient procedure that, on input* m, pk*, generates* c, w *such that* $\mathsf{Verify}(\mathsf{pk}, c, m, w)$ *outputs* 1 *with high probability.*
- Soundness*: no efficient procedure, on input* pk*,* m, m' *with* $m \neq m'$*, can generate* c, w, w' *such that* $\mathsf{Verify}(\mathsf{pk}, c, m, w)$ *and* $\mathsf{Verify}(\mathsf{pk}, c, m', w')$ *both output* 1*, except with negligible probability.*

By contrapositive, a scheme is *unexplainable* if, for any such Verify, one of completeness or soundness fails. Thus, the sense in which a classical deniable scheme is unexplainable is that a sender can never convincingly "prove" to an attacker that they encrypted a particular plaintext (even if the sender wishes to do so honestly): this is precisely because, by definition of deniability, it is always possible for a sender to efficiently generate randomness consistent with *any* plaintext of their choice. In other words, a classical deniable encryption scheme is unexplainable because the soundness condition above fails when one takes Verify to be the procedure that interprets w as the randomness used in the encryption, and simply checks that $c = \mathsf{Enc}(\mathsf{pk}, m; w)$.

The notion of unexplainability can be thought of as a rephrasing of deniability from a different perspective. In fact, we show that the appropriate variation on the definition of unexplainability is *equivalent* to deniability (we refer the reader to the full version for details). However, unlike deniability, the notion of unexplainability has a very natural extension to the quantum setting, as it is not centered around randomness, but rather, more abstractly, around the notion of a proof: without modifications, the definition above makes sense even in the quantum setting (where one may choose to allow the "proof" w to be a quantum state).

Notice that for an encryption scheme with *perfect* decryption (i.e. one in which there is a unique plaintext consistent with a given ciphertext), the notion of a proof described above coincides with that of an NP (or QMA) proof: since we require that $\mathsf{Verify}(\mathsf{pk}, c, m, w) = 0$ if the triple (pk, c, m) is inconsistent, then Verify is precisely an NP-relation (or QMA-relation) for the language

$$L = \{x = (m, c, \mathsf{pk}) : c \text{ is a valid encryption of } m \text{ under } \mathsf{pk}\}.$$

The definition then has the additional requirement of *completeness*, which asks that there is an efficient procedure that takes as input pk, m, and generates valid c, w.

Notice that, for encryption schemes with perfect decryption, unexplainability is impossible to achieve classically. First notice, that for a scheme with perfect decryption, the *soundness* condition in Definition 1 is trivially satisfied. Thus, the scheme is unexplainable if and only if for all Verify (as in Definition 1) the *completeness* condition fails. However, the latter is contradicted by the following. Consider the NP-relation $\mathsf{Verify}((m, c, \mathsf{pk}), w)$ where the witness is the randomness used to encrypt, i.e. $\mathsf{Verify}((m, c, \mathsf{pk}), w) = 1$ if $\mathsf{Enc}(\mathsf{pk}, m; w) = c$, and $\mathsf{Verify}((m, c, \mathsf{pk}), w) = 0$ otherwise. Then, simply consider the efficient procedure which encrypts honestly, and outputs the randomness as the witness.

In this work, we show the following.

Theorem 2 (Informal). *There exists an unexplainable public-key encryption scheme with perfect decryption, with security in the quantum random oracle model (QROM), assuming the quantum hardness of LWE.*

The scheme that makes the theorem true is a variation on the previous one, and is inspired by the follow-up work [9] to [7], which makes use of a random oracle.

Again, the public key is a choice of trapdoor claw-free function pair $(f_{k,0}, f_{k,1})$. The encryption of a single bit b is a triple (z, d, y) such that $z = b \oplus d \cdot (x_0 \oplus x_1) \oplus H(x_0) \oplus H(x_1)$, where x_0, x_1 are the pre-images of y. In other words, now the pad is the bit $d \cdot (x_0 \oplus$

$x_1) \oplus H(x_0) \oplus H(x_1)$. The need for a random oracle stems from the need to obtain a more rigid characterization of the structure of algorithms that produce valid encryptions (we discuss at the end why obtaining a scheme from just LWE may be difficult).

We show that our scheme is unexplainable in the strongest sense: it is simply impossible, except with negligible probability, for an efficient quantum algorithm to produce both a valid encryption *and* a proof of its validity. We emphasize that the latter guarantee holds for *any* efficient quantum algorithm that attempts to produce both a valid encryption and a proof, not just for algorithms that run *honest* encryption. Using the terminology introduced earlier, we show that for any verification procedure Verify such that $\mathsf{Verify}(\mathsf{pk}, c, m, w) = 0$ if the triple (pk, c, m) is inconsistent, the *completeness* condition fails. We call this *perfect unexplainability*.

Notice that perfect unexplainability is impossible to achieve classically, since one can always take Verify to be the procedure that interprets w as the randomness in the encryption, and runs encryption forward to check consistency.

We find this behaviour to be quite striking, beyond its implications for deniable encryption, and we believe that it has the potential to find applications in other settings.

As a first example, our encryption scheme provides protection against coercion *before-the-fact*, in the following sense. In the classical world, a coercer who approaches a sender prior to sending an encrypted message can dictate which randomness shall be used by the sender when computing the encryption (and which plaintext shall be encrypted). In this way, the coercer can, at a later stage, check that the ciphertext submitted by the sender corresponds to an encryption of the desired plaintext with the prescribed randomness. This holds true even if the public key (or some other public information to be used in the encryption) is revealed *after* coercion. This type of coercion is a major concern for electronic elections with online encrypted votes. Our unexplainable encryption scheme prevents coercion before-the-fact since the randomness cannot be controlled, even by the sender themselves: it is instead the result of a carefully chosen measurement. Thus, in a scenario where the public key (or some other public information to be used in the encryption) is revealed *after* the coercion stage, a before-the-fact coercer would not be able to succeed. In particular, there is simply no way for an attacker to prescribe to the sender *how* to encrypt in a way that it can later verify, because this would immediately violate perfect unexplainability: it would imply that there exists *some* quantum algorithm that outputs ciphertexts *and* proofs of their validity. We discuss coercion before-the-fact in more detail in Section 5.2. As a further practical motivation, the fact that it is not possible for a sender, even if they wanted to, to prove that their ciphertext is a valid encryption of their plaintext provides a way to protect against "vote-selling".

Lastly, efficiently checkable proofs have been at the center of the stage in complexity and cryptography in the past four decades: proofs with a variety of surprising properties have been discovered and have been the object of intense study (e.g. PCPs, zero-knowledge, succinct). Our work provides a new perspective on the notion of proofs in a quantum world by formalizing the notion of "inability to prove": this is captured by a computational problem that can be solved efficiently, but for which it is impossible to concurrently provide an efficiently checkable proof of correctness. Classically, the transcript of a computation serves as an efficiently checkable proof that the output is correctly reported. At the heart of the new property is the fact that quantum computations, in general, lack the notion of a *transcript* or a "trajectory" of the computation. In this work, we identify a computational problem for which it is *provably intractable* to find a solution while concurrently extracting any meaningful proof that the solution is correct. Using the terminology introduced earlier, a proof is a witness to an appropriate NP (or QMA)-relation Verify.

The analysis in our security proof uses Zhandry's compressed oracle technique [23], and relies on two novel technical contributions.

(i) The first key step in the security proof is establishing that a strategy which produces valid encryptions must be close to a strategy that queries the oracle at a uniform superposition of the two pre-images (in a sense made more precise in the main text). In a bit more detail, let $H : \{0,1\}^n \to \{0,1\}$ be a uniformly random function, and let $x_0, x_1 \in \{0,1\}^n$. We prove a technical lemma that characterizes the structure of strategies that are successful at guessing $H(x_0) \oplus H(x_1)$. We use this lemma to derive a corresponding "rigidity" theorem for strategies that produce valid encryptions (equivalently, valid "equations" in the terminology of [7]). This rigidity theorem may find applications elsewhere, and may be of independent interest.

(ii) The second technical contribution is an *online* extraction argument which allows to extract a claw from any prover that produces a valid equation *and* a proof of its validity, with non-negligible probability. The extraction is *online* in the sense that no rewinding is required.

We point out that the scheme that makes Theorem 5 true can be instantiated with any 2-to-1 (noisy) trapdoor claw-free function pair (the injective invariance property is not needed here). We also remark that for both of our constructions (in Theorems 1 and 5) we do not require an *adaptive* hardcore bit property.

One final remark is in order. Although a perfectly unexplainable encryption scheme based solely on, say hardness of LWE without the use of a random oracles is desirable, we point out that it is unlikely that such a result can be achieved without any additional assumption. The reason is that it would imply a *single-round* message-response proof of quantumness protocol [7]. The following is the protocol:

- The verifier samples $(\mathsf{pk}, \mathsf{sk})$ as in the encryption scheme, together with a message m (say uniformly at random). The verifier sends pk and m to the prover.
- The prover returns an encryption c of m under public key pk.
- The verifier checks that c decrypts to m.

Since perfect unexplainability is impossible to achieve classically, it must be that the encryption algorithm is quantum, and in particular that it cannot be replaced by a classical algorithm (otherwise there would be a way to "explain" by providing the input randomness).

Now, as originally pointed out in [9], a single-round proof of quantumness immediately implies a separation of the sampling classes BPP and BQP. Such a separation does not seem to be implied

by the hardness of LWE, as the current state-of-the-art suggests that LWE is equally intractable for classical and quantum computers.

