Noise-Tolerant Quantum Tokens for MAC

Amit Behera[1], Or Sattath[1], and Uriel Shinar[1]

[1]Computer Science Department, Ben-Gurion University

October 7, 2021

Abstract

Message Authentication Code, or MAC, is a well-studied cryptographic primitive that is used to authenticate communication between two parties who share a secret key. A Tokenized MAC or TMAC is a related cryptographic primitive, introduced by Ben-David & Sattath (QCrypt'17), which allows limited signing authority to be delegated to third parties via the use of single-use quantum signing tokens. These tokens can be issued using the secret key, such that each token can be used to sign *at most* one document.

We provide an elementary construction for TMAC based on BB84 states. Our construction can tolerate up to 14% noise, making it the first noise-tolerant TMAC construction. The simplicity of the quantum states required for our construction, combined with its noise tolerance, make it practically more feasible than the previous TMAC construction.

The TMAC presented is existentially unforgeable against adversaries with signing and verification oracles (i.e., it is analogous to EUF-CMA security for MAC), assuming that post-quantum one-way functions exist.

In memory of Stephen Wiesner, 1942—2021.

Contents

1 Introduction

The discovery of Wiesner's quantum money protocol [Wie83] initiated the study of quantum cryptographic primitives based on the no-cloning theorem. Ben-David and Sattath [BS17] introduced one such primitive called a Tokenized Private Digital Signature scheme, or alternatively, Tokenized Message Authentication Code (TMAC). Traditional MAC schemes allow Alice to communicate with Bob in an authenticated manner by sharing a secret key. She can sign a document and send it along with its signature to Bob. Bob would then use the shared secret key to verify the signature and could respond back in the same manner. The motivation for TMAC arises from the following scenario in which a MAC scheme is not enough. Suppose it so happens that Alice will be temporarily absent for a short time, and she would like Charlie to sign a few urgent documents on her behalf. The naïve thing that Alice can do is to send her key to Charlie. However, there are two main drawbacks with this approach: (i) The secret key would allow Charlie to forever sign an arbitrary number of documents on Alice's behalf, and (ii) even if Charlie is completely trusted, Charlie's computer can get hacked — and the secret key copied — without him even noticing. These two issues can be circumvented using a TMAC scheme. A TMAC consists of four algorithms: key-gen, token-gen, sign, and verify. Alice can run the algorithm key-gen to generate a *classical* secret key k, which can then be shared with Bob, as in the case of MAC. Next, she can run token-gen$_k$ to generate a *quantum* token, denoted[1] as $|\text{\stamp}\rangle$, that she would then send to Charlie. If needed, she could repeat this procedure r times to allow Charlie to sign r documents. Charlie can then use the algorithm sign$_{|\text{\stamp}\rangle}(m)$ to generate a *classical* signature σ for a *classical* message m. Bob can verify the authenticity of the document Charlie sends by running verify$_k(m, \sigma)$.

The security guarantee of TMAC ensures that Charlie cannot produce $r+1$ signed documents (signature, document pair) given the r signing tokens he was provided with. Hence, even if Charlie's computer is hacked, the damage would be limited to only a fixed number of documents, depending on the number of tokens Charlie had been given. One may wonder why cannot Charlie run the signing algorithm once with m_1, and then run the signing algorithm again with another document m_2. This is because the signing algorithm applies a (destructive) measurement on multiple qubits, and therefore, the token is consumed during the signing.

Alice can even confirm after her return to the office, that Charlie has not kept any of the tokens or signed any unapproved documents. If Alice gave r tokens to Charlie, and he claims to have used n tokens for signing, then

---

[1] The icon $\text{\stamp}$ represents a rubber stamp.

Alice would ask for those n signed documents, and additionally, send Charlie $r - n$ fresh random documents to sign, in order to consume the remaining tokens. She would then verify all the r signatures on those r documents for confirmation. This property is known as revocability (see [BS17] for a more rigorous definition). Note that the revocation procedure mentioned above, also allows Alice to learn the n documents that Charlie signed when she was away.

Notions of Security. The security notion that we consider for TMAC schemes is inspired by the n to $n + 1$ unforgeability of quantum money schemes and the CMA security of vanilla MAC schemes. This notion asserts that a quantum polynomial time adversary who is given $n \in \mathsf{poly}(\lambda)$ many tokens and has classical access[2] to both a signing oracle and a verification oracle cannot produce valid signatures for $n+1$ distinct documents that have not been queried to the signing oracle (except with negligible probability). We say such a scheme is $\mathsf{Unforgeable}^{\mathsf{token\text{-}gen},\widetilde{\mathsf{sign}},\mathsf{verify}}$, where the superscripts denote the oracles available to the adversary.

One might also consider a strengthened notion of security, such as the one analogous to strong unforgeability (i.e., a *different* signature for a document which was signed by the signing oracle also counts as forgery), as well as quantum access to the verification oracle. Our construction satisfies neither of the two stronger security notions mentioned above, as shown in Appendix C. An extensive discussion on the different notions of security for TMAC is given in Section 2.2.2.

Our Contributions. We consider an IID noise model $\mathsf{Noise}(\mathsf{p})$ for some $0 \leq p \leq 1$, meaning the qubit at each coordinate gets corrupted[3] with a probability of p, and remains undeterred with a probability of $(1 - p)$, independent of other qubits. A TMAC is δ-noise-tolerant for some $0 \leq \delta \leq 1$, if correctness holds for the scheme up to negligible error in the noise model $\mathsf{Noise}(\delta)$ (Definition 3). Our main result is as follows.

Theorem 1. *Let[4] $\alpha \approx \cos^2(\pi/8)$. Assuming post-quantum one-way functions exist, for every $\delta < 1 - \alpha$, there exists an* $\mathsf{Unforgeable}^{\mathsf{token\text{-}gen},\mathsf{verify},\widetilde{\mathsf{sign}}}$ δ-noise-tolerant TMAC *(see Definitions 4 and 5)*

---

[2]Classical here means that the adversary can only send classical documents and signatures to the oracles and, hence, cannot query in superposition.

[3]We say a qubit gets corrupted to mean that an arbitrary dimension-preserving CPTP map acts on it. Since we deal with BB84 states in our construction, the map in our case which (negatively) affects verification the most is the Pauli operator Y.

[4]The value of α was computed numerically, the numerical error was relatively large and differed between engines (of magnitude 10^{-5} on the "sedumi" solver and 10^{-3} on the "SDPT3" solver), leaving some room for doubt if α is exactly $\cos^2(\frac{\pi}{8})$, or a slightly larger value. Regardless, the exact value of α does not affect the results of this work, besides the precise amount of noise tolerance.

scheme Π_δ (see Algorithm 5 and Proposition 1) based on conjugate coding states.

The formal proof is given in Section 3, on Page 19.

A previous construction for TMAC [BS17] achieves the same security under the same assumption.[5] However, the construction is not known to be noise-tolerant, and it also requires entangled states as the tokens.

In comparison, our construction is noise-tolerant and only requires simple tensor product states, which improves the practicality of our construction. Hence, the implementation of our TMAC scheme requires a non-perfect quantum channel capable of transmitting a BB84 state with less than a threshold percentage of error, and a quantum memory for the third party to store the BB84 token states for the duration it is granted to sign. Transmission of BB84 states or Wiesner money states for long distances with relatively low noise has already been demonstrated in quantum key distribution experiments [HRP+06, KLH+15]. Hence, the main practical challenge that remains is that of a quantum memory capable of storing BB84 states for reasonable periods of time. The main vectors used for transmission of quantum information are photons that are difficult to store with high fidelity, and the qubit lifespan is short even on quantum computers. However, for tokenized private signatures, a short-term storage may very well be enough, as the tokens are expected to be temporary by nature.

As an application for TMAC, we also show that any TMAC scheme can be used to construct a classically verifiable quantum money scheme, see Section 6.2. The verification can even be made non-interactive, at the cost of requiring temporary memory dependence.[6] To the best of our knowledge, none of the known private quantum money schemes based on similarly simple states has non-interactive verification. The fact that a secure TMAC implies secure private quantum money, combined with a recent result [Aar20] that shows that unconditionally secure private quantum money schemes do not exist, implies that no TMAC scheme can be unconditionally unforgeable if the adversary is allowed to request a polynomial number of tokens (see Theorem 10). Since the existence of post-quantum one-way functions is one of the weakest possible computational assumptions in cryptography, this shows that our construction is close to optimal in terms of the assumptions required.

Another application of our result is that it resolves an open problem of [BGZ21] regarding the construction of one-time memories in the stateless hardware model. This is a non-standard model in which it is possible to

---

[5][BS17] actually assumes the slightly stronger assumption of collision resistant hash functions. However using a universal one-way hash function (UOWHF) instead would achieve the same result, as shown in Appendix A.2.

[6]This can be viewed as a small database that needs to be maintained for a short time frame, at the end of which the database can be deleted.

implement any program with classical input, as a stateless black-box that can be queried classically. One-time memories are an ideal modeling of 1 out of 2 non-interactive oblivious transfer, which is impossible to realize in the standard model. In [BGZ21], the authors presented a conjugate coding-based construction of one-time memories in the stateless hardware model, and conjectured it to be a universally composable realization of one-time memories against unbounded malicious receivers in that model. In [BGZ21], the authors prove this conjecture partially; they require the added assumption that the receiver makes fewer than $c\lambda$ (classical) queries to the stateless hardware where $c < 0.114$ and λ is the security parameter. In Section 6.1, we show that our results imply that the result above can be extended to any polynomial number of classical queries.[7] Moreover, our work proves that this construction of one-time memories remains secure even in a noisy setting. We note that [CGLZ19] also presented a construction of one-time memories from stateless hardware, with a full security proof, but it was previously unknown whether this could be done with a conjugate coding-based scheme.

Construction. Our construction has two main steps. First, we construct a one-restricted TMAC, meaning only single-bit documents can be signed. The construction is unconditionally unforgeable against single-token attacks in the following sense. Given a single token and access to a verification oracle, which can be queried with only with classical strings and only a polynomial number of times, a computationally unbounded adversary that is given one signing token, cannot produce valid signatures for both 0 and 1 (except with negligible probability). We say such a scheme is Unconditionally Unforgeable$^{\text{1-token-gen,verify}}$, where 1-token-gen represents the single token available to the adversary, and verify the verification oracle.

Assuming post-quantum one-way functions exist, any such single bit TMAC that is Unconditionally Unforgeable$^{\text{1-token-gen,verify}}$ can be lifted in a noise-tolerance preserving manner using standard techniques similar to the ones used in [BS17] to a scheme that can sign documents of any length, and that is Unforgeable$^{\text{token-gen},\widetilde{\text{sign}},\text{verify}}$[8] (see "Notions of security", Page 4). The approach is reminiscent of the mini-scheme to full-scheme lift used for quantum money [AC13]. The above-mentioned lifting is the only place where we need (standard) computational hardness assumptions in our entire construction. This lifting is described in detail in Appendix A.2.

The main challenge in our work is to construct a single-bit TMAC that is Unconditionally Unforgeable$^{\text{1-token-gen,verify}}$. Our construction draws inspira-

---

[7]One cannot hope for an unbounded number of queries as the entire input-output behavior of the stateless hardware can be extracted. In addition, as covered in [BGZ21], quantum queries to the stateless hardware also result in insecure schemes.

[8]Unforgeability against Quantum Polynomial Time adversaries suffices for this lift as well.

tion from the classically verifiable variant of Wiesner's quantum money [MVW12] (see also [PYJ+12]). In this variant of Wiesner's quantum money, the bill is a tensor product of random conjugate coding states (also known as BB84 states), such as $|1\rangle \otimes |-\rangle \otimes |0\rangle \otimes |-\rangle \otimes \ldots \otimes |0\rangle$, the classical representation of which is kept secret by the bank. The bank sends a uniformly random challenge bit per qubit when a customer approaches the bank to verify the money. An honest customer should measure each qubit either in the computational basis or in the Hadamard basis, according to the challenge received, and should send the result as a proof. The bank accepts if the result is consistent with the corresponding qubit at the coordinates where the challenge string agrees with the secret string representing the conjugate state. The money scheme can be made noise-tolerant by relaxing the verification to accept even if the consistency check of the result with respect to the challenge fails at a small fraction of the coordinates. The same security guarantees hold for the noise-tolerant variant, as shown in [PYJ+12, MVW12].

In our construction for the single-bit noise-sensitive TMAC scheme (Algorithm 4), the quantum money state for the above-mentioned scheme serves as the token, and its classical representation serves as the secret key. The challenges are no longer random, but instead correspond to the (single-bit) document to be signed, i.e., an honest signer measures all the token's qubits in the computational basis in order to sign 0, or measures them in the Hadamard basis in order to sign 1. In both cases, the measurement outcome is the signature. Verification of an alleged signature for the document 0 (respectively, 1) is done by checking if the signature string is consistent with the secret key at the coordinates that had computational basis (respectively, Hadamard basis) states. The construction is made noise-tolerant in a way similar to the quantum money scheme, i.e., we relax the verification to accept even if the consistency check of the signature string for the document with the secret key fail at a small fraction of the coordinates.

Forgery against the noise-sensitive (respectively, noise-tolerant) variant of the TMAC scheme — i.e., producing valid signatures of both 0 and 1 using a single token — corresponds to passing two verifications in the noise-sensitive (respectively, noise-tolerant) variant of the money scheme using a single money state. However, there is a caveat that unlike the money scheme, the challenges are chosen by the adversary. We prove that despite the mentioned caveat, the probability to forge a signature for both 0 and 1 decreases exponentially in the number of qubits in both the noise-sensitive and noise-tolerant variants. Moreover, the security guarantee needs to hold even against adversaries that have classical access to a verification oracle (which is not the case for Wiesner's quantum money). For these reasons, we cannot reduce the TMAC unforgeability to the quantum money unforgeability, and provide a complete new proof instead. The complete scheme is shown in Algorithm 1.

Algorithm 1 $\mathsf{CTMAC}^{0.07}$ - A 14% noise-tolerant TMAC scheme.

Assumes: AENC is a post-quantum classical-queries strong authenticated encryption scheme, $\{h_r : \{0,1\}^* \to \{0,1\}^{\ell'(|r|)}\}_{r \in \{0,1\}^*}$ is a universal one-way hash function family with indexing function I, $\ell(\lambda) = \ell'(\lambda) + \lambda$.

1: **procedure** key-gen(1^λ)
2:      **Return** AENC.key-gen(1^λ).
3: **end procedure**

1: **procedure** token-gen$_k$
2:      **for** $j = 1, \ldots, \ell(\lambda)$ **do**
3:          $a^i, b^j \leftarrow_\$ \{0,1\}^\lambda$.
4:      **end for**
5:      Denote $\kappa = (a^1, b^1), \ldots, (a^{\ell(\lambda)}, b^{\ell(\lambda)})$.
6:      Compute $|\widehat{\triangleq}_j\rangle = H^{b^j}|a^j\rangle$.
7:      **Return** $(\otimes_{j=1}^{\ell(\lambda)}|\widetilde{\triangleq}_j\rangle, \mathsf{AENC}.\mathsf{Enc}_k(\kappa))$.
8: **end procedure**

1: **procedure** sign$_{|\triangleq\rangle}(m)$
2:      Interpret $|\triangleq\rangle$ as $(\otimes_{j=1}^{\ell(\lambda)}|\widetilde{\triangleq}_j\rangle, e)$.
3:      $r_1 \leftarrow I(\lambda)$, $r_2 \leftarrow_\$ \{0,1\}^\lambda$. Let $m' \equiv (h_{r_1}(m\|r_2)\|r_1)$
4:      **for** $j = 1, \ldots, \ell(\lambda)$ **do**
5:          Let m'_j denote the jth bit of m' and measure $(H^{m'_j})^{\otimes\lambda}|\widetilde{\triangleq}_j\rangle$ in the computational basis to obtain the classical string s^j.
6:      **end for**
7:      **Return:**$((s^1, s^2, \ldots, s^{\ell(\lambda)}), e, r_1, r_2)$.
8: **end procedure**

1: **procedure** verify$_k(m, \sigma)$
2:      Interpret σ as $((s^1, s^2, \ldots, s^{\ell(\lambda)}), e, r_1, r_2)$.
3:      Compute $\mathsf{AENC}.\mathsf{dec}_k(e)$ to obtain $\kappa \equiv (a^1, b^1), \ldots, (a^{\ell(\lambda)}, b^{\ell(\lambda)})$ and denote $m' \equiv (h_{r_1}(m\|r_2)\|r_1)$, let m'_j be the jth bit of m'.
4:      **for** $j = 1, \ldots, \ell(\lambda)$ **do**
5:          Let a_i^j, b_i^j, s_i^j denote the ith bit of a^j, b^j, s^j respectively.
6:          Construct the subset $\mathsf{Cons}_{m'_j} = \{i \in [\lambda] \mid b_i^j = m'_j\}$.
7:          Construct the subset $\mathsf{Miss}_{m'_j, s^j} = \{i \in \mathsf{Cons}_{m'_j} \mid s_i^j \neq a_i^j\}$.
8:          **if** $\left|\mathsf{Miss}_{m'_j, s^j}\right| > 0.07\lambda$ **then**
9:             **Return** 0.
10:        **end if**
11:      **end for**
12:      **Return** 1.
13: **end procedure**

Proof Techniques. We first give the proof sketch for the security of the noise-sensitive scheme, and then explain how to reduce the security of the noise-tolerant scheme to that of the noise-sensitive one.

First, we prove unforgeability against single-token attacks for the noise-sensitive one-restricted TMAC scheme (see the paragraph "Construction" on Page 6). The main challenge in the proof is to tackle the verification oracle, and to understand what information the adversary can procure from the verification oracle. In order to do so, we define a new game for our specific scheme where we strengthen the adversary by replacing the verification oracle with a stronger oracle. This oracle not only answers verification queries but also outputs a part of the secret key after a successful query for a document. The part of the secret key models all the information that the adversary might learn using the verification oracle. Let m be a document for which a successful query was made in the game. We argue that in such a game, if a successful query is made with respect to a document m, then it is redundant to make verification queries corresponding to m, thereafter. This is because the adversary can simulate such queries on her own, using the part of the secret key that was received after the first successful query for m.

In the next step, an adversary in the new game that we described in the last paragraph is reduced to an adversary against a Quantum Encryption with Certified Deletion scheme. In a Quantum Encryption with Certified Deletion (QECD) scheme ([BI20], see also Definition 8), a user can provide proof of deletion for a quantum cipher-text such that the proof can be verified using a separate algorithm. The security guarantee is that if the adversary submits a proof of deletion for a cipher-text that passes verification, then they could not learn the message later, even if the secret key is provided.

We map any adversary against TMAC in the intermediate game described in the last paragraph to an adversary against a QECD scheme in a weaker security game that is a relaxation of the certified deletion game in [BI20] (see Section 5.2). We should point out that this is a scheme-specific reduction, and we do not know if it can be made generic.

Lastly, standard semi-definite programming techniques are used to provide a bound on the success probability of an adversary in the weak certified deletion game against the QECD scheme mentioned above–see Appendix B.

The proof of unforgeability for the noise-tolerant variant (see the paragraph "Construction" on Page 6) follows along similar lines. We consider an intermediate scheme-specific game for the noise-tolerant scheme with a strengthened adversary, similar to what we did for the noise-sensitive scheme. In the next step, however, instead of mapping the strengthened adversary to an adversary in a certified deletion game, the adversary is

mapped to a strengthened adversary in the corresponding game[9] for the noise-sensitive scheme with a smaller security parameter. In our analysis for the noise-sensitive scheme, we have shown that the winning probability of the strengthened adversary against the noise-sensitive scheme in the later game is exponentially small. The reduction then gives an exponentially small bound on the winning probability of the strengthened adversary against the noise-tolerant scheme.

Related Works. In [BS17], the authors also introduced the public variant of tokenized MACs called Public Tokenized Digital Signatures that allow anyone to verify the validity of a signature using a public key. The authors provide a construction for the private variant and a candidate construction for the public variant based on hidden sub-spaces that were originally used by Aaronson and Christiano to construct public quantum money [AC13]. A variant of the construction for public tokenized digital signature construction in [BS17] was proven to be secure based on post-quantum Indistinguishability Obfuscation (IO) in [CLLZ21]. The main advantage of the private scheme in [BS17], as well as the public variant in [CLLZ21], over our construction is that it is secure even against adversaries performing superposition queries to the verification oracle. Another advantage is that the TMAC scheme in [BS17] is a strongly unforgeable TMAC, i.e., given a token, the adversary cannot even produce two different signatures for the same document. The analysis of the construction in [BS17] is based on query-complexity theoretic lower bounds, whereas the analysis of our construction involves techniques based on semi-definite programming. However, [BS17] does not achieve noise tolerance. TMAC is an enhanced version of the much studied cryptographic primitive called Message Authentication Codes (MAC) in terms of functionality.

There are two kinds of MAC schemes that are of interest to us: post-quantum MACs for classical documents, and MACs for quantum documents. There have been quite a few works [BZ13, KLLN16, SY17] on post-quantum MACs, starting with the work of Boneh and Zhandry [BZ13]. Among other results, the authors in [BZ13] showed that quantum secure Pseudorandom functions imply an existentially unforgeable MAC that is secure against quantum-chosen message attacks, i.e., even against quantum adversaries who can query the MAC on superposition of documents. The notion of authentication of quantum data was first studied by Barnum et al. in [BCG⁺02]. Later, there were subsequent works including [BW16, AM17, GYZ17] that strengthened the security definitions and the constructions achieving them, although these results are less relevant for this work since we concentrate on the authentication of *classical* documents.

---

[9]This is the same intermediate game that we discussed above in the analysis of the noise-sensitive scheme.

Tokenized digital signatures are not the only cryptographic primitives that achieve the task of revocable signature delegation. One such primitive that achieves such delegation is called the "one-shot signature", introduced recently by Amos et al. [AGKZ20]. In one-shot signatures, everyone can generate a quantum signing token and a classical verification key that can verify alleged signatures produced by the token. The security guarantees that one cannot produce two signatures that pass verification with the same verification key. One-shot signatures allow an owner to delegate signing authority to a signer only using classical communication.[10] This is the main advantage of one-shot signatures over TMAC since a TMAC requires the owner to send the quantum tokens to the signer over a quantum channel. However, the obvious limitation is that one-shot signatures lack efficient construction. The construction given in [AGKZ20] is based on an oracle, for which we have no efficient instantiation as of yet. In contrast, tokenized digital signatures exist under standard assumptions. Another related notion is the bolt-to-certificate transformation introduced in [CS20]. In [CS20], the authors used Zhandry's quantum lightning [Zha21] to construct such a transformation. Note that the lightning bolts are quantum states that are similar to tokens for digital signatures in two ways. First, both of these are unclonable, i.e., it is not possible to make two bolts or two tokens from a single bolt or token. Second, they are revocable; bolts can be converted to certificates (as shown in [CS20]), and tokens can be spent by signing a bit. However, the main difference between the two is that unlike lightning bolts, there are at least two revocation procedures for tokens: one can either sign the document with 0 or 1 in order to spend it. Note that in order to use the bolts for signing bits, it is crucial to have two different revocation procedures: one for signing the bit 0 and the other for 1. However, lightning-to-certificate provides only a single revocation method. Nevertheless, the transformation of bolt-to-certificate has been proven useful in other areas such as in the bitcoin scalability problem [CS20] and for semi-quantum money [RS19]. There is no construction of quantum lightning based on standard assumption, and the security of the existing scheme [Zha21] was put into question by Roberts [Rob21].

TMACs also imply other important cryptographic primitives that are well studied in the literature. One such example is unforgeable private-key quantum money [Wie83, AC13], which we discuss in more detail in Section 6.2. Similarly, it is also possible to construct unforgeable public quantum money [AC13, Zha21, FGH+12] from public tokenized digital signatures (as shown in [BS17]), which is a stronger and much harder primitive

---

[10]The owner can sign a verification key vk of the signer, using a MAC or a digital signature, and send the signature of vk to the signer. The signer then can sign one document using its token and append the signature of pk that he received to it. The signature of vk made by the owner validates all signatures that pass verification with respect to vk. The security offered by the one-shot signatures restricts the signer to only produce one signature that passes verification with respect to vk.

to construct.

Another application of TMAC is that of disposable cryptographic back-doors [CGLZ19]. For a cryptographic primitive such as an encryption scheme, the task is to give one-time backdoor access to secret information, such as the message hidden under the cipher, in case of an encryption. This task is achieved by defining a disposable backdoor variant of encryption, where the key-generation algorithm additionally outputs a quantum token or backdoor that can be used only once to learn the message from the cipher. In theory, the above variant can be constructed using exotic primitives called one-time programs [GKR08], which are programs that can be executed once and then they become useless. However, these primitives cannot exist in the standard model, even in a quantum setting, see [GKR08, BGS13]. In [CGLZ19], the authors show how to construct one-time programs from TMACs, relative to classical stateless hardware. It should be noted that all queries to the oracle in the stateless hardware model are classical, and hence the reduction in [CGLZ19] only requires the TMAC scheme to be unforgeable with respect to classical queries. Unforgeability against superposition queries for the underlying TMAC does not provide any additional utility. This is yet another scenario where the unforgeability definition considered in our work (i.e., unforgeability with respect to classical queries), despite not being the strongest security notion, is the correct notion to consider under the circumstances.

The same result regarding one-time programs was considered in [BGZ21], in which the authors directly use a conjugate coding state-based construction to achieve one-time programs. However, only a partial result regarding the security of the construction was provided. In Section 6.1, the results of this work are used to prove a conjecture in [BGZ21], and to strengthen their result.

Organization. We start with some preliminaries in Section 2, where we also discuss the definition, correctness, and security notions of TMAC. An overview of the main result is given in Section 3. The construction for a 1-bit TMAC scheme that is unforgeable against single token attacks with verification oracle TMAC, $CTMAC^\eta$ is given in Section 4, and its unforgeability is proved in Section 5. Section 6 covers applications of our result: a solution to an unsolved problem in [BGZ21] regarding a construction of one-time memories from stateless hardware, and a construction of private quantum money from TMAC. In Section 7, we discuss some open questions and future directions. The expansion of $CTMAC^\eta$ to a full-blown scheme is covered in Appendix A. In Appendix B, we provide a detailed security analysis of the two QECD schemes used in the proof, which is left out of the main text. Appendix C discusses drawbacks of $CTMAC^\eta$, Appendix D shows that any TMAC satisfying a standard notion of unforgeability can also be used as a classical MAC and Appendix E discusses a lift to a (length-restricted)

TMAC scheme which is unconditionally unforgeable for a fixed number of tokens given to the adversary. A nomenclature appears on the last page (Page 81).

2 Notations and Definitions

2.1 Notations

We assume that the reader is familiar with classical cryptography (see [KL14, Gol04]), as well as quantum computing (see [NC11]).

The operation of the Hadamard gate on a single qubit is defined by the following matrix:

For a string $b \in \{0,1\}^n$, we define H^b as the n-qubit operator $H^{b_1} \otimes \cdots \otimes H^{b_n}$, where

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

We use the notation $[n]$ to denote the set $\{1, \ldots, n\}$. For any string $x = (x_1, \ldots, x_n)$ and a subset $\mathcal{J} \subseteq [n]$, let $x|_{\mathcal{J}}$ denote the string x restricted to the bits with indices in $\mathcal{J}$. We occasionally also use the notation $s(i)$ to denote the ith bit of the string s, and $x\|y$ to denote the concatenation of x and y. For every $n \in \mathbb{N}$ and $j \leq n$, let $\binom{[n]}{j}$ denote the set of all j-sized subsets of $[n]$.

We write $y \leftarrow \mathsf{alg}(x)$ to denote the probabilistic process in which y is sampled according to the distribution $\mathsf{alg}(x)$. The sampler alg takes the string x as input and outputs the string y. To define a variable x as the value y, we use the notation $x \equiv y$. For a finite set of S, the notation $x \leftarrow_\$ S$ denotes that x is sampled uniformly at random from S.

When discussing a scheme, $\mathcal{S}$, which is dependent on a security parameter λ, we assume that λ is known to all algorithms in this scheme. To avoid ambiguity, when discussing multiple schemes $\mathcal{S}_1, \mathcal{S}_2, \ldots$ for some cryptographic primitive, we use the notation $\mathcal{S}_1.\mathsf{alg}, \mathcal{S}_2.\mathsf{alg}$ to differentiate between the algorithm alg of the respective schemes.

As per usual convention, security is defined by a game consisting of two sides: a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Unless explicitly stated otherwise, it is assumed that $\mathcal{C}$ always refers to the challenger, and likewise, $\mathcal{A}$ always refers to the adversary.

