

# Quantum Tokens for Digital Signatures

Shalev Ben-David<sup>1</sup> and Or Sattath<sup>2,1</sup>

<sup>1</sup>MIT

<sup>2</sup>The Hebrew University

February 7, 2017

## Abstract

The fisherman caught a quantum fish. *Fisherman, please let me go*, begged the fish, and *I will grant you three wishes*. The fisherman agreed. The fish gave the fisherman a quantum computer, three quantum signing tokens and his classical public key. The fish explained: *to sign your three wishes, use the tokenized signature scheme on this quantum computer, then show your valid signature to the king who owes me a favor*.

The fisherman used one of the signing tokens to sign the document “give me a castle!” and rushed to the palace. The king executed the classical verification algorithm using the fish’s public key, and since it was valid, the king complied.

The fisherman’s wife wanted to sign ten wishes using their two remaining signing tokens. The fisherman did not want to cheat, and secretly sailed to meet the fish. *Fish, my wife wants to sign ten more wishes*. But the fish was not worried: *I have learned quantum cryptography following the previous story<sup>1</sup>. These quantum tokens are consumed during the signing. Your polynomial wife cannot even sign four wishes using the three signing tokens I gave you*.

*How does it work?* wondered the fisherman. *Have you heard of quantum money? These are quantum states which can be easily verified but are hard to copy. This tokenized quantum signature scheme extends Aaronson and Christiano’s quantum money scheme, which is why the signing tokens cannot be copied*.

*Does your scheme have additional fancy properties?* asked the fisherman. *Yes, the scheme has other security guarantees: revocability, testability and everlasting security. Furthermore, if you’re at sea and your quantum phone has only classical reception, you can use this scheme to transfer the value of the quantum money to shore*, said the fish, and swam away.

## The Normal Abstract

We introduce a new quantum cryptographic primitive which we call a tokenized signature scheme. Ordinary digital signatures are used to produce signed documents which are publicly verifiable but are unfeasible to forge by third parties. A tokenized signature scheme has the

---

<sup>1</sup>The Fisherman and His Wife by the brothers Grimm.

additional property that the signer can produce and distribute one-use quantum signing tokens that allow the holder to sign one (and only one) document of her choice.

We provide a candidate construction for tokenized signatures based on Aaronson and Christiano’s quantum money scheme. We also show that these schemes satisfy various other notions of security, including revocability, testability, and everlasting security. Finally, we show that any testable tokenized signature scheme can be used as a quantum money scheme with additional desirable properties, including the ability to turn a quantum coin into a classical check.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Tokens for Digital Signatures . . . . .	3
1.2	A Candidate Construction . . . . .	5
1.3	Computational requirements . . . . .	8
1.4	Application to Quantum Money . . . . .	8
1.5	Security assumptions . . . . .	9
1.6	Organization . . . . .	10
<b>2</b>	<b>Preliminaries and Notation</b>	<b>10</b>
2.1	Linear Algebra . . . . .	10
2.2	Cryptography . . . . .	11
<b>3</b>	<b>Definitions</b>	<b>12</b>
<b>4</b>	<b>Properties of Tokenized Signature Schemes</b>	<b>15</b>
<b>5</b>	<b>Reduction to a one-bit onetime Scheme</b>	<b>18</b>
<b>6</b>	<b>Security in the Oracle Model</b>	<b>25</b>
6.1	The Oracle Model . . . . .	25
6.2	Lower Bound . . . . .	26
<b>7</b>	<b>Breaking and Fixing the Aaronson-Christiano Scheme</b>	<b>30</b>
7.1	Overview . . . . .	30
7.2	Attack on the Noisy Scheme . . . . .	31
7.3	Fixing the Quantum Money Scheme . . . . .	32
<b>8</b>	<b>Sending Quantum Money over a Classical Channel</b>	<b>36</b>
8.1	Implications . . . . .	37
<b>9</b>	<b>Circumventing two faced behavior</b>	<b>38</b>

<b>10 Open Questions</b>	<b>38</b>
<b>11 Acknowledgments</b>	<b>39</b>
<b>A Nomenclature</b>	<b>43</b>
<b>B Memoryless Digital Signatures from Tokens</b>	<b>44</b>
B.1 Proving Super-Security and Unpredictability . . . . .	46
<b>C Message Authentication Codes</b>	<b>47</b>
C.1 Definitions . . . . .	47
C.2 Extending a 1-bit scheme to a full scheme . . . . .	49

## 1 Introduction

One of the main goals of cryptography is to allow an authorized party, typically holding a secret key, to perform an action which an unauthorized party (without the key) cannot. For example, in a digital signature scheme, the authorized party, Alice, holds a secret key that allows her to create digital signatures that will be accepted by a public verification algorithm. Anyone without Alice’s key cannot forge her signature.

In this work, we consider the task of delegating *limited* authorization: is it possible to provide a one-time access to the secret key to a third party? For example, if Alice goes on vacation, can she allow Bob to sign one (and only one) document of his choice?

Classically, Bob either knows the secret key or doesn’t, and there is no way to control how many times the key is used. But with quantum mechanics, the situation is different: the no cloning theorem [WZ82] allows us to create secrets that cannot be copied. Consequently, we propose the design of cryptographic schemes with two levels of secrets: one classical “master” secret, which is used only to generate any number of unclonable quantum “tokens”, each of which can be used to perform one action, and is consumed in the process due to destructive measurements. If Alice holds the secret key, she can delegate authorization to Bob by granting him a limited number of quantum tokens.

### 1.1 Tokens for Digital Signatures

This work applies the previous proposal regarding tokens specifically to digital signatures, allowing the delegation of limited authorization via the use of quantum tokens.

Digital signature is a cryptographic primitive which is arguably second in importance only to encryption. A digital signature scheme [DH76] consists of three Probabilistic Polynomial Time (PPT) algorithms: key-gen, sign, and verify. The first algorithm outputs a secret key  $sk$  and a public key  $pk$ . The signer can use  $sk$  to sign a document  $\alpha$  by calling  $\text{sign}(sk, \alpha)$ . This produces a signature  $sig$ , which can be verified by anyone holding the public key by calling  $\text{verify}(pk, \alpha, sig)$ .

We will denote the keys as subscripts, so these calls are  $\text{sign}_{sk}(\alpha)$  and  $\text{verify}_{pk}(\alpha, sig)$ . The verify algorithm returns either “accept” or “reject”. A valid signature should be accepted, so  $\text{verify}_{pk}(\alpha, \text{sign}_{sk}(\alpha))$  should accept for all  $\alpha$ . A digital signature scheme is *secure against chosen message attack* if an adversary with access to a signing oracle cannot efficiently generate any *new* signature (that is, a pair  $\alpha, sig$  for which  $\text{verify}_{pk}(\alpha, sig)$  accepts, where  $\alpha$  was not signed by the oracle), except with a negligible probability.

Our main contribution is a construction of a *tokenized* signature scheme. A tokenized signature scheme consists of 4 quantum polynomial time (QPT for short) algorithms: key-gen, token-gen, sign, and verify. In this setting there are *three* entities: a signer, a verifier and a new entity, which we will call the signing authority. In this context, even though the signer is the one signing the document, it is done in the name of the signing authority. The authority generates the pair  $(sk, pk)$  using key-gen as before. Next, it generates a quantum state  $|\underline{\sigma}\rangle$ , which we call a *signing token*, by running  $\text{token-gen}_{sk}$ .<sup>2</sup> Note that different calls to  $\text{token-gen}_{sk}$  may produce different signing tokens. The signer, who gets one copy of a signing token from the authority, can sign a single document of her choice. The output of  $\text{sign}(\alpha, |\underline{\sigma}\rangle)$  is classical, similarly to the classical setting. The correctness property remains unchanged:  $\text{verify}_{pk}(\alpha, \text{sign}(\alpha, |\underline{\sigma}\rangle))$  must accept for all documents  $\alpha$ .

The novelty of tokenized signatures is that sign applies a measurement which collapse which *consumes*  $|\underline{\sigma}\rangle$ , and therefore it cannot be reused to sign an additional document. Informally, the security requirement is that a QPT adversary **Adv**, with access to the public key  $pk$  and to  $\ell$  signing tokens  $|\underline{\sigma}_1\rangle \otimes \dots \otimes |\underline{\sigma}_\ell\rangle$ , cannot generate valid signatures for  $\ell + 1$  different documents.

Here are two motivating examples. A manager wants to hedge the embezzlement risks that an accountant introduces. This can be achieved by agreeing with the bank that any signed wire is limited to \$1000. Then, the manager can grant the accountant a number of tokens according to her level of trust in the accountant. Online computers are more prone to hacks than offline computers. A system administrator can hold the secret keys on an offline computer, and generate signing tokens to be used on the online computer. The common denominator of these examples is that the usage of tokens reduces risk, as the potential damage of abusing a token is smaller than that of the secret key.

A Tokenized signature scheme can be used as a digital signature scheme, up to some minor technicalities (see Theorem 11). Every tokenized digital scheme is also *revocable*: the signer can destroy a token in a publicly-verifiable way. A motivating example is an employee who receives signing tokens which were generated by the system administrator. Once the employee resigns, the employer can ask her to revoke her tokens. The employer only needs the public key (not the secret key, perhaps kept only by the system administrator) to run *revoke*.<sup>3</sup> If the (proclaimed) signing token passes revocation it is guaranteed that no unauthorized document was or will be signed by the employee.<sup>4</sup> The algorithm *revoke* is extremely simple (see text 4

---

<sup>2</sup>The symbol  $\underline{\sigma}$  represents a rubber stamp.

<sup>3</sup>The employee can provide the signed documents for the signing tokens that were already used, and the employer can determine whether they are valid and meet the firm’s policies.

<sup>4</sup>For this to work, the employee must cooperate and provide the previously signed documents and remaining signing

An even stronger notion of revocability is testability. In this case, the algorithm `verify-token` allows testing the signing token without consuming it. Not every tokenized signature scheme is necessarily testable, but the scheme we construct is. We show that every testable tokenized signature scheme can also be used as a public quantum money scheme (see Theorem 12).

If the adversary has no access to the public key, `revoke` and `verify-token` in some of our schemes have *everlasting security*, which means that a computationally unbounded quantum adversary  $\text{Adv}$  with access to a single copy of  $|\underline{\alpha}\rangle$  but without access to  $pk$ , cannot pass revocation, while also generating a valid signature for some document  $\alpha$ . Here is a motivating example. Consider a king who is going on a mission. The king provides the signing token to the heir to be used in case of an emergency, and the public key to the lawyer, accountant, and family doctor (but *not* the heir). Recall that the public key (which the heir does not have access to) is not needed for signing – it is only required for the verification. When the king returns from his mission, he may ask the heir to return the signing token. If `revoke` (or `verify-token`) passes, it is guaranteed that even if the heir is computationally unbounded, she can only generate a valid signature with negligible probability.

## 1.2 A Candidate Construction

We first reduce the problem of constructing a tokenized digital signature scheme to the problem of constructing a *single-bit* tokenized signature scheme, which can only sign a single document consisting of a single bit.

This extension is done in three steps. The first step is to extend the scheme to support a single document consisting of  $r$  bits. This can be done by essentially duplicating the single-bit protocol  $r$  times. In the second step, we extend the scheme to support documents of arbitrary size – rather than a size  $r$  which is fixed in advance – by employing a hash function (which we require to be collision-resistant against a quantum adversary). The idea is simply to hash the document down to  $r$  bits, and then to sign the hash. This is known as the hash-and-sign paradigm.

At this point, the scheme can sign a single document using a single token. Next, we need to extend it to a scheme that can support an arbitrary number of tokens. We do this by employing a classical digital signature scheme, which we require to be secure against chosen message attacks by a quantum adversary. The public key of the final tokenized scheme will be the public key of the classical signature scheme. Then, every time a token is required, it is generated by restarting the one-time tokenized scheme from scratch (using a new public and secret key each time), signing the public key, using the classical signature scheme, and attaching it to the token.

In Section 5, we prove the security of these three steps, and also show that they preserve several desirable properties. This reduces the problem of constructing a tokenized signature scheme to the problem of constructing a single-bit, single-use tokenized signature scheme.

The main challenge is to construct the single-bit tokenized scheme. Our construction for this is based on Aaronson and Christiano’s quantum money scheme [AC12, AC13]. In general, we do not know how to use a public quantum money scheme to provide a tokenized signature scheme

---

tokens. This mechanism is not suitable for workers that do not cooperate, or other scenarios such as an attacker who gains access to the signing tokens.

in a black-box way; however, we can use the specific construction of [AC12], based on hidden subspaces, to construct a tokenized signature scheme with many desirable properties.

Roughly speaking, the construction of Aaronson and Christiano works as follows. They consider a random subspace  $A$  of  $\mathbb{F}_2^n$  of dimension  $n/2$ , and the money state (in our case, this would be the signing token) uses  $n$  qubits, which are in the uniform superposition over  $A$ , denoted by  $|A\rangle$ . For example, for  $n = 4$ , the space  $\mathbb{F}_2^4$  consists of 16 vectors 0000, 0001,  $\dots$ , 1111, and one choice for the subspace  $A$  could be the vectors 0000, 0011, 1110, 1101. The addition operation is bitwise addition modulo 2 (XOR), for example:  $0011 \oplus 1110 = 1101$ . The state  $|A\rangle$  in this case is the 4 qubit state (a unit vector in  $\mathbb{C}^{2^4}$ ):  $\frac{1}{2}(|0000\rangle + |0011\rangle + |1110\rangle + |1101\rangle)$ .

Another important subspace of  $\mathbb{F}_2^n$  is  $A^\perp$ :

$$A^\perp = \{b \in \mathbb{F}_2^n \mid \forall a \in A, a \cdot b = \sum_{i=1}^n a_i b_i \pmod 2 = 0\}.$$

In the example above,  $A^\perp$  consists of the vectors 0000, 0111, 1011, 1100. Applying  $H^{\otimes n}$  (which can be implemented efficiently on a quantum computer by a circuit of depth 1) on  $|A\rangle$  gives the state  $|A^\perp\rangle$ :

$$H^{\otimes n}|A\rangle = |A^\perp\rangle = \frac{1}{2^{n/4}} \sum_{b \in A^\perp} |b\rangle,$$

where  $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ .

We can use the above properties to construct a *private* tokenized signature scheme. In the classical literature, there are two types of digital signatures: the standard one, which was described above; the other is a private (symmetric) digital signature scheme, known as *message authentication code* (MAC), in which verification requires the secret key. The difference between (public) digital signature scheme and (symmetric) MAC is very similar to the difference between public and symmetric key encryption.<sup>5</sup>

The signing authority samples a random  $n/2$  dimensional subspace  $A$  as the secret key, and generates  $|A\rangle$  as the signing token. The idea is to let any non-zero vector in  $A$  correspond to a signature for the bit 0, and a non-zero vector in  $A^\perp$  correspond to a signature for the bit 1. An adversary holding  $|A\rangle$  can either measure it in the standard basis to get a (uniformly random) element of  $A$ , or can measure  $|A^\perp\rangle$  to get an element of  $A^\perp$ . We show that an adversary with a single copy of  $|A\rangle$  cannot do both, since measuring the state collapses it. This is a manifestation of the no-cloning theorem[WZ82]: if the adversary could clone the state  $|A\rangle$ , and hold two copies of it, he could use the first copy to find an element of  $A$ , and the second copy to find an element of  $A^\perp$ , and break the security of the scheme. The verifier, which in the private setting knows the secret key (the choice of  $A$  and  $A^\perp$ ), can verify whether the signature is valid, by testing for the relevant subspace membership. This gives us a private tokenized signature scheme whose only

---

<sup>5</sup>An alternative term would be tokenized MAC, but we use private tokenized signature scheme, to follow the same conventions that were made between public and private, quantum money.

security assumption is a collision-resistant hash function secure against quantum adversaries, with all the analogous properties of the public tokenized signature scheme – see Appendix C.

How can we turn this into a public tokenized signature scheme? Suppose the adversary has one copy of  $|A\rangle$  as before, but additionally can ask questions of the form: is  $x$  in  $A$ ? Is  $y$  in  $A^\perp$ ? We use quantum query complexity techniques to prove that such an adversary still cannot efficiently find both a non-zero element of  $A$  and of  $A^\perp$  (see Section 6). These questions are sufficient to verify signatures. Therefore, the problem we face is the following: is there a way to obfuscate the membership for  $A$  and  $A^\perp$ ? The obfuscated program should allow the verifier to test whether an element is in the relevant subspace, but should not leak any additional information about  $A$  and  $A^\perp$ . Aaronson and Christiano faced the same challenge and suggested an ad-hoc approach for obfuscating these subspaces. Unfortunately, a variant of their scheme, known as the noise-free case, was broken in [PFP15]. This attack can be extended to the (main) noisy scheme, so that it is also broken; this observation was made by Paul Christiano, and is reported here in Section 7.2 for completeness.

is to let an element in  $A$  correspond to a signature for the bit 0, and to let an element in the dual space  $A^\perp$  correspond to a signature for the bit 1. An adversary holding  $|A\rangle$  can either measure it (in the standard basis) to get an element of  $A$ , or can measure  $|A\rangle$  in a different (Hadamard) basis to get an element of  $A^\perp$ . Intuitively, however, such an adversary cannot do both, since measuring the state collapses it.

and Christiano that proves the impossibility of cloning  $|A\rangle$  is not strong enough to prove the impossibility of finding an element from  $A$  and another from  $A^\perp$ . We fix this by strengthening their oracle result in Section 6.

of the oracle in Aaronson and Christiano’s quantum money scheme is now broken. A variant of their scheme, known as the noise-free case, was broken in [PFP15]. This attack can be extended to the (main) noisy scheme, so that it is also broken; this observation was made by Paul Christiano, and is reported here in Section 7.2 for completeness.

described earlier also holds for a private tokenized signature scheme, though some minor modifications are required. This gives us a private tokenized signature scheme whose only security assumption is a collision-resistant hash function secure against quantum adversaries, with all the analogous properties of the public tokenized signature scheme – see Appendix C.

This approach is based on ideas related to *obfuscation*, and the security of our construction is on shakier ground, as we will discuss in Section 1.5. We define a version of virtual black-box obfuscation for subspaces which would be sufficient to prove our scheme secure, and argue that the existence of such an obfuscation scheme is plausible (for instance, there are no known impossibility results that apply). Unfortunately, there are no candidate constructions which satisfy this requirement. We then conjecture that the indistinguishability obfuscation ( $i\mathcal{O}$ ) satisfies this definition; under this conjecture, we can prove our scheme secure. See Section 7 for discussion.

### 1.3 Computational requirements

One of the functions of money is store of value. From an engineering perspective, this makes quantum money (which our construction is based on) a challenging task, as it requires a long term quantum memory to store the quantum money. Signing tokens do not necessarily need to be stored for long periods (perhaps, only the signed documents are), and therefore some applications may require relatively short term memory, and will thus be easier to implement in practice. Furthermore, signing requires a very simple (depth 1) quantum circuit, and verification is done on a classical computer. Our tokens can be generated using Clifford circuits, which do not require a universal quantum computer, and therefore may be easier to implement. The only task which requires a universal quantum computer is verify-token – which is not part of the minimal requirements for tokenized signatures, but is required for constructing public quantum money out of it. The security of the weak scheme is exponential in the number of qubits; this means, among other things, that the depth of token-gen in all our schemes is roughly logarithmic (in the security parameter).