1.2 Related Work and Concepts

In the classical setting, *receiver-deniable* encryption has also been studied. In the latter, an attacker coerces the *receiver* after-the-fact into revealing their secret key. While non-interactive receiver-deniable encryption is impossible classically, there exists a generic transformation that compiles any sender-deniable encryption scheme into a receiver-deniable one, at the cost of one additional message [11]. While we do not formalize this explicitly, such a transformation also applies to the quantum setting.

In an interactive setting, one can also consider the notion of *bideniable* encryption, i.e. an encryption scheme that is simultaneously sender and receiver-deniable. Classically, this setting has been considered by Canetti, Park and Poburinnaya [12], who show that bideniable encryption can be realized from iO. We leave it as an open question to improve this result (i.e. realize bideniable encryption from weaker assumptions) in the quantum setting using *classical* communication.

Although we do not formalize this, we remark that, if one allows for *quantum* communication, deniable encryption (including bideniable) becomes immediate in the interactive setting (with information-theoretic security!). The reason is that sender and receiver can share an information-theoretically secure key by running a quantum key distribution protocol, and then use this key as a one-time pad. If the attacker approaches the parties *before* the key distribution protocol is completed, then the parties can trivially deny since no information about the message has been revealed yet. If the attacker approaches the parties *after* the key distribution protocol is completed, then the key is information-theoretically hidden from the attacker, and the parties can trivially find a key that is consistent with any message of their choice.

2 TECHNICAL OVERVIEW

We introduced two notions of deniability in the quantum setting along with two constructions. The first construction, which achieves the notion of *quantum deniability* is relatively straightforward to explain in full detail, along with a proof of security, and this is carried out in Section 4. The second construction, which achieves the notion of *unexplainability*, and its proof of security are more involved. In this section, we give an overview of the latter construction, and the proof of security.

2.1 Notation

In this overview, we let $\{(f_{k,0}, f_{k,1})\}_k$ be a family of trapdoor claw-free function pairs. We assume that $f_{k,0}, f_{k,1}$ are injective with identical range, and that they map n-bit strings to m-bit strings. In our actual scheme, we will instead use *noisy* trapdoor claw-free functions, since these can be constructed from LWE. However, this distinction is immaterial for the purpose of this overview.

2.2 An Unexplainable Encryption Scheme

The construction is simple, and is inspired by the "proof of quantumness" construction in [9]. To obtain an encryption scheme, the idea is to use the "hardcore" bit in their construction as a one-time pad.

In more detail, the public key in the encryption scheme is a choice of trapdoor claw-free function k, and the secret key is a trapdoor t_k.

- *Encryption*: To encrypt a bit m, under public key k, Compute a triple (z, d, y), where $d \in \{0,1\}^n$ and $y \in \mathcal{Y}$, and $z = d \cdot (x_0^y \oplus x_1^y) \oplus H(x_0^y) \oplus H(x_1^y)$. This can be done as follows:
  - Create the uniform superposition over $n+1$ qubits
    $$\sum_{b\in\{0,1\},x\in\{0,1\}^n} |b\rangle\,|x\rangle\ .$$
    Then, compute $f_{k,0}$ and $f_{k,1}$ in superposition, controlled on the first qubit. The resulting state is
    $$\sum_{b\in\{0,1\},x\in X} |b\rangle\,|x\rangle\,|f_{k,b}(x)\rangle\ .$$
  - Measure the image register, and let y be the outcome. As a result, the state has collapsed to:
    $$\frac{1}{\sqrt{2}}(|0\rangle\,|x_0^y\rangle + |1\rangle\,|x_1^y\rangle)\,.$$
  - Query the phase oracle for H, to obtain:
    $$\frac{1}{\sqrt{2}}((-1)^{H(x_0)}\,|0\rangle\,|x_0\rangle + (-1)^{H(x_1)}\,|1\rangle\,|x_1\rangle)\,.$$
  - Apply the Hadamard gate to all $n+1$ registers, and measure. Parse the measurement outcome as $z||d$ where $z \in \{0,1\}$ and $d \in \{0,1\}^n$.

  The ciphertext is $c = (m \oplus z, d, y)$.
- *Decryption*: On input $c = (\tilde{z}, d, y)$, use the trapdoor t_k to compute the pre-images x_0^y, x_1^y. Output $\tilde{z} \oplus d \cdot (x_0^y \oplus x_1^y) \oplus H(x_0^y) \oplus H(x_1^y)$.

The actual scheme will be a parallel repetition of this, i.e. the plaintext m is encrypted many times using the single-shot scheme described here. However, for the purpose of this overview, we will just consider the single-shot scheme.

2.3 Security

It is straightforward to see that the scheme satisfies CPA security. This essentially follows from a regular "hardcore bit" property, satisfied by the trapdoor claw-free function family (more details in Section 4.3) In this overview, we focus on outlining how the scheme satisfies (a strong version of) unexplainability.

Our main result is that this scheme has the property that, although it is possible to encrypt, it is not possible to simultaneously produce both a valid encryption of a desired plaintext m *and* a "proof" or a "certificate" w that the ciphertext indeed is an encryption of m. We refer to this as *perfect unexplainability*.

More precisely, we show that, for any efficient algorithm Verify taking as input a tuple (pk, c, m, w), such that $\mathsf{Verify}(\mathsf{pk}, c, m, w) = 0$ if the triple (pk, c, m) is inconsistent, the following holds: for any efficient algorithm P, for any m,

$$\Pr[\mathsf{Verify}(\mathsf{pk}, c, m, w) = 1 : (c, w) \leftarrow P(\mathsf{pk}, m)] = \mathsf{negl}(n)\,. \quad (1)$$

Notice that perfect unexplainability is impossible to achieve with a classical encryption scheme, because one can always have w play the role of the randomness in the encryption, and have Verify be

the algorithm that simply outputs 1 if $\mathsf{Enc}(\mathsf{pk}, m; w) = c$, and 0 otherwise.

A consequence of this observation is that perfect unexplainability is an even stronger property than a "proof of quantumness": if an encryption scheme is unexplainable, then it must be that producing valid encryptions (with high enough probability) is something that only a quantum computer can do (otherwise, there would exist a classical algorithm Enc that encrypts successfully, and, just like before, we can set Verify to be the procedure that interprets w as randomness, runs Enc with this randomness, and checks consistency with the ciphertext).

For simplicity, in the rest of this discussion, we will take the plaintext to be $m = 0$, so that producing a valid ciphertext is equivalent to producing a triple (z, d, y), such that $z = d \cdot (x_0^y \oplus x_1^y) \oplus H(x_0^y) \oplus H(x_1^y)$, which is referred to as a valid "equation" in [7, 9].

So, what is the intuition for why the property above holds for our scheme? At a high level, the intuition is that in order to produce a valid equation, one *has to* query the oracle on a superposition of the two pre-images, and have the two branches interfere in just the right way, so as to produce z, d which satisfy the equation. This delicate process of interference between the two branches essentially requires that nothing is "left behind" in the workspace.

The difficulty with formalizing this intuition is that it assumes a global view of the entire computation. Whereas unlike in the classical setting, we cannot in general define the trajectory which in this case corresponds to the sequence of oracle queries that led to a particular configuration. To do so, we make use of Zhandry's compressed oracle technique, which leverages properties of uniformly random oracles. It turns out that the cryptographic assumptions interact very cleanly with the compressed oracle formalism. This provides a way of carrying key quantities from our classical intuition over into natural formalizations in the quantum context.

This reduction involves two broad steps:

(i) We establish a *rigidity* theorem which formalizes the intuition that an algorithm P which is successful at producing valid equations must query the oracle on a superposition of both pre-images, in a sense that we will make more precise below. Concisely, we appeal to Zhandry's compressed oracle technique for "recording queries" [23], which formalizes the idea that, when the oracle is uniformly random, there is a meaningful way to record the queries made by the algorithm efficiently, in a way that is well-defined and does not disrupt the run of the algorithm. In a compressed oracle simulation, the quantum state of the algorithm at any point is in a superposition over *databases* of queried inputs. What we establish is that, if P produces a valid equation with high probability, then the quantum state of P right before measurement of the equation must be close to a uniform superposition (with the appropriate phase) of two branches: one on which the first pre-image was queried, and one in which the second was queried.

(ii) We leverage (i) to construct an algorithm that extracts a claw. One crucial observation here is that, since Verify never accepts an inconsistent tuple (pk, c, m), then it must be the case that whenever Verify accepts, it *must* itself have queried at a superposition of both pre-images, in the same sense as in the rigidity theorem in point (i). At a high level, the extraction algorithm will eventually be the following:
   - Run P, followed by a measurement to obtain z, d, y, and an appropriate measurement of the database register, hoping to find a pre-image of y;
   - Then, run Verify on the leftover state, and conditioned on "accept", measure the database register, hoping to find the *other* pre-image of y.

   It is not a priori clear that this leads to successfully recovering a claw (in particular the measurement of the database right after running P, may disrupt things in a way that Verify no longer accepts). What we show is that if, to begin with, Verify accepts with high enough probability, then this extraction strategy works.

In the rest of the section, we discuss the elements of the security proof in a bit more detail.

2.3.1 Zhandry's compressed oracle technique. In this subsection, we give an exposition of Zhandry's compressed oracle technique, both because it is a building block in our proof of security, and also to encourage its broader use. A reader who is familiar with the technique should feel free to skip this subsection.

Let $H : \{0,1\}^n \to \{0,1\}$ be a fixed function. For simplicity, in this overview we restrict ourselves to considering boolean functions (since this is also the relevant case for our scheme).

While classically it is always possible to record the queries of the algorithm, in a way that is undetectable to the algorithm itself, this is not possible in general in the quantum case. The issue arises because the quantum algorithm can *query in superposition*. We illustrate this with an example.

