

Noise-Tolerant Quantum One-Time Programs Using Stateless Trusted Hardware, Revisited

Prabhanjan Ananth*
UCSB

Divyanshu Bhardwaj†
UCSB

Abstract

A quantum one-time program scheme is a functionality-preserving compiler that transforms a program into a quantum state that can be used to obtain only one output of the program. Our goal is to design quantum one-time programs that can be realized on existing quantum technologies. Our contributions are two-fold:

- We design a BB84 state-based quantum one-time program scheme in the stateless trusted hardware model that is resilient to single-qubit corruption errors and detectable erasure errors. To illustrate, under independent noise with a 90% detectable erasure rate and a 3% Pauli Y -error rate on surviving qubits, a signature token of 39,801 BB84 qubits achieves forging probability at most 2^{-128} against at most 2^{32} classical verification queries, with honest failure probability below 10^{-6} . Prior to our work, [Behera, Sattath, and Shinar TQC'22] studied a BB84 state-based quantum one-time program scheme resilient to independent single-qubit corruption errors; our construction treats detectable erasures separately.
- We give two extensions to our scheme. First, we study the setting wherein the same trusted hardware can be reused across different independently generated quantum one-time programs. Second, we study the setting wherein multiple trusted hardware devices are used and the security holds as long as at least one of them is not compromised.

*prabhanjan@cs.ucsb.edu

†dbhardwaj@ucsb.edu

Contents

1	Introduction	3
1.1	Overview	6
2	Preliminaries	12
2.1	Notation	12
2.2	Message Authentication Code	12
2.3	Quantum Computing	13
2.4	Noise Model	13
2.5	Noise-Tolerant Tokenized Digital Signature	14
2.6	Noise-Tolerant Quantum One-Time Programs	15
3	Noise-Tolerant Tokenized Digital Signatures	16
3.1	Correctness and Security	17
3.2	Concrete Parameters and Security	23
4	Noise-Tolerant Quantum One-Time Programs	24
4.1	Correctness and Security	26
5	Extensions	30
5.1	Reusable Trusted Hardware	30
5.2	Security against Compromised Hardware	36
6	AI Disclosure	47
7	Acknowledgements	47
A	Optimized Noise-Tolerant Scheme	50

1 Introduction

One-time programs, introduced by Goldwasser, Kalai and Rothblum [GKR08], are defined via a compiler that takes as input a program P and outputs a functionally equivalent program $\tilde{P}$ that can only be evaluated on one input. If $\tilde{P}$ has a classical description, then clearly this primitive cannot be achieved. Hence, there have been two different approaches undertaken to implement P :

- **Trusted hardware:** Trusted hardware allows for embedding secret functionality into hardware such that users only receive input-output access to the trusted hardware. There are two types of trusted hardware studied in the classical cryptography literature: stateful and stateless. The stateful hardware allows for a limited number of evaluations after which the user will not be able to use the hardware anymore. On the other hand, stateless trusted hardware allows for an unlimited number of evaluations.
 - [GKR08] showed that one-time programs can be achieved using one-time stateful hardware.
 - On the other hand, it can be seen that classical information alone cannot be combined with stateless trusted hardware to obtain one-time programs.

Stateful hardware used to enforce one-time access must prevent an adversary from restoring its pre-use state, for example by resetting the device or interrupting its power supply [DKMN16; BGZ21]. Stateless hardware keeps no record of previous interactions and avoids the need to maintain a protected record of prior use [BGZ21]; this motivates our use of stateless trusted hardware.

- **Quantum implementation:** The no-cloning principle of quantum mechanics stipulates that there does not exist a universal cloner that can clone quantum states. A sequence of works have leveraged this principle to investigate the feasibility and infeasibility of building quantum one-time programs (i.e., when the compilation and the evaluation procedures can be quantum algorithms). Barring the work of [BSS21] which develops noise-tolerant BB84-based signing tokens and uses stateless hardware to obtain noise-tolerant one-time memories (and hence one-time programs), the other works [CGLZ19; BS23; GLR+25; AB26] make use of other types of unclonable states which (a) are highly entangled and (b) require more resources to prepare the state (unlike BB84 states which are just prepare-and-measure). A few other works also consider noise, but they either only handle depth-bounded adversaries [Liu23], only achieve probabilistic one-time programs [RKB+18], or require the evaluator to run a complex quantum computation to generate semi-quantum tokens [Shm22; Sta25].

This raises the following question: *Can these BB84-based one-time program constructions tolerate high rates of detected erasures alongside errors on surviving qubits?* For instance, building one-time programs from BB84 states would bring us one step closer to implementation since we can repurpose existing quantum key distribution technologies to realize one-time programs. Towards addressing this question, two issues arise:

- **Plain-model barrier:** General simulation-secure one-time programs for deterministic classical functionalities cannot be realized by noninteractive protocols in the plain quantum model, even with computational security [BGS13]. This limitation also applies to BB84-based encodings and motivates the use of trusted hardware.

- **Handling noise:** Noise affects both quantum communication channels and local quantum operations. Practical cryptographic protocols must therefore tolerate noise in both settings. There are two types of noise that commonly arise:

- **Erasure errors:** some of the qubits sent over the communication channel could be lost forever. The erasure rate can be quite high in some settings, as high as even 80% to 90%. Fortunately, in some systems, it is possible to detect which positions were erased.
- **Corruption errors:** some qubits could be completely corrupted. Corruption rates are typically lower although the exact positions that are corrupted cannot typically be detected.

Our Results. Building on the noise-tolerant signing tokens and one-time programs of [BSS21], we handle detected erasures separately from corruption errors and explore one-time program constructions and hardware extensions based solely on BB84 states and stateless hardware. As mentioned earlier, stateless hardware cannot be combined with classical information alone. For general deterministic classical functionalities, removing the hardware would encounter the plain-model impossibility of simulation-secure quantum one-time programs [BGS13]. An evaluator can compute the output coherently, copy the classical result, and reverse the evaluation to recover the program state for further evaluations; trivial functionalities whose outputs can already be recovered from a single ideal query are exceptions.

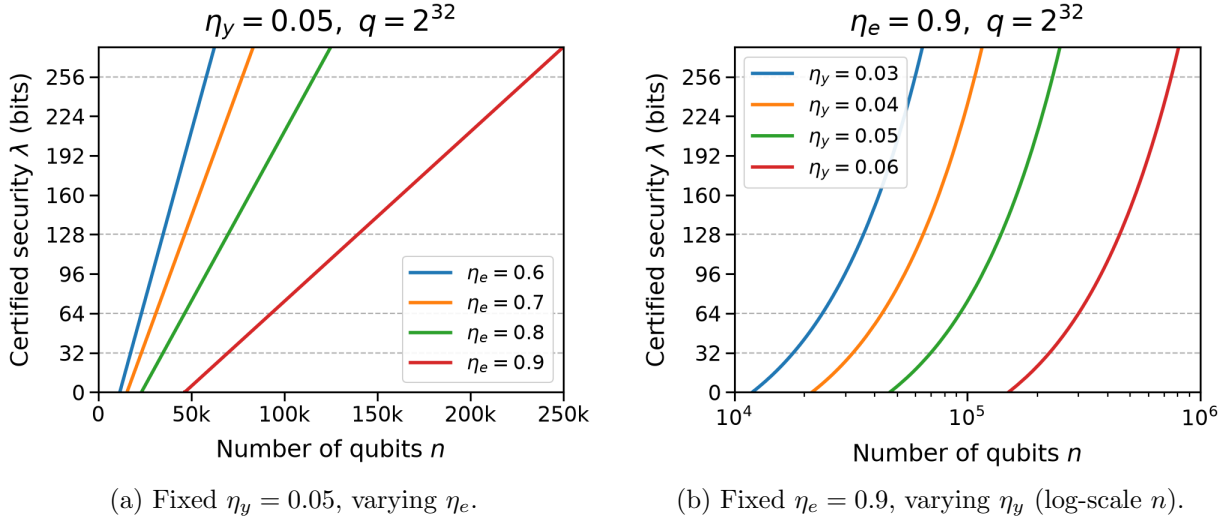


Figure 1: Certified security in bits (forging probability $\leq 2^{-\lambda}$) of the noise-tolerant tokenized signature scheme as a function of the token length n , using (3). The verifier thresholds are η_e for erasures and η_y for checked errors; the adversary makes at most $q = 2^{32}$ classical verification queries, including rejected queries.

Our schemes tolerate the two kinds of noise found in practical quantum channels and are parametrized by two thresholds $\eta_e \in [0, 1)$ and $\eta_y \in [0, (1 - \alpha)/2)$, where $\alpha = \cos^2(\pi/8)$. All our schemes use only BB84 states and stateless hardware, and hardware queries are classical throughout. We state the computational assumptions, where needed, with each result.

Table 1: Smallest signature token size n certified by (3) for $q = 2^{32}$ and forging probability at most 2^{-128} . The verifier thresholds η_e and η_y bound erasures and checked errors, respectively.

η_e	η_y			
	0.03	0.04	0.05	0.06
0.6	8,956	16,153	34,878	112,711
0.7	11,941	21,537	46,504	150,281
0.8	17,911	32,306	69,756	225,421
0.9	35,821	64,611	139,511	450,841

Intermediate construction: Noise-Tolerant Tokenized Signatures. We extend the noise-tolerant tokenized signature scheme of [BSS21] to handle detected erasures separately from errors in surviving qubits. The resulting scheme $\text{NT-TDS}(n, \eta_e, \eta_y)$ has signing tokens consisting of n BB84 qubits. The construction satisfies *one-time unforgeability* (Definition 2.2) for arbitrarily large erasure thresholds η_e , as long as $(1 - \eta_e)n = \omega(\log n)$ and $\eta_y < (1 - \alpha)/2$. We derive a concrete bound on the forging probability for any efficient adversary and plot the number of qubits n against bits of security in Figures 1a and 1b. We also give concrete token sizes certified for 128-bit forging security in Table 1. For verifier thresholds $(\eta_e, \eta_y) = (0.6, 0.05)$ and $(0.9, 0.05)$, our bound certifies $n = 34,878$ and $139,511$, respectively, against at most $q = 2^{32}$ classical verification queries. These sizes certify the forging bound; correctness at physical noise rates (p_e, p_y) must be checked separately for a specified honest-failure budget. This intermediate construction yields the following one-time program result.

Theorem 1.1 (Noise-Tolerant One-Time Programs (Informal)). *For any efficiently computable keyed function family with ℓ_{in} -bit inputs, there exists a noise-tolerant one-time program scheme $\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ whose quantum tokens consist of $n\ell_{\text{in}}$ BB84 qubits. Assuming $\text{NT-TDS}(n, \eta_e, \eta_y)$ is one-time unforgeable, the scheme is simulation secure against malicious evaluators making polynomially many queries to the stateless hardware.*

Intermediate construction: Reusable Hardware for Tokenized Signatures. To support many signing tokens with hardware initialized only once, we construct $\text{rNT-TDS}(n, \eta_e, \eta_y)$ from post-quantum one-way functions. The scheme uses a single stateless hardware device and supports any polynomial number of signing tokens. It satisfies *many-token unforgeability* (Definition 5.1), assuming the single-token scheme $\text{NT-TDS}(n, \eta_e, \eta_y)$ is one-time unforgeable. This intermediate construction leads to reusable hardware for one-time programs.

Theorem 1.2 (Noise-Tolerant One-Time Programs with Reusable Hardware (Informal)). *Assuming post-quantum one-way functions, there exists a noise-tolerant one-time program scheme $\text{rNT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ using a single stateless hardware device that supports any polynomial number of one-time program tokens. Assuming the single-token scheme $\text{NT-TDS}(n, \eta_e, \eta_y)$ is one-time unforgeable, $\text{rNT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ is simulation secure against any QPT adversary given $N = \text{poly}(n)$ one-time program tokens.*

Theorem 1.3 (Noise-Tolerant One-Time Programs with Security against Compromised Hardware (Informal)). *Assume a constant-round post-quantum secure MPC protocol in the common reference string (CRS) model, a statistically binding and computationally hiding commitment scheme, a collision-resistant hash function, an unforgeable (EUF-CMA) classical signature scheme, and a one-time unforgeable scheme $\text{NT-TDS}(n, \eta_e, \eta_y)$. There exists a noise-tolerant one-time program scheme $\text{dNT-QOTP}(n, t, \ell_{\text{in}}, \eta_e, \eta_y)$ that distributes its secret among t hardware devices and is simulation secure against a malicious evaluator corrupting all but one of them (Definition 5.7).*

We discuss the main technical ideas of our work next.

1.1 Overview

Our starting point is the conjugate tokenized message authentication code (CTMAC) of [BSS21]. A tokenized message authentication scheme is a one-time program for a signing functionality. That is, a signing functionality is compiled into a quantum state such that this state can only be used to compute a MAC signature on a single message.

For simplicity, let us first see how to design a quantum one-time program from CTMAC under the assumption that there is no noise. The construction is simple:

- To compile a program P of input length ℓ_{in} into a one-time program, first, prepare a signature token $|\text{sk}\rangle$. The trusted hardware contains a program that takes as input (x, σ) , checks if σ is a valid signature on x and only then it outputs $P(x)$. The signature token $|\text{sk}\rangle$ along with the trusted hardware together will become the one-time program.
- To evaluate on an input x , first sign x using the signature token to obtain (x, σ) . Then evaluate the trusted hardware on (x, σ) to obtain the output $P(x)$.

The correctness and security follow from the corresponding properties of the tokenized MAC scheme.

[BSS21] already provide noise-tolerant CTMAC and one-time program constructions. Our task is to extend their approach to handle detected erasures separately from errors on surviving qubits. We first recall the CTMAC scheme, which consists of key generation, signing and verification.

- **Key Generation.** Sample $\theta, x \xleftarrow{\$} \{0, 1\}^n$. The secret key is (θ, x) and the signing token is the BB84 state $|x^\theta\rangle = \otimes_{i \in [n]} H^{\theta_i} |x_i\rangle$.
- **Signing.** To sign $\mu \in \{0, 1\}$, measure every qubit of the token in the computational basis if $\mu = 0$ and in the Hadamard basis if $\mu = 1$. The measurement output $\sigma \in \{0, 1\}^n$ is the signature.
- **Verification.** Given (μ, σ) and the secret key (θ, x) , count the mismatches between σ and x at the positions where $\theta_i = \mu$

$$\text{Miss}_\mu(\sigma) = \{i \in [n] : \theta_i = \mu \wedge \sigma_i \neq x_i\}$$

and accept if and only if $|\text{Miss}_\mu(\sigma)| = 0$.

An honest signer who measures in basis μ recovers x_i exactly on the positions with $\theta_i = \mu$ and $|\text{Miss}_\mu(\sigma)| = 0$. A forger, on the other hand, must produce two signatures, one that matches x at all positions where $\theta_i = 0$ and the other that matches x at positions with $\theta_i = 1$ without

knowing which positions are which. For a single qubit, the best probability of answering correctly in whichever basis is later checked is at most $\alpha = \cos^2(\pi/8)$. So the probability of answering correctly for n -qubit signing token is α^n . [BSS21] also lift their scheme to noise-tolerant CTMAC_{η_y} that can handle Pauli Y errors by relaxing the verification to accept as long as $|\text{Miss}_\mu(\sigma)| \leq \eta_y n$, i.e., at most η_y fraction of positions, where $\theta_i = \mu$, in σ, x are a mismatch. The forging probability for CTMAC_{η_y} decays exponentially in n as long as $\eta_y < (1 - \alpha)/2 \approx 0.073$ given polynomially many queries to the verification oracle.