### 1.4 Application to Quantum Money

We show that a testable tokenized signature scheme can be used as quantum money [Wie83, BBBW83, TOI03, MS10, Aar09, Gav12, AC13, FGH<sup>+</sup>12, GK15]. Interestingly, the quantum money scheme we get this way has a combination of desirable properties that no previous scheme had, not even the Aaronson-Christianano scheme on which our construction is based.

To get a public quantum money scheme from a testable tokenized signature scheme, simply use the signing tokens as money. We show in Theorem 12 that this satisfies the definition of a public quantum money scheme. An interesting property of this scheme is that the money can be converted into a publicly-verifiable signature for a document. This can be used to send the money over a classical channel (vis-à-vis “standard” quantum money, which can only be sent via a quantum channel).

In particular, suppose Alice wants to send money to Bob, but she is stranded and cannot get quantum internet reception on her quantum cellphone. Can she send the money using only classical communication? With our scheme, Alice can use her money as a signing token to sign the document “I’m sending this money, with the serial number 03217, to Bob”. Bob can then take the signed document and present it to his bank. The bank knows that Alice must have burned her money to produce the signed document, and so it can safely issue Bob a new money state. In essence, Alice can convert a quantum coin into a classical “check”.

Another important property is that traceability of money transactions is optional. In a public money scheme, transactions are generally traceless, meaning that third parties do not need to know the identities of the buyer and seller. This is a double-edged sword: it is desirable for privacy reasons, but it also prevents resolution for payment disputes; a seller can always deny being paid even after receiving the money. In our scheme, however, the buyer has a choice of the payment method: she can either hand over the physical quantum state (which is traceless), or use the money state to sign a payment check, as before. If she opts for the latter, she gains the ability to prove to



a third party that she has paid the merchant.

This alternate payment method is also secure against a variety of attacks. For instance, anyone eavesdropping on the transaction cannot get access to the money (indeed, Alice may as well broadcast her payment publicly). Furthermore, the bank branch needs only the public key in order to verify that the payment is valid, meaning that individual branches do not need to store the secret key at all – so if the branch is hacked, it loses the quantum money it stored in its safe, but no further damage to the monetary system is inflicted; the thief does not gain the ability to mint new money.

## 1.5 Security assumptions

In light of the failure of various recent cryptographic schemes, we should carefully discuss our security assumptions.

First, our construction suffices for a private tokenized signature scheme, under very mild cryptographic assumptions: the existence of a collision-resistant hash function (secure against quantum adversaries). There are various candidates for collision-resistant hash functions that are believed to be secure, so this construction appears to be on solid ground. All of our main results apply equally well for the private scheme – see Appendix C.

Second, several of our results, such as the construction of public quantum money (that can be turned into classical “checks”) from any public tokenized signature scheme, as well as many others (Theorems 10, 11, 13, 14, 15) hold without any assumptions. For this reason, we believe that a public tokenized signature scheme is an interesting primitive even if our current construction fails.

For the construction of a *public* tokenized digital signature scheme, we can show our protocol is provably secure relative to an oracle. Instantiating the oracle can be done via a certain type of black-box obfuscation for a specific class of functions<sup>6</sup>; we hence prove a reduction from tokenized digital signatures to a certain type of obfuscation. The obfuscation part is the only problematic step. There has been a virtual black-box obfuscation proposal for a class of function that are similar to what we need, but it does not exactly match our requirements (see Section 7.3). Instead, we argue that some recent *iO* constructions likely satisfy our obfuscation definition. We explain the plausibility of this assumption in Section 7.3, though it remains significantly shakier than the assumptions we need for the private scheme.

We stress that even if general *iO* does not exist, it seems plausible that *some* virtual black-box obfuscation exists for our class of functions. But even if you are a die-hard skeptic that disbelieves all obfuscation claims, it at least remains plausible that some *other* construction for a tokenized digital signature scheme exists – if only because it exists relative to an oracle. One of our main contributions in this work is the idea of tokenized signature schemes, and the various applications of this primitive, which is independent of the security of our candidate construction. Furthermore, we hope it will spur further research on the unresolved topics which are outside the scope of this paper, such as explicit virtual black-box obfuscation constructions for subspaces, etc.

---

<sup>6</sup>Recall that while general black-box obfuscation is impossible [BGI<sup>+</sup>12], the impossibility result does not rule out black-box obfuscations for restricted classes of functions.

## 1.6 Organization

In Section 2 we provide preliminary definitions and notation, in Section 3 we introduce new definitions, in Section 4 we shows several properties of tokenized signatures schemes. Section 5 strengthens a very weak form of tokenized signatures to a full blown tokenized signature scheme. We show a query complexity lower bound in Section 6, which is used later to construct the weak form of tokenized signatures. In Section 7 we revisit Aaronson and Christiano’s quantum money scheme, report on an attack by Christiano which makes their original proposal insecure, and then propose how to fix it. This section also contains the weak form of tokenized signature protocol, and we discuss the security of it. In Section 8 we show an application: how tokenized signatures can be used to transfer the value of quantum money, without using quantum communication, and another application in the context of distributed algorithms, in Section 9.

Nomenclature is provided in Appendix A. In Appendix C we show how to construct a *private* tokenized signatures scheme, under weaker assumptions than required for the public scheme.

## 2 Preliminaries and Notation

### 2.1 Linear Algebra

For any subspace  $A \prec \mathbb{F}_2^n$ , let

$$A^\perp = \{b \in \mathbb{F}_2^n \mid \forall a \in A, a \cdot b = \sum_{i=1}^n a_i b_i \pmod{2} = 0\}.$$

A basis for  $A$  will be denoted  $\langle A \rangle$ . For any subspace  $A$ ,  $A^\perp$  is also a subspace, which satisfies

$$(A^\perp)^\perp = A$$

and

$$\dim(A) + \dim(A^\perp) = n. \tag{1}$$

We denote by  $\chi_A$  the indicator function for  $A$ , and similarly for  $\chi_{A^\perp}$ :

$$\chi_A(a) = \begin{cases} 1 & a \in A \\ 0 & \text{otherwise.} \end{cases}$$

We denote the unified membership function  $\chi_{A^*}$ , which combines  $\chi_A$  and  $\chi_{A^\perp}$ :

$$\chi_{A^*}(a, p) = \begin{cases} \chi_A(a) & p = 0 \\ \chi_{A^\perp}(a) & p = 1 \end{cases} \tag{2}$$

For even  $n$ , let  $S(n)$  be the set of all subspaces of  $\mathbb{F}_2^n$  with dimension  $\frac{n}{2}$ . When  $n$  is clear from the context we simply write  $S$ . For any  $A \in S(n)$ , we define the state  $|A\rangle$ :

$$|A\rangle = \frac{1}{\sqrt{2^{n/2}}} \sum_{a \in A} |a\rangle, \tag{3}$$

and similarly for  $|A^\perp\rangle$  (here we use Eq. (1) to justify the normalization coefficient):

$$|A^\perp\rangle = \frac{1}{\sqrt{2^{n/2}}} \sum_{a \in A^\perp} |a\rangle.$$

The process in which  $x$  is sampled uniformly from a finite set  $X$  is denoted by  $x \stackrel{R}{\leftarrow} X$ , and  $output \leftarrow \text{Alg}(input)$  denotes sampling  $output$  according to the distribution generated by running  $\text{Alg}(input)$ .

## 2.2 Cryptography

For a function  $f : \mathbb{N} \mapsto \mathbb{R}^+$  we use the shorthand  $f(\kappa) \leq \text{negl}(\kappa)$  to say that  $f$  is negligible: for every polynomial  $p(\kappa)$ , there exists a large enough  $\kappa_0$  such that  $f(\kappa) \leq p(\kappa)$  for all  $\kappa \geq \kappa_0$ . We use the shorthand  $f(\kappa) \geq \text{non-negl}(\kappa)$  to say that  $f$  is non-negligible. We say that  $f$  is super-logarithmic if  $f(n) = \omega(\log n)$ .

**Definition 1** (Collision resistant hashing scheme, adapted from [Gol04]). Let  $r(\kappa) : \mathbb{N} \mapsto \mathbb{N}$ . A collection  $\{h_s : \{0, 1\}^* \mapsto \{0, 1\}^{r(|s|)}\}_{s \in \{0, 1\}^*}$  is called a collision resistant hashing scheme against quantum adversaries if there exists a PPT algorithm  $\text{index}$  such that the following holds:

1. There exists a polynomial time algorithm, that given  $s$  and  $x$ , returns  $h_s(x)$ .
2. We say that the pair  $x \neq x'$  forms a collision under the function  $h$  if  $h(x) = h(x')$ . We require that for every QPT  $\text{Adv}$ , given  $\text{index}(1^\kappa)$  as an input, outputs a collision under  $h_{\text{index}(1^\kappa)}$  with negligible probability:

$$\Pr \left[ \text{Adv}(\text{index}(1^\kappa)) \text{ is a collision under } h_{\text{index}(1^\kappa)} \right] \leq \text{negl}(\kappa)$$

3. For some polynomial  $p$ , all sufficiently large  $\kappa$ 's, and every  $s$  in the range of  $\text{index}(1^\kappa)$ , it holds that  $\kappa \leq p(|s|)$ . Furthermore,  $\kappa$  can be computed in polynomial time from  $s$ .<sup>7</sup>

**Definition 2** (Digital Signature Scheme). A digital signature (DS) scheme consists of 3 PPT algorithms  $\text{key-gen}$ ,  $\text{sign}$  and  $\text{verify}$  satisfying, for every  $\alpha \in \{0, 1\}^*$ :

1. When a document is signed using the private key, the signature is accepted by the verification algorithm using the public key:

$$\Pr \left[ \text{verify}_{pk}(\alpha, \text{sign}_{sk}(\alpha)) = T \right] = 1$$

---

<sup>7</sup> The main purpose of this technical requirement is to allow the adversary to run in time which is polynomial in  $\kappa$ .

2. The scheme is secure against existential forgery under chosen message attacks; that is, an adversary with the capability of adaptively requesting documents to be signed by a signing oracle cannot generate a signature for any document they did not ask the oracle about:

$$\Pr \left[ (\alpha, sig) \leftarrow \text{Adv}^{\text{sign}_{sk}}(pk), \text{verify}_{pk}(\alpha, sig) = T \text{ and } \alpha \notin Q_{\text{Adv}}^{\text{sign}_{sk}} \right] \leq \text{negl}(\kappa), \quad (4)$$

where  $\text{Adv}^{\text{sign}_{sk}}$  is a QPT algorithm with access to the signing oracle, and  $Q_{\text{Adv}}^{\text{sign}_{sk}}$  is the set of queries it made to the oracle.

In the above definition, the probabilities range over  $(pk, sk) \leftarrow \text{key-gen}(1^\kappa)$  as well as the internal randomness of all the algorithms. This will also be the case in all our future definitions.

We say a digital signature scheme is deterministic if  $\text{verify}$  is deterministic. We also define a related notion called a *memory-dependent signature scheme (MDS)*. Such a scheme is defined similarly to the above, except the chosen messages of the adversary must all be unique. It is clear that every DS is an MDS. The difference comes down to a technicality; it is generally easier to prove a scheme is an MDS than a DS, and it provides almost as much security (the signing protocol of an MDS scheme can be made secure against all chosen message attacks if it had memory, allowing it to respond to two signing requests for the same document with the same signature).

*Remark 1.* We will always assume (as we did in a Eq. (4)) that the public key  $pk$  and the secret key  $sk$  are the output of  $\text{key-gen}(1^\kappa)$  of the relevant protocol, and that  $|\underline{\omega}\rangle$  is the output of  $\text{token-gen}_{sk}$  (to be defined later) and the probabilities always take that into account.

### 3 Definitions

**Definition 3** (Public quantum money scheme<sup>8</sup>). A public quantum money scheme consists of three QPT algorithms:

1.  $\text{key-gen}$ , which takes as input the security parameter  $1^\kappa$  and probabilistically generates a key pair  $pk, sk$ .
2.  $\text{token-gen}$ , which takes as an input  $sk$  and generates a quantum state  $\$$  called a (bank) token.
3.  $\text{verify-token}$  which takes as input  $pk$  and an alleged bank token  $\tau$  and either accepts or rejects.

$\text{verify-token}$  should accept a valid bank tokens  $\$$ :

$$\Pr \left[ \$ \leftarrow \text{token-gen}_{sk}, \text{verify-token}_{pk}(\$) = T \right] = 1.$$

Let  $\text{verify-token}_{k,pk}$  take as an input  $pk$ , as well as a collection of (possibly entangled)  $k$  alleged bank tokens  $\tau_1, \dots, \tau_k$  and accepts if and only if for every  $1 \leq i \leq k$ ,  $\text{verify-token}_{pk}(\tau_i)$  accepts.

---

<sup>8</sup>Also known as locally verifiable quantum money [MS10]

We say that the scheme is weakly secure if for every QPT Adv which maps  $\ell = \text{poly}(\kappa)$  valid tokens  $\$1, \dots, \$\ell$  to  $\ell + 1$  (possibly entangled) alleged tokens  $\tau_1, \dots, \tau_{\ell+1}$ :<sup>9</sup>

$$\Pr \left[ \text{verify-token}_{\ell+1, pk}(\tau_1, \dots, \tau_{\ell+1}) = T \right] \leq \text{negl}(\kappa) \quad (5)$$

The scheme is secure if additionally, for every QPT Adv with access to  $pk$  and polynomially many quantum money states, which generates a state  $\sigma$ :

$$\Pr \left[ \text{verify-token}_{pk}(\tau) = T \mid (T, \tau) \leftarrow \text{verify-token}_{pk}(\sigma) \right] \geq 1 - \text{negl}(\kappa) \quad (6)$$

*Remark 2.* To motivate the requirement in Eq. (6) (which had not appeared in the literature, as far as we know), consider an adversary which aims to harm a specific merchant. Without the above requirement, one possible attack for the adversary (and there are other attacks with similar flavor) is to distort a quantum money state so that it will pass the verification test *exactly* once. That means that when the merchant will use that state again, say, to buy goods, the verification will fail (since it is the second verification). Even worse, the merchant might be accused of fraud. Eq. (6) guarantees that if a state passes the verification, it will continue to pass further verifications (except with a negligible probability).

Aaronson and Christiano used a projective scheme which is a stronger requirement than Eq. (6) (see [AC12] for the definition of a projective scheme), but they did not consider it as part of the security requirement for quantum money.

**Definition 4** (Tokenized signature scheme). A tokenized signature (TS) scheme consists of 4 QPT algorithms, key-gen, token-gen, sign, and verify with the following properties:

1. On input  $1^\kappa$  where  $\kappa$  is the security parameter, key-gen outputs a classical public keys  $pk$  and a secret key  $sk$ .
2. token-gen $_{sk}$  generates a signing token  $|\underline{\$}\rangle$ . We emphasize that if token-gen $_{sk}$  is called  $\ell$  times it may (and in our construction, would) output different states  $|\underline{\$}_1\rangle, \dots, |\underline{\$}_\ell\rangle$ .
3. For every document  $\alpha \in \{0, 1\}^*$ ,

$$\Pr \left[ |\underline{\$}\rangle \leftarrow \text{token-gen}_{sk}, \text{verify}_{pk}(\alpha, \text{sign}(\alpha, |\underline{\$}\rangle)) = T \right] = 1.$$

---

<sup>9</sup>Aaronson and Christiano used a slightly different definition described next. These definitions are clearly equivalent for *perfect completeness* schemes, which are our main focus, and the way we defined tokenized signatures and quantum money. Their definition is as follows. Let count take as input  $pk$ , as well as a collection of (possibly-entangled) alleged tokens  $\tau_1, \dots, \tau_\ell$  and output the number of indices  $i \in [r]$  such that verify-token $_{pk}(\tau_i)$  accepts. We say that the scheme is secure if for every QPT Adv which maps  $\ell = \text{poly}(\kappa)$  valid tokens  $\$1, \dots, \$\ell$  to  $m$  (possibly entangled) alleged tokens  $\tau_1, \dots, \tau_m$ :

$$\Pr [\text{count}(\tau_1, \dots, \tau_m) > \ell] \leq \text{negl}(\kappa)$$

In an imperfect scheme, the above requirement is relaxed:

$$\Pr \left[ |\mathfrak{D}\rangle \leftarrow \text{token-gen}_{sk}, \text{verify}_{pk}(\alpha, \text{sign}(\alpha, |\mathfrak{D}\rangle)) = T \right] \geq 1 - \text{negl}(\kappa).$$

All the results in this paper can be easily extended to imperfect schemes – this is usually omitted to improve the presentation. Our main construction is imperfect.

**Definition 5** (Length restricted TS scheme). A TS scheme is  $r$ -restricted if Item 3 holds only for  $\alpha \in \{0, 1\}^r$ .

To define unforgeability, we introduce the algorithm  $\text{verify}_{k,pk}$ . This algorithm takes as an input  $k$  pairs  $(\alpha_1, s_1), \dots, (\alpha_k, s_k)$  and accepts if and only if (i) all the documents are distinct, i.e.  $\alpha_i \neq \alpha_j$  for every  $1 \leq i \neq j \leq k$ , and (ii) all the pairs pass the verification test  $\text{verify}_{pk}(\alpha_i, s_i)$ .

**Definition 6** (Unforgeability and onetime unforgeability). A TS scheme is unforgeable if for every  $\ell = \text{poly}(\kappa)$  a QPT adversary cannot sign  $\ell + 1$  different documents by using the public key and  $\ell$  signing tokens:

$$\Pr \left[ \text{verify}_{\ell+1,pk}(\text{Adv}(pk, |\mathfrak{D}_1\rangle \otimes \dots \otimes |\mathfrak{D}_\ell\rangle)) = T \right] \leq \text{negl}(\kappa) \quad (7)$$

Onetime unforgeability requires the above only for  $\ell = 1$ .

Note that the above requirement for  $\ell \neq 0$  is stronger than the requirement for key-only attack ( $\ell = 0$ ): it implies the that an adversary with access to the public key (and without signing tokens) cannot find a pair  $(\alpha, s)$  which would pass verification:

$$\Pr \left[ (\alpha, s) \leftarrow \text{Adv}(pk), \text{verify}_{pk}(\alpha, s) \right] \leq \text{negl}(\kappa)$$

Otherwise, the adversary could use his  $\ell$  signing tokens to sign additional  $\ell$  different documents, and violate the security assumption in Eq. (7).

We define  $\text{revoke}_{k,pk}$  in a similar way to  $\text{verify}_{k,pk}$ : it receives as an input  $k$  registers, preforms  $\text{revoke}_{pk}$  on each one of them, and accepts iff all of the revocations accepted.