We discuss both computationally bounded and computationally unbounded adversaries. A computationally unbounded adversary is a family of quantum circuits $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$. A Quantum Polynomial Time (QPT) adversary is a family of uniform quantum circuits $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ such that for every λ, circuit $\mathcal{C}_\lambda$ has $p(\lambda)$ input nodes and size $S(\lambda)$ for both $q(\lambda), S(\lambda) \in \mathsf{poly}(\lambda)$. Likewise,

a Probabilistic Polynomial Time (PPT) adversary denotes a probabilistic Turing Machine running in polynomial time.

A function is negligible in λ if it decreases to 0 faster than an inverse polynomial, with $\lambda \to \infty$. We also use the shorthand $f(\lambda) \leq \mathsf{negl}(\lambda)$ to state that f is negligible. For a function $f : \{0,1\}^n \to \{0,1\}^m$, we define the unitary operator U_f as the linear operator such that $\mathsf{U}_f(|x\rangle|y\rangle) = |x\rangle|f(x)\oplus y\rangle$ where $x \in \{0,1\}^n, y \in \{0,1\}^m$. The notation $\mathcal{BB}84$ is used to denote the set of quantum states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$.

2.2 Tokens for MAC

2.2.1 Definition and Correctness

We define tokenized MAC, also known as a tokenized private digital signature, as in [BS17]:

Definition 1 (Tokenized MAC). *A Tokenized Message Authentication Code, or* TMAC *scheme consists of four* QPT *algorithms:* $\mathsf{key\text{-}gen}$, $\mathsf{token\text{-}gen}$, sign, *and* verify, *with the following syntax.*

1. *Upon input* 1^λ, *where* λ *is the security parameter,* $\mathsf{key\text{-}gen}$ *outputs a classical key* k *known as the secret key.*

2. *The algorithm* $\mathsf{token\text{-}gen}$ *receives* k *as input and generates some quantum state* $|�she\rangle$, *a signing token. We stress that, in the general case, if* $\mathsf{token\text{-}gen}$ *is called* r *times, it may output different states* $|☎_1\rangle, \ldots, |☎_r\rangle$.

3. *The algorithm* sign *receives* $|☎\rangle$ *and a classical document* m *and outputs the signature* σ, *which is a classical string.*

4. *The algorithm* verify *receives* k, *a classical document* m *and a classical signature* σ, *and outputs a Boolean answer.*

We say a TMAC *scheme is correct if for every document* $m \in \{0,1\}^*$, *every* k *in the range of* $\mathsf{key\text{-}gen}$, *and every token* $|☎\rangle$ *generated by* $\mathsf{token\text{-}gen}_k$:

$$\Pr[\mathsf{verify}_k(\mathsf{sign}_{|☎\rangle}(m)) = 1] = 1.$$

We also define a length-restricted version of the scheme:

Definition 2 (Length-Restricted TMAC). *A* TMAC *is* ℓ *restricted if the document space is restricted to* $\{0,1\}^\ell$ *for some integer function* $\ell(\lambda)$.

Next, we define the noise model we will be working with as the following noise model $\mathsf{Noise}(\mathsf{p})$ for some $0 \leq p \leq 1$.

Definition 3 (Noise model $\mathsf{Noise}(\mathsf{p})$)**.** $\mathsf{Noise}(\mathsf{p})$ *is the noise model in which the noise acts on each qubit as an IID Boolean random variable with a probability of p, i.e., at each coordinate, with a probability of p, the qubit gets corrupted, meaning an arbitrary CPTP map that preserves dimensions acts on the qubit, and with a probability of* $1-p$, *the qubit remains undeterred. Since we deal with BB84 states in our construction, the CPTP map in our setting that negatively affects verification the most is the Pauli operator* Y.

Definition 4 (Noise-tolerant scheme and noise-tolerance preserving lift)**.** *A* TMAC *is* δ-$\mathsf{noise\text{-}tolerant}$ *for some* $0 \leq \delta \leq 1$ *if correctness holds for the scheme up to a negligible error in the noise model* $\mathsf{Noise}(\delta)$ *(Definition 3). A transformation that maps every* δ-$\mathsf{noise\text{-}tolerant}$ TMAC *scheme to a* δ-$\mathsf{noise\text{-}tolerant}$ TMAC *for every* $0 \leq \delta \leq 1$ *is called a noise-tolerance preserving transformation or a noise-tolerance preserving lift.*

2.2.2 Notions of Security

In a vanilla MAC scheme, we say forgery occurs when the adversary submits a valid signature (i.e., they pass verification) of a document that was not previously signed by a signing oracle. Clearly for TMAC, this notion of forgery does not make sense since an adversary given r tokens should be able to sign r different documents. However, we expect that an adversary would not be able to sign more than r documents. Moreover, we would like to give the adversary access to some kind of signing oracle in order to model the fact that the adversary might have access to other signed documents which were legitimately signed by others. It is natural to define forgery in the following manner that is similar in spirit to the forgery of quantum money: we say forgery occurs if an adversary who is given $r \in \mathsf{poly}(\lambda)$ tokens submits valid signatures for $r + 1$ distinct documents, none of which were previously signed by the signing oracle. However, there is a syntactic issue that needs to be taken care of. The signing procedure of a TMAC scheme receives a token as input, rather than a secret key. Therefore, a small change is needed to define the signing oracle. The analogue in TMAC for the signing oracle is the procedure $\widetilde{\mathsf{sign}}_k(m)$, which first generates a token $|\underline{\mathfrak{Q}}\rangle$ by $\mathsf{token\text{-}gen}_k$, and then signs the document m with $\mathsf{sign}_{|\underline{\mathfrak{Q}}\rangle}(m)$.

Similarly to quantum money, an adversary that received r tokens may attempt to submit any $w > r$ ($w \in \mathsf{poly}(\lambda)$) number of documents with the hope that at least $r + 1$ fresh[11] distinct documents pass verification. Hence, we allow the adversary to submit signatures for polynomially many documents in the unforgeability game, and they win if at least $r + 1$ of them pass verification.

---

[11]Recall that a signature is considered fresh if the document was not signed by the $\widetilde{\mathsf{sign}}_k$ oracle.

Furthermore, an adversary may wish to submit a signature, check if it gets accepted, and improve future guesses based on the result. This is modeled in the unforgeability game by giving the adversary access to a verification oracle. Readers familiar with MAC might find it peculiar that we provide the adversary with a verification oracle. Many MAC schemes have deterministic signing (and in fact, this is w.l.o.g.–see [Gol04, Section 6.1.5.2]). In such schemes, there is no need for a verification algorithm altogether: Verification of a document m and an alleged signature σ is done by testing whether $\sigma \stackrel{?}{=} \mathsf{sign}_k(m)$. This type of verification is called *canonical verification*. It negates the need for a verification oracle, as explained in Ref. [Gol04, Section 6.1.5.1]. Therefore, the verification oracle is often ignored. Since the TMAC presented in this work does not have deterministic signing, the verification oracle is necessary to model the capabilities of the adversary.

There is some subtlety in determining to which of the oracles the adversary is allowed quantum access to, i.e., allowed to make superposition queries. While there is strong motivation to allow quantum access to the verification oracle,[12] there is less so for quantum access to a signing oracle. In addition, even defining security with regard to quantum access to the signing oracle is a contrived task (see also related works on Page 10). On the other hand, security with regard to quantum access to verification is concisely defined for TMACs with deterministic verification, but likewise, this is complicated to define in the most general scenario. While our construction does have deterministic verification, it does not achieve this stronger security notion. For the sake of generality, we present here a definition allowing for non-deterministic or even quantum verification, and postpone the definition of unforgeability against adversaries with quantum access to verification to Appendix C.1, along with the appropriate attack on our construction.

To ease notation, we provide the tokens to the adversary by the oracle $\mathsf{token\text{-}gen}_k$ that the adversary can query polynomially many times. However, it can be assumed, without loss of generality, that an adversary always makes exactly $r(\lambda) \in \mathsf{poly}(\lambda)$ calls to $\mathsf{token\text{-}gen}_k$ immediately after receiving the security parameters. In an intermediate step of our analysis, we will also discuss unforgeability against adversaries holding a single token, i.e., the adversary can access the $\mathsf{token\text{-}gen}$ oracle, but only once.

With the above motivations in mind, we define the following unforgeability game in Game 2.

Definition 5. *A* TMAC *scheme* $\Pi = (\mathsf{key\text{-}gen}, \mathsf{token\text{-}gen}, \mathsf{sign}, \mathsf{verify})$ *is said*

---

[12]The verification procedure could be potentially given as an obfuscated circuit [BGI+12] in a potential lift to a public tokenized digital signatures scheme. A quantum adversary could then run the obfuscated circuit in superposition.

Game 2 $\mathsf{Forge}_{\mathcal{A},\Pi}^{\mathsf{token\text{-}gen},\widetilde{\mathsf{sign}},\mathsf{verify}}(\lambda)$

1: $\mathcal{C}$ creates a secret key by running $k \leftarrow \mathsf{key\text{-}gen}(1^\lambda)$ and sends 1^λ to $\mathcal{A}$.
2: $\mathcal{A}$ is given classical oracle access to $\widetilde{\mathsf{sign}}_k$, verify_k and $\mathsf{token\text{-}gen}_k$ and, and can query each oracle a polynomial number of times. Let r denote the number of times $\mathsf{token\text{-}gen}$ is called, and let Q denote the set of all queries that are made to $\widetilde{\mathsf{sign}}_k$.
3: $\mathcal{A}$ sends the challenger w documents and their signature $(m_i, \sigma_i)_w$.
4: $\mathcal{C}$ runs $c_i \leftarrow \mathsf{verify}_k(m_i, \sigma_i)$.

Let S be the set of all $i \in [w]$ such that $c_i = 1$, and let *count* $= |\{m_i \mid i \in S \wedge m_i \notin Q\}|$. Then the output of the game is 1 if and only if *count* $\geq r + 1$.

---

to be Existentially Unforgeable$^{\mathsf{token\text{-}gen},\mathsf{verify},\widetilde{\mathsf{sign}}}$ *if for any* QPT *adversary* $\mathcal{A}$

$$\Pr[\mathsf{Forge}_{\mathcal{A},\Pi}^{\mathsf{token\text{-}gen},\mathsf{verify},\widetilde{\mathsf{sign}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Likewise, various other types of unforgeability can be defined by preventing access to some of the oracles, which is notated by removing the corresponding oracles from the superscript in the definition and security game. The notation $(\cdot)^{\mathsf{1\text{-}token\text{-}gen}}$ *instead of* $(\cdot)^{\mathsf{token\text{-}gen}}$ *denotes that only a single access to the oracle* $\mathsf{token\text{-}gen}_k$ *is allowed. The functionality of the oracles is summarized in Table 1.*

Oracle	Functionality
$(\cdot)^{\mathsf{verify}}$	$\mathcal{A}$ has classical access to verify_k
$(\cdot)^{\widetilde{\mathsf{sign}}}$	$\mathcal{A}$ has classical access to $\widetilde{\mathsf{sign}}_k$
$(\cdot)^{\mathsf{token\text{-}gen}}$	$\mathcal{A}$ has access to $\mathsf{token\text{-}gen}_k$
$(\cdot)^{\mathsf{1\text{-}token\text{-}gen}}$	$\mathcal{A}$ can make at most one query to $\mathsf{token\text{-}gen}_k$

Table 1: Possible oracles an adversary may have access to in Game 2

The definition above can easily be adjusted to allow weaker forms of unforgeability such as random unforgeability, selective unforgeability, and universal unforgeability. In this work, we only address existential unforgeability; hence, the prefix Existentially is omitted for the remainder of the paper.

An Unconditionally Unforgeable$^{(\cdot)}$ TMAC is defined similarly, but where the above holds even against computationally unbounded adversaries. It is emphasized that a computationally unbounded adversary could still only query the oracles a polynomial number of times.

2.2.3 Sabotage Attacks

Another notion often discussed in relation to MAC is that of sabotage. An adversary might not be able to forge a fresh signature, but still harm other users in other ways. Sabotage is the act of creating a malicious signature, that successfully passes one verification, but might not pass a second verification.

Game 3 $\mathsf{Sabotage}_{\mathcal{A},\Pi}^{\mathsf{token\text{-}gen},\widetilde{\mathsf{sign}},\mathsf{verify}}(\lambda)$

1: $\mathcal{C}$ creates a secret key by running $k \leftarrow \mathsf{key\text{-}gen}(1^\lambda)$ and sends 1^λ to $\mathcal{A}$.
2: $\mathcal{A}$ is given classical oracle access to $\widetilde{\mathsf{sign}}_k$, verify_k and $\mathsf{token\text{-}gen}_k$ and, and can query each oracle a polynomial number of times.
3: $\mathcal{A}$ sends the challenger a signed document (m,σ)
4: $\mathcal{C}$ runs $\mathsf{verify}_k(m,\sigma)$ *twice.*

The output of the game is 1 if and only if the first verification was successful, but the second was not.

Definition 6. *A* TMAC *scheme* $\Pi = (\mathsf{key\text{-}gen}, \mathsf{token\text{-}gen}, \mathsf{sign}, \mathsf{verify})$ *is said to be* $\mathsf{Unsabotagable}^{\mathsf{token\text{-}gen},\mathsf{verify},\widetilde{\mathsf{sign}}}$ *if for any* QPT *adversary* $\mathcal{A}$

$$\Pr[\mathsf{Sabotage}_{\mathcal{A},\Pi}^{\mathsf{token\text{-}gen},\mathsf{verify},\widetilde{\mathsf{sign}}}(\lambda) = 1] \le \mathsf{negl}(\lambda).$$

Likewise, various other types of sabotage security can be defined by preventing access to some of the oracles, which is represented in the same way as in Algorithm 2.

If a party can use a token to create such a signature, the party does not gain anything from it, but can harm others: Say Bob makes a transaction with Charlie's company, Charlie's assistant Alice uses one of Charlie's tokens to maliciously sign the transaction on his absence, and to verify the signature Bob turns to the trusted Dave, who knows Charlie's key. Upon Charlie's return, Bob shows him the signature, but to his surprise verification fails, and the annoyed Charlie calls for security to deport Bob, while Alice enjoys her stolen loot. Of course, if Bob is allowed to verify his signature an unlimited amount of times, verification will eventually pass, as the signature is a classical string. However even then, Bob would not necessarily know the number of times he would need to verify the signature in order to succeed with high probability. One could imagine that the verifier would lose patience and become suspicious after 100 failed verifications.

It is clear that sabotage is not an issue in any TMAC that has deterministic verification, since in that case, signatures would either be accepted or rejected with probability 1. As the construction presented in this paper

has deterministic verification,[13] we will not discuss security against sabotage any further.

3 Main Result

We say that a TMAC scheme is based on conjugate coding states if every token is a tensor product of states from $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$, perhaps in addition to some classical string. Our main result is the following:

Theorem 1. *Let[14] $\alpha \approx \cos^2(\pi/8)$. Assuming post-quantum one-way functions exist, for every $\delta < 1 - \alpha$, there exists an* Unforgeable$^{\text{token-gen,verify,}\widetilde{\text{sign}}}$ *δ-noise-tolerant TMAC (see Definitions 4 and 5) scheme Π_δ (see Algorithm 5 and Proposition 1) based on conjugate coding states.*

Proof. Given $\eta \geq 0$, we first construct a 1-restricted (that is for $m \in \{0, 1\}$) $2c\eta$-noise-tolerant TMAC (see Definition 4) scheme CTMAC$^\eta$ (see Algorithm 5 and Theorem 3) based on conjugate states for any constant $0 \leq c < 1$. In Section 5.4, we show that the scheme CTMAC$^\eta$ is Unforgeable$^{\text{1-token-gen,verify}}$ (see Theorem 2), for $0 \leq \eta \leq \frac{1-\alpha}{2}$.

Hence, given $\delta < 1 - \alpha$, there exists $\widetilde{\eta}$ such that $\delta < \widetilde{\eta} < 1 - \alpha$. Let $\widetilde{c} = \frac{\delta}{\widetilde{\eta}}$, which is clearly smaller than 1. Since $\frac{\widetilde{\eta}}{2} < \frac{1-\alpha}{2}$, by Theorem 2, CTMAC$^{\frac{\widetilde{\eta}}{2}}$ is Unforgeable$^{\text{1-token-gen,verify}}$ and by Theorem 3 is $2\widetilde{c}\frac{\widetilde{\eta}}{2}$-noise-tolerant, i.e., δ-noise-tolerant. Next, we lift CTMAC$^{\frac{\widetilde{\eta}}{2}}$ to a full-blown Unforgeable$^{\text{token-gen,verify,}\widetilde{\text{sign}}}$ TMAC scheme $\widetilde{\text{CTMAC}}^{\frac{\widetilde{\eta}}{2}}_{\text{Poly-times}}$, while preserving the parameter of noise tolerance using the following proposition, which is proven in Appendix A.2 on Page 55.

Proposition 1. *Assuming the existence of post-quantum one-way functions, there is a noise-tolerance preserving lift (see Definition 4) of any 1-restricted* Unforgeable$^{\text{1-token-gen,verify}}$ *TMAC (see Definitions 2 and 5) to an unrestricted* Unforgeable$^{\text{token-gen,verify,}\widetilde{\text{sign}}}$ *TMAC. Moreover, if the 1-restricted scheme is based on conjugate states then the same holds for the resulting unrestricted scheme.*

Hence, $\widetilde{\text{CTMAC}}^{\frac{\widetilde{\eta}}{2}}_{\text{Poly-times}}$ is the required Π_δ. $\qquad\square$

---

[13]Assuming that the lift in Lemma 5 is instantiated with a deterministic verification authenticated encryption, as most practical authenticated encryptions are.

[14]The value of α was computed numerically, the numerical error was relatively large and differed between engines (of magnitude 10^{-5} on the "sedumi" solver and 10^{-3} on the "SDPT3" solver), leaving some room for doubt if α is exactly $\cos^2(\frac{\pi}{8})$, or a slightly larger value. Regardless, the exact value of α does not affect the results of this work, besides the precise amount of noise tolerance.

In particular, since $0.14 < 1 - \alpha$, we get the following corollary of Theorem 1.

Corollary 1. *Assuming post-quantum one-way functions exist, there exists an* Unforgeable$^{\text{token-gen,verify},\widetilde{\text{sign}}}$ 14%-noise-tolerant *(see Definitions 4 and 5)* TMAC *scheme (see Algorithm 5 and Proposition 1) based on conjugate coding states.*

4 Construction of a 1-Restricted TMAC Based on Conjugate Coding States

4.1 A Noise-Sensitive Scheme

In Algorithm 4, a 1-restricted TMAC scheme (that is for $m \in \{0, 1\}$) based on the classical verification variant of Wiesner's money is described. We refer to this scheme as the Conjugate TMAC, and write CTMAC.

Algorithm 4 CTMAC - The 1-bit Conjugate TMAC

1: **procedure** key-gen(1^λ)
2:     $a \leftarrow\!\!\$ \{0,1\}^\lambda$, $b \leftarrow\!\!\$ \{0,1\}^\lambda$.
3:     **Return** $k \equiv (a, b)$.
4: **end procedure**

1: **procedure** token-gen$_k$
2:     Interpret $k = (a, b)$.
3:     **Return** $|\underline{\&}\rangle \equiv H^b |a\rangle$.
4: **end procedure**

1: **procedure** sign$_{|\underline{\&}\rangle}(m)$ $\triangleright$ $m \in \{0, 1\}$
2:     Measure $(H^m)^{\otimes \lambda} |\underline{\&}\rangle$ in the standard basis to obtain a string σ.
3:     **Return** σ.
4: **end procedure**

1: **procedure** verify$_k(m, \sigma)$
2:     Interpret k as (a, b).
3:     Construct the subset $\mathsf{Cons}_m = \{i \in [\lambda] \mid b_i = m\}$.
4:     **if** $a|_{\mathsf{Cons}_m} = \sigma|_{\mathsf{Cons}_m}$ **then**
5:         **Return** 1.
6:     **else**
7:         **Return** 0.
8:     **end if**
9: **end procedure**

We first prove the correctness of CTMAC.

Proposition 2 (Correctness of CTMAC). *The 1-restricted Conjugate* TMAC, CTMAC, *given in Algorithm 4, is correct.*

Proof. Let $k \equiv (a, b) \leftarrow$ key-gen(1^λ) be the secret key, where λ is the security parameter, fixed arbitrarily. Hence, the token $|\text{⚷}\rangle$ generated by token-gen$_k$ is $H^b|a\rangle$. Fix $m \in \{0, 1\}$. Let, $\sigma \leftarrow$ sign$_{|\text{⚷}\rangle}(m)$. By definition, σ is the (standard basis) measurement outcome of

$$(H^m)^{\otimes \lambda}|\text{⚷}\rangle = (H^m)^{\otimes \lambda}(H^b|a\rangle).$$

Let $\mathsf{Cons}_m \equiv \{i \in [\lambda] \mid b_i = m\}$. Due to the action of the Hadamard gate twice, for every $i \in \mathsf{Cons}_m$, the ith qubit of the state $(H^m)^{\otimes \lambda}(H^b|a\rangle)$ remains $|a_i\rangle$. Hence, with probability 1, $\sigma|_{\mathsf{Cons}_m} = a|_{\mathsf{Cons}_m}$. Since, verify$_k(m, \sigma)$ simply checks if $\sigma|_{\mathsf{Cons}_m} = a|_{\mathsf{Cons}_m}$, the verification accepts it with certainty. Therefore, for every k in the range of key-gen, and every token $|\text{⚷}\rangle$ generated by token-gen$_k$

$$\Pr[\text{verify}_k(m, \text{sign}_{|\text{⚷}\rangle}(m)) = 1] = 1,$$

for all $m \in \{0, 1\}$. Since λ and m were fixed arbitrarily, we conclude that CTMAC is correct. □

Theorem 2. *The conjugate* TMAC *scheme* CTMAC *given in Algorithm 4, is* Unconditionally Unforgeable$^{\text{1-token-gen,verify}}$.

Theorem 2 is proven in Section 5.2. In Appendix C, it is shown that CTMAC does not satisfy some stronger notions of unforgeability, namely, strong unforgeability and unforgeability against adversaries with quantum verification queries.

4.2 A Noise-Tolerant Scheme

In this section, we follow ideas similar to the constructions in [PYJ+12, MVW12] to extend our 1-restricted TMAC scheme CTMAC (Algorithm 4) scheme to a noise-tolerant variant which we call CTMAC$^\eta$ (see Algorithm 5), where η represents an error threshold. The construction of CTMAC$^\eta$ is the same as CTMAC, but with a lenient verification procedure where we accept an alleged signature for a bit b even if the signature is not consistent with the secret key, up to a constant fraction of the coordinates. The full construction is given in Algorithm 5.

Theorem 3. *The* 1-*restricted (see Definition 2)* TMAC *scheme* CTMAC$^\eta$ *is correct. Moreover, for any constant* $0 \le c < 1$, CTMAC$^\eta$ *is* $2c\eta$-noise-tolerant *(see Definition 4).*

Proof. The correctness follows from the correctness of the noise-sensitive scheme CTMAC (see Algorithm 4) because the two schemes have the same key-gen, token-gen, and sign algorithms, and the verification[15] is more lenient

---

[15]Note that the verifications of both CTMAC (Algorithm 4) and CTMAC$^\eta$ (Algorithm 5) are deterministic.

Algorithm 5 CTMAC$^\eta$ - $2c\eta$-noise-tolerant Conjugate TMAC, for all constant $0 \leq c < 1$

1: **procedure** key-gen(1^λ)
2:     The same as CTMAC.key-gen.
3: **end procedure**

1: **procedure** token-gen$_k$
2:     The same as CTMAC.token-gen$_k$.
3: **end procedure**

1: **procedure** sign$_{|\text{\textcircled{s}}\rangle}(m)$
2:     The same as CTMAC.sign$_{|\text{\textcircled{s}}\rangle}$.
3: **end procedure**

1: **procedure** verify$_k(m, \sigma)$
2:     Interpret k as (a, b).
3:     Construct the subset $\mathsf{Cons}_m = \{i \in [\lambda] \mid b_i = m\}$.
4:     Construct the subset $\mathsf{Miss}_{m,\sigma} = \{i \in \mathsf{Cons}_m \mid \sigma_i \neq a_i\}$.
5:     **if** $|\mathsf{Miss}_{m,\sigma}| \leq \eta\lambda$ **then**
6:         **Return** 1.
7:     **else**
8:         **Return** 0.
9:     **end if**
10: **end procedure**

---

in CTMAC$^\eta$, i.e., for the same secret key k,

$$\mathsf{CTMAC}^\eta.\mathsf{verify}_k(m, \sigma) = 1 \implies \mathsf{CTMAC}.\mathsf{verify}_k(m, \sigma) = 1,$$

for any alleged document-signature pair (m, σ).

Next, we show noise tolerance for CTMAC$^\eta$. Let $0 \leq c < 1$ be an arbitrary constant. We will consider the noise model $\mathsf{Noise}(2c\eta)$ (Definition 3) from here on. Fix $m \in \{0, 1\}$. Let $k \leftarrow \mathsf{CTMAC}^\eta.\mathsf{key\text{-}gen}(1^\lambda)$ and $|\text{\textcircled{s}}\rangle = \otimes_{j=1}^\lambda |\text{\textcircled{s}}_j\rangle \leftarrow$ CTMAC$^\eta$.token-gen$_k$. Suppose some qubits of $|\text{\textcircled{s}}\rangle$ got corrupted due to noise, and let the resulting state be $|\text{\textcircled{s}}'\rangle$.

For every $j \in [\lambda]$, let W_j, Y_j be Boolean random variables such that $Y_j = 1$ if and only if the jth qubit of $|\text{\textcircled{s}}\rangle$ was corrupted by noise, and $W_j = 1$ if and only if the quantum state of the jth qubit of $|\text{\textcircled{s}}\rangle$ is in the relevant basis with respect to m, i.e.,

$$|\text{\textcircled{s}}_j\rangle \in \{H^m|0\rangle, H^m|1\rangle\}.$$

Let the Boolean random variable X_j be defined as the product of W_j and Y_j.

$$X_j = W_j \cdot Y_j.$$

By definition of the CTMAC$^\eta$.key-gen algorithm, $\{W_j\}_{j \in [\lambda]}$ are IID random variables with parameter $\frac{1}{2}$ meaning, $\forall j \in [\lambda], \Pr[W_j = 1] = \frac{1}{2}$, moreover, $\{Y_j\}_{j \in [\lambda]}$ are IID with parameter $2c\eta$, by definition of the noise model

22

Noise($2c\eta$) (Definition 3). Clearly, for every $j \in [\lambda]$, W_j and Y_j independent random variablesbecause the noise acts on the qubits independent of the key-gen algorithm. Therefore, by definition, $\{X_j\}_{j\in[\lambda]}$ are IID random variables with parameter $\frac{1}{2} \cdot 2c\eta = c\eta$. Let $X \equiv \sum_{i=j}^{\lambda} X_j$.

Note that $X_j = 1$ denotes the event that the qubit at the jth coordinate is in the relevant basis with respect to the message m and that the qubit was corrupted by the noise. Hence, the quantum states of the qubits of $|♘'\rangle$ at the relevant coordinates with respect to m, differ from that of $|♘\rangle$ at exactly X coordinates. Since there are a total of λ qubits in each token, and $\{X_j\}_{j\in[\lambda]}$ are IID with parameter $c\eta$, $E[X] = E[\sum_{i=1}^{\lambda} X_i] = c\eta\lambda$.

Moreover, using a standard Chernoff-bound argument, we get,

$$\Pr[X > \eta\lambda] = d^\lambda \implies \Pr[X \leq \eta\lambda] \geq 1 - d^\lambda,$$

where $d \equiv e^{-2(1-c)^2\eta^2} < 1$. Let $\sigma' \leftarrow \mathsf{CTMAC}^\eta.\mathsf{sign}_{|♘'\rangle}(m)$. Conditioned on the event that $X \leq \eta\lambda$, the signature σ' would be inconsistent with the secret key at less than $\eta\lambda$ of the relevant coordinates with respect to m, and hence, $\mathsf{CTMAC}^\eta.\mathsf{verify}_k(\sigma', m) = 1$ with certainty. Therefore,

$$\Pr[\mathsf{CTMAC}^\eta.\mathsf{verify}_k(\sigma', m) = 1] \geq \Pr[X \leq \eta\lambda] \geq 1 - d^\lambda.$$

Since d^λ is negligible in λ, and $m \in \{0, 1\}$ was arbitrary, we conclude that CTMAC^η is correct up to a negligible error in the noise model, Noise($2c\eta$), and is hence $2c\eta$-noise-tolerant. Since $0 \leq c < 1$ was arbitrary, we conclude that CTMAC^η is $2c\eta$-noise-tolerant for every $0 \leq c < 1$. $\qquad\square$

Theorem 4. *The 1-restricted (Definition 2) scheme CTMAC^η (Algorithm 5) is* Unforgeable$^{\text{1-token-gen,verify}}$ *for any $0 < \eta < \frac{1-\alpha}{2}$, where $\alpha = \cos^2(\frac{\pi}{8})$.*

The proof of Theorem 4 is given in Section 5.4.