Handling erasure errors. CTMAC_{η_y} only handles Y errors on the qubits that arrive but does not deal with the erasures at all. In photonic implementations, however, the dominant noise is loss: standard telecom fiber attenuates at roughly 0.2 dB/km [PLOB17; XMZ+20], so about 90% of transmitted photons are lost after only 50 km and 99% after 100 km, before even accounting for imperfect detector efficiency [XMZ+20]. A signer cannot measure a lost qubit, so it must guess the corresponding bit with probability 1/2. Suppose a qubit is erased with probability p_e and if it is not erased, it has Y error with probability p_y . The noise model is formalized in Section 2.4. An honestly generated signature will have about $(p_e/4 + (1 - p_e)p_y/2)n$ mismatches. Security of CTMAC_{η_y} requires this threshold to be less than $(1 - \alpha)/2 \approx 0.073$, so treating erasures simply as Y errors fails once $p_e \geq 0.3$, even if we consider $p_y = 0$. This is a major challenge in deploying the tokenized signature scheme on current quantum hardware and suggests that erasures have to be handled separately. We discuss our idea to handle arbitrarily large erasures next.

Unlike Y errors, erasures can be detected: the receiver knows which positions were lost. The signer (receiver) can report which qubits were erased by sending an erasure mask $M \in \{0, 1\}^n$ such that $M_i = 1$ if qubit i was not received and $M_i = 0$ otherwise. Let $\text{HW}(M)$ be the Hamming weight of M . Verification can ignore the erased positions and normalize the threshold of Y errors tolerated in signature by the number of surviving positions. It accepts (μ, σ, M) if and only if

$$\text{HW}(M) \leq \eta_e n, \quad |E_\mu^M(\sigma)| \leq \eta_y(n - \text{HW}(M)), \quad E_\mu^M(\sigma) = \{i \in [n] : \theta_i = \mu \wedge M_i = 0 \wedge \sigma_i \neq x_i\}$$

Since the honest signature has $p_y(n - \text{HW}(M))/2$ mismatches in expectation, coming from the Y error on surviving qubits, the verification will pass with overwhelming probability as long as $p_y < 2\eta_y$, $p_e < \eta_e$ and the erasure rate no longer consumes the error budget. An arbitrarily large erasure rate can be tolerated as long as $(1 - p_e)n = \omega(\log n)$.

Binding the erasure mask. The erasure mask $M \in \{0, 1\}^n$ is supplied by the signer, who may be adversarial, and it could choose a mask in such a way that the security could be compromised. Suppose a forger could choose a different mask for each signature. It measures the first half of the token in the computational basis and the second half in the Hadamard basis. To sign 0, it reports the second half of token as erased. To sign 1, it reports the first half erased. Each signature then has no mismatches on the positions that are checked so both are accepted (provided $\eta_e \geq 1/2$). The mask must therefore be fixed once, before any signature is computed. To do this, we make the token delivery an interactive algorithm called **TokenSend**. The generator sends the qubits, and the signer replies with M . The generator aborts if $\text{HW}(M) > \eta_e n$, otherwise it returns a MAC tag τ on M . The verification oracle only accepts signatures that carry M along with its valid tag τ . Since only one message M is authenticated and verification queries are classical, an information theoretic one-time MAC suffices. A forger making q queries to the verification oracle produces a valid tag

on a different erasure mask with probability at most $q2^{-n}$. The threshold η_e ensures that every accepted mask M leaves at least $L_0 = n - \lfloor \eta_e n \rfloor = \omega(\log n)$ surviving positions, and the security depends on L_0 rather than on n . Let $\mathcal{N}$ be a noisy quantum channel. Our noise-tolerant tokenized signature scheme $\text{NT-TDS}(n, \eta_e, \eta_y)$ is as follows:

- **Key Generation.** Sample $\theta, x \xleftarrow{\$} \{0, 1\}^n$ and MAC key $\text{sk} \leftarrow \text{MAC.KeyGen}(1^n)$.
- **Token Send.** The generator sends the BB84 state $|x^\theta\rangle$ qubit by qubit via quantum channel $\mathcal{N}$. The signer sends $M \in \{0, 1\}^n$. If $\text{HW}(M) \leq \eta_e n$, generator sends tag $\tau \leftarrow \text{MAC.Tag}(\text{sk}, M)$ over classical communication channel.
- **Signing.** To sign a message $\mu \in \{0, 1\}$, the signer computes $\sigma = \sigma_1 \| \dots \| \sigma_n \in \{0, 1\}^n$ where $\sigma_i = 0$ if $M_i = 1$ or σ_i is the output of measurement of i^{th} qubit of signing token in computational basis ($\mu = 0$) or Hadamard basis ($\mu = 1$).
- **Verification.** Given (μ, σ, M, τ) and (θ, x, sk) , accept if and only if (1) $\text{HW}(M) \leq \eta_e n$, (2) $\text{MAC.Verify}(\text{sk}, M, \tau) = 1$ and (3) $|E_\mu^M(\sigma)| \leq \eta_y(n - \text{HW}(M))$.

When $\eta_e = 0$, the only admissible mask is $M = 0^n$ and the generator can send the tag τ on M along with the signing token in one shot, making **TokenSend** non-interactive. This special case $\text{NT-TDS}(n, 0, \eta_y)$ is exactly the CTMAC_{η_y} scheme of [BSS21]. $\text{NT-TDS}(n, \eta_e, \eta_y)$ is formally defined in Definition 2.2 and the construction is given in Section 3.

Reducing unforgeability to the erasure-free setting. We say that $\text{NT-TDS}(n, \eta_e, \eta_y)$ is one-time unforgeable if no QPT adversary given a signing token and classical query access to the verification oracle, can output valid signatures on both $\mu = 0$ and 1. Once M is fixed, the erased positions are never checked in verification. A forging adversary $\mathcal{A}$ against $\text{NT-TDS}(n, \eta_e, \eta_y)$ is therefore effectively forging $\text{NT-TDS}(L, 0, \eta_y)$ on the $L = n - \text{HW}(M) \geq L_0 = n - \lfloor \eta_e n \rfloor$ surviving qubits. We reduce the one-time unforgeability of $\text{NT-TDS}(n, \eta_e, \eta_y)$ to one-time unforgeability of $\text{NT-TDS}(L, 0, \eta_y)$ for every $L \geq L_0$. The noise-tolerant scheme with $\eta_e = 0$ is exactly the conjugate tokenized MAC scheme CTMAC_{η_y} from [BSS21]. It is proved in [BSS21] that CTMAC_{η_y} with $L \geq L_0$ qubits is one-time unforgeable for $L_0 = \omega(\log n)$ and $\eta_y < (1 - \alpha)/2$ ($\alpha = \cos^2(\pi/8)$). This is enough to conclude that $\text{NT-TDS}(n, \eta_e, \eta_y)$ is also one-time unforgeable for $(1 - \eta_e)n = \omega(\log n)$, $\eta_y < (1 - \alpha)/2$. We additionally improve the analysis of [BSS21] and derive better concrete bounds on the forging advantage for CTMAC_{η_y} in Appendix A. Finally, we give a concrete comparison of the number of qubits n in the token and the bits of security for different erasure and Y error thresholds in Section 3.2.

Consider a forging adversary $\mathcal{A}$ of $\text{NT-TDS}(n, \eta_e, \eta_y)$ that makes at most q classical queries to the verification oracle and outputs $\{(\mu_b, \sigma_b, M_b, \tau_b)\}_{b \in \{0, 1\}}$ such that $\mu_0 \neq \mu_1$ and both pass verification. Let $\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n)$ denote the probability that $\mathcal{A}$ succeeds. Let M be the erasure mask authenticated in **TokenSend** and $L = n - \text{HW}(M) \geq L_0$ the number of surviving qubits. The proof can be divided into four parts.

1. *Fixing the mask.* By the unforgeability of MAC, all accepted queries and both final outputs of $\mathcal{A}$ carry the mask M authenticated in **TokenSend**, except with probability at most $(q + 2)2^{-n}$. We therefore move to a hybrid in which verification rejects every tuple carrying a mask $M' \neq M$. From then on, the erased positions are fixed and never checked.

2. *Embedding an erasure-free token.* For a fixed L , we would like to construct an adversary $\mathcal{B}_L$ for NT-TDS($L, 0, \eta_y$) that uses $\mathcal{A}$ to output valid signatures on $\mu_0 \neq \mu_1$ with respect to the verification oracle of NT-TDS($L, 0, \eta_y$). The natural approach is to embed the L -qubit challenge token in the n -qubit token given to $\mathcal{A}$, at a set of positions $\mathcal{I}$, and to generate random BB84 states on the remaining $n - L$ positions. $\mathcal{B}_L$ can extract a forgery for NT-TDS($L, 0, \eta_y$) from $\mathcal{A}$'s forgery only if $\mathcal{I} \cap M = \emptyset$. The difficulty is that $\mathcal{B}_L$ has to fix $\mathcal{I}$ when sending the token and learns M only afterward, so $\mathcal{I} \cap M = \emptyset$ happens with negligible probability. To get around this, we use EPR pairs and teleportation so that $\mathcal{B}_L$ can embed the challenge after M is announced.

1. $\mathcal{B}_L$ prepares n EPR pairs on registers $(A_1, R_1), \dots, (A_n, R_n)$, sends $A_1, \dots, A_n$ to $\mathcal{A}$, and keeps $R_1, \dots, R_n$.
2. On receiving M with $\text{HW}(M) = n - L$, $\mathcal{B}_L$ obtains the challenge token $|x^{\theta'}\rangle$ on registers $C_1, \dots, C_L$ from the NT-TDS($L, 0, \eta_y$) challenger. Let $\varphi(j)$ be the j -th surviving position. For each j , $\mathcal{B}_L$ performs a Bell measurement on $(C_j, R_{\varphi(j)})$. This teleports the j -th challenge qubit into $A_{\varphi(j)}$, up to a Pauli correction $X^{a_j}Z^{b_j}$ that $\mathcal{B}_L$ learns.
3. For each erased position i , $\mathcal{B}_L$ measures R_i in a random basis θ_i and obtains x_i . This leaves A_i in the state $H^{\theta_i}|x_i\rangle$.

Two observations make this a perfect simulation. First, $\mathcal{B}_L$'s operations act only on registers that $\mathcal{A}$ never touches, so they commute with operations $\mathcal{A}$ performs on its registers. $\mathcal{A}$'s view is therefore identical to the case where the teleportations and measurements happened before $\mathcal{A}$ received anything. Second, Pauli corrections map BB84 states to BB84 states in the same basis:

$$X^a Z^b H^\theta |x\rangle = H^\theta |x \oplus c_\theta\rangle \quad (\text{up to a global phase}), \quad c_0 = a, \quad c_1 = b.$$

Hence $\mathcal{A}$ holds $|x^\theta\rangle$ with $\theta_{\varphi(j)} = \theta'_j$ and $x_{\varphi(j)} = x'_j \oplus c_{\varphi(j), \theta'_j}$, which is a uniformly random BB84 token exactly as in the real experiment. Since $\mathcal{B}_L$ knows the corrections, it can translate signatures. Given a signature σ on μ under mask M , it forwards σ' with $\sigma'_j = \sigma_{\varphi(j)} \oplus c_{\varphi(j), \mu}$. Position j is a mismatch for σ' exactly when position $\varphi(j)$ is a mismatch for σ . So the two mismatch sets are in bijection, and the thresholds agree because $\eta_y(n - \text{HW}(M)) = \eta_y L$. Every accepted query and every forgery of $\mathcal{A}$ thus becomes one for NT-TDS($L, 0, \eta_y$), with the same number of queries.

3. *Averaging over the number of surviving positions.* The mask, and hence L , is chosen by $\mathcal{A}$ and can be anything with $\text{HW}(M) \leq n - L_0$. Rather than having the reduction guess $L \xleftarrow{\$} \{L_0, \dots, n\}$, which would lose a factor of $n - L_0 + 1$, we use the fact that everything before the embedding step is independent of the fresh L -qubit challenge. This includes the mask M , the registers $R_1, \dots, R_n$, and $\mathcal{A}$'s residual state. Conditioned on $n - \text{HW}(M) = L$, this state can be given to $\mathcal{B}_L$ as a challenge-independent auxiliary state. The bound on forging advantage for NT-TDS($L, 0, \eta_y$) holds information-theoretically for any such auxiliary state, so no efficient postselection is needed. If $\varepsilon_{\eta_y}(L, q)$ bounds the forging probability against NT-TDS($L, 0, \eta_y$), then $\Pr[\mathcal{A} \text{ wins} \mid n - \text{HW}(M) = L] \leq \varepsilon_{\eta_y}(L, q)$. Since $\varepsilon_{\eta_y}(L, q)$ decreases in L , averaging over $L \geq L_0$ bounds the forging probability by $\varepsilon_{\eta_y}(L_0, q)$, plus the MAC term.

4. *Improved security analysis for the erasure-free scheme.* When $\eta_e = 0$, the MAC plays no role, and NT-TDS($L, 0, \eta_y$) is exactly CTMAC $_{\eta_y}$ of [BSS21] on L qubits. Rather than invoking their asymptotic statement, we refine the concrete analysis of their reduction (Appendix A). [BSS21] reduces a noise-tolerant forger to a noiseless forger on a random subset of m positions that avoids all

checked errors, and then fixes a single m . We keep the exact sampling loss for every $m \in \{0, \dots, L\}$ and average the resulting inequalities with binomial weights. This gives

$$\varepsilon_{\eta_y}(L, q) = 2 \binom{q+4}{2} \exp\{-L D(2\eta_y \parallel \delta)\}, \quad \delta = 1 - \alpha,$$

where $D(a \parallel b) = a \ln \frac{a}{b} + (1-a) \ln \frac{1-a}{1-b}$ is the binary relative entropy. Combining everything above, we have for every adversary $\mathcal{A}$ making at most q classical queries:

$$\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n) \leq 2 \binom{q+4}{2} e^{-L_0 D(2\eta_y \parallel \delta)} + (q+2)2^{-n},$$

which is negligible for polynomial q , $\eta_y < (1 - \alpha)/2$ and $L_0 = \omega(\log n)$. We formally prove the one-time unforgeability of NT-TDS(n, η_e, η_y) in Section 3.1.

Noise-Tolerant Quantum One-Time Programs. We define noise-tolerant quantum one-time programs NT-QOTP($n, \ell_{\text{in}}, \eta_e, \eta_y$) in Definition 2.5 and give a construction from our noise-tolerant tokenized signature scheme in Section 4. For a function f with input $v \in \{0, 1\}^{\ell_{\text{in}}}$, the generator can send ℓ_{in} signing tokens of NT-TDS(n, η_e, η_y) via `TokenSend` also authenticating an erasure mask $(\{M_i\}_i, \tau)$, one for each bit of the input as well as a stateless hardware $\mathcal{O}_k$. The evaluator can measure i^{th} token according to the bit v_i and query $\mathcal{O}_k$ on $(v, \{r_i\}_i, \{M_i\}_i, \tau)$. $\mathcal{O}_k$ checks that τ is valid tag for $\{M_i\}_i$ and for all $i \in [\ell_{\text{in}}]$, r_i is a valid noise-tolerant signature on v_i . If these checks pass, $\mathcal{O}_k$ outputs $f_k(v)$.

We define simulation security for a malicious evaluator. It stipulates that for all QPT adversaries $\mathcal{A}$, there exists a QPT `Sim` that can simulate the output of $\mathcal{A}$ given only classical query access to one-time oracle $f_{k,1}$. $f_{k,1}$ outputs $f_k(v)$ on the first query and outputs $\perp$ on every subsequent query. We formally prove the simulation security in Section 4.

Towards Deployment. Our constructions of NT-TDS(n, η_e, η_y) and NT-QOTP($n, \ell_{\text{in}}, \eta_e, \eta_y$) treat the trusted hardware as an idealized component which is instantiated to verify only a single signature or compute a single function value. For many applications in practice, it could be desirable that the hardware is initialized once and the generator only issues fresh quantum tokens for each signing/evaluation. Moreover, our security proofs model the hardware as an ideal classical oracle. Deployed TEEs such as Intel SGX, AMD SEV and ARM TrustZone fall short of this ideal. They have been subject to a long line of attacks, including transient-execution attacks [BMW+18; CCX+19; BMS+20], software-based fault injection [MOG+20], voltage glitching [BJKS21], architectural and microarchitectural data leakage [SKGY20; BKS+22], ciphertext side channels [LZW+21], and physical memory interposition [DOVV26]; see [NBB20; CSFP20] for surveys. Several of these attacks extract long-term sealing or attestation keys, or arbitrary enclave memory, and thus completely expose the enclave’s secrets. In our constructions, this could be fatal: an adversary that knows (θ, x) or k can evaluate the function f_k everywhere.