**Definition 7** (Revocability). A revocable tokenized signature scheme adds a fifth algorithm,  $\text{revoke}$  which satisfies:

1.  $\Pr [\text{revoke}_{pk}(|\mathfrak{D}\rangle) = T] = 1$ .
2. For every  $\ell \leq \text{poly}(\kappa)$ ,  $t \leq \ell$ , and every QPT  $\text{Adv}$  with  $\ell$  signing tokens  $|\mathfrak{D}_1\rangle \otimes \dots \otimes |\mathfrak{D}_\ell\rangle$  and  $pk$  which generates  $\alpha_1, \text{sig}_1, \dots, \alpha_t, \text{sig}_t, \sigma$

$$\Pr \left[ \text{verify}_{t,pk}(\alpha_1, \text{sig}_1, \dots, \alpha_t, \text{sig}_t) = T \wedge \text{revoke}_{\ell-t+1,pk}(\sigma) = T \right] \leq \text{negl}(\kappa).$$

In imperfect schemes, we relax item 1 so that the r.h.s. is at least  $1 - \text{negl}(\kappa)$ .

As was mentioned in the introduction, every TS scheme (but not necessarily length restricted TS scheme) is revocable.

A testable TS scheme has an additional algorithm `verify-token`, which, unlike `revoke`, does not consume the signing token. If a state passes this test, it can be used to sign a document.

**Definition 8** (Testability). A testable TS scheme adds a fifth QPT algorithm to a TS scheme, `verify-token`, which satisfies:

1.  $\Pr \left[ \text{verify-token}_{pk}(|\underline{\sigma}\rangle) = (T, |\underline{\sigma}\rangle) \right] = 1.$
2. For every QPT  $\text{Adv}$  with access to  $pk$  and  $\text{poly}(\kappa)$  many signing tokens, which generates  $\alpha$  and a state  $\tau$ :

$$\Pr \left[ \text{verify}_{pk}(\alpha, \text{sign}(\alpha, \sigma)) = T \mid (T, \sigma) \leftarrow \text{verify-token}_{pk}(\tau) \right] \geq 1 - \text{negl}(\kappa) \quad (8)$$

$$\Pr \left[ \text{verify-token}_{pk}(\sigma) = T \mid (T, \sigma) \leftarrow \text{verify-token}_{pk}(\tau) \right] \geq 1 - \text{negl}(\kappa) \quad (9)$$

These requirements are added for the same reasons as in Remark 2.

**Definition 9** (Everlasting Security.). A revocable TS scheme has everlasting security if a computationally unbounded quantum adversary  $\text{Adv}$  with access to  $\ell = \text{poly}(\kappa)$  signing tokens  $|\underline{\sigma}_1\rangle \otimes \dots \otimes |\underline{\sigma}_\ell\rangle$  but without access to  $pk$ ,

$$\Pr \left[ (\rho, \alpha, s) \leftarrow \text{Adv}(|\underline{\sigma}_1\rangle \otimes \dots \otimes |\underline{\sigma}_\ell\rangle), \text{revoke}_{\ell, pk}(\rho) = T \text{ and } \text{verify}_{pk}(\alpha, s) = T \right] \leq \text{negl}(k) \quad (10)$$

A testable scheme has everlasting security if `revoke` in Eq. (10) is replaced with `verify-token`. One-time everlasting security requires the above only for  $\ell = 1$ .

## 4 Properties of Tokenized Signature Schemes

In this section we prove several properties of tokenized signature schemes.

**Theorem 10.** *Every unforgeable tokenized signature scheme is also revocable (see Definition 7) by the algorithm `revoke` given in Algorithm 1.*

---

### Algorithm 1 `revoke`

---

- 1: **procedure** `revoke`( $pk, \sigma$ )
  - 2:      $\alpha \xleftarrow{R} \{0, 1\}^{f(\kappa)}$   $\triangleright f$  can be any super-logarithmic function.
  - 3:     **return** `verify` $_{pk}(\alpha, \text{sign}(\alpha, \sigma))$
  - 4: **end procedure**
-

Next, we show that a variant of a tokenized signature scheme can be used as a digital signature scheme.

Let TS be an tokenized signature scheme. We define the scheme DS which uses TS in a black box manner. In DS, key-gen and verify are the same as in TS, and  $\text{DS.sign}_{sk}(\alpha)$  is done by:  $\text{TS.sign}(\alpha, \text{TS.key-gen}_{sk})$ .

**Theorem 11.** *If TS is a tokenized signature scheme then the scheme DS as defined above is a memory-dependent digital signature scheme (see Definition 2).*

While this only gives us a memory-dependent digital signature scheme, it can be strengthened to a general digital signature scheme given additional technical assumptions about the tokenized signature scheme; see Appendix B.

**Theorem 12.** *Every testable (see Definition 8) tokenized signature scheme is a public quantum money scheme (see Definition 3).*

*Proof of Theorem 10.* Assume towards a contradiction that revoke given in Algorithm 1 does not satisfy Definition 7, i.e. there exists  $\ell = \text{poly}(\kappa)$  and  $t$ , and a QPT Adv which generates  $\alpha_1, \text{sig}_1, \dots, \alpha_t, \text{sig}_t$  and a state  $\sigma$  such that

$$\Pr \left[ \text{verify}_{t,pk}(\alpha_1, \text{sig}_1, \dots, \alpha_t, \text{sig}_t) = T \wedge \text{revoke}_{\ell-t+1,pk}(\sigma) = T \right] \geq \text{non-negl}(\kappa).$$

We define Adv' which uses Adv and simulates revoke, which can be used to break the unforgeability of the TS scheme. The adversary Adv' starts by running Adv, and then uses the state  $\sigma$  to produce signatures for additional  $\ell$  random documents, see Algorithm 2.

---

**Algorithm 2** Adv': the adversary used in the proof of Theorem 10

---

```

1: procedure ADV'(pk,  $|\underline{\sigma}_1\rangle \otimes \dots \otimes |\underline{\sigma}_\ell\rangle$ )
2:    $(\alpha_1, \text{sig}_1, \dots, \alpha_t, \text{sig}_t, \sigma_{t+1}, \dots, \sigma_{\ell+1}) \leftarrow \text{Adv}(pk, |\underline{\sigma}_1\rangle \otimes \dots \otimes |\underline{\sigma}_\ell\rangle)$ 
3:   for all  $i \in \{t+1, \dots, \ell+1\}$  do
4:      $\alpha_i \xleftarrow{R} \{0, 1\}^{f(\kappa)}$   $\triangleright f$  can be any super-logarithmic function .
5:      $\text{sig}_i \leftarrow \text{sign}(\alpha_i, \sigma_i)$ 
6:   end for
7:   return  $(\alpha, \text{sig})$ 
8: end procedure

```

---

By construction,



$$\begin{aligned}
& \Pr[\text{verify}_{\ell+1,pk}(\text{Adv}'(pk, |\mathfrak{D}_1\rangle \otimes \dots \otimes |\mathfrak{D}_\ell\rangle)) = T] \\
&= \Pr\left[\text{verify}_{t,pk}(\alpha_1, \text{sig}_1, \dots, \alpha_t, \text{sig}_t) = T \wedge \text{revoke}_{\ell-t+1,pk}(\sigma) = T \wedge \forall i \neq j, \alpha_i \neq \alpha_j\right] \\
&\geq \Pr\left[\text{verify}_{t,pk}(\alpha_1, \text{sig}_1, \dots, \alpha_t, \text{sig}_t) = T \wedge \text{revoke}_{\ell-t+1,pk}(\sigma) = T\right] - \binom{\ell+1}{2} 2^{-f(\kappa)} \\
&\geq \text{non-negl}(\kappa)
\end{aligned}$$

which contradicts unforgeability (see Definition 6).  $\square$

*Proof of Theorem 11.* It's clear that a valid signature will be accepted by this scheme, so we only need to prove security. Suppose there was an adversary who used *distinct* adaptive queries to the signing oracle and produced a signature for a new document, violating Eq. (4). Then this adversary produced valid signatures for  $\ell + 1$  different<sup>10</sup> documents using only  $\ell$  calls to the oracle. But this means it could produce  $\ell + 1$  valid signatures of distinct documents using only  $\ell$  signing tokens, contradicting the security assumption of the tokenized signature scheme.  $\square$

*Proof of Theorem 12.* A testable tokenized signature scheme already has the correctness property of quantum money, i.e.  $\Pr(\text{verify-token}_{pk}(|\mathfrak{D}\rangle)) = 1$ . Also Eq. (6) holds. The only property which is left to show is Eq. (5). Suppose it does not, i.e. there exists a QPT  $\text{Adv}$  which maps  $pk, |\mathfrak{D}_1\rangle, \dots, |\mathfrak{D}_\ell\rangle$ , for  $\ell = \text{poly}(\kappa)$  to a state  $\tau$  with  $\ell + 1$  registers such that

$$\Pr\left[\text{verify-token}_{\ell+1,pk}(\tau) = T\right] \geq \text{non-negl}(\kappa). \quad (11)$$

$\square$

Let  $\text{Adv}'$  be the adversary as defined in Algorithm 3.

$$\begin{aligned}
& \Pr\left[\text{verify}_{\ell+1,pk}(\text{Adv}'(pk, |\mathfrak{D}_1\rangle \otimes \dots \otimes |\mathfrak{D}_\ell\rangle)) = T\right] \\
&= \Pr\left[\text{verify-token}_{\ell+1,pk}(\tau) = (T, \sigma) \wedge \text{verify}_{\ell+1,pk}(\alpha, \text{sig})\right] \\
&= \Pr\left[\text{verify-token}_{\ell+1,pk}(\tau) = (T, \sigma) \bigwedge_{i \in [\ell+1]} \text{verify}_{pk}(\alpha_i, \text{sign}(\alpha_i, \sigma_i))\right] \\
&\geq \text{non-negl}(\kappa)
\end{aligned} \quad (12)$$

where in the last step we used Eq. (11) and Eq. (8). Eq. (12) contradicts unforgeability (Definition 6).

<sup>10</sup>Here we used the property that the documents given for the signing oracle are distinct.

---

**Algorithm 3**  $\text{Adv}'$ : the adversary used in the proof of Theorem 12

---

```

1: procedure  $\text{ADV}'(pk, |\mathfrak{D}_1\rangle \otimes \dots \otimes |\mathfrak{D}_\ell\rangle)$ 
2:    $\tau \leftarrow \text{Adv}(pk, |\mathfrak{D}_1\rangle \otimes \dots \otimes |\mathfrak{D}_\ell\rangle)$ 
3:   if  $\text{verify-token}_{\ell+1, pk}(\tau) = (T, \sigma_1, \dots, \sigma_{\ell+1})$  then
4:     for all  $i \in [\ell + 1]$  do
5:        $\alpha_i \leftarrow i$ 
6:        $\text{sig}_i \leftarrow \text{sign}(\alpha_i, \sigma_i)$ 
7:     end for
8:   end if
9:   return  $(\alpha, \text{sig})$ 
10: end procedure

```

---

## 5 Reduction to a one-bit onetime Scheme

In this section we show how a 1-bit length restricted (see Def. 5) onetime (see Def. 6) tokenized signature scheme, denoted OT1, can be strengthened to a full-fledged tokenized signature scheme (i.e used to sign an arbitrary number of documents, of arbitrary length).

This is done using three reductions, which construct three additional schemes. The second scheme, OTR (see Algorithm 4), is onetime  $r$ -restricted: it is achieved by using the OT1 scheme  $r$  times. The third scheme, OT (see Algorithm 5), removes the length restriction by using the hash-and-sign paradigm: instead of signing the document  $\alpha$  (which might be arbitrarily long), we sign the document  $h_s(\alpha)$  where  $h_s$  is a collision resistant hash function. The proof is the same as the classical security proof of the hash-and-sign paradigm (see, for example, [Gol04]). The fourth scheme, TS (see Algorithm 6), uses a (quantum resistant) digital signature scheme (denoted  $DS$ ). In this scheme the public key used in the previous OT scheme is signed using a (quantum resistant) digital signature scheme, and added to the quantum money state. The public and secret keys are generated using the digital signature scheme.

In rest of this section we show that all the reductions maintain unforgeability (see Definition 6), testability (Definition 8), and that the first two reductions (but not the last) maintain everlasting security (see Definition 9). We will always assume that  $r \leq \text{poly}(\kappa)$ .

- Theorem 13.**
1. *If OT1 is a 1-restricted onetime unforgeable TS scheme, then OTR is  $r$ -restricted onetime TS scheme.*
  2. *If OTR is  $r$ -restricted onetime TS scheme and  $h_s$  is a collision resistant hash function, then OT is onetime TS scheme.*
  3. *If OT is a onetime TS scheme and DS is a digital signature scheme, then TS is a (full blown) TS scheme.*

*Proof.* We start by proving item 1. The correctness property for documents of length  $r$  follows

---

**Algorithm 4** OTR: Onetime  $r$ -restricted scheme

---

```
1: procedure key-gen( $1^\kappa, 1^r$ )
2:   for all  $i \in [r]$  do
3:      $(pk_i, sk_i) \leftarrow \text{OT1.key-gen}(1^n)$ 
4:   end for
5:   return  $(pk, sk)$ 
6: end procedure
7: procedure token-gen( $sk$ )
8:   for all  $i \in [r]$  do
9:      $(|\underline{\sigma}_i\rangle) \leftarrow \text{OT1.token-gen}(sk_i)$ 
10:  end for
11:  return  $|\underline{\sigma}\rangle = |\underline{\sigma}_1\rangle \otimes \dots \otimes |\underline{\sigma}_r\rangle$ 
12: end procedure
13: procedure sign( $\alpha \in \{0, 1\}^r, |\underline{\sigma}\rangle$ )
14:  for all  $i \in [r]$  do
15:     $sig_i = \text{OT1.sign}(\alpha_i, |\underline{\sigma}_i\rangle)$ 
16:  end for
17:  return  $sig$ 
18: end procedure
19: procedure verify $_{pk}(\alpha \in \{0, 1\}^r, sig)$ 
20:  for all  $i \in [r]$  do
21:    if  $\text{OT1.verify}(\alpha_i, sig_i) = F$  then return F
22:    end if
23:  end for
24:  return T
25: end procedure
26: procedure verify-token $_{pk}(\tau)$ 
27:  if for all  $i \in [r]$   $\text{OT1.verify-token}_{pk_i}(\tau_i) = (T, \sigma_i)$  then
28:    return  $(T, \sigma)$ 
29:  else
30:    return  $F$ 
31:  end if
32: end procedure
```

---

---

**Algorithm 5** OT: onetime scheme

---

```
1: procedure key-gen( $1^\kappa$ )
2:    $s \leftarrow \text{index}(1^\kappa)$ 
3:    $(pk', sk') \leftarrow \text{OTR.key-gen}(1^\kappa, 1^{r(\kappa)})$ 
4:    $sk \leftarrow (sk', s)$ 
5:    $pk \leftarrow (pk', s)$ 
6:   return  $(pk, sk)$ 
7: end procedure
8: procedure token-gen( $sk$ )
9:    $|\underline{\sigma}'\rangle \leftarrow \text{OTR.token-gen}(sk')$ 
10:   $|\underline{\sigma}\rangle \leftarrow (s, |\underline{\sigma}'\rangle)$ 
11:  return  $|\underline{\sigma}\rangle$ 
12: end procedure
13: procedure sign( $\alpha \in \{0, 1\}^*, (s, |\underline{\sigma}\rangle)$ )
14:  return  $\text{OTR.sign}(h_s(\alpha), |\underline{\sigma}\rangle)$ 
15: end procedure
16: procedure verify $_{pk}$ ( $\alpha \in \{0, 1\}^*, sig$ )
17:  return  $\text{OTR.verify}_{pk'}(h_s(\alpha), sig)$ 
18: end procedure
19: procedure verify-token $_{pk}(s', \tau)$ 
20:  if  $s = s'$  and  $\text{OTR.verify-token}_{pk}(\tau) = (T, \sigma)$  then
21:    return  $(T, s, \sigma)$ 
22:  else
23:    return  $F$ 
24:  end if
25: end procedure
```

---

▷ See Definition 1 for index.  
▷ See Definition 1 for  $r(\kappa)$ .

---

**Algorithm 6** TS: Tokenized Signature scheme

---

```
1: procedure key-gen( $1^\kappa$ )
2:    $(pk, sk) \leftarrow \text{DS.key-gen}(\kappa)$ 
3:   return  $(pk, sk)$ 
4: end procedure
5: procedure token-gen( $sk$ )
6:    $(\text{OT}.pk, \text{OT}.sk) \leftarrow \text{OT.key-gen}(1^\kappa, r)$ 
7:    $|\underline{\alpha}'\rangle \leftarrow \text{OT.token-gen}(\text{OT}.sk)$ 
8:    $|\underline{\alpha}\rangle \leftarrow (\text{OT}.pk, \text{DS.sign}_{sk}(\text{OT}.pk), |\underline{\alpha}'\rangle)$ 
9:   return  $|\underline{\alpha}\rangle$ 
10: end procedure
11: procedure sign( $\alpha \in \{0, 1\}^*, |\underline{\alpha}\rangle$ )
12:   Assume  $|\underline{\alpha}\rangle$  has the form  $(\text{OT}.pk, \text{DS.sign}_{sk}(\text{OT}.pk), |\underline{\alpha}'\rangle)$ 
13:    $sig' \leftarrow \text{OT.sign}(\alpha, |\underline{\alpha}'\rangle)$ 
14:    $sig \leftarrow (\text{OT}.pk, \text{DS.sign}_{sk}(\text{OT}.pk), sig')$ 
15:   return  $sig$ 
16: end procedure
17: procedure verify $_{pk}(\alpha \in \{0, 1\}^*, sig)$ 
18:   Assume  $sig$  has the form  $(\text{OT}.pk, \text{DS.sign}_{sk}(\text{OT}.pk), sig')$ 
19:   return  $\text{DS.verify}_{pk}(\text{OT}.pk, \text{DS.sign}_{sk}(\text{OT}.pk)) \wedge \text{OT.verify}_{\text{OT}.pk}(\alpha, sig')$ 
20: end procedure
21: procedure verify-token $_{pk}(\tau)$ 
22:   Assume  $\tau$  has the form  $(\text{OT}.pk, \text{DS.sign}_{sk}(\text{OT}.pk), \tau')$ 
23:   if  $\text{DS.verify}_{pk}(\text{OT}.pk, \text{DS.sign}_{sk}(\text{OT}.pk)) = T$  and  $\text{OT.verify-token}_{\text{OT}.pk}(\tau') = (T, \sigma)$ 
24:     then
25:       return  $(T, \text{OT}.pk, \text{DS.sign}_{sk}(\text{OT}.pk), \sigma)$ 
26:     else
27:       return  $F$ 
28:     end if
29: end procedure
```

---

trivially from the correctness property of OT1 with respect to documents of length 1 (see Definition 5).