Consider an algorithm that prepares the state $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)\,|y\rangle$, and then makes an oracle query to H. The state after the query is:

$$\frac{1}{\sqrt{2}}\,|x_0\rangle\,|y \oplus H(x_0)\rangle + \frac{1}{\sqrt{2}}\,|x_1\rangle\,|y \oplus H(x_1)\rangle \tag{2}$$

Suppose we additionally "record" the query made, i.e. we copy the queried input into a third register. Then the state becomes:

$$\frac{1}{\sqrt{2}}\,|x_0\rangle\,|y \oplus H(x_0)\rangle\,|x_0\rangle + \frac{1}{\sqrt{2}}\,|x_1\rangle\,|y \oplus H(x_1)\rangle\,|x_1\rangle \tag{3}$$

Now, suppose that $H(x_0) = H(x_1)$, then it is easy to see that, in the case where we didn't record queries, the state of the first register after the query is exactly $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)$. On the other hand, if we recorded the query, then the third register is now entangled with the first, and as a result the state of the first register is no longer $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)$ (it is instead a mixed state). Thus, recording queries is not possible in general without disturbing the state of the oracle algorithm.

Does this mean that all hope of recording queries is lost in the quantum setting? It turns out, perhaps surprisingly, that there is a way to record queries when H is a *uniformly random* oracle.

When thinking of an algorithm that queries a uniformly random oracle, it is useful to purify the quantum state of the algorithm via an oracle register (which keeps track of the function that is being queried). An oracle query is then a unitary that acts in the following way on a standard basis element of the query register (where we

omit writing normalizing constants):

$$|x\rangle|y\rangle\sum_H|H\rangle \mapsto \sum_H|x\rangle|y\oplus H(x)\rangle|H\rangle\ .$$

It is well-known that, up to applying a Hadamard gate on the y register before and after a query, this oracle is equivalent to a "phase oracle", which acts in the following way:

$$|x\rangle|y\rangle\sum_H|H\rangle \mapsto \sum_H(-1)^{y\cdot H(x)}|x\rangle|y\rangle|H\rangle \tag{4}$$

Now, to get a better sense of what is happening with each query, let's be more concrete about how we represent H using the qubits in the oracle register.

A natural way to represent H is to use 2^n qubits, with each qubit representing the output of the oracle at one input, where we take the inputs to be ordered lexicographically. In other words, if $|H\rangle = |t\rangle$, where $t \in \{0,1\}^{2^n}$, then this means that $H(x_i) = t_i$, where x_i is the i-th n-bit string in lexicographic order. Using this representation, notice that

$$\frac{1}{\sqrt{2^n}}\sum_H|H\rangle = |+\rangle^{\otimes 2^n}\ .$$

Now, notice that we can write the RHS of (4) as

$$|x\rangle|y\rangle\sum_H(-1)^{y\cdot H(x)}|H\rangle\ ,$$

i.e. we can equivalently think of the phase in a phase oracle query as being applied to the oracle register.

Thus, when a phase oracle query is made on a standard basis vector of the query register $|x\rangle|y\rangle$, all that happens is:

$$\sum_H|H\rangle \mapsto \sum_H(-1)^{y\cdot H(x)}|H\rangle\ .$$

Notice that, using the representation for H that we chose above, the latter transformation is:

- When $y=0$, $|+\rangle^{\otimes 2^n} \mapsto |+\rangle^{\otimes 2^n}$.
- When $y=1$, $|+\rangle^{\otimes 2^n} \mapsto |+\rangle\cdots|+\rangle_{i-1}|-\rangle_i|+\rangle_{i+1}\cdots|+\rangle$, where i is such that x is the i-th string in lexicographic order.

In words, the query does not have any effect when $y=0$, and the query flips the appropriate $|+\rangle$ to a $|-\rangle$ when $y=1$. Then, when we query on a general state $\sum_{x,y}\alpha_{xy}|x\rangle|y\rangle$, the state after the query can be written as:

$$\sum_{x,y}\alpha_{xy}|x\rangle|y\rangle|D_{xy}\rangle\ ,$$

where D_{xy} is the all $|+\rangle$ state, except for a $|-\rangle$ corresponding to x if $y=1$.

The crucial observation now is that all of these branches are *orthogonal*, and thus it makes sense to talk about "the branch on which a particular query was made": the state of the oracle register reveals exactly the query that has been made on that branch. More generally, after q queries, the state will be in a superposition of branches on which at most q of the $|+\rangle$'s have been flipped to $|-\rangle$'s. These locations correspond exactly to the queries that have been made.

Moreover, the good news is that there is a way to keep track of the recorded queries *efficiently*: one does not need to store all of the (exponentially many) $|+\rangle$'s, but it suffices to keep track only of the locations that have flipped to $|-\rangle$ (which is at most q). If we know that the oracle algorithm makes at most q queries, then we need merely $n\cdot q$ qubits to store the points that have been queried. We will refer to the set of queried points as the *database*. Formally, there is a well-defined isometry that maps a state on 2^n qubits where q of them are in the $|-\rangle$ state, and the rest are $|+\rangle$, to a state on $n\cdot q$ qubits, which stores the q points corresponding to the $|-\rangle$'s in lexicographic order.

Let D denote an empty database of queried points. Then a query to a uniformly random oracle can be thought of as acting in the following way:

$$\begin{cases}|x\rangle|y\rangle|D\rangle \mapsto |x\rangle|y\rangle|D\rangle\ , \text{ if } y=0\\ |x\rangle|y\rangle|D\rangle \mapsto |x\rangle|y\rangle|D\cup\{x\}\rangle\ , \text{ if } y=1\ .\end{cases}$$

Such a way of implementing a uniformly random oracle is referred to as a *compressed phase oracle* simulation [23]. Formally, the fact that the original and the compressed oracle simulations are *identical* from the point of view of the oracle algorithm (which does not have access to the oracle register) is because at any point in the execution of the algorithm, the states in the two simulations are both purifications of the same mixed state on the algorithm's registers.

We point out that there are two properties of a uniformly random oracle that make a compressed oracle simulation possible:

- The query outputs at each point are independently distributed, which means that the state of the oracle register is always a product state across all of the 2^n qubits.
- Each query output is uniformly distributed. This is important because in general $\alpha|0\rangle+\beta|1\rangle \not\perp \alpha|0\rangle-\beta|1\rangle$ unless $|\alpha|=|\beta|$.

Notice that the above compressed oracle simulation does not explicitly keep track of the value of the function at the queried points (i.e. a database is just a set of queried points). In the following slight variation on the compressed oracle simulation, also from [23], a database is instead a set of pairs (x,w) representing a queried point and the value of the function at that point. This variation will be more useful for our analysis.

Here D is a database of pairs (x,v), which is initially empty. A query acts as follows on a standard basis element $|x\rangle|y\rangle|D\rangle$:

- If $y=0$, do nothing.
- If $y=1$, check if D contains a pair of the form (x,v) for some v.
  - If it does not, add $(x,|-\rangle)$ to the database, where by this we formally mean: $D\mapsto\sum_v(-1)^v|D\cup(x,v)\rangle$
  - If it does, apply the unitary that removes $(x,|-\rangle)$ from the database.

One way to understand this compressed simulation is that our database representation only keeps track of pairs $(x,|-\rangle)$ (corresponding to the queried points), and it does not keep track of the other unqueried points, which in a fully explicit simulation would correspond to $|+\rangle$'s. One can think of the outputs at the unqueried points as being "compressed" in this succinct representation.

It is easy to see that the map above can be extended to a well-defined unitary. In the rest of this overview, we will take this to be our compressed phase oracle. For an oracle algorithm A, we will denote by A^{CPhO} the algorithm A run with a compressed phase oracle.

2.3.2 The structure of strategies that produce valid equations. Let P be an efficient prover, which takes as input a choice k of claw-free function pair, and outputs an equation (z, d, y). Suppose P outputs a valid equation, i.e. $z = d \cdot (x_0^y \oplus x_1^y) \oplus H(x_0^y) \oplus H(x_1^y)$, with high probability. Suppose we run a compressed phase oracle simulation P^{CPhO}. We argue that the state of P^{CPhO} right before measurement, conditioned on output (z, d, y), must be such that:

- Almost all the weight is on databases containing *exactly one* of the two pre-images of y.
- Up to a phase, the weights on databases containing x_0^y and x_1^y are approximately symmetrical.

In this subsection, we provide some intuition as to why this is true.

In general, we can write the state of P^{CPhO} right before measurement of the output as:

$$\sum_{z,d,y,w,D} \alpha_{z,d,y,w,D} \left|z\right\rangle \left|d\right\rangle \left|y\right\rangle \left|w\right\rangle \left|D\right\rangle ,$$

for some $\alpha_{z,d,y,w,D}$, where the z, d, y registers correspond to the output equation, the w register is a work register (which includes the query registers), and the D register is the database register. Here D is a database of pairs (x, v).

This expression can be simplified in a couple of ways. First, up to negligible weight, no database can contain *both* pre-images, since otherwise this would yield an efficient algorithm to extract a claw. Hence, we can write the state as a superposition over z, d, y and over databases that either: do not contain any pre-image y, they contain only x_0^y, or they contain only x_1^y:

$$\begin{aligned}
&\sum_{\substack{z,d,y,w \\ D \not\ni x_0^y, x_1^y}} \alpha_{z,d,y,w,D,0} \left|z,d,y,w\right\rangle \sum_{v_0} (-1)^{v_0} \left|D \cup (x_0^y, v_0)\right\rangle \\
+ &\sum_{\substack{z,d,y,w \\ D \not\ni x_0^y, x_1^y}} \alpha_{z,d,y,w,D,1} \left|z,d,y,w\right\rangle \sum_{v_1} (-1)^{v_1} \left|D \cup (x_1^y, v_1)\right\rangle \\
+ &\sum_{\substack{z,d,y,w \\ D \not\ni x_0^y, x_1^y}} \beta_{z,d,y,w,D} \left|z,d,y,w\right\rangle \left|D\right\rangle \qquad (5)
\end{aligned}$$

for some $\alpha_{z,d,y,w,D,b}, \beta_{z,d,y,w,D}$, where we have made the notation a little more compact. Recall that the reason for the presence of the phases $(-1)^{v_0}$ and $(-1)^{v_1}$ is that, by definition of the compressed oracle simulation, the output at each queried point in the database is in a $|-\rangle$ state.