We therefore discuss extensions of our protocols that address these issues in Section 5.

Extension: Reusable Trusted Hardware. In the single token signature construction, the BB84 secrets and the MAC key are hard-wired into the hardware. The obvious way to serve many

tokens with one trusted hardware, reusing a single secret, is fatal: given two copies of $|x^\theta\rangle$, an adversary measures one in each basis and can sign both bits. Instead, the hardware stores PRF keys (K, K') . Each session starts with a fresh seed s , sent in the clear. The session's BB84 secret is derived pseudorandomly $(\theta, x) = F_K(s)$, and the mask is authenticated as $\tau = G_{K'}(s, M)$, which binds the mask to the seed. On a query, the hardware recomputes $F_K(s)$ and runs the single-token check. Hardware queries are classical, so F and G only need to be secure against quantum adversaries making classical queries, which follows from post-quantum one-way functions [Zha21]. The security requirement, which we call *many-token unforgeability*, is that any efficient adversary with N signing tokens should not be able to output signatures on $N + 1$ messages. Similarly, for one-time programs, we require that any efficient adversary with N quantum tokens should be simulatable given query access to $f_{k,N}$, an oracle that outputs $f_k(\cdot)$ for at most N queries.

We give the construction and proof of security of reusable variants $\text{rNT-TDS}(n, \eta_e, \eta_y)$ and $\text{rNT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ that support (unbounded) polynomially many quantum tokens with a single hardware device in Section 5.1 from post-quantum one-way functions.

Extension: Security against Compromised Hardware. In our single token one-time program construction, compromising the single trusted hardware is fatal: the hardware stores k , and an adversary who learns k can evaluate f_k everywhere. To mitigate this issue, we would like to split trust across t trusted hardware devices, ideally from different vendors or architectures, so that the one-time program stays simulation secure even if the adversary learns the full internal state of any $t - 1$ of them. We denote this primitive by $\text{dNT-QOTP}(n, t, \ell_{\text{in}}, \eta_e, \eta_y)$.

The idea is to use a $t + 1$ party MPC protocol Π secure against corruption of t parties. The generator XOR-shares $k = k_1 \oplus \dots \oplus k_t$. Party P_i has input k_i , its random tape rand_i and some public parameters. In each round r , each party P_i computes $\text{NxtMsg}(r, k_i, \Gamma_{<r}, \text{rand}_i)$ and broadcasts it to all other parties. Here $\Gamma_{<r}$ is the transcript of all messages exchanged until the previous round. The generator can generate Rt one-time programs, one each for party $P_{i \in [t]}$ and round $r \in [R]$ implementing $\text{NxtMsg}(r, k_i, \cdot, \text{rand}_i)$. The evaluator gets these one-time programs and plays the role of party P_{t+1} with input v . The evaluator can simulate the protocol Π : it computes its own messages, relays all messages between the trusted hardware devices to learn the next messages of all parties, and computes the output from the final transcript. The one-time program security ensures that the evaluator cannot learn more than one message of each party in each round. Now any malicious evaluator that corrupts $t - 1$ tokens controls t parties in Π and we invoke the simulation security of Π to show that any such adversary is simulatable. Note that this is a very high level intuition of our construction and the actual construction (Section 5.2) is much more involved as one-time guarantees on the next message functions alone are not enough to compile a malicious evaluator into an adversary for Π . We defer the details to the main section.

By instantiating Π with a constant-round post-quantum secure MPC protocol in the CRS model that satisfies some additional properties (Definition 5.8), we achieve simulation security against a malicious evaluator that learns the complete description, including secrets, of at most $t - 1$ hardware devices. Since a BB84 token has to be issued for each round and party, we require Π to be constant round so that the total quantum communication only depends on t . We present our construction of $\text{dNT-QOTP}(n, t, \ell_{\text{in}}, \eta_e, \eta_y)$ and proof of simulation security in Section 5.2.

2 Preliminaries

2.1 Notation

The security parameter is denoted by $n \in \mathbb{N}$. The set of binary strings of length $\ell \in \mathbb{N}$ is denoted by $\{0, 1\}^\ell$. A binary string $x \in \{0, 1\}^\ell$ can be decomposed as $x = x_1 x_2 \dots x_\ell$ where $x_i \in \{0, 1\}$ for all $i \in [\ell]$. The Hamming weight of a string x is denoted by $\text{HW}(x)$. A uniformly random sample x from a set X is denoted by $x \xleftarrow{\$} X$. $\text{negl}(\cdot)$ denotes a negligible function and similarly, $\text{poly}(\cdot)$ denotes a polynomial.

2.2 Message Authentication Code

We use a *classical* one-time message authentication code (MAC) in our construction of noise-tolerant tokenized signatures.

Definition 2.1. A one-time message authentication code MAC is a tuple of PPT algorithms $(\text{KeyGen}, \text{Tag}, \text{Verify})$:

- $\text{KeyGen}(1^n, m) \rightarrow \text{sk}$: Takes as input security parameter n , message length m and outputs an authentication key sk .
- $\text{Tag}(\text{sk}, \mu) \rightarrow \tau$: Takes as input authentication key sk , message $\mu \in \{0, 1\}^m$ and outputs a tag τ .
- $\text{Verify}(\text{sk}, \mu, \tau) \rightarrow \{0, 1\}$: Takes as input authentication key sk , message μ , tag τ and outputs 1 (accept) or 0 (reject).

The scheme satisfies the following properties:

- **Correctness:** for all $\mu \in \{0, 1\}^m$ and $\text{sk} \leftarrow \text{KeyGen}(1^n, m)$, we have $\text{Verify}(\text{sk}, \mu, \text{Tag}(\text{sk}, \mu)) = 1$.
- **EUF-1CMA with access to verification oracle:** Consider the following experiment between adversary $\mathcal{A}$ with initial state ρ_0 and challenger Ch:
 1. Ch computes $\text{sk} \leftarrow \text{KeyGen}(1^n, m)$ and gives classical query access to verification oracle $\text{Verify}(\text{sk}, \cdot)$ to $\mathcal{A}$.
 2. $\mathcal{A}$ sends $\mu \in \{0, 1\}^m$.
 3. Ch computes $\tau \leftarrow \text{Tag}(\text{sk}, \mu)$ and sends τ to $\mathcal{A}$.
 4. $\mathcal{A}$ outputs (μ^*, τ^*) .
 5. The experiment outputs 1 if $\mu^* \neq \mu$ and $\text{Verify}(\text{sk}, \mu^*, \tau^*) = 1$. Otherwise the experiment outputs 0.

Let $\text{MACForge}_{\mathcal{A}}(n)$ be the probability that the experiment outputs 1. For all (unbounded) $\mathcal{A}$ that make at most $q = \text{poly}(n)$ queries to the verification oracle and all initial states ρ_0 , $\text{MACForge}_{\mathcal{A}}(n) = \text{negl}(n)$.

Such a MAC scheme can be realized information theoretically. $\text{KeyGen}(1^n, m)$ samples $2m$ random strings $\tau_{i,b} \xleftarrow{\$} \{0,1\}^n$ and outputs $\text{sk} = (\tau_{i,b})_{i \in [m], b \in \{0,1\}}$. The tag on any message $\mu \in \{0,1\}^m$ will be $\tau = (\tau_{i,\mu_i})_{i \in [m]}$ and Verify simply checks that tag corresponds to the message given sk . Any adversary making $q = \text{poly}(n)$ queries to the verification oracle can forge a tag on $\mu^* \neq \mu$ with at most $(q+1)2^{-n} = \text{negl}(n)$ probability. This scheme is essentially Lamport signature scheme without one-way functions and we have $|\text{sk}| = 2mn, |\tau| = mn$ bits.

2.3 Quantum Computing

We assume that the reader is familiar with the basics of quantum information. Let $\mathbf{R}$ be a quantum register (or a quantum system). We denote the Hilbert space associated with $\mathbf{R}$ (or the system) by $\mathcal{H}_{\mathbf{R}}$. The dimension of the Hilbert space is denoted by $d_{\mathbf{R}} = \dim(\mathcal{H}_{\mathbf{R}})$.

The state of a system associated with the Hilbert space $\mathcal{H}$ is described by a density operator ρ which is a positive semidefinite linear operator on $\mathcal{H}$ with unit trace and $\rho = \rho^\dagger$. The space of density operators on $\mathcal{H}$ is denoted by $\mathcal{D}(\mathcal{H})$. A state ρ is *pure* if $\rho = |\psi\rangle\langle\psi|$ for some $|\psi\rangle \in \mathcal{H}$ and *mixed* otherwise. Alternatively, a state ρ is pure if and only if $\text{Tr}(\rho^2) = 1$. For a bipartite state $\rho_{AB} \in \mathcal{D}(\mathcal{H}_{AB})$, the *reduced state* on subsystem B is given by the partial trace $\rho_B = \text{Tr}_A(\rho_{AB})$.

The evolution of a closed quantum system is described by a unitary operator $U : \mathcal{H} \rightarrow \mathcal{H}$, i.e., a linear operator satisfying $U^\dagger U = UU^\dagger = \mathbb{I}$. If the system is in the pure state $|\psi\rangle \in \mathcal{H}$, its state after the evolution is $U|\psi\rangle$. More generally, a system in state $\rho \in \mathcal{D}(\mathcal{H})$ evolves to $U\rho U^\dagger$. A *quantum channel* is a completely positive trace preserving map $\phi : \mathcal{D}(\mathcal{H}_A) \rightarrow \mathcal{D}(\mathcal{H}_B)$ that describes the most general physically admissible evolution of a quantum system.

2.4 Noise Model

We model noise during the transmission of the quantum state by a memoryless quantum channel with parameters $p_e, p_y \in [0,1)$. The input space is $\mathcal{H}_{\text{in}} = \mathbb{C}^2$ and the output space is $\mathcal{H}_{\text{out}} = \mathbb{C}^2 \oplus \text{span}\{|\perp\rangle\}$ where $|\perp\rangle$ is orthogonal to the qubit subspace $\mathcal{H}_{\text{in}}$. We define the noise channel $\mathcal{N}_{p_e, p_y} : \mathcal{D}(\mathcal{H}_{\text{in}}) \rightarrow \mathcal{D}(\mathcal{H}_{\text{out}})$ as:

$$\mathcal{N}_{p_e, p_y}(\rho) = p_e \text{Tr}(\rho)|\perp\rangle\langle\perp| + (1 - p_e)p_y Y \rho Y^\dagger + (1 - p_e)(1 - p_y)\rho$$

where Y is the Pauli Y operator and $\text{Tr}(\rho) = 1$ when ρ is a density operator. Thus p_e is the erasure probability and p_y is the probability of a Y error *conditioned on non-erasure*. p_e and p_y are properties of the channel and independent of the quantum token being transmitted. Informally, the application of the channel $\mathcal{N}$ can be interpreted as follows: upon receiving a single qubit state ρ ,

- With probability p_e , replace that qubit by the state $|\perp\rangle$.
- Else, do the following:
 - With probability p_y , apply Pauli Y to ρ .
 - Else, apply identity operator on ρ .

For an n -qubit token, each qubit is independently corrupted and the noise channel is $\mathcal{N}_{p_e, p_y}^{\otimes n}$. The receiving party can detect when the qubit is erased (i.e., when the qubit is $|\perp\rangle$) but it cannot detect Y errors. We assume that the classical communication channel is noiseless¹.

We note that we only consider Pauli Y single-qubit errors. However, for the specific constructions based on BB84 states, this is without loss of generality as shown by [BSS21].

2.5 Noise-Tolerant Tokenized Digital Signature

We define a noise-tolerant tokenized digital signature scheme $\text{NT-TDS}(n, \eta_e, \eta_y)$, where the signing token is transmitted over a noisy channel. We model stateless trusted hardware as a classical-accessible oracle; that is, any query to the classical oracle is measured first.

Definition 2.2 (Noise-Tolerant One-Time Tokenized Signature Scheme). *Let $n \in \mathbb{N}, \eta_e \in [0, 1]$ and $\eta_y \in [0, 1/2]$. Let $\mathcal{N}$ be a quantum channel. A noise-tolerant one-time tokenized signature scheme $\text{NT-TDS}(n, \eta_e, \eta_y)$ in the classical-accessible oracle model consists of QPT algorithms (KeyGen, TokenSend, Sign, Verify) where TokenSend is an interactive protocol between the generator and the signer. We describe each algorithm:*

- **KeyGen(1^n)**: Takes as input security parameter 1^n , outputs classical master secret key msk and specifies the verification oracle $\mathcal{O}_{\text{vk}}(\cdot)$.
- **TokenSend($\text{msk}, \mathcal{N}$)**: An interactive protocol between the generator (holding msk) and the signer, where quantum communication happens via $\mathcal{N}$. At the end of the protocol, the signer either receives a (noisy) quantum signing token ρ_{msk} along with auxiliary classical information, or the protocol aborts.
- **Sign(state, μ)**: Takes as input signer's state state (consisting of the noisy quantum token and auxiliary classical information) and message $\mu \in \{0, 1\}$, outputs a signature σ .
- **Verify $^{\mathcal{O}_{\text{vk}}(\cdot)}(\mu, \sigma)$** : Takes as input a message μ , signature σ and (classical) query access to vk , outputs either $\text{accept}(1)$ or $\text{reject}(0)$.

Additionally, it satisfies the following properties:

- **Correctness**: We have:

$$\Pr[\text{abort} \leftarrow \text{TokenSend}(\text{msk}, \mathcal{N}) \mid (\text{msk}, \text{vk}) \leftarrow \text{KeyGen}(1^n)] = \text{negl}(n)$$

and for every message $\mu \in \{0, 1\}$, conditioned on $\text{TokenSend}(\text{msk}, \mathcal{N})$ not aborting:

$$\Pr \left[\text{Verify}^{\mathcal{O}_{\text{vk}}(\cdot)}(\mu, \sigma) = 1 \mid \begin{array}{l} (\text{msk}, \text{vk}) \leftarrow \text{KeyGen}(1^n) \\ \text{state} \leftarrow \text{TokenSend}(\text{msk}, \mathcal{N}) \\ \sigma \leftarrow \text{Sign}(\text{state}, \mu) \end{array} \right] \geq 1 - \text{negl}(n)$$

- **One-time Unforgeability**: Consider the following experiment between a QPT adversary $\mathcal{A}$ with initial state ρ_0 and challenger Ch :

¹If the classical communication channel has noise then we can use standard error correction tools to detect and correct the noise.

1. Ch runs $(\text{msk}, \text{vk}) \leftarrow \text{KeyGen}(1^n)$ and gives $\mathcal{A}$ classical query access to $\mathcal{O}_{\text{vk}}(\cdot)$.
2. Ch and $\mathcal{A}$ execute **TokenSend** using a perfect/noiseless channel: If **TokenSend** does not abort, continue.
3. $\mathcal{A}$ outputs $\{(\mu_b, \sigma_b)\}_{b \in \{0,1\}}$.
4. The experiment outputs 1 if $\mu_0 \neq \mu_1$ and $\text{Verify}(\text{vk}, \mu_b, \sigma_b) = 1$ for both $b \in \{0,1\}$. Otherwise the experiment outputs 0.

Let $\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n)$ be the probability that the experiment outputs 1. $\text{NT-TDS}(n, \eta_e, \eta_y)$ is one-time unforgeable if for all QPT adversaries $\mathcal{A}$,

$$\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n) = \text{negl}(n)$$

Remark 2.3. When defining the one-time unforgeability above, we assumed that $\mathcal{A}$ receives the quantum token over a noiseless channel. This is because if there is an adversary $\mathcal{A}'$ who is able to succeed in the above unforgeability experiment over a noisy channel then we can use $\mathcal{A}'$ to design a new adversary $\mathcal{A}$ that can succeed in the same experiment over a noiseless channel: $\mathcal{A}$ simply applies the noisy channel over the quantum token before forwarding it to $\mathcal{A}'$.