To show onetime unforgeability, we show how to map a successful attack given by  $\text{Adv}$  on OTR to a successful attack  $\text{Adv}'$  on OT1, hence a contradiction. The algorithm for  $\text{Adv}'$  is given in Algorithm 7. In this algorithm,  $\text{Adv}'$  prepares  $r - 1$  public keys and secret keys to simulate the

---

**Algorithm 7**  $\text{Adv}'$ : the adversary used in the proof of Theorem 13

---

```

1: procedure  $\text{ADV}'(pk, |\mathfrak{D}\rangle)$ 
2:    $j \xleftarrow{R} [r]$ .
3:    $(pk'_j, |\mathfrak{D}'_j\rangle) \leftarrow (pk, |\mathfrak{D}\rangle)$ 
4:   for all  $i \in [r] \setminus \{j\}$  do
5:      $(pk'_i, sk'_i) \leftarrow \text{OT1.key-gen}(\kappa)$ .
6:      $|\mathfrak{D}'_i\rangle \leftarrow \text{OT1.token-gen}(sk'_i)$ 
7:   end for
8:    $(\alpha_0, \alpha_1, s_0, s_1) \leftarrow \text{Adv}(\mathbf{pk}', |\mathfrak{D}'_1\rangle \otimes \dots \otimes |\mathfrak{D}'_r\rangle)$ 
9:   if  $\alpha_0[j] \neq \alpha_1[j]$  then
10:    return  $(\alpha_0[j], \alpha_1[j], s_0[j], s_1[j])$ 
11:  end if
12: end procedure

```

---

additional public keys that are part of the OTR scheme. Suppose the success probability of  $\text{Adv}$  is greater than  $f(\kappa)$  for some non-negligible function. The success probability of  $\text{Adv}'$  is at least  $\frac{f(\kappa)}{r} \geq \text{non-negl}(\kappa)$ , contradicting the unforgeability assumption for OT1:

$$\begin{aligned}
& \Pr \left[ \text{OT1.verify}_{2,pk} (\text{Adv}'(pk, |\mathfrak{D}\rangle)) = T \right] \\
& \geq \Pr \left[ \text{OTR.verify}_{2,pk'} (\text{Adv}(pk', |\mathfrak{D}'_1\rangle \otimes \dots \otimes |\mathfrak{D}'_r\rangle)) = T \text{ and } \alpha_0[j] \neq \alpha_1[j] \right] \\
& \geq \frac{1}{r} \Pr \left[ \text{OTR.verify}_{2,pk'} (\text{Adv}(pk', |\mathfrak{D}'_1\rangle \otimes \dots \otimes |\mathfrak{D}'_r\rangle)) = T \right] = \frac{f(\kappa)}{r} = \text{non-negl}(\kappa)
\end{aligned}$$

Here, the second inequality is justified by the fact that in order for  $\text{OTR.verify}_{2,pk'}$  to accept,  $\alpha_0$  must differ from  $\alpha_1$ . Therefore, the probability that they differ on the random index  $j$  may be smaller by at most a factor of  $\frac{1}{r}$ .

We now show item 2. The proof is essentially the same proof of the hash-and-sign paradigm for onetime length restricted digital signature scheme, see, for example, [Gol04, Proposition 6.4.7].

The proof is straightforward and given for the sake of completeness: suppose the adversary can find two documents  $\alpha_1$  and  $\alpha_2$  and signatures  $sig_1$  and  $sig_2$  which both pass the verification for OT. There are two cases: if  $h_s(\alpha_1) = h_s(\alpha_2)$ , the adversary found a collision of the (collision resistant) hashing scheme – contradiction. If  $h_s(\alpha_1) \neq h_s(\alpha_2)$ , then  $(\alpha'_1 \equiv h_s(\alpha_1), sig_1)$  and  $(\alpha'_2 \equiv h_s(\alpha_2), sig_2)$  are valid document-signature pairs, and violate the (assumed) unforgeability property of the scheme OTR – contradiction.

Next, we prove item 3. The proof is similar to the proof of the *standard construction* of Aaronson and Christiano, which elevates a quantum money mini-scheme to a (full blown) public-key quantum money scheme. A sketch proof is provided here for completeness.

Suppose towards a contradiction that there exists a QPT Adv which uses  $\ell$  signing tokens,  $|\underline{\alpha}_1\rangle \otimes \dots \otimes |\underline{\alpha}_\ell\rangle$  generated by  $\text{TS.token-gen}$  to prepare  $(\alpha_1, \text{sig}_1, \dots, \alpha_{\ell+1}, \text{sig}_{\ell+1})$  such that  $\text{TS.verify}_{\ell+1, pk}(\alpha_1, \text{sig}_1, \dots, \alpha_{\ell+1}, \text{sig}_{\ell+1})$  accepts with non-negligible probability. Assume that each  $\text{sig}_i$  has the form  $p_i, \text{DS.sign}_{sk}(p_i), \text{sig}'_i$ . Recall that each  $|\underline{\alpha}_i\rangle$  has the form  $\text{OT.pk}_i, \text{DS.sign}_{sk}(pk_i), |\underline{\alpha}'_i\rangle$ . Let  $PK = \{\text{OT.pk}_i\}_{i \in \ell}$ . We claim that for each  $i \in [\ell + 1]$ ,  $p_i \in PK$ . This is justified since the first step in  $\text{OT.verify}$  is to check for the validity of the digital signature scheme. The only documents for which the adversary has access to valid signatures are  $PK$ , and, therefore, creating any other signature would break the unforgeability property of DS.

By the pigeonhole principle, there exists  $k \in [\ell]$  and  $i \neq j \in [\ell + 1]$  such that  $p_i = p_j = \text{OT.pk}_k$ . Furthermore,  $\text{OT.verify}_{2, \text{OT.pk}_k}(\alpha_i, \text{sig}'_i, \alpha_j, \text{sig}'_j)$  accepts with non-negligible probability as part of the verification test (see line 19 in Algorithm 6). Only a single signing token associated with  $\text{OT.pk}_k$  was given to the adversary (as part of  $|\underline{\alpha}_k\rangle$ ), and therefore we reach a contradiction to the onetime-unforgeability property of OT.  $\square$

**Theorem 14.** 1. If OT1 is testable, then OTR is testable.

2. If OTR is testable then OT is testable.

3. If OT is testable and DS is a deterministic digital signature scheme then TS is testable.

*Proof.* The correctness requirement in Def. 8.1 follows trivially from the properties of the construction for (all) cases 1-3.

Proof of item 1. We show that the requirement in Eq. (8) holds. For every QPT Adv with access to  $pk$  and  $\text{poly}(\kappa)$  many signing tokens, which generates  $\alpha$  and a state  $\tau$ :

$$\begin{aligned}
& \Pr \left[ \text{OTR.verify}_{pk}(\alpha, \text{OTR.sign}(\alpha, \sigma)) = T \mid (T, \sigma) \leftarrow \text{OTR.verify-token}_{pk}(\tau) \right] \\
&= \Pr \left[ \bigwedge_{i=1}^r \text{OT1.verify}_{pk_i}(\alpha_i, \text{OT1.sign}(\alpha_i, \sigma_i)) = T \mid \bigwedge_{i=1}^r (T, \sigma_i) \leftarrow \text{OT1.verify-token}_{pk_i}(\tau_i) \right] \\
&\geq 1 - \sum_{i=1}^r \Pr \left[ \text{OT1.verify}_{pk_i}(\alpha_i, \text{OT1.sign}(\alpha_i, \sigma_i)) \neq T \mid \bigwedge_{j=1}^r (T, \sigma_j) \leftarrow \text{OT1.verify-token}_{pk_j}(\tau_j) \right] \\
&= 1 - r \cdot \text{negl}(\kappa) = 1 - \text{negl}(\kappa)
\end{aligned}$$

where we use the union bound and the assumption that OT1 is testable in the inequality.

The requirement in Eq. (9) can be shown similarly:

$$\begin{aligned}
& \Pr \left[ \text{OTR.verify-token}_{pk}(\sigma) = T \mid (T, \sigma) \leftarrow \text{OTR.verify-token}_{pk}(\tau) \right] \\
&= \Pr \left[ \bigwedge_{i=1}^r \text{OT1.verify-token}_{pk_i}(\sigma_i) = T \mid \bigwedge_{i=1}^r (T, \sigma_i) \leftarrow \text{OT1.verify-token}_{pk_i}(\tau_i) \right] \\
&\geq 1 - \sum_{i=1}^r \Pr \left[ \text{OT1.verify-token}_{pk_i}(\sigma_i) \neq T \mid \bigwedge_{j=1}^r (T, \sigma_j) \leftarrow \text{OT1.verify-token}_{pk_j}(\tau_j) \right] \\
&= 1 - r \cdot \text{negl}(\kappa) = 1 - \text{negl}(\kappa)
\end{aligned}$$

Proof of item 2: immediate. Proof of item 3: We show that the requirement in Eq. (8) holds. For every QPT Adv with access to  $pk$  and  $\text{poly}(\kappa)$  many signing tokens, which generates  $\alpha$  and a state  $\tau$ ,

$$\begin{aligned}
& \Pr \left[ \text{TS.verify}_{pk}(\alpha, \text{sign}(\alpha, (\text{OT}.pk, sig, \sigma))) = T \mid (T, (\text{OT}.pk, sig, \sigma)) \leftarrow \text{TS.verify-token}_{pk}(\tau) \right] \\
&= \Pr[\text{DS.verify}_{pk}(\text{OT}.pk, sig) = T \wedge \text{OT.verify}_{\text{OT}.pk}(\alpha, \text{sign}(\alpha, \sigma)) = T \mid \\
&\quad \text{DS.verify}_{pk}(\text{OT}.pk, sig) = T \wedge (T, \sigma) \leftarrow \text{OT.verify-token}_{\text{OT}.pk}(\tau)] \\
&\geq 1 - \text{negl}(\kappa)
\end{aligned}$$

In the last inequality we used the deterministic property of the digital signature scheme DS (see Definition 2). The requirement in Eq. (9) is shown using the same reasoning:

$$\begin{aligned}
& \Pr \left[ \text{TS.verify-token}_{pk}(\text{OT}.pk, sig, \sigma) = T \mid (T, (\text{OT}.pk, sig, \sigma)) \leftarrow \text{TS.verify-token}_{pk}(\tau) \right] \\
&= \Pr[\text{DS.verify}_{pk}(\text{OT}.pk, sig) = T \wedge \text{OT.verify-token}_{\text{OT}.pk}(\sigma) = T \mid \\
&\quad \text{DS.verify}_{pk}(\text{OT}.pk, sig) = T \wedge (T, \sigma) \leftarrow \text{OT.verify-token}_{\text{OT}.pk}(\tau)] \geq 1 - \text{negl}(\kappa)
\end{aligned}$$

□

**Theorem 15.** 1. If OT1 has onetime everlasting security, then so does OTR.

2. If OTR has onetime everlasting security, then so does OT.

*Proof.* Item 1 is trivial since OTR is simply a repetition of OT.

Proof of item 2. We map an attack by Adv on OT to an attack by Adv' on OTR. Adv' repeats the same procedure as Adv, except that the document which it outputs is  $h_s(\alpha)$  instead of  $\alpha$  (Adv'



has access to  $s$  as it is part of the signing token).

$$\begin{aligned}
& \Pr \left[ (\rho, \alpha, sig) \leftarrow \text{Adv}(|\underline{\mathcal{L}}\rangle), \text{OT.revok}_{pk}(\rho) \wedge \text{OT.verify}_{pk}(\alpha, sig) \right] \\
&= \Pr \left[ (\rho, \alpha, sig) \leftarrow \text{Adv}(|\underline{\mathcal{L}}\rangle), \text{OTR.revok}_{pk'}(\rho) \wedge \text{OTR.verify}_{pk'}(h_s(\alpha), sig) \right] \\
&= \Pr \left[ (\rho, h_s(\alpha), sig) \leftarrow \text{Adv}'(|\underline{\mathcal{L}}\rangle), \text{OTR.revok}_{pk'}(\rho) \wedge \text{OTR.verify}_{pk'}(h_s(\alpha), sig) \right] \\
&\leq \text{negl}(\kappa). \quad \square
\end{aligned}$$

## 6 Security in the Oracle Model

In this section, we prove the security of the scheme relative to an oracle, proving the following theorem. We will use the notation  $\Lambda(A)$  to denote the target set  $A \setminus \{0\} \times A^\perp \setminus \{0\}$ .

**Theorem 16.** *Let  $A$  be a uniformly random subspace from  $S(n)$ , and let  $\epsilon > 0$  be such that  $1/\epsilon = o(2^{n/2})$ . Given one copy of  $|A\rangle$  and a quantum membership oracle for  $A$  and  $A^\perp$ , a counterfeiter needs  $\Omega(\sqrt{\epsilon}2^{n/4})$  queries to output a pair  $(a, b) \in \Lambda(A)$  with probability at least  $\epsilon$ .*

Our approach is similar to the inner-product adversary method of Aaronson and Christiano. We start by reviewing the oracle model.

### 6.1 The Oracle Model

In the oracle model, we assume that we start with a single copy of the state  $|A\rangle$  and have quantum query access to an oracle for membership in the spaces  $A$  and  $A^\perp$ .

To provide quantum query access, we allow the algorithm to apply the reversible (unitary) version of the unified membership function  $\chi_{A^*}$  (see Eq. (2)). Formally,  $U_{A^*}$  acts on a Hilbert space whose basis states are triples  $(a, x, p)$  with  $a \in \mathbb{F}_2^n$  and  $x, p \in \{0, 1\}$ ; we then define  $U_{A^*}$  by:  $U_{A^*}|a, p, x\rangle = |a, p, \chi_{A^*}(a, p) \oplus x\rangle$ . In other words,  $p$  controls whether the access is to the  $A$  oracle or the  $A^\perp$  oracle,  $a$  dictates the vector whose membership is to be checked, and the output is XORed with  $x$ .

We wish to use a quantum algorithm to produce two elements  $(a, b) \in \Lambda(A)$ . The cost of the algorithm will be the number of calls it makes to the oracle  $U_{A^*}$ .

We will think of a quantum algorithm as keeping a superposition over four registers: the query register (specifying the next oracle call), a work register, and two output registers (which are measured at the end). The query register has the form  $|a, p, x\rangle$  where  $a \in \mathbb{F}_2^n$  and  $p, x \in \{0, 1\}$ ; the unitary  $U_{A^*}$  will be applied to this register. The work register has an arbitrary (finite) dimension and is initialized to a fixed state  $|0\rangle_W$ . The two output registers get measured at the end of the algorithm, and each stores an element of  $\mathbb{F}_2^n$ . Since we want the algorithm to have access to one copy of  $|A\rangle$ , we will initialize one of the output registers to  $|A\rangle$  and the other to  $|0\rangle$ . The initial state will therefore be

$$|\psi_{init}^A\rangle := |0, 0, 0\rangle|0\rangle_W|A\rangle|0\rangle.$$

A quantum algorithm is fully described by a sequence of unitaries  $U_0, U_1, \dots, U_T$ . Applying such an algorithm is achieved by alternately applying the  $U_i$  unitaries and the query unitary  $U_{A^*}$ , starting from the initial state  $|\psi_{init}^A\rangle$ . The state at the end of the algorithm is therefore

$$|\psi_f^A\rangle := U_T U_{A^*} U_{T-1} U_{A^*} \dots U_{A^*} U_1 U_{A^*} U_0 |\psi_{init}^A\rangle.$$

The two output registers are then measured, and the result constitutes the output of the algorithm.

## 6.2 Lower Bound

We will use the following theorem.

**Theorem 17** (Adapted from [AC12]). *Let  $\mathcal{R}$  be a symmetric, anti-reflexive relation on  $S(n)$  such that each space in  $S(n)$  related to at least one other space. Suppose that initially  $|\langle \psi_{init}^A | \psi_{init}^B \rangle| \geq c$  for all  $(A, B) \in \mathcal{R}$ , whereas at the end we need  $\mathbb{E}_{(A,B) \in \mathcal{R}} [|\langle \psi_f^A | \psi_f^B \rangle|] \leq d$ . Then any algorithm achieving that must make  $\Omega((c-d)2^{n/2})$  oracle queries.*

*Remark 3.* Aaronson and Christiano claimed a slightly weaker result. Their assumption is  $\forall A, B \in \mathcal{R}, |\langle \psi_f^A | \psi_f^B \rangle| \leq d$ , while ours is  $\mathbb{E}_{(A,B) \in \mathcal{R}} [|\langle \psi_f^A | \psi_f^B \rangle|] \leq d$ . However, their original proof is valid for this stronger version.

We also fix  $\mathcal{R} \subset S(n) \times S(n)$  to be the relation that contains all  $(A, B)$  such that  $\dim(A \cap B) = \frac{n}{2} - 1$ . We use  $\mathcal{R}_A$  to denote the set of  $B \in S(n)$  such that  $(A, B) \in \mathcal{R}$ .