Second, we expect intuitively that P should not be able to produce valid equations if it does not query any pre-image at all. Thus, we expect that if P produces a valid equation with high probability, then there should be only a small weight on the third branch in expression (5), i.e. the $\beta_{z,d,y,w,D}$ coefficients should be small. In our security proof, we formalize this intuition, and we show that any weight on the third branch contributes precisely $1/2$ to the probability of producing a valid equation, i.e. any weight on the third branch amounts to guessing an equation uniformly at random. Note that it is not a priori clear that this is true, and a more delicate analysis is required to establish that, when calculating the probability of outputting a valid equation, there is no interference between the branches that do not contain any pre-image, and the ones that contain one. We refer the reader to the full proof in Section 6 for more details.

So, suppose now for simplicity that P produces a valid equation with probability 1, then, from what we have discussed, up to negligible weight, the state just before measurement of the output is in a superposition of branches that contain exactly one of the two pre-images:

$$\begin{aligned}
&\sum_{\substack{z,d,y,w \\ D \not\ni x_0^y, x_1^y}} \alpha_{z,d,y,w,D,0} \left|z,d,y,w\right\rangle \sum_{v_0 \in \{0,1\}} (-1)^{v_0} \left|D \cup (x_0^y, v_0)\right\rangle \\
+ &\sum_{\substack{z,d,y,w \\ D \not\ni x_0^y, x_1^y}} \alpha_{z,d,y,w,D,1} \left|z,d,y,w\right\rangle \sum_{v_1 \in \{0,1\}} (-1)^{v_1} \left|D \cup (x_1^y, v_1)\right\rangle \qquad (6)
\end{aligned}$$

Finally, we wish to argue that the coefficients on the two branches are uniform, i.e. $|\alpha_{z,d,y,w,D,0}| = |\alpha_{z,d,y,w,D,1}|$ for all z, d, y, w, D. For this, we again appeal to the fact that P produces a valid equation with probability 1.

What does this probability correspond to in terms of expression (6)? In order to calculate this probability, we need to first *decompress* the database at both pre-images. What we mean by decompressing at x is the following:

- If the database already contains x, then do nothing.
- If the database does not contain x, then add $(x, |+\rangle)$ to it.

The reason why this makes sense is the following. Recall that points that are not present in the compressed database correspond to $|+\rangle$'s in the fully explicit "uncompressed" database. Since the condition for a valid equation depends on the value at both pre-images, we need to keep track of these values, and uncompress at the two pre-images in order to talk about the probability of a valid equation.

The state after decompressing at x_0^y and x_1^y is:

$$\begin{aligned}
&\sum_{\substack{z,d,y,w \\ D \not\ni x_0^y, x_1^y}} \alpha_{z,d,y,w,D,0} \left|z,d,y,w\right\rangle \sum_{v_0,v_1} (-1)^{v_0} \left|D \cup (x_0^y, v_0) \cup (x_1^y, v_1)\right\rangle \\
+ &\sum_{\substack{z,d,y,w \\ D \not\ni x_0^y, x_1^y}} \alpha_{z,d,y,w,D,1} \left|z,d,y,w\right\rangle \sum_{v_0,v_1} (-1)^{v_1} \left|D \cup (x_0^y, v_0) \cup (x_1^y, v_1)\right\rangle \qquad (7)
\end{aligned}$$

Now, by the equivalence of the regular and compressed oracle simulations, we have that the probability that P outputs a valid equation is equal to:

$$\Pr[z = d \cdot (x_0^y \oplus x_1^y) \oplus v_0 \oplus v_1]$$

In order for this probability to be close to 1, it must be that, for any z, d, y, w, D, the amplitudes of the two branches $\alpha_{z,d,y,w,D,0}$ and $\alpha_{z,d,y,w,D,1}$ interfere precisely constructively when $z = d \cdot (x_0^y \oplus x_1^y) \oplus v_0 \oplus v_1$ and interfere destructively otherwise. Hence, they have to be equal up to an appropriate phase.

2.3.3 Extracting a claw. We now provide some intuition about how the structure that we derived in the previous subsection can be leveraged to extract a claw.

Throughout this subsection, let P be such that, on input a choice of trapdoor claw-free function pair k, it simultaneously produces valid equations and proofs which are accepted with high probability,

i.e. it outputs (z, d, y, w) such that $\mathsf{Verify}(k, z, d, y, w) = 1$ with high probability. For concreteness we take this probability to be $9/10$ in this overview.

The first key observation is that, because by definition Verify never accepts an invalid equation (or except with negligible probability), then it must be the case that whenever Verify accepts, it *must* itself have queried at a superposition of both pre-images in the following more precise sense.

For a choice of claw-free functions k, denote by Verify_k the algorithm Verify where we fix the choice of claw-free functions to be k. Let A be *any* efficient oracle algorithm that, on input a choice of claw-free functions, outputs tuples (z, d, y, w). Suppose we run a compressed oracle simulation of A, followed by Verify, i.e. we run $(\mathsf{Verify}_k \circ A(k))^{\mathsf{CPhO}}$. Let the state right before measurement of Verify's output be:

$$\alpha \left|0\right\rangle \left|\phi_0\right\rangle + \beta \left|1\right\rangle \left|\phi_1\right\rangle ,$$

where the first qubit is the output qubit of Verify, and $|\phi_0\rangle$, $|\phi_1\rangle$ are some states on the remaining registers (including the database register). Then, except with negligible probability over the choice of k, the following holds: if β is non-negligible, then $|\phi_1\rangle$ has weight only on databases containing either x_0^y or x_1^y, and moreover, the weights for each pre-image are approximately equal. Such structure on the action of Verify follows from a similar argument as in the previous subsection, combined with the fact that Verify never accepts an invalid equation.

Even with this observation in hand, it is not a priori clear how we can extract a claw: ideally, we would like to say that, because of the observation above, if we were to run $(\mathsf{Verify}_k \circ P(k))^{\mathsf{CPhO}}$ *twice* and measure the database register, there would be a noticeable chance of obtaining distinct pre-images of some y. However, the issue is that if we run the computation twice, nothing guarantees that we will obtain the same y both times. In fact, if one thinks about the honest strategy for producing valid equations, each y has only an exponentially small probability of being the outcome.

To overcome this issue, the key observation is that, if P is successful with high enough probability, then there is a way to extract a claw with noticeable probability in a single run! The extraction algorithm is the following:

(i) Run $P^{\mathsf{CPhO}}(k)$, and measure the output registers to obtain z, d, y. Moreover, check if the database register at this point contains a pre-image of y. If so, measure it. Denote this by x_b^y.

(ii) Run $\mathsf{Verify}_k^{\mathsf{CPhO}}$ on the leftover state from the previous step, and measure the output register. Conditioned on "accept", measure the database register. If the database contains x_b^y, output the claw (x_0^y, x_1^y).

The idea behind this algorithm is that, thanks to the first observation, the state conditioned on obtaining "accept" in step (ii) has weight only on databases containing either x_0^y or x_1^y, and moreover, the weights for each pre-image are approximately equal. This implies that, conditioned on observing "accept" in step (ii), the final measurement of the algorithm is guaranteed to produce one of the two pre-images approximately *uniformly at random.*

Now, notice that the algorithm already has a constant probability of obtaining one of the two pre-images in step (i) (assuming P succeeds with probability $\frac{9}{10}$). To see this, notice that if this weren't the case, then P would be producing valid equations at most with probability close to $\frac{1}{2}$ (since this is the probability of producing a valid equation when the database register does not contain any of the two pre-images), and therefore the probability that Verify accepts P's output would also be at most close to $\frac{1}{2}$ (instead of being $\frac{9}{10}$). Thus, what is left to show is that the probability of obtaining "accept" in step (ii) *conditioned on finding a pre-image in step* (i) is still noticeable.

It is not a priori clear that this is the case. However, what we show is that if, to begin with, Verify accepts P's output with high enough probability (in fact any probability non-negligibly higher than $1/2$), then this extraction strategy works.

We remark that for our actual construction, we will want that, for any efficient P, the probability of simultaneously producing a valid equation *and* a proof is *negligible* (not just smaller than $\frac{1}{2}$+ negligible). Thus, our encryption scheme will be a parallel repetition of the single-shot encryption scheme described in this overview, i.e. the encryption of a single bit will consist of many single-shot encryptions of that bit.

3 PRELIMINARIES

3.1 Notation

We use the acronyms PPT and QPT for *probabilistic polynomial time* and *quantum polynomial time* respectively.

For a classical probabilistic algorithm $\mathcal{A}$, we write $\mathcal{A}(x; r)$ to denote running $\mathcal{A}$ on input x, with input randomness r.

For a finite set S, we use $x \leftarrow S$ to denote uniform sampling of x from the set S. We denote $[n] = \{1, 2, \cdots, n\}$. We denote by $\mathsf{Bool}(n)$ the set of functions from n bits to 1.

3.2 Deniable Encryption

We recall the classical notion of sender-deniable encryption.