2.6 Noise-Tolerant Quantum One-Time Programs

We first define an efficiently computable function family.

Definition 2.4 (Function Family). Let $n \in \mathbb{N}$. $\mathcal{F}_n = \{f_k : \{0,1\}^{\ell_{\text{in}}(n)} \rightarrow \{0,1\}^{\ell_{\text{out}}(n)} : k \in \mathcal{K}(n)\}$ is a keyed function family where $\mathcal{K}(n) \subseteq \{0,1\}^{\kappa(n)}$ is the key (index) space of $\mathcal{F}_n$ and $\ell_{\text{in}}(n), \ell_{\text{out}}(n), \kappa(n)$ are polynomial in n . There is a PPT algorithm that, given $n \in \mathbb{N}, k \in \mathcal{K}(n)$ and $v \in \{0,1\}^{\ell_{\text{in}}(n)}$, outputs $f_k(v) \in \{0,1\}^{\ell_{\text{out}}(n)}$.

Now, we define noise-tolerant quantum one-time programs for a keyed function family below.

Definition 2.5 (Noise-Tolerant Quantum One-Time Program). Let $n \in \mathbb{N}, \eta_e \in [0,1)$ and $\eta_y \in [0,1/2)$. Let $\mathcal{N}$ be a quantum channel and $\mathcal{F}_n$ be a keyed function family with key space $\mathcal{K} = \mathcal{K}(n)$, domain $\{0,1\}^{\ell_{\text{in}}}$ and codomain $\{0,1\}^{\ell_{\text{out}}}$ where $\ell_{\text{in}} = \ell_{\text{in}}(n), \ell_{\text{out}} = \ell_{\text{out}}(n)$. A noise-tolerant quantum one-time program for $\mathcal{F} = \mathcal{F}_n$ in the classical-accessible oracle model, $\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$, consists of QPT algorithms (**Generate**, **TokenSend**, **Evaluate**) where **TokenSend** is an interactive protocol:

- **Generate** $(1^n, k)$: Takes as input security parameter 1^n , function key $k \in \mathcal{K}$; outputs a classical secret key msk and specifies a classical oracle $\mathcal{O}_k(\cdot)$.
- **TokenSend** $(\text{msk}, \mathcal{N})$: An interactive protocol between the generator (holding msk) and the evaluator, where quantum communication happens via channel $\mathcal{N}$. At the end of the protocol, the evaluator either receives a noisy quantum token ρ_{msk} along with auxiliary classical information, or the protocol aborts.
- **Evaluate** $^{\mathcal{O}_k(\cdot)}(\text{state}, v)$: On input the evaluator's state state (noisy quantum token and auxiliary information), $v \in \{0,1\}^{\ell_{\text{in}}}$ and (classical) query access to $\mathcal{O}_k(\cdot)$, it outputs $y \in \{0,1\}^{\ell_{\text{out}}}$ or $\perp$.

Additionally, it satisfies the following properties:

- **Correctness:** For all $n \in \mathbb{N}, k \in \mathcal{K}$, we have:

$$\Pr[\text{abort} \leftarrow \text{TokenSend}(\text{msk}, \mathcal{N}) \mid (\text{msk}, \mathcal{O}_k) \leftarrow \text{Generate}(1^n, k)] = \text{negl}(n)$$

For all $k \in \mathcal{K}$ and $v \in \{0, 1\}^{\ell_{\text{in}}}$, conditioned on TokenSend not aborting:

$$\Pr \left[\text{Evaluate}^{\mathcal{O}_k(\cdot)}(\text{state}, v) = f_k(v) \mid \begin{array}{l} (\text{msk}, \mathcal{O}_k) \leftarrow \text{Generate}(1^n, k) \\ \text{state} \leftarrow \text{TokenSend}(\text{msk}, \mathcal{N}) \end{array} \right] \geq 1 - \text{negl}(n)$$

- **Simulation security against a malicious evaluator.** Let $f_{k,1}$ be the stateful oracle that on its first query v returns $f_k(v)$, and returns $\perp$ on every later query. For $k \in \mathcal{K}$, QPT adversary $\mathcal{A}$ with initial state ρ_0 and QPT simulator Sim , consider the following two experiments.

- $\text{Real}_{\mathcal{A},k}(\rho_0)$: The generator runs $(\text{msk}, \mathcal{O}_k) \leftarrow \text{Generate}(1^n, k)$ and provides $\mathcal{A}$ with classical query access to $\mathcal{O}_k$. Starting from ρ_0 , the adversary $\mathcal{A}$ executes TokenSend with the generator over a noiseless channel, possibly deviating from the protocol. The output of the experiment is the final state of $\mathcal{A}$.
- $\text{Ideal}_{\text{Sim},k}(\rho_0)$: The simulator Sim gets ρ_0 and classical query access to the stateful oracle $f_{k,1}(\cdot)$. The output of the experiment is the output of $\text{Sim}^{f_{k,1}(\cdot)}(\rho_0)$.

$\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ is simulation secure if for every QPT adversary $\mathcal{A}$ there exists a QPT simulator Sim such that for all $k \in \mathcal{K}$, all initial states ρ_0 and every QPT distinguisher $\mathcal{D}$,

$$\left| \Pr[\mathcal{D}(\text{Real}_{\mathcal{A},k}(\rho_0)) = 1] - \Pr[\mathcal{D}(\text{Ideal}_{\text{Sim},k}(\rho_0)) = 1] \right| = \text{negl}(n).$$

Remark 2.6. As in Definition 2.2, the security definition gives the adversary the token over a noiseless channel, which only increases its winning probability.

3 Noise-Tolerant Tokenized Digital Signatures

Let $\mathcal{N}_{p_e, p_y}$ be a quantum noisy channel defined in Section 2.4 and let $\text{MAC} = (\text{MAC.KeyGen}, \text{MAC.Tag}, \text{MAC.Verify})$ be a classical one-time MAC scheme defined in Definition 2.1. We give the construction of $\text{NT-TDS}(n, \eta_e, \eta_y)$ below.

KeyGen(1^n):

1. Sample $\theta \xleftarrow{\$} \{0, 1\}^n$, $x \xleftarrow{\$} \{0, 1\}^n$.
2. Generate $\text{sk}_{\text{MAC}} \leftarrow \text{MAC.KeyGen}(1^n, n)$.
3. Set $\text{msk} = (\theta, x, \text{sk}_{\text{MAC}})$.
4. vk is a trusted hardware device (e.g., a TEE) implementing the following program P :

On input message bit $\mu \in \{0, 1\}$, $\tilde{\sigma}, M \in \{0, 1\}^n$ and a tag τ :

- (a) Check $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M, \tau) = 1$. If not, output reject.
- (b) Check $\text{HW}(M) \leq \eta_e \cdot n$. If not, output reject.
- (c) Let $E_\mu^M(\tilde{\sigma}) = \{i \in [n] : \theta_i = \mu \wedge M_i = 0 \wedge x_i \neq \tilde{\sigma}_i\}$.
- (d) If $|E_\mu^M(\tilde{\sigma})| \leq \eta_y \cdot (n - \text{HW}(M))$, output accept. Otherwise output reject.

TokenSend($\text{msk}, \mathcal{N}_{p_e, p_y}$): This is an interactive protocol between the generator (holding $\text{msk} = (\theta, x, \text{sk}_{\text{MAC}})$) and the signer:

1. The generator prepares the BB84 state $|x^\theta\rangle = \otimes_{i \in [n]} H^{\theta_i} |x_i\rangle$ and sends it qubit-by-qubit to the signer via channel $\mathcal{N}_{p_e, p_y}$.
2. For each $i \in [n]$, the signer sets $M_i = 0$ if qubit i was received and $M_i = 1$ if qubit i was erased. The signer sends $M \in \{0, 1\}^n$ to the generator.
3. The generator checks that $\text{HW}(M) \leq \eta_e n$. If the check fails, abort.
4. The generator computes $\tau \leftarrow \text{MAC.Tag}(\text{sk}_{\text{MAC}}, M)$ and sends τ to the signer.
5. The signer stores $\text{state} = (\rho_{\text{msk}}, M, \tau)$, where $\rho_{\text{msk}} \leftarrow \mathcal{N}_{p_e, p_y}^{\otimes n}(|x^\theta\rangle\langle x^\theta|)$ is the noisy signing token over n qubits.

Sign(state, μ):

1. Parse $\text{state} = (\rho_{\text{msk}}, M, \tau)$.
2. For each $i \in [n]$:
 - (a) If $M_i = 1$: set $\tilde{\sigma}_i = 0$.
 - (b) If $M_i = 0$:
 - If $\mu = 0$: measure i^{th} qubit of ρ_{msk} in the computational basis to obtain $\tilde{\sigma}_i$.
 - If $\mu = 1$: measure i^{th} qubit of ρ_{msk} in the Hadamard basis to obtain $\tilde{\sigma}_i$.
3. Let $\tilde{\sigma} = \tilde{\sigma}_1 \dots \tilde{\sigma}_n$. Output $\sigma = (\tilde{\sigma}, M, \tau)$.

Verify(vk, μ, σ):

1. Parse $\sigma = (\tilde{\sigma}, M, \tau)$.
2. Query the trusted hardware vk on $(\mu, \tilde{\sigma}, M, \tau)$. Output the result.

Remark 3.1. *The noise-tolerant tokenized digital signature scheme is a generalization of noise-tolerant quantum tokenized MACs from [BSS21]. If we set $\eta_e = 0$, then $M = 0^n$ is the only valid erasure string and **TokenSend** becomes non-interactive: the generator can send both the quantum token and the tag on 0^n at once.*

Parameters.

- Choose $\eta_e \in (p_e, 1)$ such that $\eta_e - p_e$ is constant and $(1 - \eta_e)n = \omega(\log n)$.
- Let $\alpha = \cos^2(\frac{\pi}{8})$ and $p_y < 1 - \alpha$. Choose $\eta_y \in (\frac{p_y}{2}, \frac{1-\alpha}{2})$ such that $2\eta_y - p_y$ is constant.

3.1 Correctness and Security

Theorem 3.2. *The noise-tolerant tokenized signature scheme NT-TDS(n, η_e, η_y) satisfies Definition 2.2.*

Correctness. We use Hoeffding's inequality: if $X = \sum_{i=1}^n X_i$ for independent random variables $X_i \in [0, 1]$, then $\Pr[X - \mathbb{E}[X] \geq tn] \leq e^{-2t^2n}$ for every $t > 0$. Let $\epsilon_e, \epsilon_y > 0$ be constants with $p_e + \epsilon_e \leq \eta_e$ and $p_y + \epsilon_y \leq 2\eta_y$. Since $\eta_e > p_e$ and $2\eta_y > p_y$ in our parameter choice, we can always find such ϵ_e, ϵ_y .

The channel $\mathcal{N}_{p_e, p_y}$ erases each qubit independently with probability p_e , so $\text{HW}(M)$ is a sum of n independent Bernoulli(p_e) variables and $\mathbb{E}[\text{HW}(M)] = p_e n$. Hence

$$\begin{aligned} \Pr[\text{TokenSend}(\text{msk}, \mathcal{N}_{p_e, p_y}) \text{ aborts}] &= \Pr[\text{HW}(M) > \eta_e n] \\ &\leq \Pr[\text{HW}(M) - p_e n \geq \epsilon_e n] \\ &\leq e^{-2\epsilon_e^2 n} \\ &= \text{negl}(n), \end{aligned}$$

where the first inequality uses $\eta_e \geq p_e + \epsilon_e$, second inequality follows from Hoeffding's inequality and the last inequality uses the fact that ϵ_e is a constant.

If TokenSend does not abort, the signer holds a valid tag τ on M and $\text{HW}(M) \leq \eta_e n$. For an honestly generated signature $\tilde{\sigma}$ on μ , Verify rejects only if $|E_\mu^M(\tilde{\sigma})| > \eta_y(n - \text{HW}(M))$, where

$$E_\mu^M(\tilde{\sigma}) = \{i \in [n] : \theta_i = \mu \wedge M_i = 0 \wedge \tilde{\sigma}_i \neq x_i\}$$

Fix any M with $\text{HW}(M) \leq \eta_e n$ and let $N = n - \text{HW}(M) \geq (1 - \eta_e)n$. For $M_i = 0$, the qubit i was not erased. The signer measures it in basis μ , and if $\theta_i = \mu$ this is its encoding basis, so $\tilde{\sigma}_i \neq x_i$ iff $\mathcal{N}_{p_e, p_y}$ applied Y to it. Hence, conditioned on M , the indicators $\mathbf{1}[i \in E_\mu^M(\tilde{\sigma})]$ for $M_i = 0$ are independent, with $\Pr[i \in E_\mu^M(\tilde{\sigma})] = \Pr[\theta_i = \mu] \cdot p_y = p_y/2$, and $\mathbb{E}[|E_\mu^M(\tilde{\sigma})| \mid M] = p_y N/2$. Since $\eta_y \geq (p_y + \epsilon_y)/2$,

$$\begin{aligned} \Pr[|E_\mu^M(\tilde{\sigma})| > \eta_y N \mid M] &\leq \Pr[|E_\mu^M(\tilde{\sigma})| - \frac{p_y N}{2} \geq \frac{\epsilon_y N}{2} \mid M] \\ &\leq e^{-\epsilon_y^2 N/2} \\ &\leq e^{-\epsilon_y^2 (1 - \eta_e) n/2} \\ &= \text{negl}(n) \end{aligned}$$

where the last step uses the fact that ϵ_y is constant and $(1 - \eta_e)n = \omega(\log n)$. The bound holds for every M with $\text{HW}(M) \leq \eta_e n$, and hence conditioned on TokenSend not aborting.

One-Time Unforgeability. Let $\mathcal{A}$ be a QPT adversary that makes at most q queries to $\mathcal{O}_{\text{vk}}(\cdot)$. Let $(\theta, x) \in \{0, 1\}^{2n}$ be the secret and let $M \in \{0, 1\}^n$ be the erasure string sent by the adversary in TokenSend . Define

$$L_0 = n - \lfloor \eta_e n \rfloor, \quad N_M := \{i \in [n] : M_i = 0\}, \quad n_M := |N_M| = n - \text{HW}(M), \quad t_M := \lfloor \eta_y n_M \rfloor.$$

If TokenSend does not abort, then $\text{HW}(M) \leq \lfloor \eta_e n \rfloor$ and hence $n_M \geq L_0$. For $\mu \in \{0, 1\}$ and $\tilde{\sigma} \in \{0, 1\}^n$, define the error set $E_\mu^M(\tilde{\sigma})$ as follows:

$$E_\mu^M(\tilde{\sigma}) = \{j \in N_M : \theta_j = \mu \wedge \tilde{\sigma}_j \neq x_j\}$$

The verification oracle $\mathcal{O}_{\text{vk}}$ accepts $(\mu, \tilde{\sigma}, M', \tau')$ if and only if

$$\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M', \tau') = 1, \quad \text{HW}(M') \leq \eta_e n, \quad |E_\mu^{M'}(\tilde{\sigma})| \leq t_{M'}.$$

EPR PAIRS AND TELEPORTATION. Let $|\Phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ be an EPR pair on registers (R, A) . We use the following two observations about EPR pairs and the teleportation protocol:

- (F1) Measuring R in the basis $\{H^\theta|y\rangle\}_{y \in \{0,1\}}$ gives a uniformly random y and leaves A in the state $H^\theta|y\rangle$.
- (F2) A Bell measurement on (C, R) applied to $|\psi\rangle_C|\Phi\rangle_{RA}$ gives a uniformly random outcome $(a, b) \in \{0,1\}^2$, independent of $|\psi\rangle$, and leaves A in the state $X^a Z^b|\psi\rangle$ up to a global phase. For a BB84 state with $x, \theta \in \{0,1\}$,

$$X^a Z^b H^\theta|x\rangle = H^\theta|x \oplus c_\theta\rangle, \quad \text{where } c_0 := a, \ c_1 := b.$$

up to a global phase. For $\theta = 0$ this is $X^a Z^b|x\rangle = (-1)^{bx}|x \oplus a\rangle$. For $\theta = 1$, we have $X^a Z^b H|x\rangle = X^a H X^b|x\rangle = H Z^a|x \oplus b\rangle$.