**Lemma 18.** *For  $A \in S(n)$ , let  $|\psi_f^A\rangle$  be the final state of the algorithm when the oracle and initial state encode the space  $A$ . If the algorithm outputs a pair  $(a, b) \in \Lambda(A)$  with probability at least 0.99, Then*

$$\mathbb{E}_{(A,B) \in \mathcal{R}} [|\langle \psi_f^A | \psi_f^B \rangle|] \leq 0.2 + \max_{A \in S(n), (a,b) \in \Lambda(A)} \Pr_{B \sim \mathcal{R}_A} [(a, b) \in \Lambda(B)].$$

*Proof.* We will decompose the final state  $|\psi_f^A\rangle$  as

$$|\psi_f^A\rangle = \sum_{a,b \in \mathbb{F}_2^n} \beta_{ab}^A |\phi_{ab}^A\rangle |a\rangle |b\rangle = \sum_{(a,b) \in \Lambda(A)} \beta_{ab}^A |\phi_{ab}^A\rangle |a\rangle |b\rangle + \sum_{(a,b) \notin \Lambda(A)} \beta_{ab}^A |\phi_{ab}^A\rangle |a\rangle |b\rangle.$$

Note that  $\sum_{a,b \in \mathbb{F}_2^n} |\beta_{ab}^A|^2 = 1$  and that  $\sum_{(a,b) \notin \Lambda(A)} |\beta_{ab}^A|^2 \leq 0.01$ . Also, the two sums above form orthogonal vectors; denoting them  $|v_A\rangle$  and  $|u_A\rangle$ , and similarly for  $B$ , we have  $|\psi_f^A\rangle = |v_A\rangle + |u_A\rangle$  and  $\| |v_A\rangle \| \leq 1$ ,  $\| |u_A\rangle \| \leq 0.1$ . It follows that

$$\begin{aligned} |\langle \psi_f^A | \psi_f^B \rangle| &\leq |\langle v_A | \psi_f^B \rangle| + |\langle u_A | \psi_f^B \rangle| \leq |\langle v_A | \psi_f^B \rangle| + \| |u_A\rangle \| \cdot \| \psi_f^B \| \leq |\langle v_A | \psi_f^B \rangle| + 0.1 \\ &\leq |\langle v_A | v_B \rangle| + |\langle v_A | u_B \rangle| + 0.1 \leq |\langle v_A | v_B \rangle| + \| |v_A\rangle \| \| |u_B\rangle \| + 0.1 \leq |\langle v_A | v_B \rangle| + 0.2 \\ &= 0.2 + \left| \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} (\beta_{ab}^A)^* \beta_{ab}^B \langle \phi_{ab}^A | \phi_{ab}^B \rangle \right| \leq 0.2 + \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} |\beta_{ab}^A| |\beta_{ab}^B|. \end{aligned}$$

Let  $\chi_{\Lambda(A)}$  be defined by  $\chi_{\Lambda(A)}(a, b) = 1$  if  $(a, b) \in \Lambda(A)$  and  $\chi_{\Lambda(A)}(a, b) = 0$  otherwise. We have

$$\begin{aligned}
& \mathbb{E}_{(A,B) \sim \mathcal{R}} \left( |\langle \psi_f^A | \psi_f^B \rangle| \right) - 0.2 \leq \mathbb{E}_{(A,B) \sim \mathcal{R}} \left[ \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} |\beta_{ab}^A| |\beta_{ab}^B| \right] \\
&= \frac{1}{|\mathcal{R}|} \sum_{(A,B) \in \mathcal{R}} \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} |\beta_{ab}^A| |\beta_{ab}^B| \\
&\leq \frac{1}{|\mathcal{R}|} \sqrt{\sum_{(A,B) \in \mathcal{R}} \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} |\beta_{ab}^A|^2 \sum_{(A,B) \in \mathcal{R}} \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} |\beta_{ab}^B|^2} \quad (\text{Cauchy-Schwartz}) \\
&= \frac{1}{|\mathcal{R}|} \sum_{(A,B) \in \mathcal{R}} \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} |\beta_{ab}^A|^2 \quad (\text{since } \mathcal{R} \text{ is symmetric}) \\
&= \mathbb{E}_{(A,B) \sim \mathcal{R}} \left[ \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} |\beta_{ab}^A|^2 \right] \\
&= \mathbb{E}_{A \sim S(n)} \left[ \mathbb{E}_{B \sim \mathcal{R}_A} \left[ \sum_{(a,b) \in \Lambda(A) \cap \Lambda(B)} |\beta_{ab}^A|^2 \right] \right] \\
&= \mathbb{E}_{A \sim S(n)} \left[ \mathbb{E}_{B \sim \mathcal{R}_A} \left[ \sum_{(a,b) \in \Lambda(A)} |\beta_{ab}^A|^2 \chi_{\Lambda(B)}(a, b) \right] \right] \\
&= \mathbb{E}_{A \sim S(n)} \left[ \sum_{(a,b) \in \Lambda(A)} |\beta_{ab}^A|^2 \mathbb{E}_{B \sim \mathcal{R}_A} [\chi_{\Lambda(B)}(a, b)] \right] \quad (\text{linearity of expectation}) \\
&\leq \mathbb{E}_{A \sim S(n)} \left[ \sum_{(a,b) \in \Lambda(A)} |\beta_{ab}^A|^2 \right] \max_{A \in S(n), (a,b) \in \Lambda(A)} \left( \mathbb{E}_{B \sim \mathcal{R}_A} [\chi_{\Lambda(B)}(a, b)] \right) \\
&\leq \max_{A \in S(n), (a,b) \in \Lambda(A)} \left( \mathbb{E}_{B \sim \mathcal{R}_A} [\chi_{\Lambda(B)}(a, b)] \right) \\
&= \max_{A \in S(n), (a,b) \in \Lambda(A)} \Pr_{B \sim \mathcal{R}_A} [(a, b) \in \Lambda(B)]. \quad \square
\end{aligned}$$

**Lemma 19.**

$$\max_{A \in S(n), (a,b) \in \Lambda(A)} \Pr_{B \sim \mathcal{R}_A} [(a, b) \in \Lambda(B)] \leq \frac{1}{4}.$$

*Proof.* Fix  $A \in S(n)$  and  $(a, b) \in \Lambda(A)$ , so  $a \neq b$ ,  $a \in A$ ,  $b \in A^\perp$ , and  $a, b \neq 0$ . Picking  $B \sim \mathcal{R}_A$  is picking a vector space whose intersection with  $A$  has dimension  $n/2 - 1$ . One way of picking such a vector space uniformly at random is by picking a random basis for  $A$ , then discarding one of the vectors in the basis and adding a vector outside of  $A$ . This gives a new set of  $n/2$  independent vectors, which we take to be a basis for  $B$ .

Let  $G(m, k)$  be the number of subspaces of  $\mathbb{F}_2^m$  of dimension  $k$ . We can write down an expression for  $G(m, k)$  using the following argument: there are  $(2^m - 1)(2^m - 2) \dots (2^m - 2^{k-1})$  ways to pick an ordered list of  $k$  independent vectors in  $\mathbb{F}_2^m$ , and each such list corresponds to a subspace. However, this over-counts the subspaces by a factor of  $(2^k - 1)(2^k - 2) \dots (2^k - 2^{k-1})$ . We conclude that<sup>11</sup>

$$G(m, k) = \prod_{i=0}^{k-1} \frac{2^{m-i} - 1}{2^{k-i} - 1}.$$

We are interested in the probability that  $(a, b) \in \Lambda(B)$ . This is simply the probability that  $a \in B$  and  $b \in B^\perp$ .  $a$  is in  $B$  if  $a$  is in the span of the  $n/2 - 1$  vectors we chose from  $A$ , and  $b$  is in  $B^\perp$  if  $b$  is orthogonal to the vector we chose from outside of  $A$ . It is clear that these events are independent.

The former happens with probability  $G(n/2-1, n/2-2)/G(n/2, n/2-1) = \frac{1}{2} \left(1 - \frac{1}{2^{n/2-1}}\right) \leq 1/2$ . For the latter, note that exactly half the vectors in  $\mathbb{F}_2^n$  are orthogonal to  $b$ , and  $2^{n/2}$  of those are in  $A$ . So the probability of picking an orthogonal vector to  $c$  outside of  $A$  is

$$\frac{2^{n-1} - 2^{n/2}}{2^n - 2^{n/2}} = \frac{2^{n/2-1} - 1}{2^{n/2} - 1} \leq 1/2.$$

Thus, the probability of both events occurring is at most  $1/4$ . □

**Theorem 20.** *Any algorithm which takes  $|A\rangle$  as input and outputs (with success probability at least 0.99) a pair  $(a, b) \in \Lambda(A)$  must make  $\Omega(2^{n/4})$  oracle queries.*

*Proof.* The algorithm starts with the initial state  $|\psi_{init}^A\rangle = |0, 0\rangle|0\rangle_W|A\rangle|0\rangle$  and ends with  $|\psi_f^A\rangle$ . Note that if  $(A, B) \in \mathcal{R}$ , then  $|\langle \psi_{init}^A | \psi_{init}^B \rangle| = |\langle A | B \rangle| = 1/2$ . Also, by Lemma 18 and Lemma 19, we have  $\mathbb{E}_{(A,B) \sim \mathcal{R}} |\langle \psi_f^A | \psi_f^B \rangle| \leq \frac{1}{4} + 0.2 = 0.45$ . Therefore, by Theorem 17, the algorithm must make  $\Omega(2^{n/4})$  oracle queries. □

The main theorem shows that there is no way to find such pairs, with probability at least 0.99, for all  $A$ s. We need to show that there is no way to find the pairs even with exponentially small success probability, and even for an average  $A$ . We do that in two steps. The first step reduces exponentially small probability to constant probability, and the second step reduces average  $A$  to worst case  $A$ .

For the first step, we will need the following theorem from [AC12].

**Theorem 21** ([AC12, Theorem 8]). *Let  $U_{Init} = I - 2|Init\rangle\langle Init|$ , and for a subspace  $G$ ,  $U_G|v\rangle = -|v\rangle$  for all  $|v\rangle \in G$  and acts as the identity on all  $|v\rangle$  orthogonal to  $G$ . Let  $\delta \geq 2\epsilon > 0$ . Suppose  $F(|Init\rangle, G) \geq \epsilon$ . There exists an algorithm which starts with the state  $|Init\rangle$ , uses  $O(\frac{\log 1/\delta}{\epsilon\delta^2})$  oracle calls to  $U_{Init}$  and  $U_G$ , and prepares a state  $\rho$  such that  $F(\rho, G) \geq 1 - \delta$ .*

---

<sup>11</sup>In more generality, the number of subspaces of dimension  $k$  in a vector space of dimension  $m$  over a field with  $q$  elements is the Gaussian binomial coefficient [Pra10],  $\binom{m}{k}_q = \prod_{i=0}^{k-1} \frac{q^{m-i} - 1}{q^{k-i} - 1}$ .

Here<sup>12</sup>  $F(\rho, G) = \max_{|\psi\rangle \in G} \sqrt{\langle \psi | \rho | \psi \rangle}$ . Note that  $F(\rho, G)^2 = \max_{|\psi\rangle \in G} \text{Tr}(|\psi\rangle\langle\psi| \rho) \leq \text{Tr}(\Pi_G \rho)$ .

Equipped with this tool, we now reduce security against exponentially small success probability to security against constant success probability.

**Corollary 22.** *Let  $1/\epsilon = o(2^{n/2})$ . Given one copy of  $|A\rangle$  and oracle access to  $U_{A^*}$ , a counterfeiter needs  $\Omega(\sqrt{\epsilon}2^{n/4})$  queries to prepare a state  $\rho$  such that  $\text{Tr}(\Pi_{C(A)}\rho) \geq \epsilon$  for all  $A$ .*

*Proof.* Suppose we have a counterfeiter  $C$  that, on input  $|A\rangle$ , makes  $T = o(\sqrt{\epsilon}2^{n/4})$  queries and produces a pair  $(a, b) \in \Lambda(A)$  with probability at least  $\epsilon$ . We will construct another counterfeiter  $C'$  which violates Theorem 20 by producing a pair  $(a, b) \in \Lambda(A)$  with probability 0.99.

If the success probability was exactly  $\epsilon$ , we could have used amplitude amplification. The problem, which is the same problem Aaronson and Christiano had to deal with, is the so called soufflé problem [Bra97] of amplitude amplification (and also Grover search): the algorithm must run (the soufflé must be in the oven) just for the right time. This can be easily circumvented if  $|Init\rangle$  can be prepared free of charge – which is *not* our case. Algorithms which do not have this property are called *fixed point quantum search*. Previously, the fixed point quantum search algorithms [TGP06, CRR05] lost their quantum speedup, though they had an exponential convergence. Aaronson and Christiano constructed an algorithm which has the quadratic speedup but loses the exponential convergence.

Specifically, we apply Theorem 21 with  $|init\rangle = |\psi_f^A\rangle$ , the final state of the counterfeiter  $C$  when run on  $A$ , and  $G$  being the subspace spanned by the computational basis states of the algorithm that contain outputs in  $\Lambda(A)$ : that is, states of the form  $|c, p, x\rangle|s\rangle_W|a\rangle|b\rangle$  with  $(a, b) \in \Lambda(A)$ . We note that  $U_G$  can be implemented using two queries to  $U_{A^*}$ . Implementing the operator  $I - 2|A\rangle\langle A|$  can be done using one query to  $U_A$  and one query  $U_{A^\perp}$ , as shown by Aaronson and Christiano (more details can also be found in Section 7.1). Since  $|\psi_{init}^A\rangle$  contains only the state  $|A\rangle$  plus some blank registers, we can also implement the operator  $I - 2|\psi_{init}^A\rangle\langle\psi_{init}^A|$  in this way. Therefore  $U_{Init}$  can be implemented using one call for  $C$ , one call for  $C^\dagger$ , and two calls for  $U_{A^*}$ :

$$U_{Init} = I - 2|\psi_f^A\rangle\langle\psi_f^A| = I - 2C|\psi_{init}^A\rangle\langle\psi_{init}^A|C^\dagger = C(I - 2|\psi_{init}^A\rangle\langle\psi_{init}^A|)C^\dagger.$$

Note that the success probability of  $C$  is equal to  $F(|\psi_f^A\rangle, G)^2$ . Since this probability is at least  $\epsilon$ , we get  $F(|\psi_f^A\rangle, G) \geq \sqrt{\epsilon}$ . We apply 21 with  $\delta = 0.005$  and  $\epsilon$  (in the theorem) set to  $\sqrt{\epsilon}$  (we can assume without loss of generality that  $\sqrt{\epsilon} \leq 0.0025$ ). Theorem 21 gives us an algorithm  $C'$  which outputs a state  $\rho$  with  $F(\rho, G) \geq 0.99$ , and which uses  $O(\frac{1}{\sqrt{\epsilon}})$  calls to the counterfeiter  $C$  and  $U_{A^*}$ . Note that  $0.99 \leq 0.995^2 \leq F(\rho, G)^2 \leq \text{Tr}(\Pi_G \rho)$ , which means that measuring  $\rho$  provides a pair  $(a, b) \in \Lambda(A)$  with probability at least 0.99. Moreover, since each call for  $C$  uses  $T$  queries to

---

<sup>12</sup>The fidelity between a mixed state and a pure state is  $F(\rho, |\psi\rangle) = \sqrt{\langle \psi | \rho | \psi \rangle}$ . The above definition is a natural extension to the distance between a mixed state and a subspace.

$U_{A^*}$ , the total number of queries used by  $C'$  is

$$O\left(\frac{1}{\sqrt{\epsilon}}\right) \cdot (1 + T) = O\left(\frac{1}{\sqrt{\epsilon}}\right) + O\left(\frac{T}{\sqrt{\epsilon}}\right).$$

However,  $C'$  must use  $\Omega(2^{n/4})$  queries by Theorem 20. Since  $1/\sqrt{\epsilon} = o(2^{n/4})$ , it follows that  $T/\sqrt{\epsilon} = \Omega(2^{n/4})$ , or  $T = \Omega(\sqrt{\epsilon}2^{n/4})$ , as desired.  $\square$

We now complete the proof of Theorem 16, which we restate here for convenience.

**Theorem 16.** *Let  $A$  be a uniformly random subspace from  $S(n)$ , and let  $\epsilon > 0$  be such that  $1/\epsilon = o(2^{n/2})$ . Given one copy of  $|A\rangle$  and a quantum membership oracle for  $A$  and  $A^\perp$ , a counterfeiter needs  $\Omega(\sqrt{\epsilon}2^{n/4})$  queries to output a pair  $(a, b) \in \Lambda(A)$  with probability at least  $\epsilon$ .*

*Proof.* The only missing ingredient is a worst case to average case reduction. Suppose we had a counterfeiter  $C$  which violated the above. We will construct another counterfeiter  $C'$  which violates Corollary 22.  $C'$  first chooses a random linear invertible map  $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ . Let  $f(A)$  be the image of  $A$  under  $f$ . For any fixed  $A$ ,  $f(A)$  is a uniformly random element of  $S$ . Let  $U_f, U_{f^{-1}}$  be the unitary maps  $U_f|x\rangle = |f(x)\rangle$ ,  $U_{f^{-1}}|x\rangle = |f^{-1}(x)\rangle$ . Since  $U_{f(A)^*} = U_f U_{A^*} U_f^\dagger$ , the oracle  $U_{f(A)^*}$  can be efficiently implemented using 1 query of  $U_{A^*}$ , and  $|f(A)\rangle$  can be prepared using  $|A\rangle$ . It then applies the counterfeiting procedure to  $|f(A)\rangle$ , and since  $f(A)$  is random in  $S(n)$ , it finds  $(a, b) \in \Lambda(f(A))$  using  $t$  queries with error probability  $\epsilon$ . Using  $f^{-1}$ , it can find  $(f^{-1}(a), f^{-1}(b)) \in \Lambda(A)$ , which contradicts Corollary 22.  $\square$

## 7 Breaking and Fixing the Aaronson-Christiano Scheme

### 7.1 Overview

We start with an overview of AC's scheme. The quantum money state issued by the bank is  $|A\rangle$  (see Eq. (3)) for a random  $A \in S(n)$ , together with a classical string which is used for verifying that state – additional details on this classical string will be given shortly. They define  $\mathbb{P}_A$  and  $\mathbb{P}_{A^\perp}$  as the projection to the span of  $A$  and  $A^\perp$  respectively,

$$\mathbb{P}_A = \sum_{a \in A} |a\rangle\langle a|, \quad \mathbb{P}_{A^\perp} = \sum_{a \in A^\perp} |a\rangle\langle a|$$

and show that

$$H^{\otimes n} \mathbb{P}_{A^\perp} H^{\otimes n} \mathbb{P}_A = |A\rangle\langle A| \tag{13}$$

Eq. (13) implies that the state  $|A\rangle$  can be verified efficiently using 2 queries to  $U_{A^*}$ . They show that although (quantum) oracle access to  $U_{A^*}$  is sufficient for quickly verifying  $|A\rangle$ , it still requires exponential time to clone  $|A\rangle$ . This scheme, which requires an oracle or quantum communication with the bank to provide this service, does not suffer from the attacks which will be described next, and is proved to be unconditionally secure.

They also construct an explicit public quantum money scheme (see Definition 3) which does not require an oracle, or communication with the bank, for verification. The main challenge is how to provide an explicit program which allows the calculation of  $\chi_{A^*}$  (which is sufficient to implement  $U_{A^*}$ ), but on the other hand does not leak more information which would be sufficient to forge  $|A\rangle$  (such as a basis for  $A$ ); ideally, we would like this program to be unintelligible in the same way an oracle is. This primitive which comes in different flavors, is known as program obfuscation – more details are provided in Section 7.3. Aaronson and Christiano constructed an ad-hoc obfuscation for subspace membership. The bank picks a random element  $A \in S(n)$  as before. Next, it picks  $p_1, \dots, p_{\beta n}$  at random from the set of multi-variate polynomials with a fixed degree  $d$  that vanish on  $A$ . This can be preformed efficiently.  $\beta$  is chosen so that with exponentially high probability, the elements of  $A$  are the only ones which vanish on all polynomials, i.e.  $\{p_i\}_{i \in [\beta n]}$  can be used to compute  $\chi_A$ :

$$a \in A \iff p_i(a) = 0 \quad \forall i \in [\beta n] \tag{14}$$

The same process is done for  $A^\perp$ , with polynomials  $q_1, \dots, q_{\beta n}$ .

In the noise-free version, these polynomials are published by the bank<sup>13</sup>, and provide a means to compute  $\chi_{A^*}$ , using Eq. (14), and its analogue for  $A^\perp$ .

In the noisy version, again,  $\beta n$  polynomials are provided. This time, an  $\epsilon$  fraction of them are “decoy” polynomials: these are degree  $d$  polynomials that vanish on a random  $A' \in S$  (where each polynomial has a different  $A'$  associated with it). The other  $(1 - \epsilon)\beta n$  are as before (uniformly random from the set of all degree  $d$  polynomials that vanish on  $A$ ). The polynomials are provided at a random order. The parameters  $\beta$  and  $\epsilon$  are picked so that the vanishing polynomials can still be used to compute  $\chi_{A^*}$  using the following property:

$$a \in A \iff |\{p_i(a) = 0\}_{i \in [\beta n]}| \geq (1 - \epsilon)\beta n.$$

Pena, Faugère and Perret showed how to totally break (that is, they find a basis for  $A$ , and therefore reconstruct the secret key) the noise-free version, using Gröbner’s basis analysis [PFP15].