In particular, since $0.07 < \frac{1-\alpha}{2}$, we get the following corollary of Theorems 3 and 4.

Corollary 2. *The 1-restricted scheme (see Definition 2) $\mathsf{CTMAC}^{0.07}$ (see Algorithm 5), which is tolerant to noise up to 14% noise, is* Unconditionally Unforgeable$^{\text{1-token-gen,verify}}$.

5 Proving Single Token Unforgeability for CTMAC^η

In this section, we prove that CTMAC^η, the Conjugate TMAC (Algorithm 4), is Unconditionally Unforgeable$^{\text{1-token-gen,verify}}$ (see Definition 5).

Recall that in the security game $\mathsf{Forge}_{\mathcal{A},\Pi}^{\text{1-token-gen,verify}}(\lambda)$ for a 1-restricted scheme (see Definition 2), the adversary wins if it submits valid signatures for both 0 and 1, given a single token $|♘\rangle$ and oracle access to verify.

A naïve attempt for a proof would be to follow the blueprint used to prove the security for classically verifiable private quantum money schemes in Refs. [MVW12, PYJ$^+$12] because the bill, challenges, and answers to the challenges there roughly correspond to the tokens, documents, and signatures in our construction, respectively. The blueprint proceeds as follows. In the first step, unforgeability is proven against 1-to-2 counterfeiting (called "simple counterfeiting" in [MVW12]), meaning an adversary with a single bill (without any access to the verification oracle) cannot submit two bills that pass verification. Next, a reduction is shown from an adversary with oracle access to verification, submitting polynomially many money-states to a 1-to-2 adversary. We can define 1-to-2 unforgeability in a manner similar to $\mathsf{Unforgeable}^{\text{1-token-gen}}$ with a small change that the adversary is forced to submit only two documents, and hence, we can follow the first step of the blueprint with respect to CTMAC^η without an issue. The analysis would almost be the same as in [MVW12, Section 4.2]. However, there is an issue in mimicking the second step in the blueprint, i.e., it is not clear in case of CTMAC^η, how to simulate an adversary with access to verification oracles using an adversary mentioned without the verification oracle. In order to shed more light on this issue, we briefly review the reduction used in [MVW12]. The reduction constructs a 1-to-2 adversary (without any oracle access) that simulates the adversary augmented with a verification oracle by guessing the responses of the oracle queries. The decrease in success probability is only polynomial because it can be assumed that at most two of the queries of the augmented adversary were successful. This property holds in the context of [MVW12] because the challenger chooses the challenges (representing measurements) which means with overwhelming probability, the challenges are all distinct, and all queries to the oracle would be with regard to distinct challenges.

On the contrary, in the setting of a 1-restricted TMAC, the adversary in the unforgeability game (corresponding to $\mathsf{Forge}^{\text{1-token-gen,verify}}(\lambda)$) gets to choose the "challenges" themselves, and hence the challenges need not be random. In particular, the adversary might query the signature of the same document successfully queried multiple times with different signatures. Hence, a uniformly random guess for the set of successful queries would be correct only with exponentially small probability.[16] It is easy to see that the issue discussed above cannot occur in a strong TMAC (see Definition 17) scheme because it is not possible to successfully query different signatures of the same document in a strong TMAC, except with negligible probability. However, our construction is not a strong TMAC. For instance, if we consider

---

[16]In fact, there are TMAC schemes which are $\mathsf{Unforgeable}^{\text{1-token-gen}}$, but are not $\mathsf{Unforgeable}^{\text{1-token-gen,verify}}$. In order to see this, consider an $\mathsf{Unforgeable}^{\text{1-token-gen,verify}}$ TMAC such as our construction or [BS17], and modify its verification procedure to also accept valid signatures appended with a prefix of the secret key. Repeated tries and fails of verification could then uncover the key.

the noise-sensitive scheme CTMAC^0, then we can construct an adversary that given one token and access to verification oracle, produces two different signatures of a single bit, as described in Appendix C.2. The main idea is that an adversary with a verification oracle could easily measure the token in one of the two bases (computational or Hadamard), and obtain a valid signature as described in Appendix C.2. Then she could perform queries to the oracle changing one bit at a time to uncover the quantum states of all coordinates in the chosen basis which would enable her to create multiple signatures of the bit corresponding to the chosen basis.

Ergo, we argue for security against attacks augmented with verification oracle via a scheme specific approach. The crucial point to observe is that in the attack mentioned above, the adversary only learns the quantum state of the coordinates that are in one particular basis. Informally, we claim that this is all the information the adversary can learn using the verification oracle. This statement is made formal in Section 5.1.

The structure of the proof is given in Fig. 1. We first consider the security game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda)$ that is the same as the security game for $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\text{verify}}(\lambda)$, but with a different verification oracle that along with the result, also returns additional information after a successful query. The additional information that the oracle provides, represents all the information that the adversary could have obtained by making repeated successful queries to the same message. Hence, there is no need for the adversary to make repeated successful queries to the same message, and the game is equivalent in power if the adversary is not allowed to do so, which is the definition of the game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$. $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda)$ clearly gives more power to the adversary compared to the security game corresponding to $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\text{verify}}(\lambda)$. Hence, if the success probability for any adversary in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ is negligible, then the same holds in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\text{verify}}(\lambda)$. Next, it is shown that for $\eta = 0$, an adversary in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ can be reduced to an adversary in the "weak" certified deletion game for a tailor-made quantum encryption with certified deletion scheme [BI20].

Finally, we conclude by showing a reduction from $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ with respect to the noise-tolerant scheme to the same game but with respect to the noise-sensitive scheme (i.e $\eta = 0$), with a decrease in the security parameter. The reduction is unusual in the sense that the success probability decreases by a multiplicative exponential factor, but the bound on the success probability of the adversary in the reduced game is small enough that we still get a meaningful bound on the success probability of the adversary in the original game.

The arguments used in all the steps of the proof are information-theoretic and hence, we conclude that CTMAC^η is $\mathsf{Unconditionally\ Unforgeable}^{\text{1-token-gen},\text{verify}}$.

This is a stronger result in the sense that we require unforgeability only against QPT adversaries to lift the scheme to an $\mathsf{Unforgeable}^{\mathsf{token\text{-}gen,verify,\widetilde{sign}}}$ TMAC.

E_0, E_1 (Algorithm 7) are secure

in the WEAK-DEL-IND game (Game 8).

$\downarrow$ Proposition 5

CTMAC (Algorithm 4) is 1-restricted

$\mathsf{Unforgeable}^{1\text{-}\mathsf{token\text{-}gen}, \mathcal{VR}^*}$.

$\downarrow$ Lemma 1 and Proposition 7

CTMAC^η (Algorithm 5) is 1-restricted

$\mathsf{Unforgeable}^{1\text{-}\mathsf{token\text{-}gen}, \mathcal{VR}^{\eta *}}$.

$\downarrow$ Proposition 3

CTMAC^η is 1-restricted

$\mathsf{Unforgeable}^{1\text{-}\mathsf{token\text{-}gen}, \mathcal{VR}^\eta}$.

$\downarrow$ Follows from the definition of $\mathcal{VR}^\eta$

CTMAC^η is 1-restricted

$\mathsf{Unforgeable}^{1\text{-}\mathsf{token\text{-}gen}, \mathsf{verify}}$.
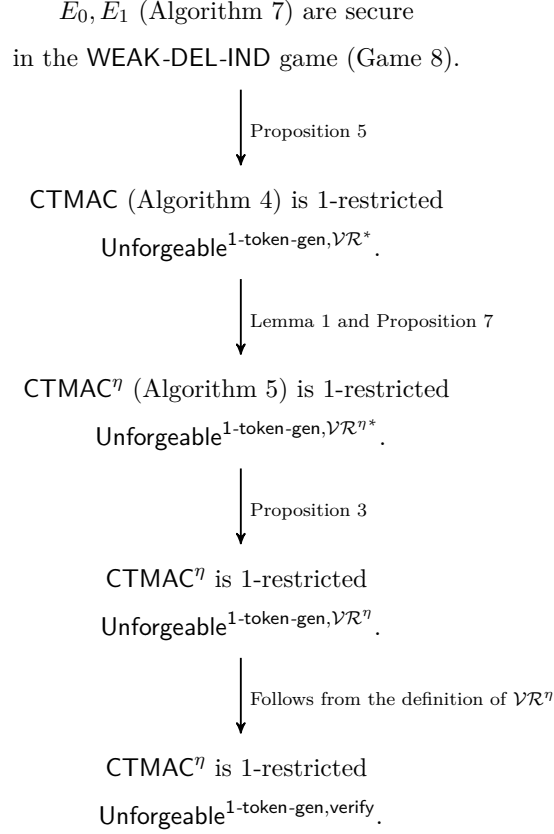
Figure 1: The above diagram summarizes the main stages of the proof of Theorem 4.

5.1 The Revealed Basis Setting

In this section, we consider a modified version of the security game $\mathsf{Forge}^{1\text{-}\mathsf{token\text{-}gen,verify}}$ that we get by replacing the verification oracle verify with an oracle $\mathcal{VR}_k^\eta$ that provides more information to the adversary.

For a key $k = (a, b)$ representing an n qubit length state $|\text{⌖}\rangle = H^b|a\rangle$, the oracle $\mathcal{VR}_k^\eta$ (Verify and Reveal) receives input of the form (m, σ), runs the verification of CTMAC^η and returns a bit 0 or 1 accordingly, called the result bit. In addition, if the verification is successful, then along with the result bit, it also responds with the sets $\mathsf{Cons}_m = \{i \in [n] | b_i = m\}$, and the set $\mathsf{Miss}_{m,\sigma} = \{i \in \mathsf{Cons}_m \mid \sigma_i \neq a_i\}$. Queries whose result bit is 0 are referred

to as failed queries, and the others are referred to as successful queries.

The corresponding security game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^\eta}(\lambda)$, described in Algorithm 6 is defined by replacing the verification oracle in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathsf{verify}}(\lambda)$ with $\mathcal{VR}_k^\eta$. As $\mathcal{VR}_k^\eta$ is scheme specific, $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^\eta}(\lambda)$ is defined only in the context of CTMAC^η.

Game 6 $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^\eta}(\lambda)$:

1: $\mathcal{C}$ generates a secret key k by $\mathsf{CTMAC}^\eta.\mathsf{key\text{-}gen}(\lambda)$.

2: $\mathcal{A}$ is given input 1^λ as well as a single access to $\mathsf{CTMAC}^\eta.\mathsf{token\text{-}gen}_k$ and classical oracle access to $\mathcal{VR}_k^\eta$, which it can query a polynomial number of times. $\mathcal{A}$ then outputs (σ_0, σ_1).

The output of the game is defined to be 1 if and only if (1) $\mathsf{CTMAC}^\eta.\mathsf{verify}_k(0, \sigma_0) = 1$ and (2) $\mathsf{CTMAC}^\eta.\mathsf{verify}_k(1, \sigma_1) = 1$.

We also define a stricter variant of the game in Definition 7.

Definition 7. *The security game,* $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ *is defined to be the same as* $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^\eta}(\lambda)$, *except for the added restriction that the value of the game is* 0 *if the adversary performs two successful queries to* $\mathcal{VR}_k^\eta$ *for the same document.*

Next, we will show that an adversary in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ does not gain additional information by repeated successful queries with respect to the same message, and hence, the two games are equivalent.

Proposition 3. *For every (computationally unbounded)* $\mathcal{A}$ *that wins* $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^\eta}(\lambda)$ *with probability* $\epsilon(\lambda)$ *using at most* $q(\lambda)$ *queries to* $\mathcal{VR}_k^\eta$, *there exists a (computationally unbounded) adversary* $\mathcal{B}$ *winning* $\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ *with probability* $\epsilon(\lambda)$ *making at most* $q(\lambda)$ *queries to* $\mathcal{VR}_k^\eta$.

Proof. Let $\mathcal{A}$ be an adversary who makes at most $q(\lambda)$ queries to the oracle $\mathcal{VR}_k^\eta$ in the security game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^\eta}(\lambda)$, and wins the game with probability $\epsilon(\lambda)$. We construct an adversary $\mathcal{B}$ to $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{1\text{-token-gen},\mathcal{VR}^\eta}(\lambda)$, who makes at most $q(\lambda)$ queries. $\mathcal{B}$ runs $\mathcal{A}$ on the security parameter she receives. $\mathcal{B}$ answers any oracle query (m, σ) that $\mathcal{A}$ makes to $\mathcal{VR}^\eta$, as follows.

- If m has not been queried in any previous successful oracle query, $\mathcal{B}$ queries its own oracle with (m, σ) and passes the answer to $\mathcal{A}$. If the query is successful, $\mathcal{B}$ stores the result Cons_m and $\mathsf{Miss}_{\sigma,m}$, and uses these sets to uncover x_m, a string describing the quantum state of the token at the coordinates Cons_m.

- Else, $\mathcal{B}$ uses the stored result to check if $\mathsf{CTMAC}^\eta.\mathsf{verify}_k(m, \sigma)$ would pass (checking that $\sigma|_{\mathsf{Cons}_m} = x_m$ for all but $\eta\lambda$ indices), and answers accordingly to $\mathcal{A}$, re-forwarding the stored result if necessary.

The view of $\mathcal{A}$ is clearly identical to that in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda)$. Note that, once $\mathcal{B}$ successfully queries the oracle for a particular document, it would not query for the same document again. Hence, $\mathcal{B}$ wins $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ if and only if $\mathcal{A}$ wins $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda)$. Therefore, $\mathcal{B}$ wins $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ with a probability of $\epsilon(\lambda)$. $\qquad\square$

Note that any adversary in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathsf{verify}}(\lambda)$ (Game 2) is also an adversary in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda)$. Therefore, if the winning probability for any QPT adversary $\mathcal{A}$ in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda)$ is bounded by $\epsilon(\lambda)$, then the same would hold for the security game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathsf{verify}}(\lambda)$.

By taking the contra-positive of the Proposition 3, we reach the following corollary:

Corollary 3. *If for every λ and for every computationally unbounded $\mathcal{B}$ making at most $q(\lambda)$ queries to $\mathcal{VR}_k^\eta$*

$$\Pr[\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda) = 1] \le \epsilon(\lambda),$$

then for every λ and for any computationally unbounded adversary $\mathcal{A}$ making at most $q(\lambda)$ queries to verify_k

$$\Pr[\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathsf{verify}}(\lambda) = 1] \le \epsilon(\lambda),$$

where $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathsf{verify}}(\lambda)$ and $\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ are the security games defined in Game 2 and Definition 7, respectively.

5.2 Reduction to Certified Deletion in a Noiseless Setting

In the previous section, we saw that Corollary 3 implies that proving unforgeability can be reduced to finding a bound on

$$\Pr[\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda) = 1],$$

where $\mathsf{Forge}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}$ is the game described in Definition 7. We begin by proving the bound for the special case $\eta = 0$, meaning the underlying scheme is the simpler CTMAC. The bound for the special case is achieved by a reduction to a weaker notion of security of a Quantum Encryption with Certified Deletion scheme (QECD). Hence, we take a short detour to QECD [BI20].

Definition 8 (Quantum Encryption with Certified Deletion, [BI20])**.** *A quantum encryption with certified deletion scheme, or* QECD, *consists of five* QPT *algorithms:* $\mathsf{key\text{-}gen}, \mathsf{enc}, \mathsf{dec}, \mathsf{del}$ *and* verify *fulfilling the following:*[17]

---

[17]We defer here from the definition given previously. In [BI20], the length of the encrypted message, n, is an integer independent of λ. In addition, [BI20] uses density operator formalism, while we use pure states to preserve consistency with the rest of the paper.

1. *On input 1^λ, where λ is the security parameter, the algorithm* key-gen *outputs a classical key k.*

2. $\mathsf{enc}_k(a)$ *receives a classical string $a \in \{0,1\}^\lambda$ and outputs a quantum state $|c\rangle$ denoting its encryption.*

3. $\mathsf{dec}_k(|c\rangle)$ *receives a quantum state $|c\rangle$ and outputs a classical string $a' \in \{0,1\}^\lambda$.*

4. $\mathsf{del}(|c\rangle)$ *receives a quantum state and returns a classical string cer, which we will refer to as the deletion certificate.*

5. $\mathsf{verify}_k(cer)$ *outputs a Boolean answer.*

The motivation for a quantum encryption with certified deletion scheme is that an adversary in possession of some cipher-text could supply a trusted authority with a certificate that the adversary has deleted the cipher, which the authority can then verify. If verification of the certificate succeeds, then the trusted authority is assured that even if the key is later leaked, or intentionally given to the adversary, the adversary cannot recover the data previously encrypted. The two QECD schemes E_m, for $m \in \{0,1\}$ that are of interest to us are given in Algorithm 7.

As per the original definition in [BI20], a QECD scheme is secure if no adversary can distinguish between the encryption of a message of his choice and that of the 0^λ string, even if she is provided with the secret key after it is verified that she deleted the cipher. These schemes are not certified deletion secure in that sense.[18] However, for our needs, it is sufficient to provide a bound on the winning probability in a stricter security game (Game 8) that is tailor-made for our needs.

In Appendix B, we prove the following proposition, based on semi-definite programming methods.

Proposition 4. *The optimal success probability for an adversary in* WEAK-DEL-IND$_{\mathcal{A},E_0}(\lambda)$ *(see Game 8 and Algorithm 7), as well as in* WEAK-DEL-IND$_{\mathcal{A},E_1}(\lambda)$ *is α^λ, where $\alpha = \cos^2(\frac{\pi}{8})$.*

Next, we exhibit a reduction from the game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^{0^*}}(\lambda)$ to the weak certified deletion game (Game 8) against the schemes E_0, E_1 (Algorithm 7), to conclude the bound on the winning probability in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^{0^*}}(\lambda)$.

Note that for $\eta = 0$, $\mathsf{Miss}_{m,\sigma}$ is always the empty set and can be disregarded. For the sake of succinctness, we will omit η from the superscript $\mathcal{VR}^\eta$ and use $\mathcal{VR}$ to denote $\mathcal{VR}^0$ from now onward.

---

[18]A simple adversary could choose a string differing by 1 bit from the 0 string as m_1: measuring honestly results in a certificate r that is then provided to the adversary. If the key is 1 at the corresponding coordinate (meaning the corresponding qubit was in the Hadamard basis), then the adversary outputs 1, or else they output a uniformly random guess. It is easy to see that the attacker wins with a probability of $\frac{3}{4}$.

Algorithm 7 E_m - A QECD scheme (for $m \in \{0,1\}$)

1: **procedure** key-gen(1^λ)
2:     $k \leftarrow\$ \{0,1\}^\lambda$.
3:     **Return** k.
4: **end procedure**

1: **procedure** enc$_k(a)$
2:     **Return** $H^k(|a\rangle)$.
3: **end procedure**

1: **procedure** dec$_k(|c\rangle)$
2:     Compute $H^k|c\rangle$ to obtain $|cc\rangle$.
3:     Measure $|cc\rangle$ to obtain a'.
4:     **Return** a'.
5: **end procedure**

1: **procedure** del$(|c\rangle)$
2:     Compute $(H^m)^{\otimes\lambda}|c\rangle$ to obtain $|cc\rangle$.
3:     Measure $|cc\rangle$ to obtain cer.
4:     **Return** cer.
5: **end procedure**

1: **procedure** verify$_k(cer)$
2:     Define $\mathsf{Cons}_m = \{i \in [\lambda] | k_i = m\}$.
3:     **if** $cer = a|_{\mathsf{Cons}_m}$ **then**
4:         **Return** 1.
5:     **else**
6:         **Return** 0.
7:     **end if**
8: **end procedure**

Game 8 Weak Certified Deletion Game $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{A},\Pi}(\lambda)$:

1: The challenger $\mathcal{C}$ runs $\mathsf{key\text{-}gen}(1^\lambda)$ to generate k, and uniformly samples $a \in \{0,1\}^\lambda$. $\mathcal{C}$ then sends $|c\rangle = \mathsf{enc}_k(a)$.
2: $\mathcal{A}$ sends $\mathcal{C}$ some string cer.
3: $\mathcal{C}$ computes $V \leftarrow \mathsf{verify}_k(cer)$, and then sends k to the adversary.
4: $\mathcal{A}$ outputs a'.

We say the output of the game is 1 if and only if $V = 1$ and $a = a'$.

Proposition 5. *For every infinite set of integers S and a computationally unbounded $\mathcal{A}$ making $q(\lambda)$ queries to $\mathcal{VR}_k$, such that for all $\lambda \in S$*

$$\Pr[\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lambda) = 1] = \epsilon(\lambda),$$

where $\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta^*}}(\lambda)$ is the game defined in Definition 7, either there is a computationally unbounded adversary $\mathcal{B}_0$ such that for infinitely many $\lambda \in S$,

$$\Pr[\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{B}_0,E_0}(\lambda) = 1] \geq \frac{1}{2\binom{q(\lambda)+2}{2}}\epsilon(\lambda),$$

or there is a computationally unbounded adversary $\mathcal{B}_1$ such that for infinitely many $\lambda \in S$,

$$\Pr[\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{B}_1,E_1}(\lambda) = 1] \geq \frac{1}{2\binom{q(\lambda)+2}{2}}\epsilon(\lambda).$$

Proof. Let $\mathcal{A}$ be an arbitrary adversary for the $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lambda)$ game (see Definition 7) who makes $q(\lambda)$ queries to $\mathcal{VR}$, and wins with probability $\epsilon(\lambda)$. Since verification is deterministic, without loss of generality, it can be assumed that if $\mathcal{A}$ wins $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lambda)$, it always makes two successful queries for different documents, perhaps at the cost of forcing $\mathcal{A}$ to make at most two more queries. Let $A_m(\lambda)$ for $m \in \{0,1\}$ be the event that the adversary sent two successful queries corresponding to two different documents and the first of the two successful queries made by $\mathcal{A}$ is for the document m, and let $p_m(\lambda) \equiv \Pr[A_m(\lambda)]$. Since $p_0(\lambda) + p_1(\lambda) = \epsilon(\lambda)$, there exists a bit $\hat{m}$, such that $p_{\hat{m}}(\lambda) \geq \frac{1}{2}\epsilon(\lambda)$ for infinitely many $\lambda \in S$. Denote the corresponding set of infinitely many λ as G. We construct an adversary $\mathcal{B}_{\hat{m}}$ against $E_{\hat{m}}$ as follows. Given an encryption $|c\rangle = H^k|a\rangle$ for message $a \in \{0,1\}^\lambda$ and key $k \in \{0,1\}^\lambda$, $\mathcal{B}_{\hat{m}}$ would pass $|c\rangle$ to $\mathcal{A}$ as a token. $\mathcal{B}_{\hat{m}}$ would then simulate $\mathcal{A}$, and uniformly guess the two indices of its two successful queries $i < j$. On the ith query, $\mathcal{B}_{\hat{m}}$ will forward the queried signature $\sigma_{\hat{m}}$ to the challenger as a certificate of deletion. Let $\mathsf{Cons}_{\hat{m}} \equiv \{i \in [\lambda] | k_i = \hat{m}\}$. The challenger will check that $a|_{\mathsf{Cons}_{\hat{m}}} = \sigma_{\hat{m}}|_{\mathsf{Cons}_{\hat{m}}}$, which is exactly what is

expected of a valid signature for $\hat{m}$. Hence, for a correct guess of the indices, the signature would be deemed valid, and $\mathcal{B}_{\hat{m}}$ would get the key k from the challenger. $\mathcal{B}_{\hat{m}}$ can then use k to construct $\mathsf{Cons}_{\hat{m}}$ by itself, and answer $\mathcal{A}$'s successful oracle query. $\mathcal{B}_{\hat{m}}$ will continue to reject all other queries made by $\mathcal{A}$ until the jth query. On the jth query, it will accept the sent string $\sigma_{1-\hat{m}}$. $\mathcal{B}_{\hat{m}}$ could then reassemble the string a' as follows.

$$
a'(i) = \begin{cases} \sigma_{\hat{m}}(i) & \text{if } i \in \mathsf{Cons}_{\hat{m}} \\ \sigma_{1-\hat{m}}(i) & \text{otherwise,} \end{cases}
$$

and send a' to the challenger.

$\mathcal{B}_{\hat{m}}$ guesses the indices correctly with a probability greater than $\frac{1}{\binom{q(\lambda)+2}{2}}$ (since up to two oracle queries may have been added). In such an event, the view of $\mathcal{A}$ is the same as in $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lambda)$, meaning for all $\lambda \in G$, with a probability greater than $\frac{1}{2}\epsilon(\lambda)$, $\mathcal{A}$ outputs two successfully verified signatures, and the first successful query is for the document $\hat{m}$.

In that case, $a|_{\mathsf{Cons}_0} = \sigma_0|_{\mathsf{Cons}_0}$ and $a|_{\mathsf{Cons}_1} = \sigma_1|_{\mathsf{Cons}_1}$ and, hence, $a' = a$ (that is, $\mathcal{B}$ wins). Therefore, $\mathcal{B}_{\hat{m}}$ could win the $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{B}_{\hat{m}},E_{\hat{m}}}(\lambda)$ with a probability of at least $\frac{1}{2\binom{q(\lambda)+2}{2}}\epsilon(\lambda)$ for all $\lambda \in G$. □

An immediate result is the following:

Proposition 6. *For every computationally unbounded $\mathcal{A}$ making $q(\lambda)$ queries to $\mathcal{VR}_k$, there exists λ_0 such that for all $\lambda > \lambda_0$:*

$$
\Pr[\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lambda) = 1] \leq 2\alpha^\lambda \binom{q(\lambda)+2}{2},
$$

where $\alpha = \cos^2(\frac{\pi}{8})$ and $\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ is the game defined in Definition 7.

Proof. Assume towards contradiction that this is not so, then by Proposition 5, there exists $\mathcal{B}_0$ winning $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{B}_0,E_0}$ with a probability greater than α^λ for infinitely many λ, or there exists $\mathcal{B}_1$ winning $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{B}_1,E_1}$ with a probability greater than α^λ for infinitely many λ. In either case, this contradicts Proposition 4. □

The bound above also holds against any adversary $\mathcal{A}$ that makes *at most* $q(\lambda)$ queries to $\mathcal{VR}_k$ because $\mathcal{A}$ can always be assumed to make exactly $q(\lambda)$ queries by forcing $\mathcal{A}$ to submit extra failing queries. If $q(\lambda)$ is polynomial, this bound is negligible in λ. Combined with Corollary 3, this bound suffices to prove that CTMAC is $\mathsf{Unforgeable}^{\text{1-token-gen,verify}}$ (Theorem 2).

5.3 Noisy to Noiseless Reduction in a Revealed Basis Setting

We extend the results to the noise-tolerant setting by giving a reduction from the noise-tolerant revealed basis setting to the noise-sensitive revealed basis setting.

Lemma 1. *Let $f : \mathbb{N} \to \mathbb{N}$ be a function such that $f(\lambda) < \lambda$, and let $\mathcal{A}$ be an adversary making $q(\lambda)$ queries to $\mathcal{VR}_k^\eta$ and denote*

$$W_\eta(\lambda) \equiv \Pr[\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda) = 1],$$

where $\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ is the game defined in Definition 7. There exists an adversary $\mathcal{B}$ making at most $q(\lambda) + 2$ queries to $\mathcal{VR}_k$ such that for every λ and any $0 \le \eta \le 1$,

$$\Pr[\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{*}}(f(\lambda)) = 1] \ge \left(1 - \frac{2\eta\lambda}{\lambda - f(\lambda)}\right)^{f(\lambda)} W_\eta(\lambda).$$

Proof. Let $\mathcal{A}$ be an adversary for $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ (Definition 7), as described in Lemma 1. For simplicity of description, we assume that that $\mathcal{A}$ always makes at least two queries to $\mathcal{VR}^\eta$, and that $\mathcal{A}$ always submits the first two successful queries it makes to $\mathcal{VR}^\eta$. We also assume that if $\mathcal{A}$ wins, it always makes two successful queries for different documents, perhaps at the cost of forcing $\mathcal{A}$ to make at most two more queries. We construct an adversary $\mathcal{B}$ for the corresponding noise-sensitive security game $\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^{*}}(f(\lambda))$ (Definition 7) as follows.