Consider the following series of hybrids:

Hyb₀. This is the one-time unforgeability experiment of Definition 2.2 for NT-TDS(n, η_e, η_y) with adversary $\mathcal{A}$. By definition, $\Pr[\mathcal{A} \text{ wins in Hyb}_0] = \text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n)$.

Hyb₁. This is the same as **Hyb₀**, except that the verification oracle, and the final verification of $\mathcal{A}$'s two outputs, also reject every tuple $(\mu, \tilde{\sigma}, M', \tau')$ with $M' \neq M$. Here M is the erasure mask $\mathcal{A}$ sent in **TokenSend** and $\tau \leftarrow \text{MAC.Tag}(\text{sk}_{\text{MAC}}, M)$. Since M already passed the check $\text{HW}(M) \leq \eta_e n$, verification in **Hyb₁** accepts $(\mu, \tilde{\sigma}, M', \tau')$ if and only if

$$M' = M, \quad \text{MAC.Verify}(\text{sk}_{\text{MAC}}, M', \tau') = 1, \quad |E_\mu^M(\tilde{\sigma})| \leq t_M. \quad (1)$$

The verification oracle also rejects any query if no M has been authenticated yet.

Hyb₂. This is the same as **Hyb₁**, except that the challenger prepares the token as follows.

1. Prepare n EPR pairs $|\Phi\rangle^{\otimes n}$ on $(R_1, A_1), \dots, (R_n, A_n)$ and send $A_1, \dots, A_n$ to $\mathcal{A}$.
2. Receive M from $\mathcal{A}$. If $\text{HW}(M) > \eta_e n$, abort. Otherwise let $L := n - \text{HW}(M)$ and let $\varphi : [L] \rightarrow N_M$ be the increasing bijection.
3. Sample $x', \theta' \xleftarrow{\$} \{0,1\}^L$ and prepare $|x'^{\theta'}\rangle$ on registers $C_1, \dots, C_L$. For each $j \in [L]$, perform a Bell measurement on $(C_j, R_{\varphi(j)})$ with outcome (a_j, b_j) , and set $c_{\varphi(j),0} := a_j$, $c_{\varphi(j),1} := b_j$.
4. For each erased position $i \in [n]$ with $M_i = 1$, sample $\theta_i \xleftarrow{\$} \{0,1\}$, measure R_i in the basis $\{H^{\theta_i}|y\rangle\}_{y \in \{0,1\}}$, and let x_i be the outcome.
5. For each $i = \varphi(j) \in N_M$, set $\theta_i := \theta'_j$ and $x_i := x'_j \oplus c_{i,\theta_i}$.
6. Continue as in **Hyb₁** with the secret (x, θ) . That is, send $\tau \leftarrow \text{MAC.Tag}(\text{sk}_{\text{MAC}}, M)$ and verify with (1).

Lemma 3.3. *For all QPT adversaries $\mathcal{A}$ with initial state ρ_0 , $|\Pr[\mathcal{A} \text{ wins Hyb}_1] - \Pr[\mathcal{A} \text{ wins Hyb}_0]| \leq \text{negl}(n)$.*

Proof. Let F be the event that $\mathcal{A}$ submits, as one of its queries or outputs, a tuple $(\mu, \tilde{\sigma}, M', \tau')$ such that $M' \neq M$ or no M has been authenticated yet, $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M', \tau') = 1$ and $|E_{\mu}^{M'}(\tilde{\sigma})| \leq \eta_y(n - \text{HW}(M'))$. The oracles in the two hybrids have exactly the same output on every query that does not trigger F . Since all queries and outputs are classical, both hybrids proceed identically until F occurs. By the EUF-1CMA security of MAC, the probability that $\mathcal{A}$ queries or outputs a tuple (amongst its $q(n) = \text{poly}(n)$ queries and 2 final outputs) such that F happens is at most $(q(n) + 2)2^{-n} = \text{negl}(n)$. Thus, for all QPT $\mathcal{A}$,

$$|\Pr[\mathcal{A} \text{ wins Hyb}_1] - \Pr[\mathcal{A} \text{ wins Hyb}_0]| \leq \Pr[F] = \text{negl}(n)$$

□

Lemma 3.4. *For all adversaries $\mathcal{A}$, $\Pr[\mathcal{A} \text{ wins in Hyb}_1] = \Pr[\mathcal{A} \text{ wins in Hyb}_2]$.*

Proof. The two hybrids differ only in how the token is prepared. After **TokenSend**, both run the same procedure as a function of the classical values $(x, \theta, M, \text{sk}_{\text{MAC}})$ and $\mathcal{A}$'s state. It therefore suffices to show that, at the end of **TokenSend**, the joint distribution of (x, θ, M) and $\mathcal{A}$'s state is the same in both hybrids. The tag τ for M is generated identically.

Let $\mathcal{E}$ be the channel that $\mathcal{A}$ applies to $A_1, \dots, A_n$ and its workspace W to produce M . It outputs M in a classical register, so we can write $\mathcal{E}(\rho) = \sum_M |M\rangle\langle M| \otimes \mathcal{E}_M(\rho)$, where each $\mathcal{E}_M$ is completely positive and trace non-increasing. For fixed M , let $\mathcal{L}_M$ be the channel implementing steps 3–5 of **Hyb**₂. It acts only on $R_1, \dots, R_n, C_1, \dots, C_L$, which $\mathcal{A}$ never touches, and writes (x, θ) and the outcomes c into classical registers. The unnormalised joint state for outcome M is therefore

$$(\mathcal{L}_M \otimes \mathcal{E}_M)(\rho) = (\text{id} \otimes \mathcal{E}_M)((\mathcal{L}_M \otimes \text{id})(\rho)),$$

where $\rho = |\Phi\rangle\langle\Phi|^{\otimes n} \otimes \omega_W$.

By (F1) and (F2), applying $\mathcal{L}_M$ leaves register A_i in the state $H^{\theta_i}|x_i\rangle$ for every i . Now fix the teleportation outcomes c . The map $(x', \theta', (x_i, \theta_i)_{i \in M}) \mapsto (x, \theta)$ defined in step 5 is a bijection of $\{0, 1\}^{2n}$, and its inputs are uniform. Hence (x, θ) is uniform and independent of c , and

$$(\mathcal{L}_M \otimes \text{id})(\rho) = \sum_{x, \theta} 4^{-n} |x, \theta\rangle\langle x, \theta| \otimes |x^\theta\rangle\langle x^\theta|_{A_1 \dots A_n} \otimes \omega_W \otimes (\text{independent uniform } c).$$

Applying $\mathcal{E}_M$ and discarding c gives $\sum_{x, \theta} 4^{-n} |x, \theta\rangle\langle x, \theta| \otimes \mathcal{E}_M(|x^\theta\rangle\langle x^\theta| \otimes \omega_W)$. This is exactly the unnormalised joint state for outcome M in **Hyb**₁, where $\mathcal{A}$ receives $|x^\theta\rangle$ for uniform (x, θ) . □

STEP I: REDUCTION TO THE ERASURE-FREE SCHEME: Let $L \in \mathbb{N}$. Consider the noise-tolerant scheme **NT-TDS**($L, 0, \eta_y$), i.e., the scheme with $\eta_e = 0$.

- In **TokenSend**, the generator aborts unless $M = 0^L$ and then sends the tag τ on 0^L .
- The verification oracle accepts $(\mu, \tilde{\sigma}, M', \tau')$ if and only if $M' = 0^L$, $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, 0^L, \tau') = 1$, and $|\{j \in [L] : \theta_j = \mu, \tilde{\sigma}_j \neq x_j\}| \leq \eta_y L$.

Since the adversary is always given a valid tag on 0^L , the MAC component plays no role. What remains is exactly the conjugate tokenized MAC **CTMAC** _{η_y} (Algorithm 5 of [BSS21]) on L qubits for one-bit messages. That scheme verifies $(\mu, \tilde{\sigma})$ by checking $|\text{Miss}_{\mu, \tilde{\sigma}}| \leq \eta_y L$, where $\text{Miss}_{\mu, \tilde{\sigma}} = \{j \in [L] : \theta_j = \mu \wedge \tilde{\sigma}_j \neq x_j\}$. We make this precise.

Lemma 3.5. *For every adversary $\mathcal{A}'$ against $\text{NT-TDS}(L, 0, \eta_y)$ making q verification queries, there is an adversary $\mathcal{A}''$ against CTMAC_{η_y} on L qubits making q verification queries with the same forging probability.*

Proof. $\mathcal{A}''$ generates its own MAC key and forwards the challenge token to $\mathcal{A}'$. If $\mathcal{A}'$ returns $M \neq 0^L$, it aborts. The NT-TDS generator would abort as well when $M \neq 0^L$. Otherwise it sends $\mathcal{A}'$ the tag τ on 0^L . It answers a query $(\mu, \tilde{\sigma}, M', \tau')$ with 0 if $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M', \tau') = 0$ or $M' \neq 0^L$, and otherwise with the answer of the CTMAC_{η_y} verification oracle on $(\mu, \tilde{\sigma})$. On the final tuples $(\mu_b, \tilde{\sigma}_b, M_b, \tau'_b)_{b \in \{0,1\}}$, it first checks $M_b = 0^L$ and $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M_b, \tau'_b) = 1$ for both b . If either check fails, it outputs two pairs with the same message, forcing a losing outcome. Otherwise it outputs $(\mu_b, \tilde{\sigma}_b)_{b \in \{0,1\}}$. The answers and the winning condition coincide with those of the $\text{NT-TDS}(L, 0, \eta_y)$ experiment.

The reduction is exact for every adversary $\mathcal{A}'$, including unbounded ones, as the security of MAC is never used. The reason is that, when $\eta_e = 0$, the only erasure string that can ever pass verification is 0^L . This is exactly the string signed in `TokenSend`, and $\mathcal{A}'$ is given a valid tag on it. An unbounded $\mathcal{A}'$ could forge MAC on other strings, but verification rejects every $M' \neq 0^L$ through the check $\text{HW}(M') \leq 0$, so such forgeries give it nothing. $\square$

Let $L \in \{L_0, \dots, n\}$ where $L_0 = n - \lfloor \eta_e n \rfloor$. We can construct an adversary $\mathcal{B}_L$ against $\text{NT-TDS}(L, 0, \eta_y)$ that simulates Hyb_2 for $\mathcal{A}$ and wins whenever $\mathcal{A}$ wins in Hyb_2 and $\text{HW}(M) = n - L$.

Description of $\mathcal{B}_L$.

1. *Preliminary phase.* Answer every verification query before mask authentication with 0, as in Hyb_1 . Prepare n EPR pairs on $(R_1, A_1), \dots, (R_n, A_n)$ and send $A_1, \dots, A_n$ to $\mathcal{A}$. Receive M . If $\text{HW}(M) \neq n - L$, abort. Otherwise let $N_M = \{i \in [n] : M_i = 0\}$ and $\varphi : [L] \rightarrow N_M$ be the increasing bijection.
2. *Obtain the challenge.* Start the experiment for $\text{NT-TDS}(L, 0, \eta_y)$. Receive the token $|x^{\theta'}\rangle$ on registers $C_1, \dots, C_L$, send 0^L in `TokenSend`, and receive a tag τ^* on 0^L .
3. *Embed.* For each $j \in [L]$, perform a Bell measurement on $(C_j, R_{\varphi(j)})$ with outcome (a_j, b_j) , and set $c_{\varphi(j),0} := a_j$, $c_{\varphi(j),1} := b_j$. For each erased coordinate $i \in [n]$ with $M_i = 1$, sample $\theta_i \xleftarrow{\$} \{0,1\}$ and measure R_i in the basis $\{H^{\theta_i}|y\rangle\}_{y \in \{0,1\}}$.
4. *Finish TokenSend.* Generate $\text{sk}_{\text{MAC}} \leftarrow \text{MAC.KeyGen}(1^n, n)$ and send $\tau \leftarrow \text{MAC.Tag}(\text{sk}_{\text{MAC}}, M)$ to $\mathcal{A}$.
5. *Answer queries.* On a query $(\mu, \tilde{\sigma}, M', \tau')$, return 0 if $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M', \tau') = 0$ or $M' \neq M$. Otherwise compute $s = T_\mu(\tilde{\sigma}) \in \{0,1\}^L$, where

$$\forall j \in [L] : \quad T_\mu(\tilde{\sigma})_j := \tilde{\sigma}_{\varphi(j)} \oplus c_{\varphi(j), \mu}$$

query the challenger's verification oracle on $(\mu, T_\mu(\tilde{\sigma}), 0^L, \tau^*)$, and return its answer.

6. *Output.* When $\mathcal{A}$ outputs $(\mu_b, \tilde{\sigma}_b, M_b, \tau'_b)_{b \in \{0,1\}}$, output $(\mu_b, T_{\mu_b}(\tilde{\sigma}_b), 0^L, \tau^*)_{b \in \{0,1\}}$.

$\mathcal{B}_L$ makes one query to the verification oracle of $\text{NT-TDS}(L, 0, \eta_y)$ for each query of $\mathcal{A}$ that passes the check in step 5, so it makes at most q queries. Define (x, θ) from (x', θ', c) and the outcomes of step 3 exactly as in step 5 of Hyb_2 .

Lemma 3.6. *Suppose $M_0 = M_1 = M$ and tags τ', τ^* are valid MAC tags on $M, 0^L$ respectively. Then the challenger accepts $(\mu, T_\mu(\tilde{\sigma}), 0^L, \tau^*)$ if and only if (1) accepts $(\mu, \tilde{\sigma}, M, \tau')$ with respect to (x, θ) .*

Proof. The challenger accepts if and only if $|\{j \in [L] : \theta'_j = \mu, T_\mu(\tilde{\sigma})_j \neq x'_j\}| \leq \eta_y L$. Let $i = \varphi(j)$. If $\theta'_j = \mu$, then $\theta_i = \mu$ and $x_i = x'_j \oplus c_{i,\mu}$. So $T_\mu(\tilde{\sigma})_j = \tilde{\sigma}_i \oplus c_{i,\mu}$ differs from x'_j exactly when $\tilde{\sigma}_i \neq x_i$. Hence φ maps the challenger's error set bijectively onto $E_\mu^M(\tilde{\sigma})$. Since $L = n - \text{HW}(M)$, the two thresholds coincide. $\square$

Lemma 3.7. *For every $L \in \{L_0, \dots, n\}$,*

$$\Pr[\mathcal{A} \text{ wins in Hyb}_2 \wedge \text{HW}(M) = n - L] \leq \Pr[\mathcal{B}_L \text{ wins}].$$

Proof. The first step of $\mathcal{B}_L$ is steps 1–2 of Hyb_2 . Suppose $\text{HW}(M) = n - L$. In step 3 of Hyb_2 , the challenger samples $|x^{\theta'}\rangle$ with uniform $(x', \theta') \in \{0, 1\}^{2L}$ after receiving M . In $\mathcal{B}_L$, the same state is supplied by the $\text{NT-TDS}(L, 0, \eta_y)$ challenger. The remaining token preparation and the MAC step are identical. Hence, on the event $\text{HW}(M) = n - L$, the run of $\mathcal{A}$ inside $\mathcal{B}_L$ is distributed exactly as in Hyb_2 .