Definition 2. *A public-key encryption scheme* (Gen, Enc, Dec) *is said to be* deniable *if there exists a PPT algorithm* Fake *such that, for any messages* m_0, m_1*:*

$$(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_1; r), m_1, r) \approx_c (\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_0; r), m_1, r')$$

where r *is uniformly random,* pk *is sampled according to* Gen*, and* $r' \leftarrow \mathsf{Fake}(\mathsf{pk}, c, m_0, m_1, r)$.

3.3 Noisy Trapdoor Claw-Free Functions

In this section we introduce the notion of noisy trapdoor claw-free functions (NTCFs). This section is taken almost verbatim from [9]. Let $\mathcal{X}, \mathcal{Y}$ be finite sets and $\mathcal{K}$ a set of keys. For each $k \in K$ there should exist two (efficiently computable) injective functions $f_{k,0}, f_{k,1}$ that map $\mathcal{X}$ to $\mathcal{Y}$, together with a trapdoor t_k that allows efficient inversion from $(b, y) \in \{0, 1\} \times \mathcal{Y}$ to $f_{k,b}^{-1}(y) \in \mathcal{X} \cup \{\perp\}$. For security, we require that for a randomly chosen key k, no polynomial time adversary can efficiently compute $x_0, x_1 \in \mathcal{X}$ such that $f_{k,0}(x_0) = f_{k,1}(x_1)$ (such a pair (x_0, x_1) is called a *claw*).

Unfortunately, we do not know how to construct such 'clean' trapdoor claw-free functions. Hence, as in the previous works [7, 9, 19], we will use 'noisy' version of the above notion. For each $k \in \mathcal{K}$,

there exist two functions $f_{k,0}, f_{k,1}$ that map $\mathcal{X}$ to a distribution over $\mathcal{Y}$.

Definition 3 (NTCF family). *Let λ be a security parameter. Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Let $\mathcal{K}_{\mathcal{F}}$ be a finite set of keys. A family of functions*

$$\mathcal{F} = \left\{f_{k,b} : \mathcal{X} \to \mathcal{D}_{\mathcal{Y}}\right\}_{k\in\mathcal{K}_{\mathcal{F}}, b\in\{0,1\}}$$

is called a noisy trapdoor claw free (NTCF) family *if the following conditions hold:*

1. ***Efficient Function Generation.*** *There exists an efficient probabilistic algorithm $\textsc{Gen}_{\mathcal{F}}$ which generates a key $k \in \mathcal{K}_{\mathcal{F}}$ together with a trapdoor t_k:*
$$(k, t_k) \leftarrow \textsc{Gen}_{\mathcal{F}}(1^\lambda) .$$
2. ***Trapdoor Injective Pair.***
   (a) *Trapdoor: There exists an efficient deterministic algorithm $\textsc{Inv}_{\mathcal{F}}$ such that with overwhelming probability over the choice of $(k, t_k) \leftarrow \textsc{Gen}_{\mathcal{F}}(1^\lambda)$, the following holds:*

for all $b \in \{0,1\}, x \in \mathcal{X}$ and $y \in \textsc{Supp}(f_{k,b}(x))$, $\textsc{Inv}_{\mathcal{F}}(t_k, b, y) = x$.

   (b) *Injective pair: For all keys $k \in \mathcal{K}_{\mathcal{F}}$, there exists a perfect matching $\mathcal{R}_k \subseteq \mathcal{X}\times\mathcal{X}$ such that $f_{k,0}(x_0) = f_{k,1}(x_1)$ if and only if $(x_0, x_1) \in \mathcal{R}_k$.*
3. ***Efficient Range Superposition.*** *For all keys $k \in \mathcal{K}_{\mathcal{F}}$ and $b \in \{0,1\}$ there exists a function $f'_{k,b} : \mathcal{X} \to \mathcal{D}_{\mathcal{Y}}$ such that the following hold.*
   (a) *For all $(x_0, x_1) \in \mathcal{R}_k$ and $y \in \textsc{Supp}(f'_{k,b}(x_b))$, $\textsc{Inv}_{\mathcal{F}}(t_k, b, y) = x_b$ and $\textsc{Inv}_{\mathcal{F}}(t_k, b\oplus 1, y) = x_{b\oplus 1}$.*
   (b) *There exists an efficient deterministic procedure $\textsc{Chk}_{\mathcal{F}}$ that, on input k, $b \in \{0,1\}$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, returns 1 if $y \in \textsc{Supp}(f'_{k,b}(x))$ and 0 otherwise. Note that $\textsc{Chk}_{\mathcal{F}}$ is not provided the trapdoor t_k.*
   (c) *For every k and $b \in \{0,1\}$,*
$$\mathbb{E}_{x\leftarrow\mathcal{X}}\left[\, H^2(f_{k,b}(x),\, f'_{k,b}(x))\,\right] \le \mu(\lambda) .$$
for some negligible function μ. Here H^2 is the Hellinger distance. Moreover, there exists an efficient procedure $\textsc{Samp}_{\mathcal{F}}$ that on input k and $b \in \{0,1\}$ prepares the state
$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \sqrt{(f'_{k,b}(x))(y)}\, |x\rangle\, |y\rangle . \tag{8}$$
4. ***Claw-Free Property.*** *For any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds:*

$$\Pr[(x_0, x_1) \in \mathcal{R}_k : (k, t_k) \leftarrow \textsc{Gen}_{\mathcal{F}}(1^\lambda), (x_0, x_1) \leftarrow \mathcal{A}(k)] \le \mathsf{negl}(\lambda)$$

In our security analysis, we will make use of the following additional concepts from [19], which we recall informally here (we refer to [19] or the full version of this paper for the details). A *trapdoor injective function family*

$$\mathcal{G} = \left\{g_{k,b} : \mathcal{X} \to \mathcal{D}_{\mathcal{Y}}\right\}_{b\in\{0,1\}, k\in\mathcal{K}_{\mathcal{G}}}$$

satisfies the properties of "efficient function generation" and "efficient range superposition". However, crucially, $g_{k,0}$ and $g_{k,1}$ have disjoint range. Moreover, an NTCF family $\mathcal{F}$ (as in Definition 3) satisfies the *injective-invariance* property, if there exists a trapdoor injective function family $\mathcal{G}$ such that functions sampled from $\mathcal{F}$ and $\mathcal{G}$ are computationally indistinguishable.

Lemma 1 ([7], [19]). *Assuming the quantum hardness of LWE, there exists an injective invariant NTCF family.*

4 QUANTUM DENIABILITY

4.1 Definition

We assume without loss of generality that the (quantum) encryption algorithm Enc consists of an efficiently implementable isometry $\widetilde{\mathsf{Enc}}$ from register M, consisting of $m(\lambda)$ qubits, to register N, consisting of $n(\lambda)$ qubits, followed by a standard basis measurement of the output ciphertext register C (consisting of a subset of the qubits in N.

Definition 4 (Quantum Deniability). *Let $l : \mathbb{N} \to \mathbb{N}$ be polynomially bounded. A public-key encryption scheme* (Gen, Enc, Dec) *is l-deniable if the following holds. There exists a QPT algorithm* Fake*, and a sub-register* R *of* C *consisting of $l(\lambda)$ qubits, such that the following holds for any m_0, m_1:*

$$\left(\mathsf{pk}, m_1, (\mathsf{Meas}_{\mathsf{R}} \otimes \mathbb{I}_{\mathsf{C}\setminus\mathsf{R}} \otimes \mathsf{Fake}_{\mathsf{N}\setminus\mathsf{C}})\, \widetilde{\mathsf{Enc}}(\mathsf{pk}, m_0)\right) \tag{9}$$
$$\approx_c \left(\mathsf{pk}, m_1, (\mathsf{Meas}_{\mathsf{R}} \otimes \mathbb{I}_{\mathsf{N}\setminus\mathsf{R}})\, \widetilde{\mathsf{Enc}}(\mathsf{pk}, m_1)\right)$$

where $\mathsf{Meas}_{\mathsf{R}}$ *is the quantum channel that corresponds to a standard basis measurement of register* R*, and* pk *is sampled according to* Gen.

To keep the definition general, we included a faking algorithm Fake which is allowed to act on the leftover quantum state of the encryption algorithm. This more naturally parallels the classical definition of deniability. However, we will see that our scheme satisfies an even stronger notion of deniability, whereby the leftover state does not need any additional processing (before it gets handed to the attacker) i.e. Fake can be taken to be the identity.

4.2 Construction

We describe encryption of a single bit. This can, of course, be extended in parallel to encryptions of any number of bits.

Let $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ be finite sets. Let $\mathcal{F} = \left\{f_{k,b} : \mathcal{X} \to \mathcal{D}_{\mathcal{Y}}\right\}_{k\in\mathcal{K}, b\in\{0,1\}}$ be a family of noisy trapdoor claw-free functions (which exists assuming the quantum hardness of LWE [7]). Let $f'_{k,b} : \mathcal{X} \to \mathcal{Y}$ be functions satisfying the *efficient range superposition property* of Definition 3. For $x \in \mathcal{X}$, denote by $\mathsf{BitDecomp}(x)$ its bit decomposition.

Construction 1.

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*:*
  - *Run $(k, t_k) \leftarrow \textsc{Gen}_{\mathcal{F}}(1^\lambda)$. Output $(\mathsf{pk}, \mathsf{sk}) = (k, t_k)$.*
- $\mathsf{Enc}(m, \mathsf{pk}) \to c$*:*
  - *On input $m \in \{0,1\}$, and $\mathsf{pk} = k$, run $\textsc{Samp}_{\mathcal{F}}(k, \cdot)$ on a uniform superposition of b's, to obtain the state*
$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{b\in\{0,1\}, x\in X} \sqrt{f'_{k,b}(x)(y)}\, |b\rangle\, |x\rangle\, |y\rangle ,$$
where we assume that x and y are represented by their bit decomposition. We assume without loss of generality that $\textsc{Samp}_{\mathcal{F}}$ that any auxiliary register is returned to the $|0\rangle$ state. [2]

[2] Since the output of $\textsc{Samp}_{\mathcal{F}}$ on the output registers is a pure state, one can always have $\textsc{Samp}_{\mathcal{F}}$ coherently "uncompute" on all registers except does containing the output.