## 7.2 Attack on the Noisy Scheme

In this section we report on an attack by Paul Christiano[Chr15] which totally breaks the noisy version, by reducing the noisy version to the noise-free version.

The attack reduces the noisy version to the noise-free version by distinguishing between the polynomials that vanish on  $A$  and the “decoy” polynomials. The attack requires access to one copy of the quantum money state, and is inherently quantum. Interestingly, the noise-free version attack, unlike ours, is purely classical and does not require a copy of the quantum money state.

---

<sup>13</sup>To keep the public key of the quantum money scheme short, a standard digital signature key is used as the public key, and the classical part of the quantum money state contains a signed copy of these polynomials. We also use this construction in TS – see Algorithm 6.

The attack is based on the notion of *single copy tomography* which was introduced by Farhi et al. [FGH<sup>+</sup>10].<sup>14</sup> They showed that given a single copy of an unknown state  $|\psi\rangle$ , and a two outcome measurement  $\{|\psi\rangle\langle\psi|, I - |\psi\rangle\langle\psi|\}$ , which we call the protective measurement, it is possible to get a “good” estimate of  $\langle\psi|E_j|\psi\rangle$  efficiently for every efficiently implementable POVM  $\{E_j\}$ , without destroying the state  $|\psi\rangle$ . For our purposes, we will be interested only in a 2 outcome measurement with an additive approximation  $\frac{1}{10}$ , and error probability  $1/(\beta n)^2$ , which requires  $O(\log(\beta n))$  uses of the protective measurement. We define,  $E_0^i|x\rangle = p_i(x)|x\rangle$ , and  $E_1^i = I - E_0^i$  for every  $i \in [\beta n]$ . Recall that since  $p_i$  is a polynomial over  $\mathbb{F}_2^n$ ,  $p_i(x) \in \{0, 1\}$ , which implies that  $\{E_0^i, E_1^i\}$  is a POVM (in this case, a 2-outcome projective measurement), and furthermore, by standard techniques, since  $p_i(x)$  can be calculated efficiently classically, this POVM is also efficiently implementable for every  $i$ . Since the “decoy” polynomials are fully random,  $\langle A|E_0^i|A\rangle \approx \frac{1}{2}$  with very high probability. On the other hand, for the polynomials  $p_i$  which vanish on  $A$ ,  $\langle A|E_0^i|A\rangle = 0$ .

We use Farhi et al.’s single copy tomography to estimate  $\langle A|E_0^i|A\rangle$  for every  $i \in [\beta n]$ . By the union bound, with probability at least  $1 - \frac{1}{\beta n}$ , the estimation for all the noisy polynomials is at least 0.3, while for all the polynomials that vanish on  $A$  the estimation is at most  $\frac{1}{10}$ . After sieving the vanishing polynomials from the “decoys”, we can use Pena et al.’s noise-free attack.

### 7.3 Fixing the Quantum Money Scheme

The challenge that we are facing due to the previous attack, was described by Aaronson and Christiano as follows:

Given a subspace  $A \preceq \mathbb{F}_2^n$ , how can a bank distribute an “obfuscated program”  $\mathcal{O}_{\chi_{A^*}}$ , which legitimate buyers and sellers can use to decide membership in both  $A$  and  $A^\perp$ , but which does not reveal anything else about  $A$  that might facilitate counterfeiting?

For a gentle bird’s-eye view on program obfuscation, see Barak’s excellent, though already slightly outdated, review [Bar16]. There are several notions of program obfuscation in the literature. In all of them, an obfuscator is a (probabilistic) compiler which takes a circuit  $C$  and outputs a classical obfuscated program  $\mathcal{O}_C$  which preserves the functionality,

$$\forall x, C(x) = \mathcal{O}_C(x),$$

and  $\mathcal{O}_C$  is “unintelligible” in some way. The notion which we use for unintelligible is called Virtual Black Box (VBB), which requires that for every polynomial adversary with access to  $\mathcal{O}_C$  there exists an efficient simulator with only oracle access to  $C$ , which outputs a computationally indistinguishable distribution with respect to the adversary’s distribution. Barak et al. [BGI<sup>+</sup>12] defined this notion and showed that there is a class of circuits for which VBB obfuscation is impossible (hence *general purpose* VBB obfuscation is impossible). As far as we know, their results, as well

<sup>14</sup>A similar result, based on different techniques and under weaker assumptions was discovered earlier [AAV93, AV93].



as the other negative results related to VBB (e.g. [GK05, BCC<sup>+</sup>14]), does not rule out specific black box obfuscation for the class of functions required in this work.

Let  $\mathfrak{C}$  be a set of circuits that compute  $\chi_{A^*}$  for all  $A \in S(n)$ . On a positive note in that direction, Canetti, Rothblum and Varia [CRV10] proved VBB obfuscation for hyperplane membership of  $\mathbb{F}_p^n$  but only for constant  $n$  and a large  $p$ . Their construction is not secure against quantum adversaries, therefore we do not attempt to adapt our scheme to fit to their setting.

Another approach to avoid Barak et al’s impossibility result for VBB obfuscation is the following: their proof breaks if the obfuscator can output a quantum state as part of the description of the obfuscated program. If this route is chosen, the bank’s assistance would be needed to provide as many copies of the obfuscated program as required for verification. Alagic and Fefferman [AF16] have studied quantum obfuscation, and mainly proved impossibility results.

Next we define VBB obfuscation which will be sufficient for our needs.

**Definition 23** (VBB Obfuscator). A probabilistic algorithm  $\mathcal{O}$  is a VBB circuit obfuscator with dependent auxiliary input for the collection  $\mathfrak{F}$  of circuits if the following three conditions hold:

1. **Functionality.** For every circuit  $C \in \mathfrak{F}$ , the string  $\mathcal{O}_C$  is a description of a classical circuit that computes the same function as  $C$ .
2. **“Virtual black box” property - computational indistinguishability.**

For any QPT Adv there is a QPT Sim such that for every  $C \in \mathfrak{F}$  and auxiliary state  $|\psi\rangle$ , the distributions  $\text{Adv}(\mathcal{O}(C), |\psi\rangle)$  and  $\text{Sim}^C(1^{|C|}, |\psi\rangle)$  are computationally indistinguishable.

For our purposes, we only require classical computational indistinguishability, i.e. for every PPT distinguisher Dist,

$$|\Pr[\text{Dist}(\text{Adv}(\mathcal{O}(C), |\psi\rangle)) = 1] - \Pr[\text{Dist}(\text{Sim}^C(1^{|C|}, |\psi\rangle)) = 1]| \leq \text{negl}(|C|).$$

3. **Efficiency.** The algorithm  $\mathcal{O}$  is PPT.<sup>15</sup>

This definition differs from the VBB property given in Barak et al.[BGI<sup>+</sup>12] in several ways:

- We require security against a QPT adversary, not a PPT adversary.
- We require security even when the adversary has access to auxiliary input states, and furthermore, that auxiliary input state depends on the circuit  $C$ , see [GK05, BCC<sup>+</sup>14]. We need security with *dependent* auxiliary input because the adversary may have copies of the quantum signing token, which depends on the circuit.
- We require computational indistinguishability between the distributions that the adversary and the simulator generate. The main impossibility results in [BGI<sup>+</sup>12] assumed a weaker notion of security, in which the output of the adversary is a bit (thus, the goal of the simulator is learning a predicate).

---

<sup>15</sup>Note that efficiency implies that the slowdown by running  $\mathcal{O}_C$  instead of  $C$  is at most polynomial.

We note that our security proof holds even if the VBB property holds only for a random circuit in  $\mathfrak{F}$ .

We are now ready to describe our scheme, denoted OT1, which is a onetime 1-restricted, testable TS scheme. As we have seen in Section 5, this can be mapped to a full blown (without the onetime and length restrictions) testable TS scheme. The bank operates exactly in the same way as the scheme with the oracle, except that instead of providing oracle access to  $\chi_{A^*}$ , it publishes the obfuscated program  $\mathcal{O}_{\chi_{A^*}}$ .

---

**Algorithm 8** OT1 scheme.

---

```

1: procedure key-gen( $1^\kappa$ )
2:   Set  $n = f(\kappa)$  ▷  $f$  can be any super-logarithmic function.
3:    $A \xleftarrow{R} S(n)$ 
4:    $pk \leftarrow \mathcal{O}_{\chi_{A^*}}$ 
5:    $sk \leftarrow \langle A \rangle$  ▷  $\langle A \rangle$  is a basis for  $A$ .
6:   return  $(pk, sk)$ 
7: end procedure
8: procedure token-gen( $\langle A \rangle$ )
9:    $|\underline{\alpha}\rangle \leftarrow |A\rangle$  ▷ Given a basis for  $A$ , the state  $|A\rangle$  can be generated efficiently [AC12].
10:  return  $|\underline{\alpha}\rangle$ 
11: end procedure
12: procedure sign( $\alpha \in \{0, 1\}, |A\rangle$ )
13:  Apply  $H^{\otimes n}$  iff  $\alpha = 1$  to the state  $|A\rangle$ . ▷  $H^{\otimes n}|A\rangle = |A^\perp\rangle$ 
14:  Measure the resulting state in the standard basis. Set  $sig$  to be the outcome.
15:  if  $sig = 0 \dots 0$  then
16:    return failed
17:  else
18:    return  $sig$ 
19:  end if
20: end procedure
21: procedure verify $_{pk}(\alpha \in \{0, 1\}, sig)$ 
22:  return  $\mathcal{O}_{\chi_{A^*}}(\alpha, sig)$ . ▷ See Eq. (2) for the definition of  $\chi_{A^*}$ .
23: end procedure
24: procedure verify-token $_{pk}(\tau)$ 
25:  Using  $\mathcal{O}_{\chi_{A^*}}$ , measure the state  $\tau$  using the two outcome measurement  $\{|A\rangle\langle A|, I - |A\rangle\langle A|\}$ , following the procedure following Eq. (13). If the outcome is the first, return  $(T, |A\rangle)$ . Otherwise, return  $F$ .
26: end procedure

```

---

*Remark 4.* The procedures key-gen and verify are entirely classical. The state  $|A\rangle$  generated in token-gen can be constructed using only Clifford gates [CG97, GRB03], where the total number

of gates is polynomial in  $n$  and hence poly-logarithmic in  $\kappa$ . Clifford gates do not form a universal quantum gate set, and may be easier to implement on certain architectures, see, e.g. [Bom11].

**Theorem 24.** *If  $\mathcal{O}$  is a VBB obfuscator with dependent auxiliary input for the class  $\mathfrak{S}$ , then the scheme OT1 described in Algorithm 8 is an imperfect onetime everlasting secure, 1-restricted, testable, unforgeable tokenized signature scheme.*

*Proof.* The imperfect completeness property (see Eq. 3) for 1-restricted scheme is justified by the construction. The scheme is imperfect (and not perfect) since with probability of  $\frac{1}{2^{n/2}} = \text{negl}(\kappa)$  the outcome of the measurement in line 14 is  $0 \dots 0$  (which causes sign to fail). This is the only element that is trivial to guess in  $A$  and  $A^\perp$ , since the  $0 \dots 0$  vector is an element of every subspace of  $\mathbb{F}_2^n$ .

We show that the scheme is unforgeable. Fix  $A$ . Suppose the distinguisher runs the (classical) algorithm  $\text{verify}_{2,pk(A)}$ , and the auxiliary state is  $|A\rangle$ . By the VBB property<sup>16</sup>, there exists a QPT  $\text{Sim}$  such that

$$\left| \Pr_{\text{Adv}, \mathcal{O}} [\text{verify}_{2,pk(A)}(\text{Adv}(\mathcal{O}(\chi_{A^*}), |A\rangle)) = 1] - \Pr_{\text{Sim}} [\text{verify}_{2,pk(A)}(\text{Sim}^{\chi_{A^*}}(1^{|\chi_{A^*}|}, |A\rangle)) = 1] \right| \leq \text{negl}(\kappa).$$

Here, we were explicit over what the probability is.

The above holds for every  $A \in S$ . We now take the average over all  $A \in S$ :

$$\left| \Pr_{A \leftarrow S, \text{Adv}, \mathcal{O}} [\text{verify}_{2,pk(A)}(\text{Adv}(\mathcal{O}(\chi_{A^*}), |A\rangle)) = 1] - \Pr_{A \in R, S, \text{Sim}} [\text{verify}_{2,pk(A)}(\text{Sim}^{\chi_{A^*}}(1^{|\chi_{A^*}|}, |A\rangle)) = 1] \right| \leq \text{negl}(\kappa)$$

By Theorem 16, the second term of the above equation can be bounded. For every QPT  $\text{Sim}$ ,

$$\Pr_{A \leftarrow S, \text{Sim}} [\text{verify}_{2,pk(A)}(\text{Sim}^{\chi_{A^*}}(1^{|\chi_{A^*}|}, |A\rangle)) = 1] = \frac{\text{poly}(\kappa)}{2^{n/2}} = \text{negl}(\kappa).$$

Combining the above two equations, we conclude that onetime super-security (see Def. 25) holds:

$$\Pr_{A \leftarrow S, \text{Adv}, \mathcal{O}} [\text{verify}_{2,pk(A)}(\text{Adv}(\mathcal{O}(\chi_{A^*}), |A\rangle)) = 1] \leq \text{negl}(\kappa).$$

The testability properties (see Def. 8) also follow trivially by construction because  $\text{verify}$ -token implements the rank-1 projector of the token  $|\frac{\mathcal{O}}{2}\rangle = |A\rangle$  – see Eq. (13).

The everlasting security is much simpler to prove. All the adversary has is one copy of the state  $|A\rangle$  (and no access whatsoever to  $\chi_{A^*}$ ). Theorem 16 shows that in order to find with non-negligible probability one non-trivial element from  $A$  and another from  $A^\perp$ , exponentially many oracle queries to  $\chi_{A^*}$  are required. In particular, this task cannot be achieved with no queries to the oracle.  $\square$

<sup>16</sup>Here we took advantage of the fact that  $\text{verify}$  in our scheme is classical. This should clarify why our requirement in the VBB property is classical computational indistinguishability (and not quantum indistinguishability).

Constructing a dependent-auxiliary-input VBB obfuscator for  $\mathcal{G}$  based on some reasonable cryptographic primitive remains an interesting open problem. Since there is no VBB construction with these properties, we conjecture, instead, that any quantum secure  $i\mathcal{O}$  is a VBB obfuscator which satisfies the requirement in Definition 23. By Theorem 24, under this conjecture our scheme is provably secure. To instantiate the protocol, we can use the  $i\mathcal{O}$  candidate construction of [GGH<sup>+</sup>13] (or perhaps one which is based on one of the later candidates for multilinear maps [CLT13, GGH15, CLT15, Hal15, MSZ16]).

The plausibility arguments for this conjecture follows. First, since there is no impossibility theorem that applies, and that a related problem – hyperplane VBB obfuscation – has been shown [CRV10], it is plausible that there exists *some* VBB Obfuscation for our class. Second, any  $i\mathcal{O}$  scheme is also a VBB obfuscation scheme for our class if one exists [GR14].<sup>17</sup> If the scheme is a VBB obfuscation for our class, then it is plausibly also a VBB obfuscation with auxiliary inputs, which is sufficient to prove the security of our scheme, by Theorem 24.

## 8 Sending Quantum Money over a Classical Channel

In this section, we describe several interesting properties of a quantum money scheme that is derived from a tokenized (either private or public) signature scheme. In particular, we show how to convert a quantum coin into a classical “check”, which is addressed to a specific person and can be exchanged for a quantum coin at a bank branch.

Recall from Theorem 12, and its private analog (see Appendix C), that a testable public (private) digital signature scheme satisfies the definition of a public (private) quantum money scheme, with the signing tokens acting as the quantum coins. Now, in this setting, if Alice holds a quantum coin, one thing she can do is spend it the usual way. However, an alternative thing she can do with the coin is use it to sign a message. Such a signature will necessarily consume the coin, and hence can be used as proof that Alice has burned her coin.

In particular, consider what happens if Alice signs the message, “I wish to give a coin to Bob”. Alice can send over this classical signature to Bob, who can verify the signature using the public verification scheme. Bob can then show this signature to his bank branch. Crucially, the bank knows that Alice burned her coin, since the signature is valid; hence the bank can safely issue Bob a coin. In essence, this usage converts a quantum coin into a classical check!

There are a few potential flaws with this idea, particularly relating to double spending: what happens if Bob tries to cash his check twice? To prevent this, Bob can ask Alice to add a serial number or a time stamp to the check, signing a message like “I wish to give a coin to Bob; the current date is September 27, 2016, at 11:59pm”. The time stamp can be verified by Bob to be accurate, and ensures that Alice does not give Bob the same check twice. The bank can then keep a database of all the checks cashed, and reject duplicate cashing of the same check.

This solution requires the bank to keep a database of cashed checks, which raises a further issue: what if Bob tries to cash the check at two different branches simultaneously, without the

---

<sup>17</sup>Note, however, that this only gives VBB obfuscation without auxiliary input.

branches getting a chance to compare notes? To address this issue, we add a further modification to the scheme: Bob can tell Alice the name of the branch he plans to go to (e.g. “branch number 317”), and Alice can add this branch name to her check (“this check cashable at branch number 317”). The bank branch can then only accept checks that specify that branch in this way. This allows each branch to keep a completely separate database, without any need for communication between the branches.

## 8.1 Implications

The scheme above has several useful properties, which are listed below. The first two are relevant to both variants (public and private), and the last two are relevant only for public tokenized signature scheme.

1. If Alice loses quantum communication with the outside world, she can still use her money for payment – even without the entanglement that would be needed for quantum teleportation. The only schemes which were shown to be classically verifiable are [Gav12, GK15]. Our scheme achieves that with only one round of communication. Therefore, We call this a “check”: verification is not of the type of a challenge and response, like the previously classically verifiable quantum money schemes mentioned before.
2. The bank branches in this scheme do not need to communicate in order to cash checks; everything can be done completely offline. Furthermore, each branch’s data remain static – no data needs to be updated, or even be stored (besides the scheme’s key) [Gav12]. This can be done if, for example, each bank branch only accepts checks once a day, and only if the check’s timestamp is from the previous day. In this case the branch does not need to store a database of previous checks cashed.
3. Public quantum money has the nice property that the transactions are *untraceable*, so that third parties need not know that the transaction took place. While this is an advantage for privacy, it can also be a disadvantage: if Alice pays Bob, he can deny receiving the money, and Alice has no way to prove that she has paid. With our scheme, Alice can *choose* the method of payment: either the untraceable way, or else by using the traceable classical check, which allows Alice to prove that she has paid. Proving the payment, say, to the police, does not require the bank’s cooperation.
4. In the public setting, the bank branches need not store the secret key of the quantum money scheme. This means that if a branch is robbed or hacked, the thief gains the quantum coins stored in the branch, but does not gain the ability to mint new coins; no long-term damage to the monetary system is inflicted. None of the private money schemes have this property.