Whenever $\mathcal{A}$'s two outputs verify in Hyb_2 , their masks and tags satisfy the hypotheses of Lemma 3.6. That lemma implies that $\mathcal{B}_L$'s two outputs verify, with the same distinct messages $\mu_0 \neq \mu_1$. Thus, conditioned on $\text{HW}(M) = n - L$, $\mathcal{B}_L$ wins whenever $\mathcal{A}$ wins. $\square$

STEP II: ONE-TIME UNFORGEABILITY OF THE ERASURE-FREE SCHEME: By Lemma 3.5, $\text{NT-TDS}(L, 0, \eta_y)$ is equivalent to the conjugate tokenized MAC CTMAC_{η_y} of [BSS21] on L qubits, for every adversary, including computationally unbounded ones. In Appendix A, we refine the concrete analysis of the one-time unforgeability of CTMAC_{η_y} given in [BSS21]. Their proof reduces a forger for the noise-tolerant (but erasure-free) scheme to a forger for the noiseless scheme on a random subset of m positions. This loses the probability Δ_m that the subset avoids every checked error, and they then fix a single m . We instead keep Δ_m exact for every $m \in \{0, \dots, L\}$ and average the resulting inequalities with binomial weights (Theorem A.1). Combined with Lemma 3.5, this gives the following bound. Here $\alpha = \cos^2(\pi/8)$, $\delta = 1 - \alpha$, and $D(a\|b) = a \ln \frac{a}{b} + (1 - a) \ln \frac{1-a}{1-b}$ is the binary relative entropy, with $0 \ln 0 = 0$.

Corollary 3.8 (Corollary of Lemma 3.5 and Theorem A.1). *Let $L \geq 1$, $q \geq 0$, and $0 \leq 2\eta_y < \delta$. Every computationally unbounded adversary against $\text{NT-TDS}(L, 0, \eta_y)$ making at most q classical verification queries forges with probability at most*

$$\varepsilon_{\eta_y}(L, q) = 2^{\binom{q+4}{2}} \exp\{-LD(2\eta_y\|\delta)\}.$$

The bound holds for computationally unbounded adversaries with an arbitrary initial state ρ_0 .

Proof. Let $\mathcal{A}'$ be an adversary against $\text{NT-TDS}(L, 0, \eta_y)$ that makes at most q classical verification queries and starts from an auxiliary state ρ_0 . By Lemma 3.5, there is an adversary $\mathcal{A}''$ against CTMAC_{η_y} on L qubits that makes at most q queries, starts from the same auxiliary state, and forges with the same probability. CTMAC_{η_y} on L qubits is exactly the scheme of Appendix A with token length L and threshold η_y . Its unforgeability experiment is the experiment defined there and Theorem A.1 bounds the forging probability of $\mathcal{A}''$, and hence of $\mathcal{A}'$, by $2^{\binom{q+4}{2}} e^{-LD(2\eta_y\|\delta)}$. $\square$

For fixed q and η_y , the bound is decreasing in L , so

$$\varepsilon_{\eta_y}(L, q) \leq \varepsilon_{\eta_y}(L_0, q) \quad \text{for all } L \geq L_0. \quad (2)$$

Proof of Theorem 3.2. Let $L_0 = n - \lfloor \eta_e n \rfloor$ and $p_L = \Pr[n - \text{HW}(M) = L]$ in the preliminary phase of Hyb_2 . Runs with $L < L_0$ abort. For each $L \geq L_0$ with $p_L > 0$, normalize the full state at the end of this phase, conditioned on $n - \text{HW}(M) = L$. This state includes the mask, all registers $R_1, \dots, R_n$, and the adversary's residual workspace and reference registers; it is independent of the fresh L -qubit challenge sampled afterward. Define $\hat{\mathcal{B}}_L$ to start from this normalized auxiliary state and execute the challenge, embedding, and query phases of $\mathcal{B}_L$. This is permitted by the information-theoretic corollary, which allows arbitrary challenge-independent quantum auxiliary states; no efficient postselection claim is needed. Lemma 3.7, applied to $\hat{\mathcal{B}}_L$ conditioned on $n - \text{HW}(M) = L$, together with Corollary 3.8 gives

$$\Pr[\mathcal{A} \text{ wins Hyb}_2 \mid n - \text{HW}(M) = L] \leq \Pr[\hat{\mathcal{B}}_L \text{ wins}] \leq \varepsilon_{\eta_y}(L, q).$$

We therefore average over the mutually exclusive lengths rather than sum their unconditional bounds:

$$\begin{aligned} \Pr[\mathcal{A} \text{ wins Hyb}_2] &\leq \sum_{L=L_0}^n p_L \varepsilon_{\eta_y}(L, q) \\ &\leq \varepsilon_{\eta_y}(L_0, q), \end{aligned}$$

because $\sum_{L=L_0}^n p_L \leq 1$ and (2) holds. Including the q classical verification calls and two final candidates in the mask-authentication bad event yields

$$\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n) \leq 2 \binom{q+4}{2} e^{-L_0 D(2\eta_y \|\delta)} + (q+2)2^{-n}.$$

For fixed $\eta_y < \delta/2$, the divergence is a positive constant. Hence polynomial q and $L_0 = \omega(\log n)$ make this bound negligible, proving one-time unforgeability.

3.2 Concrete Parameters and Security

Let $L_0 = n - \lfloor \eta_e n \rfloor$ and $\delta = 1 - \cos^2(\pi/8)$. The preceding proof gives the concrete bound

$$\text{TDSForge}_{\eta_e, \eta_y}(n) \leq 2 \binom{q+4}{2} \exp\{-L_0 D(2\eta_y \|\delta)\} + (q+2)2^{-n} \quad (3)$$

where $\text{TDSForge}_{\eta_e, \eta_y}(n) = \max_{\mathcal{A}} \text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n)$ is the maximum forging success probability over all adversaries making q queries. The first term bounds forging on the fixed authenticated mask, the second counts the q classical verification calls and two terminal candidates in the mask-authentication bad event.

Number of queries q . We take $q = 2^{32}$, counting all classical verification queries, including rejected queries. The quantum term depends quadratically on this budget.

Token length for fixed thresholds. For each fixed (η_e, η_y) , evaluate (3) with $L_0 = n - \lfloor \eta_e n \rfloor$ and find the smallest integer n for which $\text{TDSForge}_{\eta_e, \eta_y}(n)$ is at most 2^{-128} . Table 1 gives the number of qubits n for different erasure (η_e) and Y -error thresholds (η_y) .

Honest correctness at physical noise rates. The bound (3) only concerns forging. Correctness depends on the physical rates (p_e, p_y) of the channel $\mathcal{N}_{p_e, p_y}$, which must satisfy $p_e < \eta_e$ and $p_y < 2\eta_y$. An honest signature is accepted if at most $\lfloor \eta_e n \rfloor$ qubits are erased and, among the L surviving qubits, at most $\lfloor \eta_y L \rfloor$ checked positions are flipped. Each surviving position is checked with probability $1/2$ and flipped with probability p_y , independently, so it is a checked mismatch with probability $p_y/2$. Hence the probability that an honest signature is accepted is exactly

$$A(n) = \sum_{L=L_0}^n \binom{n}{L} (1-p_e)^L p_e^{n-L} \Pr[\text{Bin}(L, p_y/2) \leq \lfloor \eta_y L \rfloor],$$

and the honest failure probability, including aborts of `TokenSend`, is $1 - A(n)$. Since $L \geq L_0$ whenever `TokenSend` does not abort, the Chernoff bound gives the simpler estimate

$$1 - A(n) \leq e^{-nD(\eta_e \| p_e)} + e^{-L_0 D(\eta_y \| p_y/2)}.$$

To choose parameters, we fix the physical rates, pick thresholds slightly above them, take the smallest n for which (3) is at most 2^{-128} , and then check that $1 - A(n)$ is below the honest-failure budget.

Example 1. Let $p_e = 0.90$ and $p_y = 0.03$, with thresholds $\eta_e = 0.91$ and $\eta_y = 0.03$; the verifier then tolerates physical Y -error rates up to $2\eta_y = 0.06$. For $q = 2^{32}$, the smallest n with $\text{TDSForge}_{\eta_e, \eta_y}(n) \leq 2^{-128}$ is $n = 39,801$, with $L_0 = 3,583$. The Chernoff estimate gives honest failure at most 7.64×10^{-10} , and the exact value $1 - A(n)$ is about 1.1×10^{-11} , both below 10^{-6} .

Example 2. Let $p_e = 0.5$ and $p_y = 0.05$, with thresholds $\eta_e = 5514/10487 \approx 0.5258$ and $\eta_y = 0.03591265$. The smallest certified length is $n = 10,542$, with $\lfloor \eta_e n \rfloor = 5,542$ and $L_0 = 5,000$. Its forging bound is about $0.9747 \cdot 2^{-128}$, and the exact honest failure $1 - A(n)$ is about 9.42×10^{-7} . With the same thresholds, $n = 10,487$ has honest failure about 9.98×10^{-7} , but the forging bound certifies only about 127 bits.

4 Noise-Tolerant Quantum One-Time Programs

Let $\mathcal{N}_{p_e, p_y}$ be a quantum noisy channel defined in Section 2.4 and let $\text{MAC} = (\text{MAC.KeyGen}, \text{MAC.Tag}, \text{MAC.Verify})$ be a classical MAC scheme defined in Definition 2.1. Let $n \in \mathbb{N}$ and $\mathcal{F} = \{f_k : \{0, 1\}^{\ell_{\text{in}}} \rightarrow \{0, 1\}^{\ell_{\text{out}}}\}^2$ be a function family indexed by $k \in \mathcal{K}$ defined in Definition 2.4. We construct a noise-tolerant quantum one-time program $\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$, for functionality $\mathcal{F}$, below.

`Generate` $(1^n, k)$:

1. For $i \in [\ell_{\text{in}}]$, sample $\theta_i \xleftarrow{\$} \{0, 1\}^n$, $x_i \xleftarrow{\$} \{0, 1\}^n$.
2. Generate $\text{sk}_{\text{MAC}} \leftarrow \text{MAC.KeyGen}(1^n, n\ell_{\text{in}})$.
3. Set $\text{msk} = (\{\theta_i, x_i\}_{i \in [\ell_{\text{in}}]}, \text{sk}_{\text{MAC}})$.

²We drop the parameter n and write $\mathcal{F}$ for $\mathcal{F}_n$

4. $\mathcal{O}_k$ is a trusted hardware device implementing the following program P :

On input $v \in \{0, 1\}^{\ell_{\text{in}}}$, $r \in \{0, 1\}^{n\ell_{\text{in}}}$, $M \in \{0, 1\}^{n\ell_{\text{in}}}$, and τ :

- (a) Check $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M, \tau) = 1$. If not, output $\perp$.
- (b) Parse $v = v_1 \dots v_{\ell_{\text{in}}}$, $r = r_{1,1} \dots r_{\ell_{\text{in}},n}$, and $M = M_{1,1} \dots M_{\ell_{\text{in}},n}$.
- (c) For each $i \in [\ell_{\text{in}}]$:
 - i. Check $\text{HW}(M_i) \leq \eta_e \cdot n$ where $M_i = (M_{i,1}, \dots, M_{i,n})$. If not, output $\perp$.
 - ii. Let $n_i = n - \text{HW}(M_i)$ (number of unerased qubits in group i).
 - iii. Let $E_{v_i}^M(r) = \{j \in [n] : \theta_{i,j} = v_i \wedge M_{i,j} = 0 \wedge r_{i,j} \neq x_{i,j}\}$.
 - iv. If $|E_{v_i}^M(r)| > \eta_y \cdot n_i$, output $\perp$.
- (d) If all checks pass, output $f_k(v)$. Otherwise, output $\perp$.

5. Output msk and $\mathcal{O}_k$.

$\text{TokenSend}(\text{msk}, \mathcal{N}_{p_e, p_y})$: This is an interactive protocol between the generator (holding $\text{msk} = (\{\theta_i, x_i\}_{i \in [\ell_{\text{in}}]}, \text{sk}_{\text{MAC}})$) and the evaluator:

1. Generator prepares the BB84 state $|x_i^{\theta_i}\rangle = \bigotimes_{j=1}^n H^{\theta_{i,j}} |x_{i,j}\rangle$. Let $|x^\theta\rangle = \bigotimes_{i=1}^{\ell_{\text{in}}} |x_i^{\theta_i}\rangle$.
2. The generator sends the BB84 state qubit-by-qubit via channel $\mathcal{N}_{p_e, p_y}$.
3. For each $(i, j) \in [\ell_{\text{in}}] \times [n]$, the evaluator sets $M_{i,j} = 0$ if qubit (i, j) was received and $M_{i,j} = 1$ if qubit (i, j) was erased. The evaluator sends $M \in \{0, 1\}^{n\ell_{\text{in}}}$ to the generator.
4. The generator checks that $\text{HW}(M_i) \leq \eta_e \cdot n$ for all $i \in [\ell_{\text{in}}]$. If any check fails, abort.
5. The generator computes $\tau \leftarrow \text{MAC.Tag}(\text{sk}_{\text{MAC}}, M)$ and sends τ to the evaluator.
6. The evaluator stores $\text{state} = (\rho_{\text{msk}}, M, \tau)$, where $\rho_{\text{msk}} \leftarrow \mathcal{N}_{p_e, p_y}^{\otimes n\ell_{\text{in}}}(|x^\theta\rangle\langle x^\theta|)$ is the noisy quantum token.

$\text{Evaluate}^{\mathcal{O}_k(\cdot)}(\text{state}, v)$:

1. Parse $\text{state} = (\rho_{\text{msk}}, M, \tau)$ and $v = v_1 \dots v_{\ell_{\text{in}}}$.
2. For each $(i, j) \in [\ell_{\text{in}}] \times [n]$:
 - (a) If $M_{i,j} = 1$ (erased): set $r_{i,j} = 0$.
 - (b) If $M_{i,j} = 0$ (received):
 - If $v_i = 0$: measure qubit (i, j) of ρ_{msk} in the computational basis to obtain $r_{i,j}$.
 - If $v_i = 1$: measure qubit (i, j) of ρ_{msk} in the Hadamard basis to obtain $r_{i,j}$.
3. Let $r = r_{1,1} \dots r_{\ell_{\text{in}},n}$. Output $\mathcal{O}_k(v, r, M, \tau)$.

Remark 4.1. We denote noise-tolerant tokenized digital signature on ℓ_{in} bits by $\text{NT-TDS}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ which is a special case of $\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$. The oracle $\mathcal{O}_k$ is replaced by the verification oracle $\mathcal{O}_{\text{vk}}$ that runs the same checks as steps (a)-(c) of $\mathcal{O}_k$ and outputs **accept** only if all the checks pass.

Parameters.

- Choose $\eta_e \in (p_e, 1)$ such that $(1 - \eta_e)n = \omega(\log n)$ and $\eta_e - p_e$ is constant.
- Let $\alpha = \cos^2(\frac{\pi}{8})$ and $p_y < 1 - \alpha$. Choose $\eta_y \in (\frac{p_y}{2}, \frac{1-\alpha}{2})$ such that $2\eta_y - p_y$ is constant.

4.1 Correctness and Security

Theorem 4.2. *The noise-tolerant quantum one-time program NT-QOTP($n, \ell_{\text{in}}, \eta_e, \eta_y$) satisfies Definition 2.5.*

Correctness. For $i \in [\ell_{\text{in}}]$, we have $\text{HW}(M_i) = \sum_{j=1}^n M_{i,j}$. Since $\mathcal{N}_{p_e, p_y}$ erases each qubit independently with probability p_e , $\text{HW}(M_i)$ is a sum of n independent Bernoulli(p_e) variables and $\mathbb{E}[\text{HW}(M_i)] = p_e n$. Let $\epsilon_e, \epsilon_y > 0$ be constants such that $\eta_e \geq p_e + \epsilon_e$ and $2\eta_y \geq p_y + \epsilon_y$. Since $\eta_e > p_e$ and $2\eta_y > p_y$ in our parameter choice, we can always find such ϵ_e, ϵ_y . Hence for all $i \in [\ell_{\text{in}}]$,

$$\begin{aligned} \Pr[\text{HW}(M_i) > \eta_e n] &\leq \Pr[\text{HW}(M_i) - \mathbb{E}[\text{HW}(M_i)] \geq \epsilon_e n] \\ &\leq e^{-2\epsilon_e^2 n} = \text{negl}(n) \end{aligned}$$

where the first inequality uses $\eta_e \geq p_e + \epsilon_e$, second inequality follows from Hoeffding's inequality and the final equality follows from the fact that ϵ_e is constant.