– *Measure the image register, and let y be the outcome. As a result, the state has collapsed to:*

$$\frac{1}{\sqrt{2}}(|0\rangle|x_0\rangle + |1\rangle|x_1\rangle),$$

where $x_0, x_1 \in \mathcal{X}$ are the unique elements such that y is in the support of $f'_{k,b}(x_b)$.
– *Let n be the length of* $\mathsf{BitDecomp}(x_0)$. *Apply a Hadamard gate to all of the remaining registers, and measure. Parse the measurement outcome as $z||d$ where $z \in \{0,1\}$ and $d \in \{0,1\}^n$.*
– *Output $c = (z, d, y)$.*

- $\mathsf{Dec}(c, t_k) \to m$:
  – *Let $c = (z',\ d,\ y)$. For $b \in \{0,1\}$, run $\textsc{Inv}_{\mathcal{F}}(t_k, b, y)$ to obtain pre-images x_0 and x_1.*
  – *Output $m = z' \oplus d \cdot (\mathsf{BitDecomp}(x_0) \oplus \mathsf{BitDecomp}(x_1))$.*
- $\mathsf{Fake}(m', \mathsf{aux})$: *Given as input m' (the desired plaintext to be claimed), and a quantum state* aux *(the leftover workspace after encryption), output (m', aux), i.e. output the leftover workspace unchanged.*

Theorem 3. *Assuming the quantum hardness of LWE, the scheme of construction 1 is CPA-secure and* 1*-deniable (as in Definition 4).*

Correctness of the encryption scheme is straightforward to verify. Hence, we focus on security in the next section.

4.3 Security

From now on, for ease of notation, when referring to the bit-decomposition of x, we simply write x instead $\mathsf{BitDecomp}(x)$ when the context is clear. We denote by x_0^y and x_1^y the two pre-images of y.

CPA security. CPA security follows straightforwardly from the following "hardcore bit" property satisfied by $\mathcal{F}$, which says that any quantum polynomial-time adversary $\mathcal{A}$ has negligible advantage in the following game between a challenger and $\mathcal{A}$:

- The challenger samples k, and runs $\textsc{Samp}_{\mathcal{F}}(k, \cdot)$ on a uniform superposition of b's to obtain the state

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{b\in\{0,1\}, x\in X} \sqrt{f'_{k,b}(x)(y)}\,|b\rangle|x\rangle|y\rangle\ ,$$

- Measures the last register to get y.
- Then, applies a Hadamard gate on the first two registers to get z, d such that $z = d \cdot (x_0 \oplus x_1)$, where x_0 and x_1 are the pre-images of y. Sends y, d to $\mathcal{A}$.
- $\mathcal{A}$ returns z'.

$\mathcal{A}$ wins if $z = z'$.

To see that this "hardcore bit" property holds, suppose for a contradiction that there was a quantum polynomial time $\mathcal{A}$ breaking this property. Then the following algorithm $\mathcal{A}'$ recovers a claw. We use the notation $\mathcal{A}(y, d)$ to denote the output of $\mathcal{A}$ on input y, d.

- On input k, run $\textsc{Samp}_{\mathcal{F}}(k, \cdot)$ on a uniform superposition of b's to obtain the state

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{b\in\{0,1\}, x\in X} \sqrt{f'_{k,b}(x)(y)}\,|b\rangle|x\rangle|y\rangle\ ,$$

- Measure the last two registers to obtain x, y where $y \in \textsc{Supp}(f'_{k,b}(x))$ for some $b \in \{0,1\}$.
- Run the "quantum Goldreich-Levin" extraction algorithm using $\mathcal{A}(y, \cdot)$ as an oracle [2]. Recall that the "quantum Goldreich-Levin" extraction algorithm makes a single query to $\mathcal{A}(y, \cdot)$. Let s be the output of this extraction algorithm.
- Output $(x, x \oplus s)$ as the claw.

Since $\mathcal{A}(y, d)$ guesses $d \cdot (x_0 \oplus x_1)$ with non-negligible advantage, the "quantum Goldreich-Levin" extraction algorithm outputs $s = x_0 \oplus x_1$ with non-negligible probability, which results in $\mathcal{A}'$ outputting a claw with non-negligible probability.

Note that the reduction works even if $\mathcal{A}$ is non-uniform with quantum advice, since $\mathcal{A}'$ makes a single query to $\mathcal{A}$, so there is no issue with rewinding.

With this hardcore bit property in hand, CPA security is immediate: notice that the distribution $(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 0))$ is simply (k, z, d, y), such that $z = d \cdot (x_0^y \oplus x_1^y)$, where k, d, z, y are sampled exactly from the distribution of the hardcore bit property. On the other hand, the distribution $(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 1))$ is (k, z, d, y), such that $z \neq d \cdot (x_0^y \oplus x_1^y)$, where k, z, d, y have the same distribution as before, except that z is flipped. Clearly, distinguishing the two distributions is precisely equivalent to guessing the value of $d \cdot (x_0^y \oplus x_1^y)$, which is the hardcore bit game above.

Deniability. We will now show that Construction 1 satisfies deniability, according to Definition 4. In particular, we will show that, for Construction 1, one can take the algorithm Fake to be the identity.

Using the notation of Definition 4, the output of the encryption algorithm Enc, before measurement of the output, is a state on register N, a sub-register of which, C, is eventually measured to produce the classical ciphertext. Notice that for Enc as defined in Construction 1, all qubits in $\mathsf{N} \setminus \mathsf{C}$ are eventually returned to the state $|0\rangle$. Thus, in order to prove 1-deniability, all that is left to show is that there is a subset R, of size 1, of the qubits of C, such that for any $m_0, m_1 \in \{0,1\}$,

$$\begin{aligned} &\left(\mathsf{pk}, m_1, \mathsf{Meas}_{\mathsf{R}} \otimes I_{\mathsf{C}\setminus\mathsf{R}}\left([\widetilde{\mathsf{Enc}}(m_0)]_{\mathsf{C}}\right)\right) \\ \approx_c &\left(\mathsf{pk}, m_1, \mathsf{Meas}_{\mathsf{R}} \otimes I_{\mathsf{C}\setminus\mathsf{R}}\left([\widetilde{\mathsf{Enc}}(m_1)]_{\mathsf{C}}\right)\right), \end{aligned} \tag{10}$$

where recall that $\widetilde{\mathsf{Enc}}$ denotes Enc excluding the final measurement, and we are denoting by $[\widetilde{\mathsf{Enc}}(m)]_{\mathsf{C}}$ the restriction of the state $\widetilde{\mathsf{Enc}}(m)$ to the register C.

We take R to be the qubit corresponding to the first bit of the ciphertext. Then, up to replacing $f_{k,b}$ with $f'_{k,b}$ (which affects the distribution at most negligibly - see Lemma 2.0.1 in [9] relating Hellinger distance to trace distance), conditioned on the outcome of the measurement being z, the LHS and the RHS of (10) are respectively,

$$\left(k, m_1, z\,,\ \sum_{x,d,y\,:\,d\cdot(x_0^y\oplus x_1^y)=z\oplus m_0} \sqrt{(f_{k,0}(x))(y)}\,|d\rangle|y\rangle\right)$$

and

$$\left(k, m_1, z\,,\ \sum_{x,d,y\,:\,d\cdot(x_0^y\oplus x_1^y)=z\oplus m_1} \sqrt{(f_{k,0}(x))(y)}\,|d\rangle|y\rangle\right)$$

where x_0^y and x_1^y denote the pre-images of y (and we omit writing BitDecomp when it is clear from the context). Then, it is easy to see that the two distributions are computationally indistinguishable if and only if:

$$\left(k, \sum_{x,d,y\,:\,d\cdot(x_0^y \oplus x_1^y)=0} \sqrt{(f_{k,0}(x))(y)}\,|d\rangle\,|y\rangle\right) \tag{11}$$

$$\approx_c \left(k, \sum_{d,y\,:\,d\cdot(x_0^y \oplus x_1^y)=1} \sqrt{(f_{k,0}(x))(y)}\,|d\rangle\,|y\rangle\right), \tag{12}$$

where we omit normalization constants.

To argue that these two states are indistinguishable, we appeal to the *injective invariance* property. Recall that by the latter, there exists a trapdoor injective family

$$\mathcal{G} = \{g_{k,b} : \mathcal{X} \to \mathcal{D}_{\mathcal{Y}}\}_{k\in\mathcal{K}_{\mathcal{G}}, b\in\{0,1\}}$$

such that

(i) $\textsc{Chk}_{\mathcal{F}} = \textsc{Chk}_{\mathcal{G}}$ and $\textsc{Samp}_{\mathcal{F}} = \textsc{Samp}_{\mathcal{G}}$.
(ii) It is computationally hard to distinguish whether $k \leftarrow \textsc{Gen}_{\mathcal{F}}$ or $k \leftarrow \textsc{Gen}_{\mathcal{G}}$.