Let $\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}$ be a set of indices, and also denote $\overline{\mathcal{I}} \equiv [\lambda]\backslash\mathcal{I}$. For every $\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}$, we define $\mathcal{B}^{\mathcal{I}}$, an adversary for the noise-sensitive scheme CTMAC^η as follows. Upon receiving a $f(\lambda)$-qubit token $|\underline{\omega}\rangle$ from $\mathcal{C}$, $\mathcal{B}^{\mathcal{I}}$ generates another token $|\widetilde{\underline{\omega}}\rangle$ consisting of $\lambda - f(\lambda)$ qubits. This is done by running $\mathsf{CTMAC}^\eta.\mathsf{key\text{-}gen}(1^{\lambda-f(\lambda)})$ to obtain a key $\widetilde{k} = (\widetilde{a},\widetilde{b})$ followed by running $\mathsf{CTMAC}^\eta.\mathsf{token\text{-}gen}_{\widetilde{k}}$ to get the token. $\mathcal{B}^{\mathcal{I}}$ then assembles a λ-length token $|\underline{\omega}'\rangle$ by placing the qubits of $|\underline{\omega}\rangle$ at coordinates $\mathcal{I}$, and the qubits of $|\widetilde{\underline{\omega}}\rangle$ at coordinates $\overline{\mathcal{I}}$. This also defines a fictional key $k' \equiv (a', b')$, composed of the strings a', b', which are derived by placing $\widetilde{a}, \widetilde{b}$ at coordinates $\overline{\mathcal{I}}$ and a, b at coordinates $\mathcal{I}$. $\mathcal{B}^{\mathcal{I}}$ simulates $\mathcal{A}$ on the token $|\underline{\omega}'\rangle$.

For every alleged query (m, σ) made to $\mathcal{VR}^\eta$ by $\mathcal{A}$ in the simulated game of $\mathcal{B}^{\mathcal{I}}$, let

$$\mathsf{Cons'}_m^{\mathcal{I}} \equiv \{i \in [\lambda] \mid b_i' = m\},$$

$$\mathsf{Miss'}_{m,\sigma}^{\mathcal{I}} \equiv \{i \in \mathsf{Cons'}_m^{\mathcal{I}} \mid a_i' \neq \sigma_i\},$$

which are defined with respect to the fictional key k'. Additionally, let

$$\widetilde{\mathsf{Cons}}_m^{\overline{\mathcal{I}}} \equiv \{i \in \overline{\mathcal{I}} \mid b_i' = m\},$$

$$\widetilde{\mathsf{Miss}}_{m,\sigma}^{\overline{\mathcal{I}}} \equiv \{i \in \widetilde{\mathsf{Cons}}_m^{\overline{\mathcal{I}}} \mid a_i' \neq \sigma_i\}.$$

We say a query (m,σ) to the oracle $\mathcal{VR}^\eta$ in the true game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ is "good" if $|\mathsf{Miss}_{m,\sigma}| \leq \eta\lambda$. Similarly, we say such a query in the simulation of $\mathcal{A}$ within $\mathcal{B}^{\mathcal{I}}$ is "good" if $\left|\mathsf{Miss}'_{m,\sigma}^{\mathcal{I}}\right| \leq \eta\lambda$.

We further describe the behavior of $\mathcal{B}^{\mathcal{I}}$: whenever $\mathcal{A}$ queries $\mathcal{VR}^\eta$, with a query (m,σ), $\mathcal{B}^{\mathcal{I}}$ runs $\mathsf{CTMAC.verify}_{\widetilde{k}}(m|_{\overline{\mathcal{I}}}, \sigma|_{\overline{\mathcal{I}}})$ and constructs the corresponding sets $\widetilde{\mathsf{Cons}}_m^{\overline{\mathcal{I}}}$, $\widetilde{\mathsf{Miss}}_{m,\sigma}^{\overline{\mathcal{I}}}$. If $\left|\widetilde{\mathsf{Miss}}_{m,\sigma}^{\overline{\mathcal{I}}}\right| > \eta\lambda$, meaning there are more than $\eta\lambda$ errors at coordinates in $\overline{\mathcal{I}}$ that $\mathcal{B}^{\mathcal{I}}$ rejects. Otherwise, it queries $\mathcal{VR}$ with $(m|_{\mathcal{I}}, \sigma|_{\mathcal{I}})$, and accepts if and only if $\mathcal{VR}$ accepts. Note that if $\mathcal{VR}$ accepts, $\mathcal{B}^{\mathcal{I}}$ would receive the set $\mathsf{Cons}_m^{\mathcal{I}} \equiv \{i \in \mathcal{I} | b_i' = m\}$, and since the challenger for $\mathcal{B}^{\mathcal{I}}$ is with respect to the noise-sensitive scheme, $\sigma|_{\mathcal{I}}$ must be consistent on $\mathsf{Cons}_m^{\mathcal{I}}$. $\mathcal{B}^{\mathcal{I}}$ could thus reliably accept the query and forward to $\mathcal{A}$ the sets $\mathsf{Cons}'_m^{\mathcal{I}} = \widetilde{\mathsf{Cons}}_m^{\overline{\mathcal{I}}} \cup \mathsf{Cons}_m^{\mathcal{I}}$ and $\mathsf{Miss}'_{m,\sigma}^{\mathcal{I}} = \widetilde{\mathsf{Miss}}_{m,\sigma}^{\overline{\mathcal{I}}}$. Finally, when $\mathcal{A}$ submits two signed documents (σ_0, σ_1), $\mathcal{B}^{\mathcal{I}}$ would submit to its challenger $(\sigma_0|_{\mathcal{I}}, \sigma_1|_{\mathcal{I}})$. The important observation is that $\mathcal{B}^{\mathcal{I}}$ would never accept a query (m,σ), unless it is "good", even though it might reject queries that are "good". The adversary $\mathcal{B}$, is defined as the adversary which uniformly selects $\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}$, and runs $\mathcal{B}^{\mathcal{I}}$. Informally, we claim that in events where $\mathcal{A}$ wins $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda)$, the view of $\mathcal{A}$ is identical to the view when run within the simulation of $\mathcal{B}^{\mathcal{I}}$, for sufficiently many choices of $\mathcal{I}$.

Definition 9. *The random variables M_1 and S_1 denote the document and signature, respectively, in the first "good" query, in the true game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}$. If such a query does not exist, M_1, S_1 are the document and signature, respectively, in the next-to-last query made by $\mathcal{A}$. Similarly, M_2, S_2 denote the document and signature, respectively, in the second "good" query. If such a query does not exist, M_2, S_2 are the document and signature, respectively, in the last query made by $\mathcal{A}$. The random variable J_1 represents the indices in which S_1 is inconsistent with the secret key, that is, the set Miss_{M_1,S_1}, and likewise, J_2 is the set Miss_{M_2,S_2}. The random variable T represents the token $|\underline{\mathcal{Q}}'\rangle$ that $\mathcal{A}$ receives from the challenger. In addition, for any $\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}$, we define corresponding random variables $M_1^{\mathcal{I}}, S_1^{\mathcal{I}}, M_2^I, S_2^{\mathcal{I}}, J_1^{\mathcal{I}}, J_2^{\mathcal{I}}, T^{\mathcal{I}}$, denoting the same, but when $\mathcal{A}$ is simulated within $\mathcal{B}^{\mathcal{I}}$, with $J_i^{\mathcal{I}}$ defined with regard to $\mathsf{Miss}'_{M_i^{\mathcal{I}}, S_i^{\mathcal{I}}}^{\mathcal{I}}$.*

We emphasize, that the value of $M_1^{\mathcal{I}}, S_1^{\mathcal{I}}, M_2^{\mathcal{I}}, S_2^{\mathcal{I}}, J_1^{\mathcal{I}}, J_2^{\mathcal{I}}$ is defined with regard to "good" queries, and regardless of the response of $\mathcal{B}^{\mathcal{I}}$ to the query, which could potentially reject a "good" query.

34

Denote:

$$\mathcal{E}_{good} = \{\mathcal{A} \text{ makes two "good" queries in } \mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}\},$$

$$\mathcal{E}_{view}^{j_1,j_2,b,|\circleddash\rangle} = \{J_1 = j_1, J_2 = j_2, M_1 = b, M_2 = 1 - b, T = |\circleddash\rangle\},$$

$$\overset{\mathcal{I}}{\mathcal{F}}_{good} = \{\mathcal{A} \text{ makes two "good" queries in the simulation within } \mathcal{B}^I\}$$

$$\overset{\mathcal{I}}{\mathcal{F}}_{view}^{j_1,j_2,b,|\circleddash\rangle} = \{J_1^{\mathcal{I}} = j_1, J_2^{\mathcal{I}} = j_2, M_1^{\mathcal{I}} = b, M_2^{\mathcal{I}} = 1 - b, T^{\mathcal{I}} = |\circleddash\rangle\}.$$

Lemma 2. *For all* $\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}$, $j_1, j_2 \subseteq \overline{\mathcal{I}}$, $b \in \{0,1\}$ *and* $|\circleddash\rangle \in \mathcal{BB}84^\lambda$ *such that* $|j_1| \le \eta\lambda, |j_2| \le \eta\lambda$:

$$\Pr[\mathcal{E}_{good} \wedge \mathcal{E}^{j_1,j_2,b,|\circleddash\rangle}] = \Pr[\overset{\mathcal{I}}{\mathcal{F}}_{good} \wedge \overset{\mathcal{I}}{\mathcal{F}}_{view}^{j_1,j_2,b,|\circleddash\rangle}].$$

Proof. Fix $\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}$ arbitrarily and j_1, j_2, as described in the lemma. Conditioned on the events where $T = |\circleddash\rangle$ and $T^{\mathcal{I}} = |\circleddash\rangle$ for some fixed value $|\circleddash\rangle$, in the true game $\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)$ and in the simulation within $\mathcal{B}^{\mathcal{I}}$, respectively, it is easy to argue by induction that for every "step" until a "good" query occurs, the view remains the same in both the true game and the simulation. We can thus derive that for any j_1, b:

$$\Pr[J_1^{\mathcal{I}} = j_1, M_1^{\mathcal{I}} = b \mid T^{\mathcal{I}} = |\circleddash\rangle] = \Pr[J_1 = j_1, M_1 = b \mid T = |\circleddash\rangle].$$

Continuing with similar reasoning, conditioned on $J_1 = j_1, M_1 = b, T = |\circleddash\rangle$ in the true game and $J_1^{\mathcal{I}} = j_1, M_1^{\mathcal{I}} = b, T^{\mathcal{I}} = |\circleddash\rangle$ in the simulation, in addition to conditioning on a fixed index of the first "good" query (if such a query exists) in both, we can argue that:

$$\Pr[J_2^{\mathcal{I}} = j_2, M_2^{\mathcal{I}} = 1 - b \mid J_1^{\mathcal{I}} = j_1, M_1^{\mathcal{I}} = b, T^{\mathcal{I}} = |\circleddash\rangle]$$
$$= \Pr[J_2 = j_2, M_2 = 1 - b \mid J_1 = j_1, M_1 = b, T = |\circleddash\rangle].$$

This is because the conditioning on j_1 ensures that the first "good" query (if such exists) will be answered positively even in the simulation: $j_1 \subseteq \overline{\mathcal{I}}$, meaning the signature corresponding to the query is consistent with b' on $\mathcal{I}$, and contains less than $\eta\lambda$ errors on $\overline{\mathcal{I}}$; hence, the same inductive reasoning on the view applies.

Similarly

$$\Pr[\overset{\mathcal{I}}{\mathcal{F}}_{good} \mid J_2^{\mathcal{I}} = j_2, M_2^{\mathcal{I}} = 1 - b, J_1^{\mathcal{I}} = j_1, M_1^{\mathcal{I}} = b, T^{\mathcal{I}} = |🔑\rangle]$$
$$= \Pr[\mathcal{E}_{good} \mid J_2 = j_2, M_2 = 1 - b, J_1 = j_1, M_1 = b, T = |🔑\rangle].$$

Finally, observe that the token $|🔑\rangle$ that $\mathcal{A}$ receives is distributed uniformly in both the true game and in the simulation. Using standard conditional probability, we break down the probability of the event $\mathcal{E}_{good} \wedge \mathcal{E}_{view}^{j_1,j_2,b,|🔑\rangle}$ into simpler terms, and then reassemble them.

$\Pr[\mathcal{E}_{good} \wedge \mathcal{E}_{view}^{j_1,j_2,b,|🔑\rangle}]$

$\quad = \Pr[\mathcal{E}_{good} \mid J_2 = j_2, M_2 = 1 - b, J_1 = j_1, M_1 = b, T = |🔑\rangle]$

$\quad \cdot \Pr[J_2 = j_2, M_2 = 1 - b \mid J_1 = j_1, M_1 = b, T = |🔑\rangle] \cdot \Pr[J_1 = j_1, M_1 = b \mid T = |🔑\rangle] \cdot \Pr[T = |🔑\rangle]$

$\quad = \Pr[\overset{\mathcal{I}}{\mathcal{F}}_{good} \mid J_2^{\mathcal{I}} = j_2, M_2^{\mathcal{I}} = 1 - b, J_1^{\mathcal{I}} = j_1, M_1^{\mathcal{I}} = b, T^{\mathcal{I}} = |🔑\rangle]$

$\quad \cdot \Pr[J_2^{\mathcal{I}} = j_2, M_2^{\mathcal{I}} = 1 - b \mid J_1^{\mathcal{I}} = j_1, M_1^{\mathcal{I}} = b, T^{\mathcal{I}} = |🔑\rangle] \cdot \Pr[J_1^{\mathcal{I}} = j_1, M_1^{\mathcal{I}} = b \mid T^{\mathcal{I}} = |🔑\rangle] \cdot \Pr[T^{\mathcal{I}} = |🔑\rangle]$

$\quad = \Pr[\overset{\mathcal{I}}{\mathcal{F}}_{good} \wedge \overset{\mathcal{I}}{\mathcal{F}}_{view}^{j_1,j_2,b,|🔑\rangle}],$

which proves Lemma 2. $\qquad\qquad\square$

Notice that for any $\mathcal{I}, j_1, j_2, b, |🔑\rangle$, as described in Lemma 2,

$$\overset{\mathcal{I}}{\mathcal{F}}_{good} \wedge \overset{\mathcal{I}}{\mathcal{F}}_{view}^{j_1,j_2,b,|🔑\rangle} \subseteq \{\mathsf{Forge}_{\mathcal{B}^{\mathcal{I}},\mathsf{CTMAC}}^{\text{1-token-gen,verify}}(\lambda) = 1\},$$

which implies,

$$\bigcup_{\substack{j_1,j_2 \subseteq \overline{\mathcal{I}} \\ |j_1|,|j_2| \le \eta\lambda \\ b \in \{0,1\} \\ |🔑\rangle \in \mathcal{BB}84^\lambda}} \left(\overset{\mathcal{I}}{\mathcal{F}}_{good} \wedge \overset{\mathcal{I}}{\mathcal{F}}_{view}^{j_1,j_2,b,|🔑\rangle} \right) \subseteq \{\mathsf{Forge}_{\mathcal{B}^{\mathcal{I}},\mathsf{CTMAC}}^{\text{1-token-gen,verify}}(\lambda) = 1\}. \qquad (1)$$

In addition, by the assumption that if $\mathcal{A}$ wins, it always makes two successful queries for different documents:

$$\{\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda) = 1\} = \bigcup_{\substack{b \in \{0,1\} \\ |🔑\rangle \in \mathcal{BB}84^\lambda \\ |j_1|,|j_2| \le \eta\lambda \\ j_1,j_2 \subseteq [\lambda]}} \left(\mathcal{E}_{good} \wedge \mathcal{E}^{j_1,j_2,b,|🔑\rangle} \right) \qquad (2)$$

Observe that the success probability of $\mathcal{B}$, is the average over the winning probabilities of $\mathcal{B}^{\mathcal{I}}$ for all choices of $\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}$. Combining this with Eq. (1),

and Lemma 2, we get,

$$\Pr[\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lambda) = 1]$$

$$= \frac{1}{\binom{\lambda}{f(\lambda)}} \sum_{\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}} \Pr[\mathsf{Forge}_{\mathcal{B}^{\mathcal{I}},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lambda) = 1]$$

$$\geq \frac{1}{\binom{\lambda}{f(\lambda)}} \sum_{\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}} \Pr[\biguplus_{\substack{j_1,j_2 \subseteq \overline{\mathcal{I}} \\ |j_1|,|j_2| \leq \eta\lambda \\ b \in \{0,1\} \\ |{\buildrel\circ\over{=}}\rangle \in \mathcal{BB}84^\lambda}} \mathcal{F}_{good}^{\mathcal{I}} \wedge \mathcal{F}_{view}^{\mathcal{I}^{j_1,j_2,b,|{\buildrel\circ\over{=}}\rangle}}] \qquad \text{by Eq. (1)}$$

$$= \frac{1}{\binom{\lambda}{f(\lambda)}} \sum_{\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}} \sum_{\substack{j_1,j_2 \subseteq \overline{\mathcal{I}} \\ |j_1|,|j_2| \leq \eta\lambda \\ b \in \{0,1\} \\ |{\buildrel\circ\over{=}}\rangle \in \mathcal{BB}84^\lambda}} \Pr[\mathcal{F}_{good}^{\mathcal{I}} \wedge \mathcal{F}_{view}^{\mathcal{I}^{j_1,j_2,b,|{\buildrel\circ\over{=}}\rangle}}]$$

$$= \frac{1}{\binom{\lambda}{f(\lambda)}} \sum_{\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}} \sum_{\substack{j_1,j_2 \subseteq \overline{\mathcal{I}} \\ |j_1|,|j_2| \leq \eta\lambda \\ b \in \{0,1\} \\ |{\buildrel\circ\over{=}}\rangle \in \mathcal{BB}84^\lambda}} \Pr[\mathcal{E}_{good} \wedge \mathcal{E}^{j_1,j_2,b,|{\buildrel\circ\over{=}}\rangle}] \qquad \text{By Lemma 2,}$$

$$= \frac{1}{\binom{\lambda}{f(\lambda)}} \sum_{\substack{b \in \{0,1\} \\ |{\buildrel\circ\over{=}}\rangle \in \mathcal{BB}84^\lambda \\ |j_1|,|j_2| \leq \eta\lambda}} \sum_{\substack{\mathcal{I} \in \binom{[\lambda]}{f(\lambda)} \\ \text{s.t} \\ \mathcal{I} \cap (j_1 \cup j_2) = \varnothing}} \Pr[\mathcal{E}_{good} \wedge \mathcal{E}^{j_1,j_2,b,|{\buildrel\circ\over{=}}\rangle}]. \qquad (3)$$

For any $j_1, j_2 \subseteq [\lambda]$ such that $|j_1|, |j_2| \leq \eta\lambda$, we denote by C_{j_1,j_2}, the number of subsets $\mathcal{I} \in \binom{[\lambda]}{f(\lambda)}$ such that $\mathcal{I} \cap (j_1 \cup j_2) = \varnothing$. Since $|j_1 \cup j_2| \leq 2\eta\lambda$, C_{j_1,j_2} is at least $\binom{\lambda-2\eta\lambda}{f(\lambda)}$. Plugging this bound on C_{j_1,j_2}, as well as Eq. (2), to Eq. (3)

achieves the required bound:

$$\Pr[\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lambda) = 1]$$

$$\geq \frac{1}{\binom{\lambda}{f(\lambda)}} \sum_{\substack{b \in \{0,1\} \\ |\underline{\circ}\rangle \in \mathcal{BB}84^\lambda \\ |j_1|,|j_2| \leq \eta\lambda}} C_{j_1,j_2} \Pr[\mathcal{E}_{good} \wedge \mathcal{E}^{j_1,j_2,b,|\underline{\circ}\rangle}]$$

$$\geq \frac{1}{\binom{\lambda}{f(\lambda)}} \sum_{\substack{b \in \{0,1\} \\ |\underline{\circ}\rangle \in \mathcal{BB}84^\lambda \\ |j_1|,|j_2| \leq \eta\lambda}} \binom{\lambda - 2\eta\lambda}{f(\lambda)} \Pr[\mathcal{E}_{good} \wedge \mathcal{E}^{j_1,j_2,b,|\underline{\circ}\rangle}]$$

$$= \frac{\binom{\lambda - 2\eta\lambda}{f(\lambda)}}{\binom{\lambda}{f(\lambda)}} \Pr[\biguplus_{\substack{b \in \{0,1\} \\ |\underline{\circ}\rangle \in \mathcal{BB}84^\lambda \\ |j_1|,|j_2| \leq \eta\lambda}} \mathcal{E}_{good} \wedge \mathcal{E}^{j_1,j_2,b,|\underline{\circ}\rangle}]$$

$$= \frac{\binom{\lambda - 2\eta\lambda}{f(\lambda)}}{\binom{\lambda}{f(\lambda)}} \Pr[\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^\eta}(\lambda) = 1] \qquad \text{By Eq. (2)}$$

$$= \frac{\binom{\lambda - 2\eta\lambda}{f(\lambda)}}{\binom{\lambda}{f(\lambda)}} W_\eta(\lambda) \qquad\qquad\qquad \text{By definition of } W_\eta(\lambda)$$

$$= \frac{\frac{(\lambda - 2\eta\lambda)!}{(\lambda - 2\eta\lambda - f(\lambda))!}}{\frac{\lambda!}{(\lambda - f(\lambda))!}} W_\eta(\lambda)$$

$$= \frac{\prod_{i=1}^{f(\lambda)} (\lambda - 2\eta\lambda - f(\lambda) + i)}{\prod_{i=1}^{f(\lambda)} (\lambda - f(\lambda) + i)} W_\eta(\lambda)$$

$$= \prod_{i=1}^{f(\lambda)} \left(1 - \frac{2\eta\lambda}{\lambda - f(\lambda) + i}\right) W_\eta(\lambda)$$

$$\geq \left(1 - \frac{2\eta\lambda}{\lambda - f(\lambda)}\right)^{f(\lambda)} W_\eta(\lambda)$$

which completes the proof of Lemma 1. □

5.4 Choosing an Appropriate Function for the Reduction

We achieve an exponential bound on $\Pr[\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda)]$ by instantiating Lemma 1 with an appropriate f as follows.

Proposition 7. *Let $0 \leq \epsilon < 1$, and $\eta \equiv \epsilon \frac{1-\alpha}{2}$. For every computationally unbounded $\mathcal{A}$ making $q(\lambda)$ queries to $\mathcal{VR}_k$, there exists λ_0 such that for all $\lambda > \lambda_0$*

$$\Pr[\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda) = 1] \leq \left(\frac{1}{k^z}\right)^\lambda k\hat{q}(\lambda),$$

where $k \equiv \frac{(1+\alpha)-\epsilon(1-\alpha)}{2\alpha}$, $z \equiv \frac{1-\epsilon}{1+\epsilon}$, $\hat{q}(\lambda) = 2\binom{q(\lambda)+4}{2}$, $\alpha = \cos^2(\frac{\pi}{8})$, and $\mathsf{Forge}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}$ is the game described in Definition 7.

Proof. Let $\mathcal{A}$ be as described in Proposition 7, and denote

$$W_\eta(\lambda) \equiv \Pr[\mathsf{Forge}_{\mathcal{A},\mathsf{CTMAC}^\eta}^{\text{1-token-gen},\mathcal{VR}^{\eta*}}(\lambda) = 1].$$

Let $f(\lambda) \equiv \lfloor z\lambda \rfloor$ (notice $0 < z \leq 1$, for any $0 \leq \epsilon < 1$), then by Lemma 1, there exists an adversary $\mathcal{B}$ that makes $q(\lambda) + 2$ queries to $\mathcal{VR}$ for which

$$\Pr[\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lfloor z\lambda \rfloor)] \geq \left(1 - \frac{2\eta\lambda}{\lambda - \lfloor z\lambda \rfloor}\right)^{\lfloor z\lambda \rfloor} W_\eta(\lambda). \tag{4}$$

Recall that by Proposition 6, for every computationally unbounded $\mathcal{B}$ making at most $q(\lambda) + 2$ queries to $\mathcal{VR}$ there exists λ_0 such that for all $\lambda > \lambda_0$,

$$\Pr[\mathsf{Forge}_{\mathcal{B},\mathsf{CTMAC}}^{\text{1-token-gen},\mathcal{VR}^*}(\lfloor z\lambda \rfloor) = 1] \leq 2\alpha^{\lfloor z\lambda \rfloor}\binom{q(\lambda)+4}{2} = \alpha^{\lfloor z\lambda \rfloor}\hat{q}(\lambda),$$

Combined with Eq. (4), we get $\left(1 - \frac{2\eta\lambda}{\lambda-\lfloor z\lambda \rfloor}\right)^{\lfloor z\lambda \rfloor} W_\eta(\lambda) \leq \alpha^{\lfloor z\lambda \rfloor}\hat{q}(\lambda)$. Moreover $\left(1 - \frac{2\eta\lambda}{\lambda-\lfloor z\lambda \rfloor}\right) \geq 0$, since $1 - \frac{\lfloor z\lambda \rfloor}{\lambda} \geq 1 - \frac{z\lambda}{\lambda} = 1 - z = \frac{2}{1+\epsilon}\epsilon \geq \epsilon > \epsilon(1-\alpha) = 2\eta$. Hence,

$$W_\eta(\lambda) \leq \left(\frac{\alpha}{\left(1 - \frac{2\eta}{1-\frac{\lfloor z\lambda \rfloor}{\lambda}}\right)}\right)^{\lfloor z\lambda \rfloor}\hat{q}(\lambda) \qquad \leq \left(\frac{\alpha}{\left(1 - \frac{2\eta}{1-\frac{\lfloor z\lambda \rfloor}{\lambda}}\right)}\right)^{z\lambda-1}\hat{q}(\lambda)$$

$$\leq \left(\frac{\alpha}{\left(1 - \frac{2\eta}{1-\frac{z\lambda}{\lambda}}\right)}\right)^{z\lambda-1}\hat{q}(\lambda) \qquad \overset{\eta=\epsilon\frac{1-\alpha}{2}}{=} \left(\frac{\alpha}{(1 - \frac{2\epsilon\frac{1-\alpha}{2}}{1-z})}\right)^{z\lambda-1}\hat{q}(\lambda)$$

$$\overset{z=\frac{1-\epsilon}{1+\epsilon}}{=} \left(\frac{\alpha}{1 - \left(\frac{2\epsilon\frac{1-\alpha}{2}}{1-\frac{1-\epsilon}{1+\epsilon}}\right)}\right)^{z\lambda-1}\hat{q}(\lambda) \qquad = \left(\frac{\alpha}{1 - \frac{(1-\alpha)(1+\epsilon)}{2}}\right)^{z\lambda-1}\hat{q}(\lambda)$$

$$= \left(\frac{1}{\frac{2-(1-\alpha)(1+\epsilon)}{2\alpha}}\right)^{z\lambda-1}\hat{q}(\lambda) \qquad = \left(\frac{1}{\frac{(1+\alpha)-\epsilon(1-\alpha)}{2\alpha}}\right)^{z\lambda-1}\hat{q}(\lambda)$$

$$= \left(\frac{1}{k}\right)^{z\lambda-1}\hat{q}(\lambda) \qquad = \left(\frac{1}{k^z}\right)^\lambda k\hat{q}(\lambda).$$

The second inequality holds since $\left(\frac{\alpha}{\left(1-\frac{2\eta}{1-\frac{\lfloor z\lambda \rfloor}{\lambda}}\right)}\right) \leq \left(\frac{\alpha}{\left(1-\frac{2\eta}{1-\frac{z\lambda}{\lambda}}\right)}\right) = \frac{1}{k} \leq 1$ and $\lfloor z\lambda \rfloor \geq z\lambda - 1$. $\qquad\square$

Note that for every $0 \leq \epsilon < 1$, $k > 1$ and $z > 0$. Proposition 7 thus gives an inverse exponential bound. The bound gets stronger as ϵ decreases. Indeed, when $\epsilon = 0$, we get the strongest bound $\left(\frac{2\alpha}{1+\alpha}\right)^{\lambda-1} \hat{q}(\lambda)$. However, as $\epsilon \to 1$, then $k \to 1$, $z \to 0$ and the bound $\left(\frac{1}{k^z}\right)^{\lambda} k\hat{q}(\lambda) \to \hat{q}(\lambda)$, thus becoming a pointless bound.

The bound above holds for adversaries making *exactly* $q(\lambda)$ queries to $\mathcal{VR}_k^{\eta}$. As in Section 5.2, this implies that the same bound holds for any $\mathcal{A}$ making *at most* $q(\lambda)$ queries. Combining this with Corollary 3, proves Theorem 4.

6 Applications

6.1 One-Time-Memories From Stateless Hardware

In this section, we resolve an open problem presented in [BGZ21]. In [BGZ21], a construction of a primitive called one-time-memories, similar to non-interactive oblivious transfer, was suggested to be possible when assuming a trusted setup of stateless hardware.[19] Although their terminology differs, in their protocol, they used what we denote here as CTMAC. However, the authors did not provide a full security proof for their construction. For completeness, we repeat their construction, and show that its security is implied immediately by CTMAC$^{\eta}$ being Unforgeable$^{\text{1-token-gen,verify}}$, for $0 \leq \eta < \frac{1-\alpha}{2}$ (see Theorem 2).