Now taking union bound over all $i \in [\ell_{\text{in}}]$,

$$\begin{aligned} &\Pr[\text{TokenSend}(\text{msk}, \mathcal{N}_{p_e, p_y}) \text{ aborts} \mid (\text{msk}, \mathcal{O}_k) \leftarrow \text{Generate}(1^n, k)] \\ &\leq \sum_{i=1}^{\ell_{\text{in}}} \Pr[\text{HW}(M_i) > \eta_e n] \\ &\leq \ell_{\text{in}} \text{negl}(n) = \text{negl}(n) \end{aligned}$$

as $\ell_{\text{in}} = \text{poly}(n)$.

If **TokenSend** does not abort, the evaluator holds a valid MAC tag on M and $\text{HW}(M_i) \leq \eta_e n$ for all $i \in [\ell_{\text{in}}]$. For any honestly generated $r \in \{0, 1\}^{n\ell_{\text{in}}}$ for any $v \in \{0, 1\}^{\ell_{\text{in}}}$, **Evaluate** outputs $\perp$ only if $\exists i \in [\ell_{\text{in}}]$ such that $|E_{v_i}^M(r)| > \eta_y(n - \text{HW}(M_i))$ where

$$E_{v_i}^M(r) = \{j \in [n] : \theta_{i,j} = v_i \wedge M_{i,j} = 0 \wedge r_{i,j} \neq x_{i,j}\}$$

Fix any $M, i \in [\ell_{\text{in}}]$ with $\text{HW}(M_i) \leq \eta_e n$. For $M_{i,j} = 0$, that is qubit (i, j) was not erased, the evaluator measures it in basis v_i and if $\theta_{i,j} = v_i$, $\Pr[r_{i,j} \neq x_{i,j}] = p_y$. Hence, conditioned on M , the indicators $\mathbf{1}_i[j \in E_{v_i}^M(r)]$ are independent, with $\Pr[j \in E_{v_i}^M(r)] = p_y/2$ and $\mathbb{E}[|E_{v_i}^M(r)| \mid M] = p_y n_i/2$. Since $\eta_y \geq (p_y + \epsilon_y)/2$,

$$\Pr[|E_{v_i}^M(r)| > \eta_y n_i] = \text{negl}(n)$$

and for all $i \in [\ell_{\text{in}}]$, by union bound, we have

$$\begin{aligned} \Pr[\mathcal{O}_k \text{ outputs } \perp] &\leq \sum_{i=1}^{\ell_{\text{in}}} \Pr[|E_{v_i}^M(r)| > \eta_y n_i] \\ &\leq \ell_{\text{in}} \text{negl}(n) = \text{negl}(n) \end{aligned}$$

as $\ell_{\text{in}} = \text{poly}(n)$ and NT-QOTP($n, \ell_{\text{in}}, \eta_e, \eta_y$) satisfies correctness.

Simulation security. Let $k \in \mathcal{K}$ and $\mathcal{A}$ be a QPT adversary with initial state ρ_0 that makes $q = \text{poly}(n)$ queries to oracle $\mathcal{O}_k$. Consider the following sequence of hybrids (Let ρ^{Hyb_j} be the final state output by $\mathcal{A}$ in Hyb_j):

Hyb₀: This is the real experiment $\text{Real}_{\mathcal{A},k}(\rho_0)$. The generator runs $(\text{msk}, \mathcal{O}_k) \leftarrow \text{Generate}(1^n, k)$ and gives $\mathcal{A}$ classical query access to $\mathcal{O}_k$. $\mathcal{A}$ receives $|x^\theta\rangle$ where $x, \theta \in \{0, 1\}^{n_{\text{in}}}$. $\mathcal{A}$ sends an $M \in \{0, 1\}^{n_{\text{in}}}$. If $\text{HW}(M_i) > \eta_e n$ for any $i \in [\ell_{\text{in}}]$, the experiment aborts. Otherwise $\mathcal{A}$ receives the tag τ on M . $\mathcal{A}$ makes q queries to $\mathcal{O}_k$ and outputs state ρ^{Hyb_0} .

Hyb₁: This hybrid is the same as before except $\mathcal{A}$ gets oracle access to $\mathcal{O}_k^{(1)}$ which on query (v, r, M', τ') , does the following:

- If $M' \neq M$ or $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M, \tau') = 0$, output $\perp$. If no M has been authenticated yet, output $\perp$.
- Otherwise output $\mathcal{O}_k(v, r, M, \tau)$.

Hyb₂: This hybrid is the same as before except $\mathcal{A}$ gets oracle access to $\mathcal{O}_k^{(2)}$ which only outputs $f_k(v^*)$ on the first successful query (v^*, r^*, M, τ) and outputs $\perp$ for any subsequent queries on a different input $v \neq v^*$. If $\mathcal{A}$ makes subsequent valid queries (v^*, r, M, τ) on v^* , $\mathcal{O}_k^{(2)}$ responds with $f_k(v^*)$.

Lemma 4.3. For every QPT $\mathcal{A}$ with initial state ρ_0 and QPT distinguisher $\mathcal{D}$,

$$\left| \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_0}) = 1] \right| = \text{negl}(n)$$

Proof. Let F be the event that $\mathcal{A}$ makes a query (v, r, M', τ') with $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M', \tau') = 1$ such that either no mask has been signed yet or $M' \neq M$.

Every query that does not trigger F gets the same answer from $\mathcal{O}_k$ and $\mathcal{O}_k^{(1)}$. Queries with an invalid signature are rejected by both, and queries with a valid signature on the signed mask M are answered by both as $\mathcal{O}_k$ would. So Hyb_0 and Hyb_1 are identical until F , and

$$\left| \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_0}) = 1] \right| \leq \Pr_{\text{Hyb}_0}[F]$$

If F occurs with non-negligible probability in Hyb_0 (or Hyb_1), then we can construct a forging adversary for MAC that is successful with the same probability. Therefore, by the EUF-1CMA security of MAC, we have for all QPT $\mathcal{A}$ and $\mathcal{D}$

$$\left| \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_0}) = 1] \right| = \text{negl}(n)$$

□

Let **Bad** be the event that $\mathcal{A}$ makes two queries with different inputs in Hyb_1 that are both accepted by $\mathcal{O}_k^{(1)}$. By definition of $\mathcal{O}_k^{(1)}$, all accepted queries in Hyb_1 carry the same signed mask M .

Lemma 4.4. For all QPT $\mathcal{A}$ in Hyb_1 , $\Pr_{\text{Hyb}_1}[\text{Bad}] = \text{negl}(n)$.

Proof. We show this by constructing a QPT adversary $\mathcal{B}$ against NT-TDS(n, η_e, η_y) with forging probability $\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{B}}(n)$ such that

$$\Pr_{\text{Hyb}_1} [\text{Bad}] \leq \ell_{\text{in}} \cdot \text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{B}}(n).$$

Then since $\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{B}}(n) = \text{negl}(n)$ for all QPT $\mathcal{B}$ for the parameters η_e, η_y (Theorem 3.2), we get $\Pr_{\text{Hyb}_1} [\text{Bad}] = \text{negl}(n)$ for all QPT $\mathcal{A}$.

$\mathcal{B}$ gets quantum advice ρ_0 and $k \in \mathcal{K}$ as auxiliary information and plays the one-time unforgeability experiment of Definition 2.2 with a challenger Ch. It has classical access to Ch's verification oracle $\mathcal{O}_{\text{vk}}$, and it receives Ch's token $|x_*^{\theta_*}\rangle$ on n qubits in the first step of TokenSend.

Description of $\mathcal{B}$.

1. *Setup.* Sample $z \xleftarrow{\$} [\ell_{\text{in}}]$. For every $i \neq z$, sample $\theta_i, x_i \xleftarrow{\$} \{0, 1\}^n$. Generate $\text{sk}_{\text{MAC}} \leftarrow \text{MAC.KeyGen}(1^n, n\ell_{\text{in}})$. Implicitly set $(\theta_z, x_z) := (\theta_*, x_*)$.
2. *Token.* Run $\mathcal{A}$ on ρ_0 and send it the token $\bigotimes_{i \in [\ell_{\text{in}}]} |x_i^{\theta_i}\rangle$. Group z is Ch's token; the other groups are prepared by $\mathcal{B}$.
3. *Mask.* On receiving M from $\mathcal{A}$:
 - If $\text{HW}(M_i) > \eta_e n$ for some $i \in [\ell_{\text{in}}]$, abort TokenSend and answer all further queries with $\perp$.
 - Otherwise, send M_z to Ch and receive $\tau_z \leftarrow \text{MAC.Tag}(\text{sk}_{\text{MAC}}^{(z)}, M_z)$; Ch does not abort because $\text{HW}(M_z) \leq \eta_e n$. Send $\tau \leftarrow \text{MAC.Tag}(\text{sk}_{\text{MAC}}, M)$ to $\mathcal{A}$.
4. *Oracle.* Answer a query (v, r, M', τ') as follows.
 - If no mask has been signed yet, or $M' \neq M$, or $\text{MAC.Verify}(\text{sk}_{\text{MAC}}, M', \tau') = 0$, answer $\perp$.
 - For each $i \neq z$, check $\text{HW}(M_i) \leq \eta_e n$ and $|E_{v_i}^M(r)| \leq \eta_y n_i$ using the known (θ_i, x_i) .
 - If all these checks pass, query $\mathcal{O}_{\text{vk}}$ on (v_z, r_z, M_z, τ_z) . Return $f_k(v)$ if $\mathcal{O}_{\text{vk}}$ accepts, and $\perp$ otherwise. If any check above fails, return $\perp$ without querying $\mathcal{O}_{\text{vk}}$.
5. *Output.* Let (v^*, r^*) be the input and response string of the first accepted query, and let (v', r') be those of the later accepted query with $v' \neq v^*$. If both exist and $v_z^* \neq v_z'$, output the two pairs

$$(v_z^*, (r_z^*, M_z, \tau_z)) \quad \text{and} \quad (v_z', (r_z', M_z, \tau_z)).$$

Otherwise, output $\perp$.

$\mathcal{B}$ runs $\mathcal{A}$ once plus polynomial-time bookkeeping, so it is QPT with auxiliary input (ρ_0, k) . It makes at most one query to $\mathcal{O}_{\text{vk}}$ per query of $\mathcal{A}$.

$\mathcal{B}$ *simulates* Hyb_1 *exactly*. By the verification program of NT-TDS(n, η_e, η_y), $\mathcal{O}_{\text{vk}}$ accepts (v_z, r_z, M_z, τ_z) if and only if all three of the following hold:

- τ_z is a valid tag on M_z , which it is, being produced by Ch,
- $\text{HW}(M_z) \leq \eta_e n$, which was checked in step 3,
- $|E_{v_z}^M(r)| \leq \eta_y n_z$, which is exactly the group- z check of $\mathcal{O}_k$ with $(\theta_z, x_z) = (\theta_*, x_*)$.

Hence, for every fixed value of z , $\mathcal{B}$ answers every query exactly as $\mathcal{O}_k^{(1)}$ does. Moreover, $(k, \theta, x, \text{sk}_{\text{MAC}})$ has the same distribution as in Hyb_1 , and $\mathcal{A}$ receives the honest token $|x^\theta\rangle$. So for every fixed z , the view of $\mathcal{A}$ inside $\mathcal{B}$ is distributed exactly as in Hyb_1 . In particular, z is independent of $\mathcal{A}$'s view.

Whether **Bad** occurs, and the inputs $v_z^* \neq v'_z$ in that case, are determined by $\mathcal{A}$'s view. Since z is uniform and independent of this view, and v^* and v' differ in at least one coordinate, we get $\Pr[v_z^* \neq v'_z \mid \text{Bad}] \geq 1/\ell_{\text{in}}$.

In that case $\mathcal{B}$'s two output pairs have different messages $v_z^* \neq v'_z$, and both verify under vk . Indeed, both queries of $\mathcal{A}$ were accepted, so in answering them $\mathcal{B}$ queried $\mathcal{O}_{\text{vk}}$ on exactly these tuples and $\mathcal{O}_{\text{vk}}$ accepted. Therefore

$$\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{B}}(n) \geq \Pr_{\text{Hyb}_1}[\text{Bad} \wedge v_z^* \neq v'_z] \geq \frac{1}{\ell_{\text{in}}} \cdot \Pr_{\text{Hyb}_1}[\text{Bad}].$$

□

Lemma 4.5. *For all QPT $\mathcal{D}$ and $\mathcal{A}$ with initial state ρ_0 ,*

$$|\Pr[\mathcal{D}(\rho^{\text{Hyb}_2}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1]| = \text{negl}(n)$$

Proof. Recall the **Bad** event that occurs when $\mathcal{A}$ makes two accepted queries with different inputs. As in the proof of Lemma 4.3, the view of $\mathcal{A}$ is identical in both Hyb_1 and Hyb_2 until **Bad** occurs and

$$|\Pr[\mathcal{D}(\rho^{\text{Hyb}_2}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1]| \leq \Pr_{\text{Hyb}_1}[\text{Bad}] = \text{negl}(n)$$

□

Description of Simulator. Sim has input ρ_0 and access to the one-time oracle $f_{k,1}$. It runs $\mathcal{A}$ on ρ_0 as follows.

1. Sample $\theta_i, x_i \xleftarrow{\$} \{0, 1\}^n$ for every $i \in [\ell_{\text{in}}]$, and $\text{sk}_{\text{MAC}} \leftarrow \text{MAC.KeyGen}(1^n, n\ell_{\text{in}})$. Send $|x^\theta\rangle = \bigotimes_i |x_i^{\theta_i}\rangle$ to $\mathcal{A}$.
2. On receiving M , abort **TokenSend** if $\text{HW}(M_i) > \eta_e n$ for some $i \in [\ell_{\text{in}}]$, exactly as the generator does. Otherwise send $\tau \leftarrow \text{MAC.Tag}(\text{sk}_{\text{MAC}}, M)$ to $\mathcal{A}$.
3. Answer a query (v, r, M', τ') as follows.
 - If no mask has been signed, or $M' \neq M$, or the tag is invalid, or some per-group check fails, answer $\perp$. The per-group checks are $\text{HW}(M_i) \leq \eta_e n$ and $|E_{v_i}^M(r)| \leq \eta_y n_i$ for all $i \in [\ell_{\text{in}}]$, computed with (θ, x) .
 - Otherwise the query is accepted. If no input has been cached, query $y \leftarrow f_{k,1}(v)$, cache (v, y) and answer y . If (v^*, y) is cached, answer y if $v = v^*$ and $\perp$ otherwise.
4. Output the final state of $\mathcal{A}$.

Sim is QPT and queries $f_{k,1}$ at most once. It never uses k directly.

Lemma 4.6. *For all QPT $\mathcal{A}$ with initial state ρ_0 and $k \in \mathcal{K}$, the output of $\text{Ideal}_{\text{Sim}, k}(\rho_0)$ is identical to the output of $\mathcal{A}$ in Hyb_2 . In particular, $\Pr[\mathcal{D}(\text{Ideal}_{\text{Sim}, k}(\rho_0)) = 1] = \Pr[\mathcal{D}(\rho^{\text{Hyb}_2}) = 1]$ for every distinguisher $\mathcal{D}$.*

Proof. Sim performs the generator's part of TokenSend exactly as in Hyb_2 , including the abort before tagging, with identically distributed (θ, x) and MAC key. It decides acceptance by exactly the rule of $\mathcal{O}_k^{(2)}$.