Now, suppose for a contradiction that there was a distinguisher D for the two (families of) states in (12). Then, there exists a distinguisher D' that breaks property (ii) of injective invariance:

- D' receives k as input.
- D' runs $\widetilde{\mathsf{Enc}}(\mathsf{pk}, 0)$ with $\mathsf{pk} = k$, and then measures the first qubit. Let z be the outcome. (note that this step is well-defined since $\textsc{Samp}_{\mathcal{F}} = \textsc{Samp}_{\mathcal{G}}$).
- D' gives k and the leftover state to D, which returns a guess z'. If $z' = z$ (i.e. the distinguisher for the two states succeeded), D' guesses that k was sampled from $\textsc{Gen}_{\mathcal{F}}$. Otherwise, it guesses that it was sampled from $\textsc{Gen}_{\mathcal{G}}$.

The reason why D' has non-negligible advantage is the following. In the case that $k \leftarrow \textsc{Gen}_{\mathcal{F}}$, the leftover states conditioned on $z = 0$ and $z = 1$ are (up to the negligible error incurred by replacing $f'_{k,b}$ with $f_{k,b}$) those on the LHS and the RHS of (12) respectively. On the other hand, when $k \leftarrow \textsc{Gen}_{\mathcal{G}}$, it is easy to see that the state that results from applying $\widetilde{\mathsf{Enc}}(\mathsf{pk}, 0)$ and measuring the first qubit is independent of the outcome z. In particular, the leftover state is simply

$$\sum_{x,d,y} \sum_{b\in\{0,1\}} \sqrt{(g_{k,b}(x))(y)}\,|d\rangle\,|y\rangle\ .$$

So the probability that D guesses z correctly is exactly $\frac{1}{2}$.

We refer the reader to the full version of the paper for the details of the proof.

5 UNEXPLAINABLE ENCRYPTION

In this section, we formally introduce the notions of *unexplainable* encryption, and *perfectly unexplainable* encryption. In 5.2, we discuss the notion of coercion before-the-fact in more detail, and its relationship to perfect unexplainability. We refer to the full version of the paper for a discussion of the relationship between unexplainable encryption and the standard definition of deniable encryption.

5.1 Definition

In the following definition, Explain takes as input a public key pk, a message m, and outputs a ciphertext c and a witness w; Verify takes as input a public key pk, a ciphertext c, a message m, a witness w, and outputs a bit; FakeExplain takes as input a public key pk, a pair of messages m, m', and outputs a ciphertext c, and two witnesses w, w'. In the rest of the section, we use the notation $\Pr_{\mathsf{pk}}$ to mean that the probability is over sampling a pk from the the generation algorithm $\mathsf{Gen}(1^\lambda)$ (where the security parameter λ is omitted from the notation).

Definition 5 (Unexplainable Encryption). *We say that a public-key encryption scheme* (Gen, Enc, Dec) *is* unexplainable *if the following holds. Let* Verify *and* Explain *be any (non-uniform) QPT algorithms such that, for all* pk, c, m, w, $\mathsf{Verify}(\mathsf{pk}, c, m, w) = 0$, *except with exponentially small probability, if c is not in the support of the distribution of* $\mathsf{Enc}(\mathsf{pk}, m)$.

Suppose Verify, Explain *satisfy the following* completeness *condition: there exists a non-negligible function γ, such that, for any m, for all λ,*

$$\Pr_{\mathsf{pk}}[\mathsf{Verify}(\mathsf{pk}, c, m, w) = 1 : (c, w) \leftarrow \mathsf{Explain}(\mathsf{pk}, m)] \geq \gamma(\lambda)\,.$$

Then, there exist a polynomial-time algorithm FakeExplain *and a non-negligible function* negl*, such that, for any distinct messages m, m', for all λ:*

$$\Big|\Pr_{\mathsf{pk}}[\mathsf{Verify}\,(\mathsf{pk}, c, m, w) = 1 : c, w \leftarrow \mathsf{Explain}(\mathsf{pk}, m)]$$
$$-\Pr_{\mathsf{pk}}[\mathsf{Verify}\,(\mathsf{pk}, c, m', w') = 1 : c, w \leftarrow \mathsf{Explain}(\mathsf{pk}, m),$$
$$w' \leftarrow \mathsf{FakeExplain}(\mathsf{pk}, m, m', c, w)]\Big| = \mathsf{negl}(\lambda) \tag{13}$$

We believe that this definition naturally captures the ideal of privacy desired in a deniable setting. Moreover, since this definition does not explicitly refer to input randomness, it applies naturally to the setting of a quantum encryption algorithm (with classical ciphertexts). w and w' can also be taken to be (possibly entangled) quantum states.

The following is a special case of unexplainable encryption. In words, an encryption scheme is *perfectly* unexplainable if there does not exist any pair of efficient algorithms Verify and Explain (where $\mathsf{Verify}(\mathsf{pk}, c, m, w) = 0$, except with exponentially small probability, if c is not in the range of $\mathsf{Enc}(\mathsf{pk}, m)$), for which the *completeness* condition holds.

Definition 6 (Perfectly Unexplainable Encryption). *A public-key encryption scheme* (Gen, Enc, Dec) *is said to be* perfectly unexplainable *if the following holds. Let* Explain *and* Verify *be any (non-uniform) pair of QPT algorithms such that, for all* pk, c, m, w, $\mathsf{Verify}(\mathsf{pk}, c, m, w) = 0$, *except with exponentially small probability, if c is not in the support of the distribution of* $\mathsf{Enc}(\mathsf{pk}, m)$. *Then, for any m, there exists a negligible function* negl*, such that for any λ,*

$$\Pr_{\mathsf{pk}}[\mathsf{Verify}(\mathsf{pk}, c, m, w) = 1 : (c, w) \leftarrow \mathsf{Explain}(\mathsf{pk}, m)] = \mathsf{negl}(\lambda)\,. \tag{14}$$

At first glance, perfect unexplainability seems unattainable. In fact, it is clear that a classical encryption scheme cannot be perfectly unexplainable: one can always take Explain to be the algorithm that encrypts honestly and outputs the ciphertext together with the randomness used (i.e. the witness is the randomness), and take Verify to be the algorithm that runs encryption forward and checks consistency. In contrast, the quantum encryption scheme that we will describe in Section 6 will be perfectly unexplainable.

5.2 Coercion Before-the-Fact

While protecting against coercion before-the-fact is very desirable in practice, to the best of our knowledge, a corresponding notion has not been previously formalized (most likely due to the fact that protecting against coercion before-the-fact is impossible to achieve classically in the plain model). Here, we propose a formal definition, and we observe that this is essentially equivalent to the notion of *perfect unexplainability* from Definition 6.

In a coercion before-the-fact scenario, an attacker, who has in mind a message m, wishes to prescribe to the sender *how* she should encrypt later in a way that:

- The resulting ciphertext decrypts to m with overwhelming probability.
- There is an efficient procedure for the attacker to verify that the sender's ciphertext (which the attacker obtains by intercepting) will decrypt to m with overwhelming probability.

First, notice that there is no hope of protecting against coercion before-the-fact in a model where the attacker can approach the sender before she sends her ciphertext, and knows *all* of the information that will be available to the sender at the time of encryption (e.g. the public key). In fact, in such a model, the attacker can simply generate a genuine encryption c of the desired message m, and prescribe that the sender's ciphertext later be exactly c. So, instead we consider the scenario where the public key is *not known* to the attacker at the time when he approaches the sender. Such a model captures an online election where citizens are required to encrypt their votes before sending them to the government using a public key encryption scheme, and the public key is announced publicly only on election day. The attacker is allowed to approach the sender any time *before* election day, i.e. any time before the public key is announced (more generally, one can consider a model where some additional information - not necessarily the public key - is revealed to the sender just before she encrypts).

We first formally define the notion of a coercion before-the-fact *attack*. An encryption scheme then protects against coercion before-the-fact if no such attack exists.

Definition 7 (Coercion before-the-fact attack). *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme with classical ciphertexts (where* Dec *is a classical deterministic algorithm). A coercion before-the-fact attack is a pair* $(\mathsf{Enc}', \mathsf{Verify})$ *where:*

- $\mathsf{Enc}'(\mathsf{pk}, m) \to c$ *is a non-uniform QPT algorithm*
- $\mathsf{Verify}(c, m) \to \mathsf{accept/reject}$ *is a non-uniform QPT algorithm*

They satisfy:

- *(Completeness of verification) For any* m, λ,

$$\Pr[\mathsf{Verify}(\mathsf{Enc}'(\mathsf{pk}, m), m) = \mathsf{accept} : (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)] = 1\,.$$

- *(Soundness of verification) There exists* $C > 0$*, such that the following holds for all* c, m, λ*:*

$$\Pr[\mathsf{Dec}(\mathsf{sk}, c) \neq m \ \wedge \ \mathsf{Verify}(\mathsf{pk}, c, m) = \mathsf{accept} : (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)] \leq 2^{-\lambda^C}\,.$$ [3]

Definition 8. *We say that an encryption scheme protects against coercion before-the-fact if no coercion before-the-fact attacks exists.*

It is straightforward to see that a perfectly unexplainable encryption scheme protects against coercion before-the-fact. This is because a coercion before-the-fact attack gives a way to encrypt in a way that can be later verified.

Theorem 4. *A perfectly unexplainable encryption scheme protects against coercion before-the-fact.*

Proof. Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a perfectly unexplainable encryption scheme (satisfying the non-uniform version of Definition 6). Suppose for a contradiction that a coercion before-the-fact attack $(\mathsf{Enc}', \mathsf{Verify})$ existed.