One-time-memories (OTM) is an ideal modeling of one out of two non-interactive oblivious transfer, as described in Algorithm 9. This ideal functionality allows one machine, called a transmitter, to communicate one, and only one, of two possible messages s_0, s_1 to another machine called the receiver, with the transmitter completely oblivious to which of the messages was requested by the receiver. It is emphasized that one-time-memories are not realizable in the standard model, even in a quantum setting [GKR08, BGS13].

The functionality of stateless hardware is described in Algorithm 10. This ideal functionality allows a transmitter to create an entity computing a stateless program of its choice, ensuring that the receiver can only query the program in a black-box manner. Although the environment and algorithm discussed in the security notion are unbounded, we stress that $\mathcal{F}_{\text{wrap}}$ may only be queried a polynomial number of times, and may only be queried classically. A protocol that uses $\mathcal{F}_{\text{wrap}}$ is said to be in the stateless-hardware-model.

The main question explored in [BGZ21] is: can OTM be implemented in the stateless hardware model?

---

[19] [BGZ21] uses the term "stateless hardware tokens", as this usage of the term "tokens" conflicts with our own, we refrain from using it.

Algorithm 9 Ideal Functionality $\mathcal{F}_{\mathsf{OTM}}$

Create: Upon input (s_0, s_1) from the transmitter, with $s_0, s_1 \in \{0, 1\}$, send create to the receiver and store (s_0, s_1).

Execute: Upon input $m \in \{0, 1\}$ from the receiver, send s_m to the receiver. Delete the instance (s_0, s_1).

Algorithm 10 Ideal Functionality $\mathcal{F}_{\mathsf{wrap}}$

The functionality is parameterized by a polynomial $p(\cdot)$, and an implicit security parameter λ.

Create: Upon input (create, M) from the transmitter, where M is a Turing machine, send create to the receiver and store M.

Execute: Upon input (run, msg) from the receiver, execute $M(msg)$ for at most $p(\lambda)$ steps, and let out be the response. Let $out := \perp$ if M does not halt in $p(\lambda)$ steps. Send out to the receiver.

They gave a (partial) positive answer:

Theorem 5 ([BGZ21, Main Theorem])**.** *There exists a protocol $\mathcal{P}$ in the stateless-hardware-model, which is based on conjugate coding, and implements the OTM functionality with statistical security in the universal composability framework against a corrupted receiver making at most $c\lambda$ queries to the stateless hardware for any $c < 0.114$.*

Note that in their construction, the adversary can only perform some bounded number of stateless hardware queries. In an earlier version of their manuscript, a stronger result was claimed, where the receiver could make a polynomial number of stateless hardware queries. Unfortunately, their early work [BGZ18] "was withdrawn due to an error in the main security proof", and the superseding work ([BGZ21]) reproduced the same result except for the limitation on the linear number of queries. Additionally, an open question was left unanswered as to whether noise tolerance could be introduced to the construction.

Here, we complete their work by proving the stronger result and solving the open problem by using CTMAC^{η} (instead of CTMAC).

The protocol $\mathcal{P}^{\eta}$ described in [BGZ21] is reformulated in Algorithm 11, replacing many of the details with the formalism of the scheme CTMAC^{η}. We defer from the original construction by using CTMAC^{η} instead of the plain noise-sensitive CTMAC. The correctness of the protocol in the noise model $\mathsf{Noise}(2c\eta)$ for any $0 \le c < 1$ is trivial.

The security discussed in [BGZ21, Appendix A] is that of universally composable (UC) security for the sender, as the framework of universally composable security, allows for easy lifting from one-time-memories to that of one-time-programs. We will not repeat the definition for universally com-

---

Algorithm 11 $\mathcal{P}^\eta$: an OTM protocol

$\mathcal{T}$ Input: $s_0, s_1 \in \{0, 1\}$

$\mathcal{R}$ Input: $m \in \{0, 1\}$

1: The transmitter $\mathcal{T}$ calls $k \leftarrow$ CTMAC$^\eta$.key-gen(1^λ) and $|\overset{\circ}{\unicode{x2291}}\rangle \leftarrow$ CTMAC$^\eta$.token-gen$_k$.

2: $\mathcal{T}$ prepares the program $M(s_0, s_1, k)$, as described in Algorithm 12.

3: $\mathcal{T}$ sends (create, M) to the functionality $\mathcal{F}_{\text{wrap}}$, and sends the token $|\overset{\circ}{\unicode{x2291}}\rangle$ to $\mathcal{R}$.

4: $\mathcal{R}$ runs $\sigma \leftarrow$ CTMAC$^\eta$.sign$(|\overset{\circ}{\unicode{x2291}}\rangle, m)$ and sends (run, σ) to $\mathcal{F}_{\text{wrap}}$, receiving *out*.

Algorithm 12 $\mathcal{M}$:Program for Stateless Hardware

Hard-coded values: $s_0, s_1 \in \{0, 1\}, k$.

Inputs: $m \in \{0, 1\}, \sigma$, where m is the evaluator's choice bit, and σ is the proclaimed signature.

1: **if** CTMAC$^\eta$.verify$_k(m, \sigma) = 1$ **then** Return s_m.

2: **else** Return $\perp$.

3: **end if**

---

posable security here, but refer the reader to [Unr10] for universal composability in a quantum setting.

The main claim used to prove Theorem 5 is the following:

Theorem 6 ([BGZ21, Theorem 3.2]). *For $\eta = 0$, given a single copy of $|\overset{\circ}{\unicode{x2291}}\rangle$ generated by the transmitter in $\mathcal{P}^\eta$, and the ability to make r (adaptive) queries to the stateless hardware, the probability that an unbounded quantum adversary can force the stateless hardware program M to output both bits s_0 and s_1 scales as $O(2^{2r-0.228\lambda})$.*

Notice that the ability to force the token to output both bits corresponds to the adversary submitting σ_0, σ_1 such that the former is a signature for 0, and the latter is a signature for 1. Because of this, Theorem 4 easily implies an improvement of Theorem 6:

Theorem 7. *For any $0 \le \eta < \frac{1-\alpha}{2}$, where $\alpha = \cos^2(\frac{\pi}{8})$, given a single copy of $|\overset{\circ}{\unicode{x2291}}\rangle$ generated by the transmitter, and the ability to make polynomially many (adaptive) queries to the stateless hardware, the probability that an unbounded quantum adversary can force the stateless hardware program M to output both bits s_0 and s_1 is negligible.*

Replacing Theorem 6 with Theorem 7, results in the following strengthening of Theorem 5:

Theorem 8. *The protocol $\mathcal{P}^{0.07}$ in the stateless hardware model, which is based on conjugate coding, implements the OTM functionality with statistical security in the universal composability framework against a corrupted*

receiver making a polynomial number of queries. Moreover, $\mathcal{P}^{0.07}$ is tolerant to up to 14% noise (Definition 4).

We repeat the argument in [BGZ21] for completeness and refer the reader to the original work for details regarding the security definitions.

Sketch proof. It is relatively straightforward to construct a simulator $\mathcal{S}^{\mathcal{R}^*}$ in an ideal world (having access to the functionality $\mathcal{F}_{\mathsf{OTM}}$) for any malicious receiver $\mathcal{R}^*$: $\mathcal{S}^{\mathcal{R}^*}$ will simulate $\mathcal{R}^*$, and also simulate by itself the functionality of $\mathcal{F}_{\mathsf{wrap}}$, but with no actual inputs. Instead, the first time a signature is successfully submitted for a bit b, $\mathcal{S}^{\mathcal{R}^*}$ will call $\mathcal{F}_{\mathsf{OTM}}$ to recover s_b. $\mathcal{S}^{\mathcal{R}^*}$ will keep s_b and answer the same in any subsequent submission of a successful signature for b. If no successful signatures for $1-b$ are submitted, this will result in a perfectly indistinguishable view of the transcript. By Theorem 7, this occurs with an overwhelming probability. $\qquad\square$

It is worth noting that even if instantiated with a TMAC that is secure even against superposition attacks, this would not result in allowing quantum queries to the stateless hardware. In a model where quantum superposition queries to the hardware are allowed, OTM could not be UC-realized to be secure with statistical security [BGZ21, Section 4.1].

Another similar result is covered in [CGLZ19], not only constructing one-time-memories and one-time-programs from stateless hardware, but also providing a general transformation of stateful oracles to stateless oracles via a black box usage of Unforgeable[1-token-gen,verify] TMAC,[20] with various applications, among those the construction of cryptographic disposable backdoors. The results, while very similar, are not directly comparable due to differences in the models discussed. Regardless, instantiating the construction in [CGLZ19] with CTMAC^η also results in a conjugate coding-based noise-tolerant construction for OTM in the corresponding model.

6.2 Quantum Money From TMAC

As an application of TMAC, some of the results in [BS17] are restated, showing how a TMAC scheme can be used to construct a classical verification private money scheme. In addition, the theoretical implications of that construction on TMAC schemes are also reviewed.

6.2.1 Classically Verifiable Private Quantum Money

Quantum money was first proposed in [Wie83] and has since been studied extensively. Here, we only consider the private variant and also restrict ourselves to schemes where verification of the bill is done by an interactive protocol where the bank's side of the protocol is classical.

---

[20]The terminology used in [CGLZ19] refers to those as disposable mac, or DMAC.

Definition 10. *A Classically Verifiable PRivate quantum Money (CVPRM) scheme consists of two* QPT *algorithms:* key-gen *and* mint, *and an interactive protocol* verify *fulfilling the following:*

1. *On input 1^λ, where λ is the security parameter, the algorithm* key-gen *outputs a classical key k.*

2. mint_k *produces a quantum state $|\$\rangle$, which we refer to as the bill.*

3. $\mathsf{verify}_k(|\$\rangle)$ *is an interactive protocol with polynomially many rounds between the bank which has access to k, and the user who has access to the state $|\$\rangle$. Communication between the sides is only done classically. The bank's algorithm is* PPT, *and the user's algorithm is* QPT.

We say a classical verification private quantum money scheme is correct if for every k in the range of key-gen:

$$\Pr[\mathsf{verify}_k(\mathsf{mint}(k)) = 1] = 1.$$

Definition 11. *A CVPRM Π is* Unforgeable *if for every* QPT *adversary $\mathcal{A}$*

$$\Pr[\mathsf{MONEY\text{-}FORGE}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{negl}(\lambda),$$

if the same holds even for computationally unbounded adversaries (minted only polynomially many bills), the CVPRM Π *is said to be* Unconditionally Unforgeable

where $\mathsf{MONEY\text{-}FORGE}_{\mathcal{A},\Pi}(\lambda)$ *is the security game described in Game 13.*

Game 13 Money Forge Game $\mathsf{MONEY\text{-}FORGE}_{\mathcal{A},\Pi}(\lambda)$:

1: $\mathcal{C}$ generates a key $k \leftarrow \mathsf{key\text{-}gen}(1^\lambda)$.
2: $\mathcal{A}$ gets oracle access to mint_k and verify_k. Let w be the number of successful verifications and ℓ the number of times that mint was called by the adversary.

The value of the game $\mathsf{MONEY\text{-}FORGE}_{\mathcal{A},\Pi}(\lambda) = 1$, i.e., the adversary wins if and only if $w > \ell$.

6.2.2 Classically Verifiable Quantum Money From TMAC

A TMAC scheme could be used to construct a CVPRM scheme: let TM be a TMAC scheme.

A money scheme PM is constructed as follows: The bank runs $\mathsf{TM.key\text{-}gen}(1^\lambda)$ to produce a key k. In order to mint a bill $|\$\rangle$ to a user, the bank uses $\mathsf{TM.token\text{-}gen}_k$. When the user approaches the bank to verify a bill, the bank will pick a document $m \in \{0,1\}^\lambda$ at random and request the user

to sign it with $\mathsf{TM.sign}_{|\$\rangle}(m)$. In order to verify, the bank will then run $\mathsf{TM.verify}_k(m, \sigma)$ on the signature supplied by the customer.

Proposition 8. *If* TM *is an* $\mathsf{Unforgeable}^{\mathsf{token\text{-}gen,verify}}$[21] TMAC *scheme, then* PM *is an unforgeable* CVPRM *scheme. Furthermore, an analogous result holds against unbounded adversaries.*

Proof. Correctness is immediate. For unforgeability, assume an adversary was minted $r \in \mathsf{poly}(\lambda)$ bills, so r random documents have been generated. The probability for a collision between two random documents is less than $\frac{r^2}{2^\lambda}$, which is negligible if r is polynomial in λ. Hence, except with a negligible probability, all documents are distinct, so if the adversary is able to generate w bills of which at least $r + 1$ bills pass verification, this means that the adversary is able to forge w distinct signed documents, of which at least $r + 1$ signatures pass verification, contradicting the security of TM. □

While $\widehat{\mathsf{CTMAC}}^0_{\mathsf{Poly\text{-}times}}$ is inspired by the classical verification variant of Wiesner's money [MVW12], the CVPRM resulting by applying the above construction is distinct from it: the bills in the resulting scheme can be thought of as a two-dimensional array of qubits, where each bit of the challenge dictates the measurement of an entire row, rather than of a single qubit.

6.2.3 Trade-Off Theorem for Quantum Money and TMAC

In Section 5, we proved that CTMAC (Algorithm 4) is $\mathsf{Unconditionally}$ $\mathsf{Unforgeable}^{\mathsf{1\text{-}token\text{-}gen,verify}}$. In order to construct an $\mathsf{Unforgeable}^{\mathsf{token\text{-}gen,verify}}$, computational assumptions were used, as described in Section 3. A natural question is then whether a TMAC can be constructed to be $\mathsf{Unconditionally}$ $\mathsf{Unforgeable}^{\mathsf{token\text{-}gen,verify}}$. Based on a result by Aaronson Theorem 9, it is shown that this is impossible. In fact, even an $\mathsf{Unconditionally\ Unforgeable}^{\mathsf{token\text{-}gen}}$, TMAC is impossible to achieve.

Theorem 9 (Trade-off Theorem for Quantum Money, [Aar20, Theorem 8]). *Given any private-key quantum money scheme,*[22] *with d-qubit bills and an m-bit secret key held by the bank, a counterfeiter can produce additional bills which pass verification with $1 - o(1)$ probability, given $\widetilde{O}(dm^4)$ legitimate bills and $\exp(d, m)$ computation time. No queries to the bank are needed to produce these bills.*

The construction of CVPRM given in Section 6.2.2 implies that the same holds for TMAC; otherwise, by Proposition 8, it could be used to construct unconditionally secure quantum money. This proves Theorem 10.

---

[21] In this context, unforgeability in light of a signing oracle is not required.

[22] And in particular a CVPRM.

Theorem 10 (Trade-off Theorem for TMAC)**.** *Given any* TMAC *scheme, with d-qubit tokens and an m-bit secret key, a counterfeiter can produce additional tokens which would produce signatures that pass verification with $1 - o(1)$ probability, given $\widetilde{O}(dm^4)$ legitimate tokens and $exp(d, m)$ computation time. No queries to the verification oracle are needed to produce these tokens.*

6.2.4 Temporarily Memory-Dependent Non-Interactive Quantum Money

While the construction above (Section 6.2.2) is interesting from a theoretical perspective, it does not currently offer anything not already guaranteed by other existing private money schemes based on simple states [MVW12, PYJ$^+$12, Gav12]. On the other hand, it clearly does not make full use of the power of the TMAC: A TMAC guarantees unforgeability as long as the documents are not repeated, regardless of them being chosen at random.

Using the above property, [BS17] devised an alternative non-interactive verification procedure at the expense of losing the statelessness of the scheme: If Alice holds a quantum bill, one thing she can do is spend it the usual way. However, an alternative thing she can do with the bill is to use it to sign a document. Such a signature will necessarily consume the bill and, thus, it can be used as proof that Alice has burned her bill, essentially creating a classical check.

To prevent double spending, the bank would only accept checks that sign a document consisting of the current time and date. The bank would have to keep a database of the cashed checks to prevent double-spending; however, this database could be time-limited. We could keep a short "time frame" during which the bank would need to keep a database of checks received, but it could clear that database at the end of a time frame. The bank can choose simply not to accept any checks whose date has "expired", i.e., the time stamp on the check does not fall within the current time frame.

If there are multiple bank branches, Alice could try to cash the check twice at different bank branches. In order to prevent this, the bank can require the signed document to list the bank branch number as well.

7 Discussion and Open Questions

In this work, we proved that post-quantum one-way functions imply TMAC schemes. It is natural to ask if the converse holds in the quantum setting.

Open Problem 1. *Does* TMAC *imply post-quantum one-way functions?*

Note that in the classical setting, we know that MACs exist if and only if one-way functions exist. A standard construction of one-way functions from MAC [Gol04, page 592, Exercize 7] is based on a clever engineering of

the classical randomness involved in the signing algorithm. It is not clear whether this approach can be applied in the quantum setting.

We presented a simple construction for an $\mathsf{Unforgeable}^{\mathsf{token\text{-}gen,verify,\widetilde{sign}}}$ TMAC based on simple tensor product states. However, the simplicity of the construction came at the expense of some of the stronger properties that the construction in [BS17, Appendix B.1, Section 6] achieves. One such property is security against adversaries that have quantum access to the verification oracle. An interesting question would be to ask whether a scheme based on BB84 states, or similar "simple" states, could satisfy this stronger notion.

Open Problem 2. *Is there a* TMAC *construction based on simple states that satisfies unforgeability, even against adversaries with quantum access to the verification oracle?*

Note that an $\mathsf{Unforgeable}^{\mathsf{token\text{-}gen,|verify\rangle,\widetilde{sign}}}$ (see Appendix C.1) TMAC, i.e., a TMAC secure against adversaries with quantum access to the verification oracle, could potentially result in a secure tokenized signature scheme if augmented with some form of obfuscation [BGI$^+$12]. This would immediately give rise to a public quantum money construction. A public scheme based on product states seems to be a difficult task even with a quantum-queries secure TMAC scheme and obfuscation. A result on quantum state restoration [FGH$^+$10] proves this is impossible to do with a rank-1 projection as the verification procedure. However, it is not known if a similar result applies for higher dimension projections.

Another interesting feature that can be added to a TMAC is to have classical token-generation, meaning, Alice instead of creating a quantum signing token and giving it to Bob would run an interactive protocol with Bob using classical communication, such that an honest Bob would have a quantum signing token at the end of the protocol.

Open Problem 3. *Is there a* TMAC *construction that allows classical token-generation?*

A natural candidate for such a scheme is a simple variant on semi-quantum money [RS19], which is a money scheme with both classical verification and classical minting. In analogy to how CTMAC signs the single bit message b by "responding" to the challenge b^λ in a Wiesner's money scheme, we can devise a TMAC scheme that signs the single bit message b by "responding" to the challenge b^λ in semi-quantum money. Security against an adversary with no access to verification follows from the security of the money scheme, but we have not been unable to either prove or disprove the security against an adversary that has access to verification.

Acknowledgments

References

[Aar20] S. Aaronson. Shadow Tomography of Quantum States. *SIAM J. Comput.*, 49(5), 2020, arXiv: `1711.01053`.

[AC13] S. Aaronson and P. F. Christiano. Quantum Money from Hidden Subspaces. *Theory Comput.*, 9:349–401, 2013, arXiv: `1203.4740`.

[AGKZ20] R. Amos, M. Georgiou, A. Kiayias, and M. Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, editors, *Proccedings of the Annual ACM SIGACT Symposium on Theory of Computing,*, pages 255–268. ACM, 2020, Cryptology ePrint Archive: `Report 2020/107`.

[AM17] G. Alagic and C. Majenz. Quantum Non-malleability and Authentication. In J. Katz and H. Shacham, editors, *Advances in Cryptology - CRYPTO*, volume 10402 of *Lecture Notes in Computer Science*, pages 310–341. Springer, 2017, arXiv: `1610.04214`.

[BCG+02] H. Barnum, C. Crépeau, D. Gottesman, A. D. Smith, and A. Tapp. Authentication of Quantum Messages. In *43rd Symposium on Foundations of Computer Science (FOCS*, pages 449–458. IEEE Computer Society, 2002, arXiv: `quant-ph/0205128`.

[BGI+12] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

[BGS13] A. Broadbent, G. Gutoski, and D. Stebila. Quantum one-time programs. *IACR Cryptol. ePrint Arch.*, 2013:343, 2013, arXiv: `1211.1080`.

[BGZ18] A. Broadbent, S. Gharibian, and H.-S. Zhou. Quantum One-Time Memories from Stateless Hardware, 2018, arXiv: `1511.01363`. Withdrawn.

[BGZ21]   A. Broadbent, S. Gharibian, and H. Zhou. Towards Quantum One-Time Memories from Stateless Hardware. *Quantum*, 5:429, 2021, arXiv: `1810.05226`.

[BI20]   A. Broadbent and R. Islam. Quantum Encryption with Certified Deletion. In R. Pass and K. Pietrzak, editors, *Theory of Cryptography TCC*, volume 12552 of *Lecture Notes in Computer Science*, pages 92–122. Springer, 2020, arXiv: `1910.03551`.

[BS17]   S. Ben-David and O. Sattath. Quantum Tokens for Digital Signatures. *IACR Cryptol. ePrint Arch.*, page 94, 2017.

[BV04]   S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BW16]   A. Broadbent and E. Wainewright. Efficient Simulation for Quantum Message Authentication. In A. C. A. Nascimento and P. S. L. M. Barreto, editors, *Information Theoretic Security*, volume 10015 of *Lecture Notes in Computer Science*, pages 72–91, 2016, arXiv: `1607.03075`.

[BZ13]   D. Boneh and M. Zhandry. Quantum-Secure Message Authentication Codes. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 592–608. Springer, 2013, Cryptology ePrint Archive: `Report 2012/606`.

[CGLZ19] K. Chung, M. Georgiou, C. Lai, and V. Zikas. Cryptography with Disposable Backdoors. *Cryptogr.*, 3(3):22, 2019, Cryptology ePrint Archive: `Report 2018/352`.

[CLLZ21] A. Coladangelo, J. Liu, Q. Liu, and M. Zhandry. Hidden Cosets and Applications to Unclonable Cryptography. *IACR Cryptol. ePrint Arch.*, 2021:946, 2021, Cryptology ePrint Archive: `Report 2021/946`.

[CS20]   A. Coladangelo and O. Sattath. A Quantum Money Solution to the Blockchain Scalability Problem. *Quantum*, 4:297, July 2020, arXiv: `2002.11998`.

[De 02]   E. De Klerk. *Aspects of semidefinite programming: Interior point algorithms and selected applications*. Number 65 in Applied optimization, ISSN 1384-6485. Kluwer Academic Publishers, Netherlands, 2002.

[FGH+10] E. Farhi, D. Gosset, A. Hassidim, A. Lutomirski, D. Nagaj, and P. Shor. Quantum State Restoration and Single-Copy Tomog-

raphy for Ground States of Hamiltonians. *Phys. Rev. Lett.*, 105:190503, Nov 2010, arXiv: `0912.3823`.

[FGH+12] E. Farhi, D. Gosset, A. Hassidim, A. Lutomirski, and P. Shor. Quantum money from knots. In *Proceedings of the Innovations in Theoretical Computer Science Conference*, pages 276–289. ACM, ACM, 2012, arXiv: `1004.5127`.

[Gav12] D. Gavinsky. Quantum Money with Classical Verification. In *Proceedings of the Conference on Computational Complexity, CCC*, pages 42–52. IEEE Computer Society, 2012, arXiv: `1109.0372`.

[GB08] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[GKR08] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. One-Time Programs. In D. A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008.

[Gol04] O. Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications.* Cambridge University Press, 2004.

[GYZ17] S. Garg, H. Yuen, and M. Zhandry. New Security Notions and Feasibility Results for Authentication of Quantum Data. In J. Katz and H. Shacham, editors, *Advances in Cryptology - CRYPTO*, volume 10402 of *Lecture Notes in Computer Science*, pages 342–371. Springer, 2017, arXiv: `1607.07759`.

[HRP+06] P. A. Hiskett, D. Rosenberg, C. G. Peterson, R. J. Hughes, S. Nam, A. E. Lita, A. J. Miller, and J. E. Nordholt. Long-distance quantum key distribution in optical fibre. *New Journal of Physics*, 8(9):193–193, sep 2006.

[KL14] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014.

[KLH+15] B. Korzh, C. C. W. Lim, R. Houlmann, N. Gisin, M. J. Li, D. Nolan, B. Sanguinetti, R. Thew, and H. Zbinden. Provably secure and practical quantum key distribution over 307km of optical fibre. *Nature Photonics*, 9(3), 2015.

[KLLN16] M. Kaplan, G. Leurent, A. Leverrier, and M. Naya-Plasencia. Breaking Symmetric Cryptosystems Using Quantum Period

Finding. In M. Robshaw and J. Katz, editors, *Advances in Cryptology - CRYPTO*, volume 9815 of *Lecture Notes in Computer Science*, pages 207–237. Springer, 2016, arXiv: `1602.05973`.

[Lov03] L. Lovász. Semidefinite Programs and Combinatorial Optimization. In *Recent Advances in Algorithms and Combinatorics*, pages 137–194. Springer New York, 2003.

[Lut10] A. Lutomirski. An online attack against Wiesner's quantum money, 2010, arXiv: `1010.0256`.

[MS07] R. Mittal and M. Szegedy. Product Rules in Semidefinite Programming. In E. Csuhaj-Varjú and Z. Ésik, editors, *Fundamentals of Computation Theory*, volume 4639 of *Lecture Notes in Computer Science*, pages 435–445. Springer, 2007.

[MVW12] A. Molina, T. Vidick, and J. Watrous. Optimal Counterfeiting Attacks and Generalizations for Wiesner's Quantum Money. In K. I. et al., editor, *Theory of Quantum Computation, Communication, and Cryptography, TQC*, volume 7582 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2012, arXiv: `1202.4010`.

[NC11] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[PYJ+12] F. Pastawski, N. Y. Yao, L. Jiang, M. D. Lukin, and J. I. Cirac. Unforgeable noise-tolerant quantum tokens. *Proceedings of the National Academy of Sciences*, 109(40):16079–16082, 2012, arXiv: `1112.5456`.

[Rob21] B. Roberts. Security Analysis of Quantum Lightning. In *EUROCRYPT (2)*, volume 12697 of *Lecture Notes in Computer Science*, pages 562–567. Springer, 2021.

[RS19] R. Radian and O. Sattath. Semi-Quantum Money. In *Proceedings of the Conference on Advances in Financial Technologies*, pages 132–146. ACM, 2019, arXiv: `1908.08889`.

[Son14] F. Song. A Note on Quantum Security for Post-Quantum Cryptography. In M. Mosca, editor, *Post-Quantum Cryptography*, volume 8772 of *Lecture Notes in Computer Science*, pages 246–265. Springer, 2014, Cryptology ePrint Archive: `Report 2014/709`.

[SY17] F. Song and A. Yun. Quantum Security of NMAC and Related Constructions - PRF Domain Extension Against Quantum at-

tacks. In J. Katz and H. Shacham, editors, *Advances in Cryptology - CRYPTO*, volume 10402 of *Lecture Notes in Computer Science*, pages 283–309. Springer, 2017.

[Unr10] D. Unruh. Universally Composable Quantum Multi-party Computation. In H. Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 486–505. Springer, 2010, arXiv: 0910.2912.

[VB96] L. Vandenberghe and S. P. Boyd. Semidefinite Programming. *SIAM Rev.*, 38(1):49–95, 1996.

[WC81] M. N. Wegman and L. Carter. New Hash Functions and Their Use in Authentication and Set Equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

[Wic17] D. Wichs. Graduate Cryptography-Problem Set 1. Problem-Set, 2017.

[Wie83] S. Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983.

[Zha21] M. Zhandry. Quantum Lightning Never Strikes the Same State Twice. Or: Quantum Money from Cryptographic Assumptions. *J. Cryptol.*, 34(1):6, 2021, arXiv: 1711.02276.

A Expansion To a Full-Blown Scheme

A.1 Cryptographic Definitions

A.1.1 Universal One-Way Hash Functions

Definition 12 (universal one-way hash functions – UOWHF, adapted from [Gol04, Definition 6.4.18]). *Let $\ell : \mathbb{N} \to \mathbb{N}$. A collection of functions $\{h_s : \{0,1\}^* \to \{0,1\}^{\ell(|s|)}\}_{s \in \{0,1\}^*}$ is called universal one-way hashing (UOWHF) if there exists a probabilistic polynomial-time algorithm I so that the following holds:*

1. *For all sufficiently large λ and all s in the range of I, it holds that $|s| = \lambda$.*

2. *(efficient evaluation): There exists a polynomial-time algorithm that, given s and x, returns $h_s(x)$.*

3. *(hard-to-form designated collisions): For every QPT $\mathcal{A}_0$, outputting a quantum state $|S\rangle$, and a string x, and a QPT $\mathcal{A}$ receiving $|S\rangle$, as well as the description of a hash function s sampled by $I(1^\lambda)$, that outputs a string y:*
$$\Pr[h_s(x) = h_s(y) \, and \, x \neq y] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the randomness of $I, \mathcal{A}_0$ and $\mathcal{A}$.