The main disadvantages of the scheme are:

- Alice needs the ability to perform universal quantum computation and store *entangled* quantum states long-term, unlike several private money schemes [Wie83, BBBW83, Gav12, GK15].
- Our public scheme requires computational assumptions, and even worse, non-standard ones. The private setting is comparable to the previous constructions in terms of computational assumptions: if one does not care that the key length grows linearly with the number of money states in the system, no computational assumptions are needed; otherwise, only a weak assumption, namely, a quantum secure one way function is required.
- Our scheme does not tolerate constant level of noise out of the box [PYJ<sup>+</sup>12].

## 9 Circumventing two faced behavior

Consider a typical scenario in distributed computing, which involves Alice, Bob and Charlie (although the following applies when there are  $n$  parties). Alice has some preference, which she needs to communicate to Bob and Charlie. If she is honest, she sends her preference. But Alice, as well as Bob and Charlie may be malicious. In consensus problems, it is crucial that honest Bob and Charlie will output the same outcome, even if Alice is malicious. One barrier is if Alice pretends that her preference is  $x$  to Bob, and  $y \neq x$  to Charlie – this is called *a two-faced behavior*. If Charlie tells Bob that Alice’s preference is  $x$ , and Alice tells Bob that her preference is  $y \neq x$ , Bob cannot determine whether Alice or Charlie (or both) is the cheater.

If we assume that Alice receives exactly one signing token, the two-faced behavior can be circumvented: she can only sign one message, and therefore cannot send two contradicting signed messages to Bob and Charlie. This should be compared to the more complicated solution, which only assumes digital signatures (see [Lyn96, Section 6.2.4] and references therein): 1) Alice sends her signed preference to Bob and Charlie. 2) Bob and Charlie compare the signed messages they received. If the preferences are not the same, and both pass the verification, they conclude that Alice is the cheater.

Even though the construction and analysis of the solution which uses signing tokens is simpler than the one which uses digital signatures, as far as we know, it does not improve the asymptotic running time for problems such as the Byzantine agreement and its variants.

## 10 Open Questions


Are there any other tokenized schemes? The immediate candidates are variants of digital signatures, such as blind signatures, group signatures and ring signatures. Can the techniques that were introduced in this work be used to construct quantum copy-protection [Aar09]?

Can we add the testability property to any TS scheme, similarly to the way we showed how revocability can be added (see Theorem 10)? We can show that there is a test which guarantees

that the signing token can be used to sign a random document w.h.p.; but we cannot construct a test which guarantees that the signing token can be used to sign every document w.h.p.

## 11 Acknowledgments

We thank Scott Aaronson, Dorit Aharonov, Nir Bitansky, Zvika Brakerski, Aram Harrow, Robin Kothari, Gil Segev and Vinod Vaikuntanathan for valuable discussions. We thank Paul Christiano for reporting to us the attack described in Section 7.2, and for his other comments.

This work was partially supported by NSF grant no. 2745557, and ERC Grant 030-8301. The authors acknowledge the hospitality of the Center for Theoretical Physics at MIT. The icon  was downloaded from <http://icons8.com>, and is licensed under Creative Commons Attribution-NonDerivs 3.0 Unported.

## References

- [Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 229–242, 2009. doi:10.1109/CCC.2009.42.
- [AAV93] Yakir Aharonov, Jeeva Anandan, and Lev Vaidman. Meaning of the wave function. *Phys. Rev. A*, 47:4616–4626, 1993. doi:10.1103/PhysRevA.47.4616.
- [AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 41–60, 2012. doi:10.1145/2213977.2213983.
- [AC13] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. *Theory of Computing*, 9:349–401, 2013. doi:10.4086/toc.2013.v009a009.
- [AF16] Gorjan Alagic and Bill Fefferman. On quantum obfuscation. 2016. URL: <http://arxiv.org/abs/1602.01771>.
- [AV93] Yakir Aharonov and Lev Vaidman. Measurement of the Schrödinger wave of a single particle. *Physics Letters A*, 178(1):38 – 42, 1993. doi:10.1016/0375-9601(93)90724-E.
- [Bar16] Boaz Barak. Hopes, fears, and software obfuscation. *Commun. ACM*, 59(3):88–96, 2016. doi:10.1145/2757276.
- [BBBW83] Charles H Bennett, Gilles Brassard, Seth Breidbart, and Stephen Wiesner. Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology*, pages 267–275. Springer, 1983. doi:10.1007/978-1-4757-0602-4\_26.

- [BCC<sup>+</sup>14] Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 71–89. Springer, 2014. doi:10.1007/978-3-662-44381-1\_5.
- [BGI<sup>+</sup>12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012. doi:10.1145/2160158.2160159.
- [Bom11] H Bombin. Clifford gates by code deformation. *New Journal of Physics*, 13(4):043005, 2011. URL: <http://stacks.iop.org/1367-2630/13/i=4/a=043005>.
- [Bra97] Gilles Brassard. Searching a quantum phone book. *Science*, 275(5300):627–628, 1997. doi:10.1126/science.275.5300.627.
- [CG97] Richard Cleve and Daniel Gottesman. Efficient computations of encodings for quantum error correction. *Phys. Rev. A*, 56:76–82, Jul 1997. URL: <http://link.aps.org/doi/10.1103/PhysRevA.56.76>, doi:10.1103/PhysRevA.56.76.
- [Chr15] Paul Christiano. Personal communication, 2015.
- [CLT13] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493. Springer, 2013. doi:10.1007/978-3-642-40041-4\_26.
- [CLT15] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 267–286. Springer, 2015. doi:10.1007/978-3-662-47989-6\_13.
- [CRR05] Sourav Chakraborty, Jaikumar Radhakrishnan, and Nandakumar Raghunathan. Bounds for error reduction with few quantum queries. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*,



*APPROX 2005 and RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 245–256, 2005. doi:10.1007/11538462\_21.

- [CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 72–89. Springer, 2010. doi:10.1007/978-3-642-11799-2\_5.
- [DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.
- [FGH<sup>+</sup>10] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, Daniel Nagaj, and Peter Shor. Quantum state restoration and single-copy tomography for ground states of Hamiltonians. *Phys. Rev. Lett.*, 105:190503, Nov 2010. doi:10.1103/PhysRevLett.105.190503.
- [FGH<sup>+</sup>12] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, and Peter Shor. Quantum money from knots. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 276–289. ACM, 2012. doi:10.1145/2090236.2090260.
- [Gav12] Dmitry Gavinsky. Quantum money with classical verification. In *IEEE 27th Annual Conference on Computational Complexity*, pages 42–52. IEEE, 2012. doi:10.1109/CCC.2012.10.
- [GGH<sup>+</sup>13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49, Oct 2013. doi:10.1109/FOCS.2013.13.
- [GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527. Springer, 2015. doi:10.1007/978-3-662-46497-7\_20.
- [GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 553–562, 2005. doi:10.1109/SFCS.2005.60.
- [GK15] Marios Georgiou and Iordanis Kerenidis. New constructions for quantum money. In Salman Beigi and Robert König, editors, *10th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2015, May 20-22, 2015*,

- Brussels, Belgium*, volume 44 of *LIPICs*, pages 92–110. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPICs.TQC.2015.92.
- [Gol04] Oded Goldreich. *The Foundations of Cryptography - Vol. 2, Basic Applications*. Cambridge University Press, 2004.
- [GR14] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. *J. Cryptology*, 27(3):480–505, 2014. doi:10.1007/s00145-013-9151-z.
- [GRB03] Markus Grassl, Martin Rötteler, and Thomas Beth. Efficient quantum circuits for non-qubit quantum error-correcting codes. *Int. J. Found. Comput. Sci.*, 14(5):757–776, 2003. doi:10.1142/S0129054103002011.
- [Hal15] Shai Halevi. Graded encoding, variations on a scheme. *IACR Cryptology ePrint Archive*, 2015:866, 2015. URL: <http://eprint.iacr.org/2015/866>.
- [Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [MS10] Michele Mosca and Douglas Stebila. *Quantum coins*, volume 523 of *Contemp. Math.*, pages 35–47. Amer. Math. Soc., 2010. doi:10.1090/conm/523/10311.
- [MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Secure obfuscation in a weak multilinear map model: A simple construction secure against all known attacks. *IACR Cryptology ePrint Archive*, 2016:588, 2016. URL: <http://eprint.iacr.org/2016/588>.
- [PFP15] Marta Conde Pena, Jean-Charles Faugère, and Ludovic Perret. Algebraic cryptanalysis of a quantum money scheme the noise-free case. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 194–213. Springer, 2015. doi:10.1007/978-3-662-46447-2\_9.
- [Pra10] Amritanshu Prasad. Counting subspaces of a finite vector space — 1. *Resonance*, 15(11):977–987, 2010. doi:10.1007/s12045-010-0114-5.
- [PYJ<sup>+</sup>12] Fernando Pastawski, Norman Y. Yao, Liang Jiang, Mikhail D. Lukin, and J. Ignacio Cirac. Unforgeable noise-tolerant quantum tokens. *Proceedings of the National Academy of Sciences*, 109(40):16079–16082, 2012. doi:10.1073/pnas.1203552109.
- [TGP06] Tathagat Tulsi, Lov K. Grover, and Apoorva Patel. A new algorithm for fixed point quantum search. *Quantum Information & Computation*, 6(6):483–494, 2006. URL: <http://portal.acm.org/citation.cfm?id=2011693>.
- [TOI03] Yuuki Tokunaga, Taisuaki Okamoto, and Nobuyuki Imoto. Anonymous quantum cash. In *ERATO Conference on Quantum Information Science*, 2003.

- [Wie83] Stephen Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983. doi: 10.1145/1008908.1008920.
- [WZ82] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

## A Nomenclature

### Nomenclature

- $\{E_j\}$  POVM elements, page 32
- $\langle A \rangle$  A basis for  $A$ , page 10
- Adv A QPT adversary, page 4
- $\alpha$  A document or a message, that usually needs to be signed, page 3
- $\ell$  The number of signing tokens given to the adversary, page 4
- $\chi_A$  The membership function to  $A$ , page 10
- $i\mathcal{O}$  Indistinguishability obfuscation, page 7
- $|\underline{\omega}\rangle$  A quantum signing token, page 4
- $|A^\perp\rangle$  The uniform superposition over elements of  $A^\perp$ , page 11
- $|A\rangle$  The uniform superposition over elements of  $A$ , page 10
- $\Lambda(A)$  The set  $A \setminus \{0\} \times A^\perp \setminus \{0\}$ , page 25
- $\mathcal{O}_C$  An obfuscated program of the circuit  $C$ , page 32
- $\mathcal{R}_A$  The set of  $B \in S(n)$  such that  $(A, B) \in \mathcal{R}$ , page 26
- $\mathcal{R}$  The relation that contains all  $(A, B)$  such that  $\dim(A \cap B) = \frac{n}{2} - 1$ , page 26
- $\mathbb{P}_A$  The projection to the span of the elements in  $A$ , page 30
- $\chi_{A^*}$  The unified membership to  $A$  and  $A^\perp$ , page 10
- verify $_{k,pk}$  Verification of  $k$  documents, using the public key  $pk$ , page 14
- $A$  A subspace of  $\mathbb{F}_2^n$ , usually of dimension  $\frac{n}{2}$ , page 10
- $A^\perp$  The orthogonal complement of  $A$ , with respect to the standard dot product in  $\mathbb{F}_2^n$ , page 10

$C$	A circuit, page 32
$p_1, \dots, p_{\beta n}$	Polynomials which obfuscate $\chi_A$ , page 31
$r$	The range specifier of a hashing scheme. Also the length of a length-restricted TS scheme (see Def. 5), page 11
$s$	The index of a hash function, page 11
$S(n)$	The set of all subspaces of $\mathbb{F}_2^n$ with dimension $\frac{n}{2}$ , page 10
DS	Digital Signature, page 11
PPT	Probabilistic Polynomial Time, page 3
QPT	Quantum Polynomial Time, page 4
TS	Tokenized Signature scheme, page 13
VBB	virtual black box, a form of obfuscation, page 32

## B Memoryless Digital Signatures from Tokens

In this section, we show how to get a general (not memory dependent) digital signature scheme from a tokenized signature scheme with some extra security properties, which we call super-security and unpredictability. Super-security is a stronger notion of security than unforgeability. For that we define the algorithm  $\text{verify}'_{k,pk}$ , in which requirement (i) from  $\text{verify}_{k,pk}$  is changed to: all pairs are distinct, i.e.  $(\alpha_i, s_i) \neq (\alpha_j, s_j)$  for every  $1 \leq i \neq j \leq k$ .

**Definition 25** (Super-security and Onetime Super-security). A TS scheme is super-secure if for every  $\ell = \text{poly}(\kappa)$  a QPT adversary cannot provide  $\ell + 1$  different signatures by using the public key and  $\ell$  signing tokens:

$$\Pr \left[ \text{verify}'_{\ell+1,pk}(\text{Adv}(pk, |\mathfrak{D}_1\rangle \otimes \dots \otimes |\mathfrak{D}_\ell\rangle)) = T \right] \leq \text{negl}(\kappa)$$

Onetime super-security requires the above only for  $\ell = 1$ .

This is analogous to super-secure digital signatures, see [Gol04, Section 6.5.2].

**Definition 26** (Unpredictability). A TS scheme is unpredictable if signing the same document twice, using two signing tokens, gives two different signatures except with negligible probability:

$$\Pr \left[ \text{sign}(\alpha, |\mathfrak{D}_1\rangle) = \text{sign}(\alpha, |\mathfrak{D}_2\rangle) \right] \leq \text{negl}(\kappa)$$

We note that unpredictability for two documents implies unpredictability for polynomially many documents.

**Theorem 27.** *If TS is an unpredictable (see Definition 26) and super secure (see Definition 25) tokenized signature scheme then the scheme DS as defined in Section 4 is an unforgeable digital signature scheme (see Definition 2).*

*Proof of Theorem 27.* Assume towards a contradiction that DS is forgeable: there exists a QPT Adv for which,

$$\Pr \left[ (\alpha, sig) \leftarrow \text{Adv}^{\text{DS.sign}_{sk}}(pk), \text{DS.verify}_{pk}(\alpha, sig) = T \text{ and } \alpha \notin Q_{\text{Adv}}^{\text{sign}_{sk}} \right] \geq \text{non-negl}(\kappa). \quad (15)$$

We define a different adversary  $\text{Adv}'$  which receives as an input  $\ell$  signing tokens (here  $\ell = \text{poly}(\kappa)$  is the number of oracle queries  $\text{Adv}^{\text{DS.sign}_{sk}}(1^\kappa)$  applies), and simulates  $\text{Adv}^{\text{DS.sign}}(1^\kappa)$ . Whenever  $\text{Adv}^{\text{DS.sign}}(1^\kappa)$  makes an oracle query to sign the document  $\alpha$ ,  $\text{Adv}'$  uses one of its signing tokens to sign that document. Suppose the documents that were given to the signing oracle as queries are  $\alpha_1, \dots, \alpha_\ell$  and the outputs are  $sig_1, \dots, sig_\ell$ , and we denote the outputs  $(\alpha, sig)$  of  $\text{Adv}^{\text{DS.sign}}$  by  $(\alpha_{\ell+1}, sig_{\ell+1})$ .

$$\begin{aligned} & \Pr[\text{verify}'_{\ell+1, pk}(\text{Adv}'(pk, |\mathfrak{D}_1\rangle \otimes |\mathfrak{D}_\ell\rangle)) = T] \\ &= \Pr \left[ \text{verify}'_{\ell+1, pk}(\alpha_1, sig_1, \dots, \alpha_{\ell+1}, sig_{\ell+1}) = T \right] \\ &= \Pr \left[ \text{verify}_{pk}(\alpha_{\ell+1}, sig_{\ell+1}) = T \text{ and } (\alpha_i, sig_i) \neq (\alpha_j, sig_j) \forall i \neq j \leq \ell + 1 \right] \\ &\geq \Pr \left[ \text{verify}_{pk}(\alpha_{\ell+1}, sig_{\ell+1}) = T \text{ and } \forall i \leq \ell, (\alpha_i, sig_i) \neq (\alpha_{\ell+1}, sig_{\ell+1}) \right] - \text{negl}(\kappa) \\ &\geq \Pr \left[ \text{verify}_{pk}(\alpha_{\ell+1}, sig_{\ell+1}) = T \text{ and } \forall i \leq \ell, \alpha_i \neq \alpha_{\ell+1} \right] - \text{negl}(\kappa) \\ &= \Pr \left[ (\alpha, sig) \leftarrow \text{Adv}^{\text{DS.sign}_{sk}}(pk), \text{DS.verify}_{pk}(\alpha, sig) \text{ and } \alpha \notin Q_{\text{Adv}}^{\text{sign}_{sk}} \right] - \text{negl}(\kappa) \\ &\geq \text{non-negl}(\kappa) \end{aligned} \quad (16)$$

The second step is justified by our assumption that  $(\alpha_1, sig_1), \dots, (\alpha_\ell, sig_\ell)$  are the outputs of the signing oracle and therefore pass the verification test by the correctness property. The unpredictability is used in the third step, and Eq. (15) for the last step. To conclude, Eq. (16) shows that TS is not super-secure, contradicting our assumption.  $\square$

By removing the second inequality in Eq. (16) and adapting the following step, the proof above can be strengthened to show that DS is a super-secure digital signature scheme – see [Gol04, Section 6.5.2].

## B.1 Proving Super-Security and Unpredictability

We show that our construction of tokenized signatures is super-secure and unpredictable (under the same obfuscation assumption as before).

We first claim that all our reductions maintain super-security (Definition 25) and unpredictability (Definition 26). It is easy to see that if a one-bit tokenized scheme gives different signatures each time a new pair of keys is generated (except with negligible probability), then the resulting full tokenized signature scheme will be unpredictable. Showing that a onetime super-secure tokenized signature scheme gives rise to a full super-secure tokenized signature scheme is a bit more involved, but directly follows the proof of Theorem 13.

Next, we show that the oracle scheme is onetime super-secure.

**Theorem 28.** *Let  $A$  be a uniformly random subspace from  $S(n)$ , and let  $\epsilon > 0$  be such that  $1/\epsilon = o(2^{n/2})$ . Given one copy of  $|A\rangle$  and a membership oracle for  $A$ , a counterfeiter needs  $\Omega(\sqrt{\epsilon}2^{n/4})$  queries to output a pair  $(a, b) \in (A \setminus \{0\}) \times (A \setminus \{0\})$  satisfying  $a \neq b$  with probability at least  $\epsilon$ . The same number of queries is also required to output a pair  $(a, b) \in (A^\perp \setminus \{0\}) \times (A^\perp \setminus \{0\})$  satisfying  $a \neq b$  with probability at least  $\epsilon$ .*

*Proof.* First, note that finding such a pair of elements in  $A$  and finding such a pair in  $A^\perp$  are essentially the same task (up to renaming  $A$  and  $A^\perp$ ); it suffices to prove the result for a pair in  $A$ , and the result for  $A^\perp$  will follow by symmetry.

Our proof will be extremely similar to the proof of Theorem 16. In fact, if we simply redefine  $\Lambda(A)$  to be the set  $\{(a, b) \in A^2 : a \neq b, a \neq 0, b \neq 0\}$ , the whole proof goes through except for Lemma 19. We therefore prove an analogue of this lemma with the redefined  $\Lambda(A)$ .