Define Verify' to be the non-uniform algorithm that on input pk, c, m, w, runs $\mathsf{Verify}(\mathsf{pk}, c, m)$ and ignores w. Let Explain be the non-uniform algorithm that, on input pk, m, runs $c \leftarrow \mathsf{Enc}'(\mathsf{pk}, m)$, and outputs c, w^*, for some fixed w^*. By the completeness and soundness of the coercion before-the-fact attack, it follows that the pair $(\mathsf{Explain}, \mathsf{Verify}')$ contradicts perfect unexplainability. □

Definition 7 only considers attacks where Enc' and Verify are non-uniform with *classical* advice. More generally, one could also consider attacks where Enc' and Verify have *quantum* advice. In particular, this advice could be in the form of an entangled state over two registers corresponding to the advice of Enc' and Verify respectively. This captures the scenario where an attacker coerces the sender before-the-fact by giving to the sender half of some entangled state, and prescribing what the encryption operation should be, i.e. Enc'. The verification of the sender's intercepted ciphertext then makes use of the other half of the entangled state. It is not difficult to see that a slightly more general version of the definition of perfect unexplainability, where Explain and Verify are allowed to be non-uniform with entangled quantum advice, implies this more general notion of protection against coercion before-the-fact. Our constructions satisfies such a notion of perfect unexplainability assuming the quantum hardness of LWE against QPT algorithms with polynomial quantum advice. Our security proof (contained in the full version) goes through unchanged (since our extraction algorithm does not involve any rewinding).

6 A PERFECTLY UNEXPLAINABLE ENCRYPTION SCHEME

In this section, we describe a public-key encryption scheme that is perfectly unexplainable (as in Definition 6).

6.1 Construction

Let $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ be finite sets. Let $\mathcal{F} = \left\{f_{k,b} : \mathcal{X} \to \mathcal{D}_\mathcal{Y}\right\}_{k\in\mathcal{K}, b\in\{0,1\}}$ be a family of noisy trapdoor claw-free functions (which exists assuming LWE [7]). Let $f'_{k,b} : \mathcal{X} \to \mathcal{Y}$ be functions satisfying the *efficient range superposition property* of Definition 3.

The scheme that we describe in this section is a parallel repeated version of the scheme described in the technical overview (Section 2.3.1). Here, by parallel repetition we mean that the same plaintext m is encrypted L times (with the same public key), where L is polynomial in the security parameter. Without parallel repetition, we are only able to show that the LHS of Equation (14) is upper bounded by $\frac{1}{2}$ + negl. With parallel repetition, we will be able to improve this to negl.

Construction 2. *Parameters:* $L = \mathsf{poly}(\lambda)$.

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*:*
  - *Run* $(k, t_k) \leftarrow \textsc{Gen}_{\mathcal{F}}(1^\lambda)$. *Output* $(\mathsf{pk}, \mathsf{sk}) = (k, t_k)$.
- $\mathsf{Enc}(m, \mathsf{pk}) \to c$*:*
  - *For* $i \in [L]$*, do the following:*
    - *On input* $m \in \{0,1\}$*, and* $\mathsf{pk} = k$*, run* $\textsc{Samp}_{\mathcal{F}}(k, \cdot)$ *on a uniform superposition of* b*'s, to obtain the state*

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{b\in\{0,1\}, x\in X} \sqrt{f'_{k,b}(x)(y)}\, |b\rangle\, |x\rangle\, |y\rangle \ ,$$

      where we assume that x *and* y *are represented by their bit decomposition. We assume without loss of generality that* $\textsc{Samp}_{\mathcal{F}}$ *that any auxiliary register is returned to the* $|0\rangle$ *state.* [4]
    - *Measure the image register, and let* $y_i \in \mathcal{Y}$ *be the outcome. As a result, the state has collapsed to:* $\frac{1}{\sqrt{2}}(|0\rangle\, |x_0\rangle + |1\rangle\, |x_1\rangle)$*, where* $x_0, x_1 \in \mathcal{X}$ *are the unique elements such that* y_i *is in the support of* $f'_{k,b}(x_b)$.
    - *Query the phase oracle for* H*, to obtain:*

$$\frac{1}{\sqrt{2}}(-1)^{H(\mathsf{BitDecomp}(x_0))}\, |0\rangle\, |x_0\rangle + \frac{1}{\sqrt{2}}(-1)^{H(\mathsf{BitDecomp}(x_1))}\, |1\rangle\, |x_1\rangle \ .$$

    - *Let* n *be the length of* $\mathsf{BitDecomp}(x_0)$*. Apply a Hadamard gate to all of the remaining registers, and measure. Parse the measurement outcome as* $z_i || d_i$ *where* $z_i \in \{0,1\}$ *and* $d_i \in \{0,1\}^n$*. Let* $z'_i := z_i \oplus m$.
  - *Let* $z' := z'_1 \dots z'_L$*,* $d := d_1 \dots d_L$*,* $y = y_1 \dots y_L$*. Output* $c = (z', d, y)$.
- $\mathsf{Dec}(c, t_k) \to m$*:*
  - *Let* $c = (z',\ d,\ y)$*. Parse* z' *as* $z' = z'_1 \dots z'_L$*. Similarly for* d *and* y.
  - *For* $i \in [L]$*:*
    - *For* $b \in \{0,1\}$*, run* $\textsc{Inv}_{\mathcal{F}}(t_k, b, y_i)$ *to obtain pre-images* $x_0^{y_i}$ *and* $x_1^{y_i}$.
    - *Let* $m_i = z'_i \oplus d_i \cdot (\mathsf{BitDecomp}(x_0^{y_i}) \oplus \mathsf{BitDecomp}(x_1^{y_i})) \oplus H(\mathsf{BitDecomp}(x_0^{y_i})) \oplus H(\mathsf{BitDecomp}(x_1^{y_i}))$.
  - *If* $m_1 = \dots = m_L$*, output* m_1*, otherwise output* $\perp$.

Theorem 5. *The scheme of Construction 2 is a CPA-secure perfectly unexplainable encryption scheme in the quantum random oracle model (QROM), assuming the quantum hardness of LWE.*

We refer to the technical overview for a high-level picture of the proof, and to the full version of the paper for all the details.

[4] Since the output of $\textsc{Samp}_{\mathcal{F}}$ on the output registers is a pure state, one can always have $\textsc{Samp}_{\mathcal{F}}$ coherently "uncompute" on all registers except does containing the output.

ACKNOWLEDGMENTS

This work was carried out while A.C. was a Quantum Postdoctoral Fellow at the Simons Institute for the Theory of Computing supported by NSF QLCI Grant No. 2016245. S.G. is supported by DARPA under agreement No. HR00112020023. U.V is supported by Vannevar Bush faculty fellowship N00014-17-1-3025, MURI Grant FA9550-18-1-0161, and NSF QLCI Grant No. 2016245.

REFERENCES

[1] Scott Aaronson. 2009. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*. IEEE, 229–242.

[2] Mark Adcock and Richard Cleve. 2002. A quantum Goldreich-Levin theorem with cryptographic applications. In *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 323–334.

[3] Shweta Agrawal, Shafi Goldwasser, and Saleet Mossel. 2021. Deniable Fully Homomorphic Encryption from Learning with Errors. In *Annual International Cryptology Conference*. Springer, 641–670.

[4] James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. 2021. One-way functions imply secure computation in a quantum world. In *Annual International Cryptology Conference*. Springer, 467–496.

[5] Shalev Ben-David and Or Sattath. 2016. Quantum tokens for digital signatures. *arXiv preprint arXiv:1609.09047* (2016).

[6] Charles H Bennett and Gilles Brassard. 1984. Quantum cryptography. In *Proc. IEEE Int. Conf. on Computers, Systems and Signal Processing, Bangalore, India*. 175–179.

[7] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh Vazirani, and Thomas Vidick. 2018. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 320–331.

[8] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. 2020. Candidate iO from homomorphic encryption schemes. (2020).

[9] Zvika Brakerski, Venkata Koppula, Umesh Vazirani, and Thomas Vidick. 2020. Simpler Proofs of Quantumness. In *15th Conference on the Theory of Quantum Computation, Communication and Cryptography*.

[10] Anne Broadbent and Sébastien Lord. 2020. Uncloneable Quantum Encryption via Oracles. In *15th Conference on the Theory of Quantum Computation, Communication and Cryptography*.

[11] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. 1997. Deniable encryption. In *Annual International Cryptology Conference*. Springer, 90–104.

[12] Ran Canetti, Sunoo Park, and Oxana Poburinnaya. 2020. Fully deniable interactive encryption. In *Annual International Cryptology Conference*. Springer, 807–835.

[13] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. 2021. Hidden cosets and applications to unclonable cryptography. In *Annual International Cryptology Conference*. Springer, 556–584.

[14] Claude Crépeau and Joe Kilian. 1988. Achieving oblivious transfer using weakened security assumptions. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 42–52.

[15] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. 2021. Succinct LWE Sampling, Random Polynomials, and Obfuscation. *Cryptology ePrint Archive* (2021).

[16] Artur K Ekert. 1992. Quantum Cryptography and Bell's Theorem. In *Quantum Measurements in Optics*. Springer, 413–418.

[17] Romain Gay and Rafael Pass. 2021. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 736–749.

[18] Aayush Jain, Huijia Lin, and Amit Sahai. 2021. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 60–73.

[19] Urmila Mahadev. 2018. Classical verification of quantum computations. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 259–267.

[20] Amit Sahai and Brent Waters. 2014. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 475–484.

[21] Hoeteck Wee and Daniel Wichs. 2021. Candidate obfuscation via oblivious LWE sampling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 127–156.

[22] Stephen Wiesner. 1983. Conjugate Coding. *SIGACT News* 15, 1 (Jan. 1983), 78–88. https://doi.org/10.1145/1008908.1008920

[23] Mark Zhandry. 2019. How to Record Quantum Queries, and Applications to Quantum Indifferentiability. In *Advances in Cryptology – CRYPTO 2019*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer International Publishing, Cham, 239–268.