Remark 1. The existence of post-quantum UOWHF's, for any polynomially bounded integer function ℓ is implied by the existence of post-quantum one-way functions ([Son14]).

A.1.2 Authenticated Encryption

An authenticated encryption is an encryption scheme that satisfies both chosen cipher-text indistinguishability security and authentication guarantees analogous to the unforgeability guarantees of MAC, i.e., an adversary cannot create a valid encryption for a document that it has not seen encryptions for. We first recall the definition of CCA security:

Definition 13 (Chosen-Cipher-Text Attack Security, [KL14, Definition 3.33]). *A private-key encryption scheme* $\Pi = (\mathsf{key\text{-}gen}, \mathsf{enc}, \mathsf{dec})$ *has indistinguishable encryptions under a chosen-cipher-text-attack, or is* CCA-*secure, if for all* PPT *adversaries* $\mathcal{A}$:

$$\Pr[\mathsf{PrivK}^{\mathsf{CCA}}{}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the probability is taken over all random coins used in the game.

Game 14 The CCA Indistinguishability Game $\mathsf{PrivK}^{\mathsf{CCA}}{}_{\mathcal{A},\Pi}(\lambda)$:

1: A random key k is generated by running $\mathsf{key\text{-}gen}(1^\lambda)$.
2: The adversary $\mathcal{A}$ is given input 1^λ and oracle access to enc_k and dec_k. It outputs a pair of messages (m_0, m_1) of the same length.
3: A uniform bit $b \in \{0, 1\}$ is chosen, and then a cipher-text $c \leftarrow \mathsf{enc}_k(m_b)$ is given to $\mathcal{A}$. We call c the challenge cipher-text.
4: The adversary $\mathcal{A}$ continues to have oracle access to enc_k and dec_k, but is not allowed to query the latter on the challenge cipher-text itself, Eventually, $\mathcal{A}$ outputs a bit b'.

The output of the game is defined to be 1 if $b = b'$, and 0 otherwise. If the output of the game is 1, we say that $\mathcal{A}$ succeeds.

As it is in the case of MAC, there are two variants of unforgeability: strong unforgeability where even a fresh distinct encryption for an already encrypted plain-text counts as forgery, as opposed to the weaker, albeit more common security notion where the plain-texts are required to be distinct. We require the strong variant, and so we modify the definition in [KL14] accordingly.

Definition 14 (Unforgeable Encryption Scheme, based on [KL14, Definition 4.17]). *A private-key encryption scheme* $\Pi = (\mathsf{key\text{-}gen}, \mathsf{enc}, \mathsf{dec})$ *is strongly unforgeable if for all* PPT *adversaries* $\mathcal{A}$:

$$\Pr[\mathsf{Enc\text{-}forge}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over all random coins used in the game.

Game 15 The Strong Unforgeable Encryption Game $\mathsf{Enc\text{-}forge}_{\mathcal{A},\Pi}(\lambda)$:

1: Run $\mathsf{key\text{-}gen}(1^\lambda)$ to obtain a key k.
2: The adversary $\mathcal{A}$ is given input 1^λ and access to an encryption oracle enc_k. The adversary outputs a cipher-text c.
3: Let $m \equiv \mathsf{dec}_k(c)$, and let Q denote the set of all responses to queries $\mathcal{A}$ made to the encryption oracle.

The output of the game is 1 if and only if (1) $m \neq \bot$ and (2) $c \notin Q$.

A strong authenticated encryption is an encryption scheme that fulfills both of the security notions discussed above:

Definition 15 (Strong Authenticated Encryption). *A private key encryption scheme is an authenticated encryption if it is* CCA*-secure and strongly unforgeable.*

Moreover, we call an encryption scheme a Post-Quantum Classical-Queries Strong Authenticated Encryption if it satisfies both of the security notions (CCA *indistinguishability and strong unforgeability), even against* QPT *adversaries.*

Next, we give a brief discussion regarding the assumptions required for post-quantum classical-queries strong authenticated encryption.

A standard construction for authenticated encryption, as detailed in [KL14, Theorem 4.9], is an *encrypt-then-sign* algorithm using a CPA-secure encryption scheme and an unforgeable strong MAC scheme. Careful inspection of the proof brought in [KL14] shows that the resulting authenticated encryption is actually a strong authenticated encryption, and also that quantum adversaries with classical access to the oracles, would not have any substantial advantage, as long as the MAC and encryption schemes are also secure against quantum adversaries with classical access to the oracles. A result in [BZ13] proves that post-quantum one-way functions imply the existence of a qCPA encryption scheme, and a qCMA MAC scheme, in which quantum adversaries are even allowed to perform quantum queries to the oracles, let alone classical queries. Such schemes thus suffice to construct a post-quantum classical-queries strong authenticated encryption. Hence, we have the following remark summarizing the discussion above.

Remark 2. If post-quantum one-way functions exist, so does a post-quantum classical-query strong authenticated encryption scheme.

A.2 Security Proof for the Full Scheme

In this section, we prove the lifting proposition, Proposition 1. Fig. 2 illustrates the structure of the expansion of CTMAC to a full blown scheme, $\widetilde{\mathsf{CTMAC}}_{\mathsf{Poly\text{-}times}}^{\eta}$.

Proof of Proposition 1. The proof is immediate by combining Lemmas 3 to 5 and 8, which we prove in this section. These lifting lemmas are achieved via a series of constructions, all of which preserve the tokens being BB84 states, perhaps appending them with classical data as well. □

Lemmas 3 to 5 are an adaptation of corresponding lemmas in [BS17, Appendix C], with few differences. One, unlike [BS17], we extend the results to take into account adversaries that have access to a verification oracle, and two, we only assume the existence of post-quantum one-way functions, instead of post-quantum collision resistant hash functions. Lastly the scheme in [BS17] was unforgeable even against adversaries with a signing oracle (see discussion in Appendix D), since it was a strong TMAC (see Appendix C.2). The scheme presented in this work is not such a priori, and hence we provide a general lift to unforgeability against adversaries with a signing oracle in Lemma 8.

A.2.1   Expansion to a Length-Restricted Scheme

We extend a 1-restricted scheme to an ℓ-restricted scheme (see Definition 2) for $\ell \in \mathsf{poly}(\lambda)$ using a standard construction. This is done by signing the ℓ bits of the document using ℓ independent instances of the 1-restricted scheme, each corresponding to a different bit of the document (see Algorithm 16):

Lemma 3. *There is a noise-tolerance preserving lift (Definition 4) of any 1-restricted* Unforgeable$^{\text{1-token-gen,verify}}$ TMAC *(see Definitions 2 and 5) to an ℓ-restricted* Unforgeable$^{\text{1-token-gen,verify}}$ TMAC*, where $\ell(\lambda) \in \mathsf{poly}(\lambda)$, as shown in Algorithm 16.*

Proof. Correctness follows from the correctness of the OT1 scheme, that is, the verification of a valid signature for m accepts if and only if all the ℓ verifications for the ℓ copies of the OT1 scheme accepts. Next, we show that the noise-tolerance property is preserved in the transformation from OT1 to OTL. Suppose OT1 is δ-noise-tolerant where $\delta \geq 0$ is some constant. We claim that OTL is also δ-noise-tolerant, i.e., correctness holds for OTL up to negligible error, in the noise model $\mathsf{Noise}(\delta)$ (see Definition 3). Let $m = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$ be arbitrary. Let $k \leftarrow \mathsf{OTL.key\text{-}gen}(1^\lambda)$ and $|\unicode{x1F4AE}\rangle \leftarrow \mathsf{OTL.token\text{-}gen}_k$ be a valid token with respective to the secret key k. The secret key k and $|\unicode{x1F4AE}\rangle$ can be viewed as a concatenation of ℓ OT1 secret keys and the respective tokens, i.e.,

$$k = k_1 \| \cdots \| k_\ell, \quad |\unicode{x1F4AE}\rangle = |\unicode{x1F4AE}_1\rangle \otimes \cdots \otimes |\unicode{x1F4AE}_\ell\rangle,$$

where $|\unicode{x1F4AE}_j\rangle$ is the jth OT1 token with respect to the OT1 secret key k_j. Similarly, let $|\unicode{x1F4AE}'\rangle$, and $|\unicode{x1F4AE}'_1\rangle \otimes \ldots \otimes |\unicode{x1F4AE}'_\ell\rangle$ denote the quantum state of the

E_0, E_1 (Algorithm 7) are

WEAK-DEL-IND secure (Proposition 9).

Post-Quantum One-Way Functions

Corollary 3 and Lemma 1

Propositions 3 and 6

CTMAC^η (Algorithm 5) is an 1-restricted

Unforgeable$^{\text{1-token-gen,verify}}$ TMAC.

Lemma 3

Remark 2

CTMAC^η_ℓ (Algorithm 16) is an ℓ-restricted

Unforgeable$^{\text{1-token-gen,verify}}$ TMAC.

Lemma 4

$\mathsf{CTMAC}^\eta_{\text{1-time}}$ (Algorithm 17) is an unrestricted

Unforgeable$^{\text{1-token-gen,verify}}$ TMAC.

Strong Authenticated Encryption

(Definition 15)

Lemma 5

$\mathsf{CTMAC}^\eta_{\text{Poly-times}}$ (Algorithm 18) is an unrestricted

Unforgeable$^{\text{token-gen,verify}}$ TMAC.

Lemma 8

$\widetilde{\mathsf{CTMAC}}^\eta_{\text{Poly-times}}$ (Algorithm 19) is an unrestricted

Unforgeable$^{\text{token-gen,verify,}\widetilde{\text{sign}}}$ TMAC.

Figure 2: The above diagram summarizes the expansion to a full blown scheme showing the different steps in our construction and the reductions in the security proofs. Each node represents a result, and two arrows arising from two different nodes pointing at the same destination node means that the results in the two nodes together imply the result in the destination node. For any $\eta \leq 0.07$, all the TMAC schemes in the diagram above are $2c\eta$-noise-tolerant, for every $0 \leq c < 1$. The bolded arrow is the main technical novelty of this work, and is expanded upon in Fig. 1.

56

Algorithm 16 OTL.

The resulting scheme obtained by instantiating OTL with the specific scheme CTMAC^η (Algorithm 5), is called CTMAC^η_ℓ.

Assumes: OT1 is a 1-restricted TMAC, and $\ell = \ell(\lambda) \in \mathsf{poly}(\lambda)$.

1: **procedure** key-gen(1^λ)
2:      **for** $i = 1, 2, \ldots, \ell$ **do**
3:          $k_i \leftarrow$ OT1.key-gen(1^λ)
4:      **end for**
5:      **Return** $(k_1, \ldots, k_\ell)$.
6: **end procedure**

1: **procedure** token-gen$_k$
2:      Interpret $k = (k_1, \ldots, k_\ell)$.
3:      **for** $i = 1, 2, \ldots, \ell$ **do**
4:          $|\text{⚖}_i\rangle \leftarrow$ OT1.token-gen$_{k_i}$.
5:      **end for**
6:      **Return** $(|\text{⚖}_1\rangle \otimes \ldots \otimes |\text{⚖}_\ell\rangle)$.
7: **end procedure**

1: **procedure** sign$_{|\text{⚖}\rangle}(m)$
2:      Interpret $|\text{⚖}\rangle = (|\text{⚖}_1\rangle \otimes \ldots \otimes |\text{⚖}_\ell\rangle)$ and $m = (m_1, \ldots, m_\ell)$.
3:      **for** $i = 1, 2, \ldots, \ell$ **do**
4:          $\sigma_i \leftarrow$ OT1.sign
5:      **end for**
6:      **Return** $(\sigma_1, \ldots, \sigma_\ell)$.
7: **end procedure**

1: **procedure** verify$_k(m, \sigma)$
2:      Interpret k as $(k_1, \ldots, k_\ell)$, m as $(m_1, \ldots, m_\ell)$ and σ as $(\sigma_1, \ldots, \sigma_\ell)$.
3:      **for** $i = 1, 2, \ldots, \ell$ **do**
4:          $v_i \leftarrow$ OT1.verify$_{k_i}(m_i, \sigma_i)$.
5:      **end for**
6:      **Return** $\bigwedge_i v_i$
7: **end procedure**

---

respective tokens after the corruption of some of its qubits due to noise. Let $\sigma \leftarrow \mathsf{OTL.sign}_{|\underline{\mathtt{o}}'\rangle}(m)$. Hence, $\sigma = \sigma_1 \| \cdots \| \sigma_\ell$, where $\sigma_j \leftarrow \mathsf{OT1.sign}_{|\underline{\mathtt{o}}'_j\rangle}(m_j)$, for every $j \in \ell$.

Let $Good$ be the event that $\mathsf{OTL.verify}_k(\sigma) = 1$. It is enough to show that

$$\Pr[Good] \geq 1 - \mathsf{negl}(\lambda),$$

where $\mathsf{negl}(\lambda)$ is some negligible function in the security parameter λ.

Note that by construction,

$$Good = \bigwedge_{j \in \ell} Good_j, \tag{5}$$

where for each $j \in [\ell]$, $Good_j$ denotes the event that in the OTL verification $\mathsf{OT1.verify}_{k_j}(\sigma_j) = 1$.

By the δ-noise tolerance of $\mathsf{OT1}$, for every $j \in [\ell]$,

$$\Pr[Good_j] \geq 1 - \widetilde{\mathsf{negl}(\lambda)}, \tag{6}$$

for some negligible function $\widetilde{\mathsf{negl}(\lambda)}$. Recalling Eqs. (5) and (6) and using the fact that for each of the ℓ tokens, $\mathsf{token\text{-}gen}$ and the respective $\mathsf{key\text{-}gen}$ were done independent of each other,

$$\Pr[Good] = \Pr[\bigwedge_{j \in [\ell]} Good_j] = \prod_{j \in \ell} \Pr[Good_j] \geq (1 - \widetilde{\mathsf{negl}(\lambda)})^{\ell(\lambda)} \geq 1 - \ell(\lambda) \cdot \widetilde{\mathsf{negl}(\lambda)}.$$

Since, $\ell(\lambda) \cdot \widetilde{\mathsf{negl}(\lambda)}$ is a negligible function, this concludes the proof regarding the noise tolerance.

We prove unforgeability as follows. Suppose there is an adversary $\mathcal{A}$ winning $\mathsf{Forge}_{\mathcal{A},\mathsf{OTL}}^{\text{1-token-gen,verify}}(\lambda)$ (see Game 2) with non-negligible probability $\epsilon(\lambda)$. In other words, $\mathcal{A}$ gets one signing token and submits $\{m_i, \sigma_i\}_w$ in the $\mathsf{Forge}^{\text{1-token-gen,verify}}(\lambda)$ game, where the documents m_i are of length ℓ, such that with a probability of $\epsilon(\lambda)$, there exists $m_i \neq m_j$, and verification (for all the ℓ bits) is successful for both documents. We will construct a new adversary $\mathcal{B}$ that attempts to win the $\mathsf{Forge}^{\text{1-token-gen,verify}}(\lambda)$ game against $\mathsf{OT1}$. $\mathcal{B}$ receives one signing token from the challenger. $\mathcal{B}$ then runs $\mathsf{OT1.key\text{-}gen}$ and $\mathsf{OT1.token\text{-}gen}$ $\ell - 1$ times to get additional $\ell - 1$ tokens. $\mathcal{B}$ shuffles these together, i.e., inserts the token from the challenger to an index, chosen uniformly at random from $[\ell]$, and simulates $\mathcal{A}$ to obtain $\{m_i, \sigma_i\}_w$. $\mathcal{B}$ would then extract from each of the w signed documents, the index of the document and signature corresponding to the token he received from the challenger. $\mathcal{B}$ can simulate every oracle query from $\mathcal{A}$ as follows. $\mathcal{B}$ queries the verification oracle for the $\mathsf{OT1}$ at the index corresponding to the challenge token and verifies the other indices by themselves. $\mathcal{B}$ returns 1 if and only if $\mathsf{OT1.verify}$ outputs 1, and all the other verifications $\mathcal{B}$ made were successful. Since the view of $\mathcal{A}$ is the same as in $\mathsf{Forge}_{\mathcal{A},\mathsf{OTL}}^{\text{1-token-gen,verify}}(\lambda)$, with

non-negligible probability $\epsilon(\lambda)$, *count* ≥ 2 in $\mathsf{Forge}_{\mathcal{A},\mathsf{OTL}}^{\text{1-token-gen,verify}}(\lambda)$, where *count* is the random variable counting the number of successful verifications, as defined in Game 2.

Conditioning on the event mentioned above, there exists $m_i \neq m_j$ such that both pass verification on all indices. Since $m_i \neq m_j$, they must differ on at least one of the bits. With a probability of at least $\frac{1}{\ell}$, this would be the same bit corresponding to the input token given to $\mathcal{B}$. Conditioning on the event that it is indeed the right index, $\mathcal{B}$ would have managed to get alleged signatures for two different documents with respect to $\mathsf{OT1}$, both of which would pass verification with a probability of at least $\epsilon(\lambda)$ (by the success guarantee of $\mathcal{A}$). Here, we also use the fact that the event of $\mathcal{A}$ winning the game against OTL is independent of the index in which the real token was inserted because $\mathcal{A}$ always sees the same distribution no matter how the real token was inserted. Hence, with a probability of at least $\epsilon(\lambda)/\ell$, which is non-negligible, $\mathcal{B}$ would win $\mathsf{Forge}_{\mathcal{B},\mathsf{OT1}}^{\text{1-token-gen,verify}}(\lambda)$, thus, giving us a contradiction. $\square$

A.2.2 An Unrestricted Scheme Using Hash-and-Sign

Next, we use the hash-and-sign paradigm to extend a length-restricted (see Definition 2) scheme to an unrestricted scheme that can sign documents of arbitrary length. The main idea is to hash the documents to a fixed size and sign their hash. However, a slightly more contrived approach is necessary to achieve unforgeability, since we are using a UOWHF and not a CRH as the hash function. For each message m, we sample a fresh hash function, hash the message to a string of length ℓ, sign the hash value concatenated with the description of the hash function, and then attach the description of the hash function as a part of the signature. The construction is described formally in Algorithm 17.

Lemma 4. *Assuming the existence of post-quantum one-way functions, there is a noise-tolerance preserving lift (Definition 4) of any ℓ-restricted* $\mathsf{Unforgeable}^{\text{1-token-gen,verify}}$ TMAC *for $\ell(\lambda) \in \mathsf{poly}(\lambda)$ (see Definitions 2 and 5), to an unrestricted* $\mathsf{Unforgeable}^{\text{1-token-gen,verify}}$ TMAC, *as shown in Algorithm 17.*

Proof. The correctness of the scheme is trivial and, hence, we have omitted the proof. The claim about preservation of noise tolerance is also immediate since all the algorithms are the same in OTL and OT up to hashing the message which is a classical procedure, and therefore, is unaffected by the noise in the quantum communication channel.

We now prove that the unrestricted scheme OT is $\mathsf{Unforgeable}^{\text{1-token-gen,verify}}$. Suppose by way of contradiction, there exists a QPT adversary $\mathcal{A}$ that wins $\mathsf{Forge}_{\mathsf{OT}}^{\text{1-token-gen,verify}}(\lambda)$ with non-negligible probability. Let $\{m_i, (s_i, \sigma_i')\}_w$ be the documents submitted by $\mathcal{A}$. First, notice that except with negligible

---

Algorithm 17 OT.

The resulting scheme obtained by instantiating OT with the specific scheme CTMAC_ℓ^η (Algorithm 16), is called $\mathsf{CTMAC}_{1\text{-time}}^\eta$.

Assumes: OTL is a $(\ell + \lambda)$-restricted TMAC for some $\ell = \ell(\lambda) \in \mathsf{poly}(\lambda)$, $\{h_r : \{0,1\}^* \to \{0,1\}^{\ell(|r|)}\}_{r \in \{0,1\}^*}$ is a UOWHF with indexing function I.

1: **procedure** key-gen(1^λ)
2:     **Return** OTL.key-gen(1^λ).
3: **end procedure**

1: **procedure** token-gen$_k$
2:     **Return** OTL.token-gen$_k$.
3: **end procedure**

1: **procedure** sign$_{|\underline{\textcircled{\tiny 2}}\rangle}(m)$
2:     $s \leftarrow I(1^\lambda)$
3:     **Return** $(s, \mathsf{OTL.sign}_{|\underline{\textcircled{\tiny 2}}\rangle}(s\|h_s(m)))$.
4: **end procedure**

1: **procedure** verify$_k(m, \sigma)$
2:     Interpret $\sigma = (s, \sigma')$
3:     **Return** $\mathsf{OTL.verify}_k(s\|(h_s(m)), \sigma')$.
4: **end procedure**

---

probability, for any $i < j$, $(s_i\|h_{s_i}(m_i)) \neq (s_j\|h_{s_j}(m_j))$. Otherwise, $s_i = s_j$, and $\mathcal{A}$ could be used to find a designated collision in h_{s_i}, by first uniformly guessing the index corresponding to i and outputting m_i, and then simulating the algorithm to gather m_j. Since $\mathcal{A}$ wins with non-negligible probability, the event that $\mathcal{A}$ wins, and that there is no collision between the hashes of distinct documents, occurs with non-negligible probability. Let this probability be $\epsilon(\lambda)$. Conditioning on the above-mentioned event, there is a set S' of size greater then $r + 1$ such that for all $i, j \in S'$:

$$(s_i\|h_{s_i}(m_i)) \neq (s_j\|h_{s_j}(m_j)),$$
$$\mathsf{OTL.verify}_k((s_i\|h_{s_i}(m_i)), \sigma'_i)) = 1,$$
$$\mathsf{OTL.verify}_k((s_j\|h_{s_j}(m_j)), (\sigma'_j)) = 1.$$

It is thus easy to use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for OTL winning with probability $\epsilon(\lambda)$, contradicting that OTL is $\mathsf{Unforgeable}^{\text{1-token-gen,verify}}$.

□

A.2.3 Unforgeability Against Polynomial Tokens Attacks

In this section, we show how to lift any $\mathsf{Unforgeable}^{\text{1-token-gen,verify}}$ unrestricted scheme, such as OT (Algorithm 17), to an $\mathsf{Unforgeable}^{\text{token-gen,verify}}$ unrestricted scheme TM (Algorithm 18), using an authenticated encryption

scheme (AENC). In this full-blown scheme, TM, the secret key is the key for the authenticated scheme. A token is generated by sampling a fresh secret key for OT and then running OT.token-gen on the OT secret key to generate an OT token. The OT token, along with the encrypted secret key, forms the token for the full-blown scheme TM.

A document is signed by first signing it under the OT token to get a OT signature for it, and appending the encrypted secret key to it. Verification of an alleged document-signature pair is done by first decrypting the cipher corresponding to the OT secret key, which is then used to run OT.verify on the other part of the signature. Verification accepts if and only if the authentication of the cipher accepts, and the OT verification succeeds.

We demand that the AENC scheme used here satisfies a property analogous to that of strong MAC. Assuming the existence of post-quantum one-way functions, authenticated encryptions schemes satisfying this strong property do exist, see Remark 2.

Lemma 5. *Assuming the existence of a post-quantum classical-queries strong authenticated encryption scheme, there is a noise-tolerance preserving lift (Definition 4) of any unrestricted* Unforgeable$^{\text{1-token-gen,verify}}$ TMAC *(see Definitions 2 and 5) to an unrestricted* Unforgeable$^{\text{token-gen,verify}}$ TMAC, *as shown in Algorithm 18.*

Proof. Given an unrestricted scheme Unforgeable$^{\text{1-token-gen,verify}}$ TMAC OT, we construct another unrestricted scheme TM given in Algorithm 18 that is Unforgeable$^{\text{token-gen,verify}}$.

Correctness: It is clear that in the TM scheme (Algorithm 18), the verification procedure accepts the output of the signing procedure (assuming the same holds for OT).

The claim about preservation of noise tolerance is also immediate and, hence, we omit the proof.

Next, we show that TM is Unforgeable$^{\text{token-gen,verify}}$. Assume towards contradiction, that there is a QPT adversary with oracle access to verification, $\mathcal{A}$, getting r signing tokens, and submitting w signed documents such that with non-negligible probability $r + 1$ distinct documents all pass verification, meaning $\mathsf{Forge}^{\text{token-gen,verify}}_{\mathcal{A},\text{TM}}(\lambda) = 1$ with non-negligible probability. Without loss of generality, assume that $\mathcal{A}$ always outputs a fixed (per λ) number of signed documents w.

Let k_{AENC} be the secret key generated for TM, which by construction, is a key for the authenticated encryption scheme AENC. Let $d_1, d_2, \ldots, d_w$ be the w signed documents submitted by $\mathcal{A}$, and let S be the set of indices that the challenger accepts. Without loss of generality, each d_i must have the form $d_i = (m_i, s_i, e_i)$, where m_i is the document that was signed, e_i is an encryption of the associated secret key for the OT scheme (whose decryption we will refer to as κ_i), and each s_i is the signature for m_i by OT. For all $i \in S$,

---
Algorithm 18 TM.

The resulting scheme obtained by instantiating TM with the specific scheme $\mathsf{CTMAC}^{\eta}_{\text{1-time}}$ (Algorithm 17) is called $\mathsf{CTMAC}^{\eta}_{\text{Poly-times}}$.

Assumes: OT is an unrestricted TMAC, and AENC is a post-quantum classical-queries strong authenticated encryption scheme.

1: **procedure** key-gen(1^{λ})
2:     **Return** AENC.key-gen(1^{λ}).
3: **end procedure**

1: **procedure** token-gen$_k$
2:     Run OT1.key-gen(1^{λ}) to obtain κ.
3:     Run OT1.token-gen$_{\kappa}$ to obtain $|\widetilde{\underline{\mathbb{Q}}}\rangle$.
4:     **Return** $(|\widetilde{\underline{\mathbb{Q}}}\rangle, \mathsf{AENC.Enc}_k(\kappa))$.
5: **end procedure**

1: **procedure** sign$_{|\underline{\mathbb{Q}}\rangle}(m)$
2:     Interpret $|\underline{\mathbb{Q}}\rangle$ as $(|\widetilde{\underline{\mathbb{Q}}}\rangle, e)$
3:     **Return** $(\mathsf{OT1.sign}_{|\widetilde{\underline{\mathbb{Q}}}\rangle}(m), e)$.
4: **end procedure**

1: **procedure** verify$_k(m, \sigma)$
2:     Interpret σ as (s, e)
3:     $\kappa \leftarrow \mathsf{AENC.dec}_k(e)$.
4:     **if** $\kappa = \bot$ **then**
5:         **Return** 0
6:     **else**
7:         **Return** $\mathsf{OT1.verify}_{\kappa}(m, s)$.
8:     **end if**
9: **end procedure**

---

since $\mathsf{verify_{OT}}(d_i) = 1$, e_i is a valid encryption under AENC, and $\mathsf{dec_{k_{AENC}}}(e_i)$ outputs a string p_i such that $\mathsf{OT.verify}_{p_i}(m_i, s_i)$ accepts.

Denote $\mathcal{E}_{Win}^\lambda$ to be the event that $\mathsf{Forge}_{\mathcal{A},\mathsf{TM}}^{\mathsf{token\text{-}gen,verify}}(\lambda) = 1$, i.e.,

$$count \geq r + 1,$$

where *count* is the random variable counting the number of successful verifications as defined in Game 2. Similarly, let $\mathcal{E}_{Dup}^\lambda$ be the event that for every $i \in S$, e_i appeared in one of the tokens sent by the challenger in response to the token-gen queries of $\mathcal{A}$. Let,

$$\mathcal{E}^\lambda \equiv \mathcal{E}_{Dup}^\lambda \bigwedge \mathcal{E}_{win}^\lambda. \tag{7}$$

We will prove the following lemma:

Lemma 6. $\Pr[\mathcal{E}^\lambda]$ *is non-negligible.*

Sketch proof. Suppose not, and $\Pr[\mathcal{E}^\lambda]$ is negligible. Since $\Pr[\mathcal{E}_{Win}^\lambda]$ is non-negligible by assumption, the supposition implies $\Pr[\mathcal{E}_{Win}^\lambda \wedge (\mathcal{E}_{Dup}^\lambda)^c]$ is non-negligible. Denote by $\epsilon(\lambda)$ the non-negligible probability that both of the following events occur:

1. $(\mathcal{E}_{Win}^\lambda)$: $\mathsf{Forge}_{\mathcal{A},\mathsf{TM}}^{\mathsf{token\text{-}gen,verify}}(\lambda) = 1$.