Fix  $A \in S(n)$  and  $(a, b) \in \Lambda(A)$ , so  $a \neq b$ ,  $a, b \in A$ , and  $a, b \neq 0$ . Picking  $B \sim \mathcal{R}_A$  is picking a vector space whose intersection with  $A$  has dimension  $n/2 - 1$ . One way of picking such a vector space uniformly at random is by picking a random basis for  $A$ , then discarding one of the vectors in the basis and adding a vector outside of  $A$ . This gives a new set of  $n/2$  independent vectors, which we take to be a basis for  $B$ .

We are interested in the probability that  $(a, b) \in \Lambda(B)$ . This is simply the probability that  $a, b \in B$ , which is equal to the probability that  $a$  and  $b$  are both in the span of a randomly chosen set of  $n/2 - 1$  independent vectors in  $A$ .

Now, the number of subspaces of  $A$  of dimension  $n/2 - 1$  is  $G(n/2, n/2 - 1)$ , and the number of such subspaces that contain  $a$  and  $b$  is simply  $G(n/2 - 2, n/2 - 3)$ . The probability that a random subspace contains  $a$  and  $b$  is therefore

$$\frac{G(n/2 - 2, n/2 - 3)}{G(n/2, n/2 - 1)},$$

which simplifies to

$$\frac{2^{n/2-2} - 1}{2^{n/2} - 1} = \frac{1}{4} \left( 1 - \frac{3}{2^{n/2} - 1} \right) \leq \frac{1}{4}.$$

□

Finally, we show that the construction from a VBB obfuscator with dependent auxiliary inputs is onetime super-secure, which implies that under our strong obfuscation assumption our scheme is unforgeable and super-secure (and thus can be used as a digital signature scheme).

**Theorem 29.** *If  $\mathcal{O}$  is a VBB obfuscator with dependent auxiliary input for the class  $\mathfrak{S}$ , then the scheme OT1 described in Algorithm 8 is onetime super-secure.*

*Proof.* The proof is essentially identical to the proof that the scheme is unforgeable, except we use  $\text{verify}'$  instead of  $\text{verify}$ . Fix  $A$ . Suppose the distinguisher runs the (classical) algorithm  $\text{verify}'_{2,pk(A)}$ , and the auxiliary state is  $|A\rangle$ . By the VBB property,

$$\left| \Pr_{\text{Adv}, \mathcal{O}} [\text{verify}'_{2,pk(A)}(\text{Adv}(\mathcal{O}(\chi_{A^*}), |A\rangle)) = 1] - \Pr_{\text{Sim}} [\text{verify}'_{2,pk(A)}(\text{Sim}^{\chi_{A^*}}(1^{|\chi_{A^*}|}, |A\rangle)) = 1] \right| \leq \text{negl}(\kappa).$$

Here, we were explicit over what the probability is.

The above holds for every  $A \in S$ . We now take the average over all  $A \in S$ :

$$\left| \Pr_{A \leftarrow S, \text{Adv}, \mathcal{O}} [\text{verify}'_{2,pk(A)}(\text{Adv}(\mathcal{O}(\chi_{A^*}), |A\rangle)) = 1] - \Pr_{A \in R, S, \text{Sim}} [\text{verify}'_{2,pk(A)}(\text{Sim}^{\chi_{A^*}}(1^{|\chi_{A^*}|}, |A\rangle)) = 1] \right| \leq \text{negl}(\kappa)$$

By Theorem 28, the second term of the above equation can be bounded. For every QPT  $\text{Sim}$ ,

$$\Pr_{A \leftarrow S, \text{Sim}} [\text{verify}'_{2,pk(A)}(\text{Sim}^{\chi_{A^*}}(1^{|\chi_{A^*}|}, |A\rangle)) = 1] = \frac{\text{poly}(\kappa)}{2^{n/2}} = \text{negl}(\kappa)$$

Combining the above two equations, we conclude that onetime super-security (see Def. 25) holds:

$$\Pr_{A \leftarrow S, \text{Adv}, \mathcal{O}} [\text{verify}'_{2,pk(A)}(\text{Adv}(\mathcal{O}(\chi_{A^*}), |A\rangle)) = 1] \leq \text{negl}(\kappa).$$

□

## C Message Authentication Codes

### C.1 Definitions

A message authentication code is defined the same way as a digital signature scheme, except that the verification key (the “public” key) is now only given to one party, and not to the adversary. In other words, it is a private-key version of digital signatures. In such a scheme we can assume without loss of generality that the signing and verification keys are the same. We now formally define message authentication codes.

**Definition 30** (Message Authentication Code). A message authentication code (MAC) consists of 3 PPT algorithms  $\text{key-gen}$ ,  $\text{sign}$  and  $\text{verify}$  satisfying:

1.  $\text{key-gen}(1^\kappa)$  outputs a key  $k$ , where  $\kappa$  is the security parameter.

2. When a document is signed using the signing key, the signature is accepted by the verification algorithm using the verification key. That is, for every  $\alpha \in \{0, 1\}^*$ ,

$$\Pr [\text{verify}_k(\alpha, \text{sign}_k(\alpha)) = T] = 1$$

3. The scheme is secure against quantum existential forgery under chosen message attacks; that is, a quantum adversary with the capability of adaptively requesting documents to be signed by a signing oracle (but without access to the key  $k$ ) cannot generate a signature for any document they did not ask the oracle about:

$$\Pr [(\alpha, \text{sig}) \leftarrow \text{Adv}^{\text{sign}_k}, \text{verify}_k(\alpha, \text{sig}) = T \text{ and } \alpha \notin Q_{\text{Adv}}^{\text{sign}_k}] \leq \text{negl}(\kappa),$$

where  $\text{Adv}^{\text{sign}_k}$  is a QPT algorithm with access to a classical signing oracle, and  $Q_{\text{Adv}}^{\text{sign}_k}$  is the set of documents the signing oracle has signed.

We define a private (length-restricted) tokenized signature scheme in analogy to a public (length-restricted) public tokenized digital signature scheme. The only difference occurs in the symmetric key, and the definition of unforgeability.

**Definition 31** (Private tokenized signature scheme (private TS)). A private TS scheme consists of 4 QPT algorithms, key-gen, token-gen, sign, and verify, with the following properties:

1. On input  $1^\kappa$  where  $\kappa$  is the security parameter, key-gen outputs a classical key  $k$ .
2. token-gen $_k$  generates a signing token  $|\underline{\alpha}\rangle$ . We emphasize that if token-gen $_k$  is called  $\ell$  times it may (and in our construction, would) output different states  $|\underline{\alpha}_1\rangle, \dots, |\underline{\alpha}_\ell\rangle$ .
3. For every document  $\alpha \in \{0, 1\}^*$ ,

$$\Pr [|\underline{\alpha}\rangle \leftarrow \text{token-gen}_k, \text{verify}_k(\alpha, \text{sign}(\alpha, |\underline{\alpha}\rangle)) = T] = 1.$$

In an imperfect scheme, the above requirement is relaxed:

$$\Pr [|\underline{\alpha}\rangle \leftarrow \text{token-gen}_k, \text{verify}_k(\alpha, \text{sign}(\alpha, |\underline{\alpha}\rangle)) = T] \geq 1 - \text{negl}(\kappa).$$

**Definition 32** (Length restricted private TS). A private TS is  $r$ -restricted if Item 3 holds only for  $\alpha \in \{0, 1\}^r$ .

To define unforgeability, we introduce the algorithm  $\text{verify}_{\ell, k}$ . This algorithm takes as an input  $\ell$  pairs  $(\alpha_1, s_1), \dots, (\alpha_\ell, s_\ell)$  and accepts if and only if (i) all the documents are distinct, i.e.  $\alpha_i \neq \alpha_j$  for every  $1 \leq i \neq j \leq \ell$ , and (ii) all the pairs pass the verification test  $\text{verify}_k(\alpha_i, s_i)$ .



**Definition 33** (Unforgeability and onetime unforgeability). A private TS scheme is unforgeable if for every  $\ell = \text{poly}(\kappa)$  a QPT adversary cannot sign  $\ell + 1$  different documents by using  $\ell$  signing tokens:

$$\Pr \left[ \text{verify}_{\ell+1, k}(\text{Adv}(|\mathfrak{D}_1\rangle \otimes \dots \otimes |\mathfrak{D}_\ell\rangle)) = T \right] \leq \text{negl}(\kappa)$$

Onetime unforgeability requires the above only for  $\ell = 1$ .

For the construction of the private TS scheme, we will also need the concept of private key encryption schemes which is semantically secure against quantum adversaries, which we now define.

**Definition 34** (Private key encryption [Gol04, Section 5.2.2]). A quantum secure private key encryption scheme consists of 3 PPT algorithms, key-gen, encrypt, and decrypt, with the following properties:

1. On input  $1^\kappa$  where  $\kappa$  is the security parameter, key-gen outputs a key  $e$ .
2.  $\text{encrypt}_e$  and  $\text{decrypt}_e$  are maps from  $\{0, 1\}^*$  to  $\{0, 1\}^*$  such that for all  $\alpha \in \{0, 1\}^*$ ,

$$\Pr [\text{decrypt}_e(\text{encrypt}_e(\alpha)) = \alpha] \geq 1 - \text{negl}(\kappa).$$

3. A QPT adversary cannot distinguish between the encryption of two polynomially sized documents with non-negligible probability. That is, for all  $\alpha_1, \alpha_2 \in \{0, 1\}^{\text{poly}(\kappa)}$  with  $|\alpha_1| = |\alpha_2|$  and all QPT algorithms  $\text{Adv}$ , we have

$$\Pr[\text{Adv}(\text{encrypt}_e(\alpha_1)) \neq \text{Adv}(\text{encrypt}_e(\alpha_2))] \leq \text{negl}(\kappa).$$

## C.2 Extending a 1-bit scheme to a full scheme

We now show how to extend a onetime 1-bit length restricted private TS scheme into a full private TS scheme.

**Lemma 35.** *A 1-restricted onetime private TS OT1 can be transformed into an  $r$ -restricted private TS OTR, such that if OT1 is unforgeable, so is OTR. Here  $r$  is an integer which is at most polynomial in the security parameter  $\kappa$ .*

*Proof.* As with public tokenized signature schemes, we construct OTR from OT1 by repeating it  $r$  times; that is, key-gen for OTR will run key-gen for OT1  $r$  times, and use the tuple of signing keys as its signing key and the tuple of verification keys as its verification key. Signing a document  $\alpha \in \{0, 1\}^r$  works by simply signing each bit of  $\alpha$  separately. Soundness follows from the soundness of the OT1 scheme; that is, the verification of a valid signature for  $\alpha$  fails to accept if and only if one of the verifications for the  $r$  copies of the OT1 scheme fails to accept. By the union bound, this is at most  $r$  times the probability that OT1 fails to verify a valid signature; this is 0 in a perfect scheme and  $r \cdot \text{negl}(\kappa)$  in an imperfect scheme.

We show unforgeability. Suppose there was an adversary  $\text{Adv}$  that signed two different documents of length  $r$  with non-negligible probability  $f(\kappa)$ . Consider a new adversary  $\text{Adv}'$  that attempts to sign two different documents of length 1.  $\text{Adv}'$  takes as input one signing token. It then runs  $\text{key-gen}$  and  $\text{token-gen}$   $r - 1$  times, to get  $r - 1$  additional signing tokens. It shuffles these together, and uses the  $\text{Adv}$  algorithm to sign two different documents (for OTR) with non-negligible probability  $f(\kappa)$ . Since the messages are different, they must differ on some bit; there is a  $1/r$  chance that this bit is the same bit as the “real” input token, rather than the artificially generated ones. In that case,  $\text{Adv}'$  successfully created valid signatures for two different documents relative to OT1. This happens with probability  $f(\kappa)/r$ , which is non-negligible, giving a contradiction.  $\square$

**Lemma 36.** *An  $r$ -restricted onetime private TS OTR can be transformed into a onetime unrestricted private TS OT with the help of a hash function, such that if OTR is unforgeable and the hash function is collision-resistant, then OT is unforgeable.*

*Proof.* The OT scheme works by the hash-and-sign paradigm: we first hash a document down to length  $r$ , and then sign the hash. Here  $r$  needs to be a function of the security parameter  $\kappa$ , say  $r = \kappa$ . The soundness of the scheme is clear. The unforgeability also follows easily: if an adversary produces valid signatures for two messages, then either the two messages have the same hash value (which contradicts the collision-resistance of the hash function), or else the two messages have different hash values (which contradicts the unforgeability of the OTR scheme).  $\square$

**Lemma 37.** *A onetime private TS OT can be transformed into a full-blown private TS TM with the help of a quantum secure MAC and a quantum secure classical encryption scheme.*

*Proof.* The scheme TM will work as follows. The algorithm  $\text{key-gen}$  will generate a key  $k$  for the classical MAC, as well as a key  $e$  for the classical encryption scheme, using the security parameter  $\kappa$ . The pair  $(k, e)$  will be the key of TM.

The algorithm  $\text{token-gen}_{k,e}$  will then use OT to generate a key  $k_1$  and a signing token  $|\underline{\alpha}_1\rangle$ ; it will then encrypt  $k_1$  using the encryption key  $e$ , and sign the resulting encrypted message using the MAC and  $k$ . The final signing token will be  $|\underline{\alpha}_1\rangle$  appended with the encrypted key  $\text{encrypt}_e(k_1)$ , and the its signature  $\text{sign}_k^{\text{MAC}}(\text{encrypt}_e(k_1))$ .

The new signing algorithm  $\text{sign}$  applies the signing procedure of OT, and appends to this signature the signed encrypted verification key that came as part of the token.

The algorithm  $\text{verify}$  uses  $k$  to verify the signature of the encrypted key, uses  $e$  to decrypt the key  $k_1$ , and uses  $k_1$  to verify the signature according to the OT protocol.

It’s clear that the verification procedure accepts the output of the signing procedure (assuming this holds for OT). We now show that if OT is unforgeable, then so is TM.

Suppose by contradiction that there was an adversary  $\text{Adv}$  that used  $\ell$  signing tokens and produced  $\ell + 1$  signatures  $a_1, a_2, \dots, a_{\ell+1}$  that are distinct and accepted with non-negligible probability  $f(\kappa)$  by the verification algorithm. Each  $a_i$  must have the form  $a_i = (\alpha_i, s_i, q_i, g_i)$  where  $\alpha_i$  are the distinct documents that were signed,  $g_i$  represents a signature for  $q_i$  under the classical MAC,  $q_i$  represents the encrypted verification key, and  $s_i$  represents the final signature for  $\alpha_i$ . If

the verification algorithm accepts the  $(a_1, a_2, \dots, a_{\ell+1})$  tuple, we know that the  $\alpha_i$  documents are all distinct, and that the verification algorithm accepts each  $a_i$ . In turn, this tells us that  $g_i$  is a valid signature for  $q_i$  under the MAC scheme, and that  $\text{decrypt}_e(q_i)$  outputs a string  $p_i$  with the property that  $\text{verify}_{p_i}(\alpha_i, s_i)$  accepts.

The fact that  $g_i$  is a valid signature for  $q_i$  means that  $q_i$  was signed using the classical MAC with non-negligible probability (otherwise, the MAC is broken by  $\text{Adv}$ ). Since  $\text{Adv}$  does not have access to the signing key or verification key for the MAC, the  $(q_i, g_i)$  must be signatures that were appended to the signing tokens. This means there are only at most  $\ell$  unique values for the  $q_i$ , so by the pigeonhole principle, some two are equal; without loss of generality, say  $q_1 = q_2$ .

Let  $q = q_1 = q_2$ , and let  $p$  be the decryption of  $q$  using  $e$ . Since the signature  $g_1$  of  $q$  is valid, we know that  $q$  is really one of the messages signed by the token-printing authority, so  $p$  is a key for one of the copies of the onetime protocol. The adversary received one token,  $|\underline{\Omega}\rangle$ , that corresponds to the key  $p$ . Since  $a_1$  and  $a_2$  are accepted, we know that  $\text{verify}_p(\alpha_1, s_1)$  and  $\text{verify}_p(\alpha_2, s_2)$  both accept with non-negligible probability; that is, the adversary produced two valid signatures for the onetime protocol. This is almost a contradiction, but not quite: the adversary did produce two valid signatures using one token for the onetime protocol, but it had access to a bit more information – an encryption of the verification key for that onetime protocol.

In other words, using the hypothesized adversary for the private TS TM, we've produced an adversary for OT that takes in the token and an encryption of the key, and produces two valid signatures (for different messages). To obtain a contradiction, we must show that the adversary must either break the encryption, or else break the unforgeability of OT.

We do this by considering what happens when we feed this adversary a valid token but a bogus encryption (e.g. the encryption of a random string rather than the verification key). There are two cases: either the adversary fails to produce two valid signatures with non-negligible probability, or else it succeeds. If it succeeds, then it breaks the OT scheme, which is a contradiction. So suppose the algorithm fails. In this case, the adversary has the ability to distinguish the real encryption of the verification key  $p$  from the encryption of a random string, using access to the token. But the token depends only on the key that is encrypted, and not on the key of the encryption scheme. Therefore, this breaks the security of the encryption scheme.  $\square$

We conclude that using a classical MAC scheme, a classical symmetric encryption scheme, and a collision-resistant hash function (all of which are secure against QPT adversaries), we can convert a 1-bit, onetime unforgeable private TS scheme into a full unforgeable private TS scheme. All three of these cryptographic primitives follow from a collision-resistant hash function (see, e.g. [Gol04]), so this is the only assumption our reduction uses.

For the actual construction of the 1-bit onetime private TS scheme, we have information-theoretic unforgeability. The scheme is simply the same as our tokenized signature scheme: the token is a superposition over a hidden subspace, and a bit is signed by measuring the state in either the computational basis or the Hadamard basis. The difference is that we no longer need to publish an obfuscated verification oracle. Without such an oracle, the proof of security is an easy special case of Theorem 16. Hence the 1-bit onetime scheme is unforgeable information-theoretically.

Also all the other results that apply for public tokenized signatures have analogous statements for private tokenized signature schemes.

By arguments similar to those in Appendix B, our 1-bit private TS scheme is information-theoretically supersecure, and our full TS scheme is supersecure as well (using a collision-resistant hash function). Also, again by analogous arguments to Appendix B, a supersecure private TS can be used as a regular MAC. Recall that the existence of a MAC follows from the existence of a one way function [Gol04]. What we've shown here is that using a slightly stronger assumption – specifically, a collision-resistant hash function instead of a one way function – we can construct a supersecure tokenized TS scheme, which is strictly stronger than a regular MAC.

Essentially by the same proof of their public analogs, it can be shown that every private tokenized scheme is also revocable (the analog of Theorem 10), that every private testable tokenized signature scheme is a private quantum money scheme (the analog of Theorem 12), and that the reductions we use to strengthen a 1-bit onetime private tokenized signature scheme to a full private tokenized signature scheme respect testability (analogous to Theorem 14) and everlasting security (Theorem 15).