2. $(\mathcal{E}_{Dup}^\lambda)^c$: There is an $i \in S$ such that e_i did not appear in the tokens sent by the challenger.

It is straightforward to construct a QPT $\mathcal{B}_{frg}$ with oracle access to $\mathsf{AENC.enc_{k_{AENC}}}(m, \sigma)$, that guesses the right index for i, and succeeds in winning $\mathsf{Enc\text{-}forge}_{\mathcal{B}_{frg},\mathsf{AENC}}(\lambda)$ (Game 15) with a probability that is greater than $\epsilon(\lambda)$ times an inverse polynomial, implying that $\Pr[\mathcal{E}^\lambda]$ must be non-negligible. Here we use the fact that AENC is a strong *authenticated* encryption. $\qquad\square$

Conditioning on $\mathcal{E}^\lambda$, there are at most r distinct values for the e_i's. Hence, by the pigeonhole principle for some $i, j \in S$ $e' \equiv e_i = e_j$, and since e' was appended in one of the tokens, it must be an encryption of one of the secret keys for OT generated by the challenger during $\mathsf{TM.token\text{-}gen}$. Denote $p \equiv \mathsf{dec_{k_{AENC}}}(e')$.

Since $a_i = (m_i, s_i, e_i)$ and $a_j = (m_j, s_j, e_j)$ are accepted, we know that $\mathsf{OT.verify}_p(m_i, s_i) = 1$ and $\mathsf{OT.verify}_p(m_j, s_j) = 1$. This is already very close to forgery of the OT scheme, only $\mathcal{A}$ has access to some extra information, namely the encrypted secret keys. In order to complete our proof via contradiction, we will construct an adversary $\mathcal{B}$ against the one-time scheme OT (Algorithm 17) in the $\mathsf{Forge}^{\mathsf{1\text{-}token\text{-}gen,verify}}(\lambda)$ game (see Game 15), and using the CCA encryption property of the underlying AENC scheme, show that the adversary wins with non-negligible probability.

An adversary for OT. $\mathcal{B}_{\mathsf{OT}}$ will act in the following manner: given a single token $|\text{☝}'\rangle$, $\mathcal{B}_{\mathsf{OT}}$ will run OT.key-gen and OT.token-gen $r-1$ times to produce $r-1$ secret keys and the corresponding tokens, respectively. $\mathcal{B}_{\mathsf{OT}}$ will then run AENC.key-gen to generate the secret key $\mathsf{k}_{\mathsf{AENC}}$ and use it to encrypt all the secret keys and append them accordingly to the tokens. Lastly, $\mathcal{B}_{\mathsf{OT}}$ will call OT.key-gen one more time to create a bogus secret key Bog, encrypt it, and append it to $|\text{☝}'\rangle$ (the original challenge token from the challenger) as the rth token. We denote the secret keys generated by $\mathcal{B}_{\mathsf{OT}}$ as the set $\{\kappa_i\}_i$. $\mathcal{B}_{\mathsf{OT}}$ will then shuffle those tokens and feed them to $\mathcal{A}$. Whenever $\mathcal{A}$ queries the oracle with (m, sig, c), $\mathcal{B}_{\mathsf{OT}}$ will do the following steps:

1. Obtain $\mathsf{dec}_{\mathsf{k}_{\mathsf{AENC}}}(\mathsf{c}) = \kappa$, and if this fails, reject.

2. If κ is the bogus key Bog, query the oracle on (m, sig) and return its answer. Otherwise, check that $\mathsf{OT.verify}_\kappa(m, sig) = 1$.

If all tests pass $\mathcal{A}$ answers 1.

Upon receiving $\mathcal{A}$'s answer $\{d_i\}_{i \in S}$, $\mathcal{B}_{\mathsf{OT}}$ will guess $\hat{i}, \hat{j} \in [w]$ and output $(m_{\hat{i}}, s_{\hat{i}}), (m_{\hat{j}}, s_{\hat{j}})$.

In a similar manner as before, we define analogous events in the simulation of $\mathcal{A}$ in $\mathcal{B}_{\mathsf{OT}}$. Let $\mathcal{F}^\lambda_{Win}$ be the event within the $\mathcal{B}_{\mathsf{OT}}$ simulation, $\mathcal{A}$ produces at least $r+1$ signed documents which would pass verification, i.e.,

$$\mathcal{F}^\lambda_{Win} \equiv \{\text{within the simulation, } count \geq r+1\},$$

where $count$ is the random variable counting the number of successful verifications as defined in Game 2. Similarly, let $\mathcal{F}^\lambda_{Dup}, \mathcal{F}^\lambda$ be the events defined as follows:

$$\mathcal{F}^\lambda_{Dup} \equiv \{\text{within the simulation, } \forall i \in S, \, e_i \text{ appeared in the tokens provided to } \mathcal{A}\}.$$

$$\mathcal{F}^\lambda \equiv \mathcal{F}^\lambda_{Dup} \bigwedge \mathcal{F}^\lambda_{win}$$

If $\mathcal{F}^\lambda$ occurs with non-negligible probability, then clearly $\mathcal{B}_{\mathsf{OT}}$ succeeds with non-negligible probability. The deduction holds due to the same arguments that hold for the event in Eq. (7). The main arguments are that there must be some $i \neq j$ such that $e' \equiv e_i = e_j$, and $\mathsf{OT.verify}_p(m_j, s_j) = 1, \mathsf{OT.verify}_p(m_i, s_i) = 1$. Since the tokens were shuffled by $\mathcal{B}_{\mathsf{OT}}$ before being submitted to $\mathcal{A}$, e' corresponds to any given copy of the OT scheme with probability $\frac{1}{r}$. Clearly, $\mathcal{B}_{\mathsf{OT}}$ would pick the correct i, j with a probability of at least $\frac{1}{w^2}$. Hence, $\mathsf{Forge}^{\text{1-token-gen,verify}}_{\mathcal{B}_{\mathsf{OT}}, \mathsf{OT}}(\lambda) = 1$ with probability at least $\frac{1}{rw^2} \Pr[\mathcal{F}^\lambda]$, which is non-negligible if $\Pr[\mathcal{F}^\lambda]$ is non-negligible.

Hence, the last thing that we need to complete the proof of Lemma 5 (by reaching the desired contradiction), is to prove that $\Pr[\mathcal{F}^\lambda]$ is non-negligible, which we do next.

Lemma 7. $\Pr[\mathcal{F}^\lambda]$ *is non-negligible.*

Proof. Assume to the contrary, that $\Pr[\mathcal{F}^\lambda]$ is negligible. We will construct an efficient distinguisher for AENC that wins the indistinguishability game in Definition 13 with non-negligible advantage.

A Distinguisher for AENC. $\mathcal{B}_{ind}$ is a distinguisher for Game 14. $\mathcal{B}_{ind}$ acts in the following manner: $\mathcal{B}_{ind}$ runs $\mathsf{OT.key\text{-}gen}, \mathsf{OT.token\text{-}gen}$ r times to create secret keys $\{\kappa_i\}_i$ and corresponding tokens $\{|\text{⚷}_i\rangle\}_i$. $\mathcal{B}_{ind}$ then asks the encryption oracle to encrypt all but one of the secret keys $\{\kappa_i\}_{2 \leq i \leq r}$. Then $\mathcal{B}_{ind}$ appends the corresponding encryptions to the respective tokens, barring the first token, $|\text{⚷}_1\rangle$.

$\mathcal{B}_{ind}$ runs $\mathsf{OT.key\text{-}gen}$ one last time to get a fresh secret key p, and sends p and κ_1 as the challenge messages, and gets back c which is the challenge cipher. $\mathcal{B}_{ind}$ appends c to $|\text{⚷}_1\rangle$ to get a TM token.

$\mathcal{B}_{ind}$ will then shuffle those tokens and feed them to $\mathcal{A}$. Whenever $\mathcal{A}$ queries the oracle with (m, sig, c) $\mathcal{B}_{ind}$ will:

1. If $\mathsf{c} \neq c$, call a decryption oracle to extract $\hat{\kappa} \equiv \mathsf{dec}_{\mathsf{k_{AENC}}}(\mathsf{c})$, or reject if not valid, and return $\mathsf{OT.verify}_{\hat{\kappa}}(m, sig) = 1$

2. If $\mathsf{c} = c$, check that $\mathsf{OT.verify}_{\kappa_1}(m, sig) = 1$.

When $\mathcal{A}$ submits $\{d_i\}_w$, $\mathcal{B}_{ind}$ would verify all of them in the exact same manner as that of oracle queries, construct the subset S, verify that for all $i \in S$, e_i's are such that were appended in the tokens, and that $|S| \geq r + 1$. If all of the above-mentioned verification holds, $\mathcal{B}_{ind}$ would output 1, and otherwise outputs 0.

Notice that if the challenge cipher c is an encryption of p, the key unrelated to $|\text{⚷}_1\rangle$, then $\mathcal{A}$'s view is exactly the same as it is in the simulation of $\mathcal{B}_{\mathsf{OT}}$. Hence, in that case, so $\mathcal{B}_{ind}$ will output 1 with a probability of $\Pr[\mathcal{F}^\lambda]$. On the other hand, if $c = \mathsf{Enc}_{\mathsf{k_{AENC}}}(\kappa_1)$, then the view of $\mathcal{A}$ is the same as in $\mathsf{Forge}_{\mathcal{A},\mathsf{TM}}^{\mathsf{token\text{-}gen,verify}}(\lambda)$, so it will succeed with probability $\Pr[\mathcal{E}^\lambda]$. Hence,

$$|\Pr[\mathcal{B}_{ind}(enc_{\mathsf{k_{AENC}}}(p), \lambda) = 1] - \Pr[\mathcal{B}_{ind}(enc_{\mathsf{k_{AENC}}}(\kappa_1), \lambda) = 1]| = \Pr[\mathcal{F}^\lambda] - \Pr[\mathcal{E}^\lambda]|,$$

which is non-negligible since $\Pr[\mathcal{F}^\lambda]$ is negligible as per assumption, but $\Pr[\mathcal{E}^\lambda]$ is non-negligible (Lemma 6). This is a contradiction to the security of AENC. Hence, it must be the case that $\Pr[\mathcal{F}^\lambda]$ is non-negligible, thus concluding the proof of the lemma. □

Lemma 7 in turn finishes the proof of Lemma 5 as mentioned above. □

A.2.4 Unforgeability in the Presence of a Signing Oracle

Lastly, an Unforgeable$^{\text{token-gen,verify}}$ TMAC scheme is wrapped with randomness to get an Unforgeable$^{\text{token-gen,verify,}\widetilde{\text{sign}}}$ TMAC scheme. The idea is relatively simple. One can think of a naive reduction from Unforgeable$^{\text{token-gen,verify,}\widetilde{\text{sign}}}$ to Unforgeable$^{\text{token-gen,verify}}$ by simulating signing queries using additional tokens. However, the reduction fails essentially because the signing oracle $(\widetilde{\text{sign}}_k)$ may be queried several times with the same document, providing different responses each time (see Appendix D for more details). A way to enforce that there are no multiple signing queries for the same document is to concatenate the document to be signed with fresh randomness. The randomness is then provided as part of the signature. Verification is done by simply verifying the document concatenated with the proclaimed randomness. In this way, we circumvent the issue mentioned above, while maintaining all previous properties. The full description of the construction is provided in Algorithm 19.

Lemma 8. *There is a noise-tolerance preserving lift (Definition 4) of any unrestricted* Unforgeable$^{\text{token-gen,verify}}$ *TMAC (see Definitions 2 and 5), to an unrestricted* Unforgeable$^{\text{token-gen,verify,}\widetilde{\text{sign}}}$ *TMAC, as shown in Algorithm 19.*

Algorithm 19 TMS.

The resulting scheme obtained by instantiating TMS with the specific scheme $\mathsf{CTMAC}^{\eta}_{\text{Poly-times}}$ (Algorithm 18), is called $\widetilde{\mathsf{CTMAC}}^{\eta}_{\text{Poly-times}}$.

Assumes: TM is an unrestricted TMAC.

1: **procedure** key-gen(1^{λ})
2:     **Return** TM.key-gen(1^{λ}).
3: **end procedure**

1: **procedure** token-gen$_k$
2:     **Return** TM.token-gen$_k$
3: **end procedure**

1: **procedure** sign$_{|\underline{\circ}\rangle}(m)$
2:     rnd $\leftarrow_{\$} \{0,1\}^{\lambda}$
3:     **Return** $(\mathsf{rnd}, \mathsf{TM.sign}_{|\underline{\circ}\rangle}(m\|\mathsf{rnd}))$.
4: **end procedure**

1: **procedure** verify$_k(m, \sigma)$
2:     Interpret σ as (rnd, sig).
3:     **Return** TM.verify$_k(m\|\mathsf{rnd}, sig)$.
4: **end procedure**

Proof. The correctness is immediate. The claim about preservation of noise tolerance is also immediate since all the algorithms are the same in TM and

TMS up to adding randomness to the message, which is a classical procedure and is, hence, unaffected by noise in the quantum communication channel.

For unforgeability, let $\mathcal{A}$ be an adversary winning $\mathsf{Forge}_{\mathcal{A},\mathsf{TMS}}^{\mathsf{token\text{-}gen},\mathsf{verify},\widetilde{\mathsf{sign}}}(\lambda)$ with non-negligible probability $\epsilon(\lambda)$. Without loss of generality, it can be assumed that $\mathcal{A}$ always makes exactly j oracle queries to $\widetilde{\mathsf{sign}}$ for some $j \in \mathsf{poly}(\lambda)$, and asks for $r \in \mathsf{poly}(\lambda)$ tokens.

Next, we construct a corresponding adversary $\mathcal{B}$ winning $\mathsf{Forge}_{\mathcal{B},\mathsf{TM}}^{\mathsf{token\text{-}gen},\mathsf{verify}}(\lambda)$ with non-negligible probability. $\mathcal{B}$ will use $r + j$ tokens, and run $\mathcal{A}$, supplied with r of those tokens. If $\mathcal{A}$ makes a query of the form $(m, (\mathsf{rnd}, \sigma))$ to the verification oracle, $\mathcal{B}$ would query $(m\|\mathsf{rnd}, \sigma)$ to its verification oracle in $\mathsf{Forge}_{\mathsf{TM}}^{\mathsf{token\text{-}gen},\widetilde{sign},\mathsf{verify}}$ and answer accordingly. If $\widetilde{\mathsf{sign}}_k$ is called with a document query m_i, $\mathcal{B}$ will sample a random $\mathsf{rnd}_i \in \{0,1\}^\lambda$, compute $\sigma_i \leftarrow \mathsf{sign}_{|\underline{\mathfrak{Q}}_i\rangle}(m_i\|\mathsf{rnd}_i)$ with one of its remaining tokens $|\underline{\mathfrak{Q}}_i\rangle$, and return to $\mathcal{A}$ the response $(\mathsf{rnd}_i, \sigma_i)$. As $\mathcal{A}$ would make j calls to $\widetilde{\mathsf{sign}}_k$, $\mathcal{B}$ will not run out of tokens. $\mathcal{A}$ outputs $\hat{m}_1, \dots, \hat{m}_w$ and corresponding signatures $(\hat{\mathsf{rnd}}_1, \hat{\sigma}_1), \dots, (\hat{\mathsf{rnd}}_w, \hat{\sigma}_w)$. $\mathcal{B}$ will then extract from that the documents $\hat{m}_1\|\hat{\mathsf{rnd}}_1, \dots, \hat{m}_w\|\hat{\mathsf{rnd}}_w$, and $\hat{\sigma}_1, \dots, \hat{\sigma}_w$ as the corresponding signatures for them. $\mathcal{B}$ would output those, in addition to those signed documents that he generated by itself $(m_i\|\mathsf{rnd}_i, \sigma_i)_{i\in[j]}$. Due to the randomness of $(\mathsf{rnd}_i)_{i\in[j]}$, they are all distinct with overwhelming probability $\delta(\lambda)$, meaning $(m_i\|\mathsf{rnd}_i)_{i\in[j]}$ are also distinct with that probability. These documents are bound to pass verification, as they were signed by the use of a token. In the winning event for $\mathcal{A}$, there is a $r + 1$ subset of $(\hat{m}_i\|\hat{\mathsf{rnd}}_i)_{i\in[w]}$, which are both distinct from $(m_i\|\mathsf{rnd}_i)_{i\in[j]}$ and successfully pass verification (see the winning condition for Game 2). As the view of $\mathcal{A}$ is the same as in the true game $\mathsf{Forge}_{\mathcal{A},\mathsf{TMS}}^{\mathsf{token\text{-}gen},\widetilde{sign},\mathsf{verify}}$, this occurs with probability $\epsilon(\lambda)$. $\mathcal{B}$ could only lose only if $\mathcal{A}$ loses in the simulation, or if the randomness sampled was not distinct(or both). By the union bound this means $\mathcal{B}$ loses $\mathsf{Forge}_{\mathcal{B},\mathsf{TM}}^{\mathsf{token\text{-}gen},\mathsf{verify}}(\lambda)$ with probability at most $1 - \delta(\lambda) + 1 - \epsilon(\lambda)$, or alternatively, $\mathcal{B}$ wins with probability $\epsilon(\lambda) - (1 - \delta(\lambda))$, which is a non-negligible function, meaning such $\mathcal{A}$ cannot exist. $\qquad\square$

B   Detailed Analysis of Certified Deletion Scheme

For convenience, the definitions for the schemes E_m and the security game for weak certified deletion are brought here again.

Presented ahead is one possible strategy for attacking the scheme. The main idea is that if the adversary could know both a valid result of a measurement of the token by the standard basis and a measurement in the Hadamard basis, then it could win the security game by submitting first the result of a measurement in the Hadamard basis as a certificate, and upon receiving the key, output the measurement corresponding to it as a'. There-

Algorithm 20 The QECD E_m - A QECD scheme (for $m \in \{0,1\}$)

1: **procedure** key-gen(1^λ)
2:     $k \leftarrow\$ \{0,1\}^\lambda$
3:     **Return** k.
4: **end procedure**

1: **procedure** enc$_k(a)$
2:     **Return** $H^k(|a\rangle)$.
3: **end procedure**

1: **procedure** dec$_k(|c\rangle)$
2:     Compute $H^k|c\rangle$ to obtain $|cc\rangle$
3:     Measure $|cc\rangle$ to obtain a'
4:     **Return** a'.
5: **end procedure**

1: **procedure** del$(|c\rangle)$
2:     Compute $(H^m)^{\otimes\lambda}|c\rangle$ to obtain $|cc\rangle$
3:     Measure $|cc\rangle$ to obtain cer
4:     **Return** cer.
5: **end procedure**

1: **procedure** verify$_k(cer)$
2:     Define $\mathsf{Cons}_m = \{i \in [\lambda] | k_i = m\}$.
3:     **if** $cer = a|_{\mathsf{Cons}_m}$ **then**
4:         **Return** 1.
5:     **else**
6:         **Return** 0.
7:     **end if**
8: **end procedure**

Game 21 Weak Certified Deletion Game $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{A},\Pi}(\lambda)$:

1: The challenger $\mathcal{C}$ runs $\mathsf{key\text{-}gen}(1^\lambda)$ to generate k, and uniformly samples $a \in \{0,1\}^\lambda$. $\mathcal{C}$ then sends $|c\rangle = \mathsf{enc}_k(a)$.

2: $\mathcal{A}$ sends $\mathcal{C}$ some string cer.

3: $\mathcal{C}$ computes $V \leftarrow \mathsf{verify}_k(cer)$, and then sends k to the adversary.

4: $\mathcal{A}$ outputs a'.

We say the output of the game is 1 if and only if $V = 1$ and $a = a'$.

---

fore, an adversary may act in the following manner: when receiving $|\stackrel{\circ}{\smile}\rangle$, it will create two strings corresponding to guesses for measurement by both bases. After submitting the guessed result for a measurement by the standard basis, it will receive the key and answer accordingly in a deterministic manner. The question is then limited to computing the optimal probability for such an adversary to be correct in its guesses at the first stage of the protocol. This question is exactly the one discussed in [MVW12, Section 4.2], in the form of two distinct challenges sent by the bank to an adversary having a single quantum money bill. The optimal probability achieved by [MVW12] is $\cos^2(\frac{\pi}{8}) \approx 0.85355$.

It can be concluded that there is an adversary winning $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{A},E_1}(1)$ with a probability of at least $\cos^2(\frac{\pi}{8})$. A priori, it is not clear why this is the best the adversary can do. It seems reasonable that an adversary might use a more complex strategy to improve their chances instead of guessing ahead at the first stage of the protocol. We show that, in fact, this is an optimal strategy using semi-definite programming techniques.

B.1   Semi-Definite Programming

This section discusses standard notions and results of semi-definite programming. Our analysis in this section is similar to the one in Ref. [MVW12], and we follow the conventions therein. This preliminary section is taken verbatim from [MVW12].

Semi-definite programming is a topic that has found several interesting applications within quantum computing and quantum information theory in recent years. Here, we provide just a brief summary of semi-definite programming that is focused on the narrow aspects that we use. More comprehensive discussions can, for instance, be found in [VB96, Lov03, De 02, BV04]. We first cover some notations:

For any finite-dimensional complex Hilbert space $\mathcal{X}$, we write $L(\mathcal{X})$ to denote the set of linear operators acting on $\mathcal{X}$, $\mathsf{Herm}(\mathcal{X})$ to denote the set of Hermitian operators acting on $\mathcal{X}$, $\mathsf{Pos}(\mathcal{X})$ to denote the set of positive semi-definite operators acting on $\mathcal{X}$, and $\mathsf{Pd}(\mathcal{X})$ to denote the set of positive-definite operators acting on $\mathcal{X}$. For every $A, B \in \mathsf{Herm}(\mathcal{X})$, the notation $A \succeq B$ indicates that $A - B$ is positive semi-definite. $D(\mathcal{X})$ denote the set

of density operators acting on $\mathcal{X}$.

Given operators $A, B \in L(\mathcal{X})$, one defines the inner product between A and B as $\langle A, B \rangle = \text{Tr}(A^*B)$. For Hermitian operators $A, B \in \text{Herm}(\mathcal{X})$, it holds that $\langle A, B \rangle$ is a real number and satisfies $\langle A, B \rangle = \langle B, A \rangle$. For every choice of finite-dimensional complex Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$, and for a given linear mapping of the form $\Phi : L(\mathcal{X}) \to L(\mathcal{Y})$, there is a unique mapping $\Phi^* : L(\mathcal{Y}) \to L(\mathcal{X})$ (known as the *adjoint* of Φ) that satisfies $\langle Y, \Phi(X) \rangle = \langle \Phi^*(Y), X \rangle$ for all $X \in L(\mathcal{X})$ and $Y \in L(\mathcal{Y})$.

A semi-definite program is a triple (Φ, A, B), where

- $\Phi : L(\mathcal{X}) \to L(\mathcal{Y})$ is a Hermiticity-preserving linear mapping, and

- $A \in \text{Herm}(\mathcal{X})$ and $B \in \text{Herm}(\mathcal{Y})$ are Hermitian operators,

for some choice of finite-dimensional complex Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$. We associate with the triple (Φ, A, B) two optimization problems, called the *primal* and *dual* problems, as follows:

Primal problem	Dual problem
maximize: $\langle A, X \rangle$	minimize: $\langle B, Y \rangle$
subject to: $\Phi(X) = B$,	subject to: $\Phi^*(Y) \geq A$,
$X \in \text{Pos}(\mathcal{X})$	$Y \in \text{Herm}(\mathcal{Y})$.

The optimal primal value of this semi-definite program is

$$\alpha = \sup\{\langle A, X \rangle : X \in \text{Pos}(\mathcal{X}), \; \Phi(X) = B\},$$

and the optimal dual value is

$$\beta = \inf\{\langle B, Y \rangle : Y \in \text{Herm}(\mathcal{Y}), \; \Phi^*(Y) \geq A\}.$$

(It is to be understood that the supremum over an empty set is $-\infty$ and the infimum over an empty set is ∞, so α and β are well-defined values in $\mathbb{R} \cup \{-\infty, \infty\}$. In this paper, however, we will only consider semi-definite programs for which α and β are finite.)

It always holds that $\alpha \leq \beta$, which is a fact known as *weak duality*. The condition $\alpha = \beta$, which is known as *strong duality*, does not hold for every semi-definite program, but there are simple conditions known under which it does hold. The following theorem provides one such condition (that has both a primal and dual form).

Theorem 11 (Slater's theorem for semi-definite programs)**.** *Let (Φ, A, B) be a semi-definite program and let α and β be its optimal primal and dual values.*

- *If β is finite and there exists a positive definite operator $X \in \text{Pd}(\mathcal{X})$ for which $\Phi(X) = B$, then $\alpha = \beta$ and there exists an operator $Y \in \text{Herm}(\mathcal{Y})$ such that $\Phi^*(Y) \geq A$ and $\langle B, Y \rangle = \beta$.*

- *If α is finite and there exists a Hermitian operator $Y \in \mathsf{Herm}(\mathcal{Y})$ for which $\Phi^*(Y) > A$, then $\alpha = \beta$ and there exists a positive semi-definite operator $X \in \mathsf{Pos}(\mathcal{X})$ such that $\Phi(X) = B$ and $\langle A, X \rangle = \alpha$.*

In other words, the first item of this theorem states that if the dual problem is feasible and the primal problem is *strictly feasible*, then strong duality holds and the optimal dual solution is achievable. The second item is similar, with the roles of the primal and dual problems reversed.

B.2 SDP Formulation of a 1-Fold Weak Certified Deletion Scheme

We begin by addressing the specific security game $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{A},E_1}(1)$. That is, $m = 1$, and since $\lambda = 1$, the message, the key, and the certificate are all only a single qubit.

Let $\mathcal{X}_a, \mathcal{X}_k$ represent the single-qubit registers designating the message a and the key k. Also, let $\mathcal{X}_{m_1}, \mathcal{X}_{m_2}$ be registers used to transport messages, back and forth. As during the protocol, the key is sent to the adversary, we initialize $\mathcal{X}_{m_2}$ as a copy of the key, and $\mathcal{X}_{m_1}$ as the encrypted state itself. Finally, we pad with $\mathcal{X}_p$ to get a purified state. For ease of notation, we will also denote $\overline{\mathcal{X}}_*$ when referring to all register but the one appearing in the subscript.

Denote
$$\mathcal{X} = \mathcal{X}_p \otimes \mathcal{X}_{m_1} \otimes \mathcal{X}_{m_2} \otimes \mathcal{X}_k \otimes \mathcal{X}_a,$$

$$|\psi\rangle = \frac{1}{2}\left(|00000\rangle + |11001\rangle + |0+110\rangle + |1-111\rangle\right) \in \mathcal{X},$$

$$\sigma_0 = |\psi\rangle\langle\psi|.$$

For any adversary, the security game can be described as follows: at the first stage, the challenger in the hold of σ_0, sends $\mathcal{X}_{m_1}$ to the adversary. The adversary then applies some quantum channel Q_1 on its part of the system and sends back the register $\mathcal{X}_{m_1}$ with its guess of the certificate. The challenger responds by sending over $\mathcal{X}_{m_2}$, which holds a copy of the key. While the challenger should have measured the certificate first, by the deferred measurement principle, it is clear that the outcome of the game is the same as if it were not to be touched, and measured at the end of the protocol. The adversary then applies a second quantum channel Q_2, and sends back $\mathcal{X}_{m_2}$ with a guess for a'. In order to verify, the challenger checks its registers and verifies that the results are proper. This verification could be instantiated by the following positive operator valued measurement:

$$\begin{aligned}\Pi = &I_{\mathcal{X}_p} \otimes (|0000\rangle\langle0000| + |1000\rangle\langle1000| + |0101\rangle\langle0101| \\ &+ |1101\rangle\langle1101| + |0010\rangle\langle0010| + |1111\rangle\langle1111|).\end{aligned}$$

We denote ρ_1, ρ_2 respectively as the state of the system after the adversary sends the guess for the certificate and the state of the system when the adversary sends back a guess for a'. Since ρ_1, ρ_2 must be quantum systems, their trace must be 1, and since Q_1, Q_2 cannot affect the subsystems in possession of the challenger, it is clear that the following constraints are also a necessary condition:

$$\mathrm{Tr}_{\mathcal{X}_{m_1}}(\rho_1) = \mathrm{Tr}_{\mathcal{X}_{m_1}}(\sigma_0),$$
$$\mathrm{Tr}_{\mathcal{X}_{m_2}}(\rho_2) = \mathrm{Tr}_{\mathcal{X}_{m_2}}(\rho_1).$$

While not initially clear, these 4 constraints suffice to describe the game fully. That is, for any ρ_1 and ρ_2 fulfilling this, there exist quantum channels Q_1, Q_2 operating only on the adversary's subsystem such that $Q_1(\sigma_0) = \rho_1$ and $Q_2(\rho_1) = \rho_2$. To see this, consider the following theorem.

Theorem 12 (Freedom in Purifications, [NC11, Exercise 2.81]). *Let $|\psi\rangle$ and $|\phi\rangle$ be two purifications of a state ρ to a composite system $\mathcal{A} \otimes \mathcal{B}$. There exists a unitary transformation U acting on system $\mathcal{B}$ such that:*

$$|\phi\rangle = (I_A \otimes U)|\psi\rangle$$

Let $c_1 \in D(\mathcal{X} \otimes \mathcal{C})$ be a purification of ρ_1. As both $\sigma_0 \otimes |0\rangle\langle 0|_{\mathcal{C}}$ and c_1, are purifications of $\mathrm{Tr}_{\mathcal{X}_{m_1}}(\sigma_0)$, by Theorem 12 there is a unitary U operating only on $\mathcal{X}_{m_1} \otimes \mathcal{C}$ such that $I_{\overline{\mathcal{X}}_{m_1}} \otimes U(\sigma_0 \otimes |0\rangle\langle 0|_{\mathcal{C}}) = c_1$. Dropping the extra registers, this gives a desired $Q_1(\sigma_0) = \rho_1$. The same argument applies for the existence of a channel Q_2' only acting on the adversary's subsystem such that $Q_2'(c_1) = \rho_2$, perhaps demanding an even larger space. Throwing away the extra subsystem again provides Q_2. We can now maximize over the probability of success:

<u>Primal problem</u>

maximize: $\langle \Pi, \rho_2 \rangle$

subject to: $\mathrm{Tr}_{\mathcal{X}_{m_1}}(\rho_1) = \mathrm{Tr}_{\mathcal{X}_{m_1}}(\sigma_0)$

$\mathrm{Tr}_{\mathcal{X}_{m_2}}(\rho_2) = \mathrm{Tr}_{\mathcal{X}_{m_2}}(\rho_1)$

$\mathrm{Tr}(\rho_1) = \mathrm{Tr}(\rho_2) = 1$

$\rho_1, \rho_2 \in \mathsf{Pos}(\mathcal{X})$

In order to present this SDP in the standard form, we combine the two variables into a single variable by adding an auxiliary space $\mathcal{Z} \equiv \mathbb{C}^2$. We can then identify ρ_1 with $\mathrm{Tr}_{\mathcal{Z}}\big((|0\rangle\langle 0| \otimes I_{\mathcal{X}})\rho(|0\rangle\langle 0| \otimes I_{\mathcal{X}})\big)$ and ρ_2 with $\mathrm{Tr}_{\mathcal{Z}}\big((|1\rangle\langle 1| \otimes I_{\mathcal{X}})\rho(|1\rangle\langle 1| \otimes I_{\mathcal{X}})\big)$.

<div style="text-align: center"><u>Primal problem</u></div>

$$
\begin{aligned}
\text{maximize:} \quad & \langle |1\rangle\langle 1| \otimes \Pi, \rho \rangle \\
\text{mark:} \quad & \Gamma_0 \equiv |0\rangle\langle 0| \otimes I_{\mathcal{X}} \\
& \Gamma_1 \equiv |1\rangle\langle 1| \otimes I_{\mathcal{X}} \\
& \rho_1 \equiv \mathrm{Tr}_{\mathcal{Z}}(\Gamma_0 \rho \Gamma_0) \\
& \rho_2 \equiv \mathrm{Tr}_{\mathcal{Z}}(\Gamma_1 \rho \Gamma_1) \\
\text{subject to:} \quad & \mathrm{Tr}_{\mathcal{X}_{m_1}}(\rho_1) = \mathrm{Tr}_{\mathcal{X}_{m_1}}(\sigma_0) \\
& \mathrm{Tr}_{\mathcal{X}_{m_2}}(\rho_1) - \mathrm{Tr}_{\mathcal{X}_{m_2}}(\rho_2) = 0 \\
& \mathrm{Tr}(\rho_2) = 1 \\
& \mathrm{Tr}(\rho_1) = 1 \\
& \rho \in \mathsf{Pos}(\mathcal{Z} \otimes \mathcal{X})
\end{aligned}
$$

We further simplify notations in order to succinctly present the dual: we rewrite our constraints as a linear mapping

$$
\Phi : L(\mathcal{Z} \otimes \mathcal{X}) \to L(\overline{\mathcal{X}}_{m_1} \oplus \overline{\mathcal{X}}_{m_2} \oplus \mathbb{C} \oplus \mathbb{C}),
$$

defined by

$$
\phi(\rho) = \phi_1(\rho) \oplus \phi_2(\rho) \oplus \phi_3(\rho) \oplus \phi_4(\rho),
$$

for

$$
\phi_1(\rho) = \mathrm{Tr}_{\mathcal{X}_{m_1},\mathcal{Z}}(\Gamma_0 \rho \Gamma_0)
$$

$$
\phi_2(\rho) = \mathrm{Tr}_{\mathcal{X}_{m_2},\mathcal{Z}}(\Gamma_1 \rho \Gamma_1 - \Gamma_0 \rho \Gamma_0)
$$

$$
\phi_3(\rho) = \mathrm{Tr}(\mathrm{Tr}_Z(\Gamma_0 \rho \Gamma_0)).
$$

$$
\phi_4(\rho) = \mathrm{Tr}(\mathrm{Tr}_Z(\Gamma_1 \rho \Gamma_1)).
$$

We also denote $Q \equiv |0\rangle\langle 0| \otimes \Pi$ and notice Q is positive semi-definite. For convenience, we also rename the left-hand side of the constraints: $R_1 \equiv \sigma_0, R_2 \equiv \overline{0} \in \overline{\mathcal{X}}_{m_2}, R_3 \equiv 1 \in \mathbb{C}, R_4 \equiv 1 \in \mathbb{C}$, and $R \equiv R_1 \oplus R_2 \oplus R_3 \oplus R_4$. Lastly, we define the space $\mathcal{S} \equiv \overline{\mathcal{X}}_{m_1} \oplus \overline{\mathcal{X}}_{m_2} \oplus \mathbb{C} \oplus \mathbb{C}$.

<div style="text-align: center"><u>Primal problem</u></div>

$$
\begin{aligned}
\text{maximize:} \quad & \langle Q, \rho \rangle \qquad (8) \\
\text{subject to:} \quad & \Phi(\rho) = R, \\
& \rho \in \mathsf{Pos}(\mathcal{Z} \otimes \mathcal{X})
\end{aligned}
$$

<div style="text-align: center"><u>Dual problem</u></div>

$$
\begin{aligned}
\text{minimize:} \quad & \langle R, Y \rangle \qquad (9) \\
\text{subject to:} \quad & \Phi^*(Y) \succeq Q, \\
& Y \in \mathsf{Herm}(\mathcal{S})
\end{aligned}
$$

Remark 3. The value of SDP 8 for the primal problem is α, where $\alpha = \cos^2(\frac{\pi}{8})$ is obtained numerically.[23] As the resulting solution is positive definite, by Slater's condition (Theorem 11), the optimal solution to the dual is also α.

B.3 Parallel Repetition for Weak Certified Deletion Scheme

Let us now consider the security game $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{A},E_1}(\lambda)$. This game can be thought of as the game of playing λ copies of $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{A},E_1}(1)$ games simultaneously, where $\mathcal{A}$ wins if and only if it wins in all copies. In particular, the preparation and verification of a λ-qubit encrypted message and a certificate is, from the perspective of the challenger, equivalent to the independent preparation and verification of λ independent pairs of single-qubit encrypted messages and certificates. Hence, a successful decryption attack is equivalent to a successful decryption attack against all λ of the single-qubit games.

It can be concluded from the semi-definite programming formulation above that an adversary gains no advantage whatsoever by correlating multiple qubits during an attack. The interested reader is encouraged to read [MS07] for a more comprehensive discussion on the conditions that this holds. In our game, the key factor is that the objective function Q is positive semi-definite. We formulate a SDP for the $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{A},E_1}(\lambda)$ game, which is the λ-fold repetitions of $\mathsf{WEAK\text{-}DEL\text{-}IND}_{\mathcal{A},E_1}(1)$.

$$\underline{\text{Primal problem}}$$

$$\text{maximize: } \langle Q^{\otimes\lambda}, \rho \rangle \qquad (10)$$
$$\text{subject to: } \Phi^{\otimes\lambda}(\rho) = R^{\otimes\lambda},$$
$$\rho \in \mathsf{Pos}((\mathcal{Z} \otimes \mathcal{X})^{\otimes\lambda})$$

$$\underline{\text{Dual problem}}$$

$$\text{minimize: } \langle R^{\otimes\lambda}, Y \rangle$$
$$\text{subject to: } (\Phi^{\otimes\lambda})^*(Y) \geq Q^{\otimes\lambda},$$
$$Y \in \mathsf{Herm}(\mathcal{S}^{\otimes\lambda})$$

Proposition 9. *The value of Eq. (10) is α^λ, where $\alpha = \cos^2(\frac{\pi}{8})$.*

Proof. By Remark 3, the optimal value for the primal problem (Eq. (8)) and its dual problem (Eq. (9)) is α. It is easy to argue that the solution $P = \bigotimes_{i=1}^{\lambda} P'$ for P' being an optimal primal solution for the single-repetition semi-definite program is a solution for the primal parallel problem with value α^λ. Similarly, the value of $D = \bigotimes_{i=1}^{\lambda} D'$ for D' being optimal dual solutions for the single-repetition semi-definite program is α^λ. In order to see that D

---

[23]We used the CVX [GB08] package on Matlab, and the Quantinf package by Toby Cubbit, the numerical error was relatively large and differed between engines (of magnitude 10^{-5} on the "sedumi" solver and 10^{-3} on the "SDPT3" solver), leaving some room for doubt if α is exactly $\cos^2(\frac{\pi}{8})$, or a slightly larger value. Regardless, the exact value of α does not affect the results of this work, besides the precise amount of noise tolerance. The Matlab source code is available at `https://arxiv.org/src/2105.05016/anc`.

is indeed a feasible solution, we use the fact that for all positive semi-definite A and B, $A \succeq B \succeq 0$ implies that $A^{\otimes \lambda} \succeq B^{\otimes \lambda}$, since $\Phi^*(D') \succeq Q \succeq 0$ we have that $(\Phi^{\otimes \lambda})^*)(D') = (\Phi^*(D'))^{\otimes \lambda} \succeq Q^{\otimes \lambda} \succeq 0$. This implies that the success probability of an optimal strategy adversary for WEAK-DEL-IND$_{\mathcal{A}, E_1}(\lambda)$, is α^λ. □

By symmetry, the same holds for the SDP corresponding to WEAK-DEL-IND$_{\mathcal{A}, E_0}(\lambda)$ as well. As an immediate corollary, we get the following.

Proposition 4. *The optimal success probability for an adversary in* WEAK-DEL-IND$_{\mathcal{A}, E_0}(\lambda)$ *(see Game 8 and Algorithm 7), as well as in* WEAK-DEL-IND$_{\mathcal{A}, E_1}(\lambda)$ *is α^λ, where $\alpha = \cos^2(\frac{\pi}{8})$.*

C Drawbacks of the Conjugate TMAC

In this section, we bring two attacks against CTMAC (see Section 2.2): one by quantum access to the verification oracle, and the other violating strong unforgeability.

C.1 A Quantum Superposition Attack

As briefly discussed in Section 2.2.2, for TMACs with deterministic verifications, as CTMAC is, it is straightforward to define a stronger security notion where security is preserved even if the adversary is allowed quantum access to the verification function.

Definition 16. Forge$^{(|\text{verify}\rangle, \cdot, \cdot)}(\lambda)$ *is the same as* Forge$^{(\text{verify}, \cdot, \cdot)}(\lambda)$, *except that $\mathcal{A}$ instead has access to the quantum unitary* $\mathsf{U}_{\text{verify}_k}$ *(which it can query polynomially many times) instead of* verify$_k$.

It is emphasized that Forge$^{(|\text{verify}\rangle, \cdot, \cdot)}(\lambda)$ is defined only when verification is deterministic, as U_f is only defined when f is a function.

Proposition 10. *For all λ, there exists an adversary $\mathcal{A}$ winning* Forge$_{\mathcal{A}, \text{CTMAC}}^{1\text{-token-gen}, |\text{verify}\rangle}(\lambda)$ *with certainty.*

Proof. The attack is very similar to the one in [Lut10]. The idea is to learn the token one qubit at a time: given a token $|\triangleq\rangle$, to learn the ith qubit $\mathcal{A}$ would query the verification oracle with $|(0, X_i \triangleq)\rangle|0\rangle$ for $X_i \equiv I \otimes \ldots \otimes X \otimes \ldots I$, where X operates on the ith, and I is the identity. $\mathcal{A}$ would then measure the result register. If verification passed, the ith qubit is either $|+\rangle$ or $|-\rangle$, and the signature register submitted has not been damaged at all by the measurement. If verification failed, the ith qubit is either $|0\rangle$ or $|1\rangle$. The adversary then restores $|\triangleq\rangle$ by applying X_i. In either case, measuring the qubit on the correct basis that the attacker now knows would reveal the exact quantum state of the ith qubit of $|\triangleq\rangle$. $\mathcal{A}$ could then reassemble $|\triangleq\rangle$

and repeat the process above for all qubits, essentially uncovering the secret key. □

C.2 A Break of Strong Unforgeability

We can also consider a stricter definition regarding what counts as forgery. The security definition brought in Section 2.2.2 only guarantees the inability of an adversary provided with r token to produce $r + 1$ signatures for distinct fresh documents. A stronger security notion would prohibit even the creation of $r + 1$ fresh distinct signed documents, even if the documents themselves are not distinct, or not fresh. Often this discussion is null, as vanilla MAC schemes usually have unique signatures. The TMAC schemes we present, however, have exponentially many signatures for every document. In analogy to strong MAC schemes [KL14, Definition 4.3], we can thus also consider a strong variant of all of the above, and define corresponding security notions Strong-Unforgeable$^{(\cdot,\cdot,\cdot)}$, and security games Strong-Forge$^{(\cdot,\cdot,\cdot)}(\lambda)$.

Definition 17. Strong-Forge$^{(\cdot,\cdot,\cdot)}(\lambda)$ *differs from* Forge$^{(\cdot,\cdot,\cdot)}(\lambda)$ *(Game 2) by the following:*

1. *The set Q is defined to be all query-response pairs made to $\widetilde{\text{sign}}$, instead of only the queries.*

2. *count $= |\{(m_i, \sigma_i)|i \in S \wedge (m_i, \sigma_i) \notin Q\}|$.*

Proposition 11. *For all λ, there exists an adversary $\mathcal{A}$ winning* Strong-Forge$_{\mathcal{A},\text{CTMAC}}^{\text{1-token-gen,verify}}(\lambda)$ *with probability $\frac{1}{2}$.*

Proof. For a key $k \equiv (a, b)$ generated by $\mathcal{C}$, $\mathcal{A}$ receives a token $|\overset{\circ}{\rightleftharpoons}\rangle = H^b|a\rangle$, and measures the token in the standard basis to obtain a classical string σ, representing a signature for 0. There are, however, many other valid signatures this process could have generated. A change of any bit in σ corresponding to a coordinate i for which $b_i = 1$ would result in such a signature. On average, half the coordinates are such, and all an adversary must do to uncover a fresh signature, is pick one bit of the valid signature and perturb it. The adversary thus wins with a probability of $\frac{1}{2}$. □

D TMAC as a MAC

In this section, we show how a TMAC can function as an unforgeable MAC scheme, provided the TMAC is unforgeable against attacks with signing and verification oracles:

Theorem 13. *Let* TM *be an* Unforgeable$^{\widetilde{\text{sign}},\text{verify}}$ *(see Definition 5)* TMAC. *Then the* MAC *scheme* CMAC $=$ (TM.key-gen, TM.$\widetilde{\text{sign}}$, TM.verify) *derived from* TM*, is* CMA *unforgeable.*

The proof of this theorem follows by definition and, hence, is omitted.

It is tempting to argue that an Unforgeable$^{\text{token-gen},\text{verify}}$ TMAC is sufficient for the above theorem. An equivalent question is whether an Unforgeable$^{\text{token-gen},\text{verify}}$ TMAC is also an Unforgeable$^{\widetilde{\text{sign}},\text{verify}}$ TMAC. The naïve intuition being that any document signed by the signing oracle could simply be signed by an extra token instead. However, this turns out to be not necessarily true.

Proposition 12. *There exists an* Unforgeable$^{\text{token-gen},\text{verify}}$ TMAC *scheme which is not* Unforgeable$^{\widetilde{\text{sign}},\text{verify}}$.

Proof. Let Π be a deterministic verification Strong-Unforgeable$^{\text{token-gen},\text{verify}}$ scheme such that for some message m_1, there are at least two distinct signatures which $\widetilde{\text{sign}}$ generates with non-negligible probability (The scheme presented in [BS17] satisfies these requirements; Our scheme is not *strongly* unforgeable). We define a new TMAC scheme Π' which is the same as Π up to a modification to the verification algorithm. Fix $m_0 \neq m_1$. In addition to what Π.verify accepts, Π'.verify also accepts two distinct signatures for m_1 concatenated to each other, as a valid signature for m_0. Π' is clearly not Unforgeable$^{\widetilde{\text{sign}},\text{verify}}$. An adversary could easily request the oracle to sign m_1 twice, resulting (with non-negligible probability) in two distinct signatures for m_1. Concatenating those is a valid signature for m_0.

Next we claim that Π' is Unforgeable$^{\text{token-gen},\text{verify}}$. Suppose not, then there exists $\mathcal{A}$ asking for ℓ tokens, and submitting $(a_1, \sigma_1), \ldots, (a_{\ell+1}, \sigma_{\ell+1})$ such that with non-negligible probability all are valid signed documents, and the documents are all distinct. Let $p_0(\lambda)$ denote the probability that $\mathcal{A}$ succeeds, and among the signatures submitted there is a signature for m_0 which is not valid in Π, and $p_1(\lambda)$ the probability that $\mathcal{A}$ succeeds, but all submitted signatures are also valid in Π. Consider $\mathcal{B}$ is an adversary against Π, simulating $\mathcal{A}$ until it receives its final reply. With probability $p_1(\lambda)$, the signatures submitted by $\mathcal{A}$ are all also valid in Π, immediately implying $p_1(\lambda)$ is negligible. With probability $p_0(\lambda)$ and W.L.O.G, $a_{\ell+1} = m_0$ and $\sigma_{\ell+1}$ is a concatenation of two signatures for m_1. In such a scenario, (which is easily recognizable by $\mathcal{B}$), $\mathcal{B}$ could simply submit the two distinct signatures for m_1, alongside $(a_1, \sigma_1) \ldots (a_{\ell-1}, \sigma_{\ell-1})$. Here W.L.O.G., it is assumed that if there originally were some $a_i = m_1$, then it was the ℓth document. As all the signed documents submitted by $\mathcal{B}$ are valid and distinct, this comprises a win in Strong-Forge$_{\mathcal{B},\Pi}^{\text{token-gen},\text{verify}}$ (Definition 17); hence, $p_1(\lambda)$ must be negligible as well, contradicting the assumption that $p_0(\lambda) + p_1(\lambda)$ was non-negligible, and providing a separation between the security notions. □

The flaw with the intuitive reduction is that the documents queried to the signing oracle might not be distinct. The intuitive reduction can be

formalized if we can prevent this event from occurring.[24] In practice, any Unforgeable$^{\text{token-gen,verify}}$ TMAC can be used to construct an Unforgeable$^{\widetilde{\text{sign}},\text{verify}}$ TMAC, which is in fact, an Unforgeable$^{\text{token-gen},\widetilde{\text{sign}},\text{verify}}$ TMAC, as discussed in Appendix A.2.4.

E Unconditional Unforgeability for a Fixed Number of Tokens

In previous sections, we saw constructions of Unconditionally Unforgeable$^{\text{1-token-gen,verify}}$ and Unforgeable$^{\text{token-gen,verify}}$ TMACs. Moreover, we know by Theorem 10 that unconditional security cannot be achieved for polynomially many tokens. This leads us to an intermediate question of achieving unconditional security for a fixed constant number of tokens.

Definition 18. *For $j \in \mathbb{N}$,* Forge$^{\text{j-token-gen,verify}}_{\mathcal{A},\Pi}$ TMAC *is identical to* Forge$^{\text{1-token-gen,verify}}_{\mathcal{A},\Pi}$, *but the adversary in the security game is allowed to make up to j queries to* token-gen. *The corresponding security notion is called* Unforgeable$^{\text{j-token-gen,verify}}$ TMAC.

We use the same ideas as in Appendix A.2.3 to achieve this notion of security unconditionally, for length-restricted schemes. The idea is to replace (computationally) secure authenticated encryption used in Appendix A.2.3 with a direct application of the "encrypt then sign" paradigm, using MAC and encryption schemes that satisfy information-theoretic variants of secrecy and unforgeability respectively, for a fixed number of documents, i.e., up to j documents.

Definition 19. *For $j \in \mathbb{N}$, a deterministic encryption scheme $\Pi = (\text{key-gen}, \text{Enc}, \text{Dec})$ is said to be* statistically j-secure *if for any two j-tuples of messages $(m_1, m_2, ...m_j)$ and $(m'_1, m'_2, ...m'_j)$ and two j-tuples of cipher-texts $(c_1, c_2, ...c_j)$ and $(c'_1, c'_2, ...c'_j)s$*

$$\left| \Pr[\text{Enc}_k(m_1) = c_1, ...\text{Enc}_k(m_1) = c_1] - \Pr[\text{Enc}_k(m'_1) = c'_1, ...\text{Enc}_k(m'_1) = c'_1] \right| \le \text{negl}(\lambda),$$

where the probability is taken over the distribution of $k \leftarrow \text{key-gen}(1^\lambda)$, as well as the randomness of the encryption.

Definition 20. *For $j \in \mathbb{N}$, a* MAC *scheme $\Pi = (\text{key-gen}, \text{sign}, \text{verify})$ is said to be* Unconditionally j-Unforgeable *if for any λ, and every key k generated by*

---

[24]In [BS17], the authors point out that such events cannot occur except with negligible probability, in TMAC schemes that satisfy the strong variant of unforgeability as well as a property called unpredictability. Unpredictability means that if we sign (by $\widetilde{\text{sign}}$) a document twice, this results in two different signatures with overwhelming probability. While the second property is fulfilled by our construction, the first is not; hence, this approach is of little use in this work.

key-gen(1^λ) *and any* $j' < j$, *the probability of any computationally unbounded adversary in hold of* j' *signed document, cannot submit a signature for a fresh document except, with negligible probability.*

Remark 4. There exists a length-restricted deterministic MAC (with canonical verification) scheme satisfying Definition 20 ([WC81]). The computational cost of this scheme is polynomial in the security parameter, the length of the message and in j.

Remark 5. There exists a length-restricted encryption scheme satisfying Definition 19.[25] The computational cost of this scheme is polynomial in the security parameter, the length of the message and in j.

Lemma 9. *For any* $\ell \in \mathbb{N}$, *there is a noise-tolerance preserving lift (Definition 4) of an* ℓ-restricted Unconditionally Unforgeable$^{\text{1-token-gen,verify}}$ TMAC *(see Definitions 2 and 5), to an* ℓ-restricted Unconditionally Unforgeable$^{\text{j-token-gen,verify}}$ TMAC *(see Definition 18), as shown in Algorithm 22.*

The proof of security follows the same principles as Lemma 5. The resulting scheme obtained by instantiating TM with the specific scheme CTMAC$^{\eta}_{\text{1-time}}$ (Algorithm 17), is called CTMAC$^{\eta}_{\text{J-times}}$.

Nomenclature

α Approximately $\cos^2(\frac{\pi}{8})$, page 4

m A document or a message, that usually needs to be signed, page 14

r The number of signing tokens given to the adversary, page 3

σ A signature of a document, page 14

$|\stackrel{\triangle}{\underline{\ }}\rangle$ A quantum signing token, page 3

$[n]$ The set $\{1, \ldots, n\}$, page 13

$\binom{[n]}{j}$ The set of all j-sized subsets of $[n]$, page 13

Cons$_m$ The subset $\{i \in [\lambda] \mid b_i = m\}$, with regards to (a,b), page 20

Miss$_{m,\sigma}$ The subset $\{i \in \text{Cons}_m \mid \sigma_i \neq a_i\}$ with regards to (a,b), page 22

CTMAC   1-bit noise-sensitive Conjugate TMAC scheme, page 20

---

[25]The existence of such a scheme is a known folklore. The construction is as follows. A j-wise independent hash function h sampled as the key, the domain and range of which is same as the message space. The encryption of a message m is $(h(r) \oplus m, r)$, where r is sampled uniformly from the message space. See for example [Wic17] for a private case of this result.

Algorithm 22 TM - An Unconditionally Unforgeable$^{\text{j-token-gen,verify}}$ TMAC, for parameter j.

Assumes: OT is an ℓ-restricted TMAC, CMAC, CENC are families of encryption and MAC schemes respectively, where if key-gen is provided with the parameter j, the schemes constitute a j-Unforgeable MAC, and a j-Secret encryption, respectively. The domain of CENC is assumed to be the message space, and the domain of CMAC is assumed to be the cipher-text space CENC.

1: **procedure** key-gen$(1^\lambda, j)$
2:     **Return** $(\mathsf{CENC.key\text{-}gen}(1^\lambda, j), \mathsf{CMAC.key\text{-}gen}(1^\lambda, j))$.
3: **end procedure**

1: **procedure** token-gen$_k$
2:     Interpret $k = (k_{\mathsf{CENC}}, k_{\mathsf{CMAC}})$.
3:     $\kappa \leftarrow \mathsf{OT.key\text{-}gen}(1^\lambda)$.
4:     $\mathsf{c} \leftarrow \mathsf{CENC.Enc}_{k_{\mathsf{CENC}}}(\kappa)$, $|\widetilde{\underline{\Omega}}\rangle \leftarrow \mathsf{OT.token\text{-}gen}_\kappa$.
5:     **Return** $(|\widetilde{\underline{\Omega}}\rangle, \mathsf{c}, \mathsf{CMAC.sign}_{k_{\mathsf{CMAC}}}(\mathsf{c}))$.
6: **end procedure**

1: **procedure** sign$_{|\underline{\Omega}\rangle}(m)$
2:     Interpret $|\underline{\Omega}\rangle$ as $(|\widetilde{\underline{\Omega}}\rangle, \mathsf{c}, \mathsf{ksig})$.
3:     **Return** $(\mathsf{OT1.sign}_{|\widetilde{\underline{\Omega}}\rangle}(m), \mathsf{c}, \mathsf{ksig})$.
4: **end procedure**

1: **procedure** verify$_k(m, \sigma)$
2:     Interpret σ as $(s, \mathsf{c}, \mathsf{ksig})$, and k as $(k_{\mathsf{CENC}}, k_{\mathsf{CMAC}})$.
3:     $Res \leftarrow \mathsf{CMAC.verify}_{k_{\mathsf{CMAC}}}(\mathsf{c}, \mathsf{ksig})$.
4:     **if** $Res=0$ **then**
5:         **Return 0**.
6:     **else**
7:         $\kappa \leftarrow \mathsf{CENC.dec}_{k_{\mathsf{CENC}}}(\mathsf{c})$.
8:         **Return** $\mathsf{OT1.verify}_\kappa(m, s)$.
9:     **end if**
10: **end procedure**

CTMAC^η 1-bit noise-tolerant Conjugate TMAC scheme, page 21

$\widetilde{\mathsf{CTMAC}}^\eta_{\mathsf{Poly\text{-}times}}$ The result of the expansion of CTMAC^η to a full blown scheme, page 19

AENC Authenticated encryption scheme, page 8

η Error threshold, page 19

Forge Unforgeability game for a TMAC, page 16

$\mathsf{Forge}^{1\text{-}\mathsf{token\text{-}gen},\mathcal{VR}^{\eta*}}_{\mathcal{A},\mathsf{CTMAC}^\eta}$ A variant of the security game Forge, specific to the scheme CTMAC^η, page 27

H Hadamard gate, page 13

$\mathcal{I}$ A set of coordinates chosen by the adversary in the proof of Lemma 1, page 33

PPT Probabilistic Polynomial Time, page 14

QPT Quantum Polynomial Time, page 13

$\mathcal{VR}^\eta$ Verify and reveal oracle with parameter η , page 26

$\mathcal{VR}$ $\mathcal{VR}^0$, page 29

UOWHF Universal one-way hash functions, page 52