It remains to compare the returned values on accepted queries. On the first accepted query, $\mathcal{O}_k^{(2)}$ returns $f_k(v^*)$. Sim returns $f_{k,1}(v^*)$, which equals $f_k(v^*)$ because this is Sim 's first and only query to $f_{k,1}$. On a later accepted query, both return $f_k(v^*)$ if the input is v^* , and $\perp$ otherwise. Hence $\mathcal{A}$ interacts with the same oracle in both experiments, and the output states coincide for every $k \in \mathcal{K}$. $\square$

Proof of Theorem 4.2. Combining Lemma 4.3, 4.5 and 4.6, we have that for the parameters $\ell_{\text{in}} = \text{poly}(n)$, $\eta_e \in [0, 1)$, $\eta_y \in [0, 1/2)$ such that $(1 - \eta_e)n = \omega(\log n)$ and $\eta_y < \frac{1-\alpha}{2}$, no QPT distinguisher $\mathcal{D}$ can distinguish $\text{Real}_{\mathcal{A},k}(\rho_0)$ and $\text{Ideal}_{\text{Sim},k}(\rho_0)$ except with $\text{negl}(n)$ probability for any QPT $\mathcal{A}$, initial state ρ_0 and function key $k \in \mathcal{K}$. Therefore $\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ satisfies simulation security.

Concrete correctness and security. A one-time program for ℓ_{in} -bit inputs uses ℓ_{in} independent blocks of n BB84 qubits, i.e., $n\ell_{\text{in}}$ qubits in total. Since the blocks are independent, the honest failure probability is $1 - A(n)^{\ell_{\text{in}}} \leq \ell_{\text{in}}(1 - A(n))$, where $A(n)$ is the single-block acceptance probability of Section 3.2. By Lemma 4.4, the distinguishing advantage in the proof of Theorem 4.2 is at most

$$\ell_{\text{in}} \cdot 2 \binom{q+4}{2} e^{-L_0 D(2\eta_y \|\delta)} + (q+2)2^{-n}$$

plus the MAC term of Lemma 4.3, where $L_0 = n - \lfloor \eta_e n \rfloor$. Hence a target of 2^{-128} for the whole program is met by choosing n as in Section 3.2 with target $2^{-128}/\ell_{\text{in}}$, i.e., with 128 replaced by $128 + \log_2 \ell_{\text{in}}$.

5 Extensions

We describe two extensions of our constructions. The first (Section 5.1) allows a *single* stateless trusted hardware to serve an a priori unbounded polynomial number of quantum tokens, so that the hardware needs to be provisioned only once. The second (Section 5.2) addresses the fact that deployed trusted execution environments are not perfect black boxes and can leak secrets: we distribute the function key k across t hardware devices such that the one-time program remains simulation secure even if the evaluator learns the *entire* internal state of up to $t - 1$ of them.

5.1 Reusable Trusted Hardware

In the constructions of Sections 3 and 4, the BB84 secret (θ, x) and the MAC key are hard-wired into the hardware, so every signing token (resp. every copy of a one-time program) requires a fresh hardware device. However, for the purpose of deployment, it is more desirable to create the trusted hardware once and re-use it by creating fresh BB84 tokens for each signature. We show how one trusted hardware can be re-used across polynomially many quantum tokens.

Pseudorandom BB84 states. Each quantum token must have its own secret. If two copies of $|x^\theta\rangle$ are issued against the same hardware, it is possible to learn both x, θ by measuring one state in the computational basis and the other in the Hadamard basis allowing forgery. The one-time MAC of Definition 2.1 cannot be reused either, since a tag reveals half of the key. As the number of tokens is not bounded in advance, the hardware cannot store independent secrets for each of them, so we derive them pseudorandomly.

The hardware stores a PRF key K and a *many-time* MAC key K' . Every quantum token carries a classical seed s , sampled freshly by the generator, and its BB84 secret is $(\theta_s, x_s) = F_K(s)$. The erasure mask is authenticated *together with the seed*, $\tau = G_{K'}(s, M)$. This prevents the adversary from inventing its own seed and binds every token to a single mask and a seed issued by the generator. On a query, the hardware recomputes $(\theta_s, x_s) \leftarrow F_K(s)$ and proceeds as in the verification procedure of the single-token scheme NT-TDS(n, η_e, η_y). Since hardware queries are classical, we only need the PRF to be secure against quantum adversaries making *classical* queries, which follows from post-quantum one-way functions [Zha21]. We instantiate $G_{K'}(\cdot)$ with a PRF, which is also a deterministic MAC that is unforgeable even given a verification oracle and polynomially many tagging queries; it can likewise be instantiated from post-quantum secure one-way functions. We define many-token unforgeability next.

Definition 5.1 (Many-token Unforgeability). *Let $n \in \mathbb{N}$. Consider the following experiment with a challenger and QPT adversary $\mathcal{A}$. The challenger runs $\text{KeyGen}(1^n)$ and gives $\mathcal{A}$ classical query access to $\mathcal{O}_{\text{vk}}$. The adversary may adaptively initiate up to $N(n)$ sequential executions of TokenSend over a noiseless channel, interleaved arbitrarily with its oracle queries. Finally, $\mathcal{A}$ outputs (μ_0, σ_0) and (μ_1, σ_1) . Let $\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n, N)$ be the probability that $\text{Verify}^{\mathcal{O}_{\text{vk}}}(\mu_b, \sigma_b) = 1$ for all $b \in \{0, 1\}$, σ_0, σ_1 carry the same seed s and $\mu_0 \neq \mu_1$. The scheme is N -token unforgeable if for every QPT $\mathcal{A}$ with initial state ρ_0 and every polynomial N ,*

$$\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n, N) = \text{negl}(n)$$

Remark 5.2. *Alternatively, the many-token unforgeability experiment can be defined such that $\mathcal{A}$ outputs $N + 1$ signatures $\{\mu_i, \sigma_i\}_i$ and wins if $\text{Verify}^{\mathcal{O}_{\text{vk}}}(\mu_i, \sigma_i) = 1$ for all $i \in [N + 1]$ and all μ_i are distinct. Many works on collusion-resistant primitives work with this notion, however, this definition is only meaningful when the message space has more than N elements. We work with Definition 5.1 because it meaningfully captures the forgeability for any message space, including $\{0, 1\}$. For our construction, the two definitions are equivalent when the message space is large enough.*

Construction. Let $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ and $G : \{0, 1\}^n \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ be post-quantum secure PRFs. The scheme rNT-TDS(n, η_e, η_y) is defined as follows.

- **KeyGen(1^n):** Sample $K, K' \xleftarrow{\$} \{0, 1\}^n$ and set $\text{msk} = (K, K')$. The verification oracle $\mathcal{O}_{\text{vk}}$ is a trusted hardware implementing $P_{K, K'}$:
on input $(s, \mu, \tilde{\sigma}, M, \tau)$,
 1. Output reject if $\tau \neq G_{K'}(s, M)$ or $\text{HW}(M) > \eta_e n$.
 2. Compute $(\theta, x) \leftarrow F_K(s)$.

3. Output **accept** if and only if $|E_\mu^M(\tilde{\sigma})| \leq \eta_y(n - \text{HW}(M))$, where $E_\mu^M(\tilde{\sigma}) = \{i : \theta_i = \mu \wedge M_i = 0 \wedge \tilde{\sigma}_i \neq x_i\}$.

- **TokenSend**($\text{msk}, \mathcal{N}$): The generator samples $s \xleftarrow{\$} \{0, 1\}^n$, computes $(\theta, x) \leftarrow F_K(s)$, and sends s together with $|x^\theta\rangle$, the latter over $\mathcal{N}$. The signer replies with its erasure mask M . The generator aborts if $\text{HW}(M) > \eta_e n$ and otherwise sends $\tau = G_{K'}(s, M)$. The generator tags exactly one mask per session and keeps no state across sessions. Define **state** = $(s, \rho_{\text{msk}}, M, \tau)$ where $\rho_{\text{msk}} = \mathcal{N}(|x^\theta\rangle\langle x^\theta|)$.
- **Sign**(**state**, μ): Same as **NT-TDS**(n, η_e, η_y), output $\sigma = (s, \tilde{\sigma}, M, \tau)$.
- **Verify** $^{\mathcal{O}_{\text{vk}}(\cdot)}(\mu, \sigma)$: Query $\mathcal{O}_{\text{vk}}$ on $(s, \mu, \tilde{\sigma}, M, \tau)$ and output its answer.

For each token, correctness is the same as in Theorem 3.2 with $F_K(s)$ in place of a uniformly random (θ, x) . The correctness argument uses that θ is uniformly random so that each unerased position is checked with probability $1/2$. Replacing a uniformly random (x, θ) with $(x, \theta) = F_K(s)$ changes the honest failure probability by at most $\text{negl}(n)$. The honest experiment, which derives (x, θ) from the seed, sends the token through the channel, signs and verifies, can be run efficiently given classical oracle access to $F_K(\cdot)$, since the channel noise is efficiently samplable. If its failure probability with F_K differed non-negligibly from that with a truly random function, this experiment would distinguish F_K from a random function, contradicting PRF security. With a truly random function, (x, θ) is uniformly random for every token, so the correctness argument of Theorem 3.2 applies, and correctness follows.

Theorem 5.3. *Suppose F and G are pseudorandom functions secure against QPT adversaries with classical query access to $F_K(\cdot), G_{K'}(\cdot)$ for uniformly random K, K' . Then for every QPT forging adversary $\mathcal{A}$ for **rNT-TDS**(n, η_e, η_y) that makes at most $q = \text{poly}(n)$ classical oracle queries to $\mathcal{O}_{\text{vk}}(\cdot)$ and receives at most $N = \text{poly}(n)$ tokens,*

$$\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n, N) = \text{negl}(n)$$

Proof. Consider the following sequence of hybrids. Let $\text{Hyb}_j^{\mathcal{A}}(n, N)$ be the probability that $\mathcal{A}$ wins in Hyb_j :

- **Hyb₀** : This is the experiment from Definition 5.1. $\text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{A}}(n, N) = \text{Hyb}_0^{\mathcal{A}}(n, N)$ for all QPT $\mathcal{A}$.
- **Hyb₁** : This is the same as the previous hybrid except the challenger replaces F_K by a lazily sampled random function R . The function R on query s , samples $(\theta, x) \xleftarrow{\$} \{0, 1\}^{2n}$, records $(s, (\theta, x))$ in a lookup table and returns (θ, x) . If s has been queried before, it returns (θ, x) stored in the lookup table. Now, for every run of **TokenSend**, the challenger simply samples $s \xleftarrow{\$} \{0, 1\}^n$ and queries R to obtain (θ_s, x_s) . For any query $(s, \mu, \tilde{\sigma}, M, \tau)$ to $\mathcal{O}_{\text{vk}}$ by $\mathcal{A}$, the challenger queries R and continues as in the previous hybrid. By the security of PRF F_K for $K \xleftarrow{\$} \{0, 1\}^n$, we have

$$|\text{Hyb}_1^{\mathcal{A}}(n, N) - \text{Hyb}_0^{\mathcal{A}}(n, N)| \leq \text{negl}(n)$$

for all QPT $\mathcal{A}$.

- **Hyb₂** : This is the same as the previous hybrid except the challenger replaces $G_{K'}$ by a lazily sampled random function R' . Following the same argument as in the previous hybrid, we have for all QPT $\mathcal{A}$:

$$|\text{Hyb}_2^{\mathcal{A}}(n, N) - \text{Hyb}_1^{\mathcal{A}}(n, N)| \leq \text{negl}(n)$$

- **Hyb₃** : This is the same as the previous hybrids except the challenger aborts if two sessions of **TokenSend** sample the same seed $s \in \{0, 1\}^n$. Since s is sampled uniformly at random for each session, the probability that this happens is at most $N^2/2^{n+1} = \text{negl}(n)$ for $N = \text{poly}(n)$. Therefore for all QPT $\mathcal{A}$, we have

$$|\text{Hyb}_3^{\mathcal{A}}(n, N) - \text{Hyb}_2^{\mathcal{A}}(n, N)| \leq \text{negl}(n)$$

- **Hyb₄** : The only difference from the previous hybrid is that $\mathcal{O}_{\text{vk}}$ rejects any tuple $(s, \mu, \tilde{\sigma}, M, \tau)$ unless (s, M) is a seed and mask which the generator authenticated in one of the **TokenSend** executions. For any pair (s, M) not authenticated by the challenger, $R'(s, M)$ is uniformly random and independent of the view of $\mathcal{A}$. **Hyb₃** and **Hyb₄** are identical unless $\mathcal{A}$ queries $(s, \mu, \tilde{\sigma}, M, \tau)$ where $\tau = R'(s, M)$ and (s, M) was not authenticated by the challenger. Since $R'(s, M)$ is uniformly random in the view of $\mathcal{A}$, the probability that $\mathcal{A}$ submits such a query is at most $(q + 2)/(2^n - q - 2) = \text{negl}(n)$ for polynomial N and q . Therefore, we have for all QPT $\mathcal{A}$,

$$|\text{Hyb}_4^{\mathcal{A}}(n, N) - \text{Hyb}_3^{\mathcal{A}}(n, N)| \leq \text{negl}(n)$$

In **Hyb₄**, every accepted tuple has the form $(s_i, \mu, M_i, R'(s_i, M_i))$ for some (s_i, M_i) authenticated in the i^{th} execution of **TokenSend**, $i \in [N]$. Whether it is accepted depends only on $(\theta_i, x_i) = R(s_i)$, M_i and $(\mu, \tilde{\sigma})$. (θ_i, x_i) are uniformly random and independent as R is a random function. Let W_i be the event that $\mathcal{A}$ wins **Hyb₄** with $(s_i, \mu_0, \tilde{\sigma}_0, M_i, R'(s_i, M_i))$ and $(s_i, \mu_1, \tilde{\sigma}_1, M_i, R'(s_i, M_i))$ i.e. it forges signature with respect to i^{th} token. Then

$$\text{Hyb}_4^{\mathcal{A}}(n, N) \leq \sum_{i \in [N]} \Pr[W_i]$$

Lemma 5.4. *For every $i \in [N]$, $\Pr_{\text{Hyb}_4}[W_i] \leq \text{TDSForge}_{\eta_e, \eta_y}(n)$ where $\text{TDSForge}_{\eta_e, \eta_y}(n)$ is the forging probability for **NT-TDS**(n, η_e, η_y) defined in Definition 2.2.*

Proof. We can construct an adversary $\mathcal{B}_i$ against one-time unforgeability of **NT-TDS**(n, η_e, η_y) (Definition 2.2). $\mathcal{B}_i$ runs $\mathcal{A}$ and simulates **Hyb₄**, lazily sampling R, R' itself, except in session i :

Description of $\mathcal{B}_i$:

1. When $\mathcal{A}$ initiates the i^{th} session of **TokenSend**, $\mathcal{B}_i$ receives $|x_{\star}^{\theta_{\star}}\rangle$ from its challenger, samples a fresh $s_i \xleftarrow{\$} \{0, 1\}^n$ and sends $(s_i, |x_{\star}^{\theta_{\star}}\rangle)$ to $\mathcal{A}$.
2. Upon receiving M_i from $\mathcal{A}$, $\mathcal{B}_i$ forwards M_i to its challenger. If the challenger aborts, which happens when $\text{HW}(M_i) > \eta_e n$, $\mathcal{B}_i$ also aborts. Otherwise, $\mathcal{B}_i$ receives a tag $\tau_{\star}$ and sends a fresh uniform $\tau_i \xleftarrow{\$} \{0, 1\}^n$ to $\mathcal{A}$.
3. On a query $(s_j, \mu, \tilde{\sigma}, M_j, \tau_j)$ to $\mathcal{O}_{\text{vk}}$ by $\mathcal{A}$, $\mathcal{B}_i$ first checks that (s_j, M_j) is one of the seed and erasure masks authenticated during **TokenSend** and $\tau_j = R'(s_j, M_j)$. If not, it outputs $\perp$.

4. Otherwise $\mathcal{B}_i$ answers the query using $(s_j, R(s_j), R'(s_j, M_j))$ sampled by $\mathcal{B}_i$ for $j \neq i$. For $j = i$, $\mathcal{B}_i$ forwards the query $(\mu, \tilde{\sigma}, M_i, \tau_\star)$ to challenger and returns whatever it outputs.
5. $\mathcal{A}$ outputs $(s_j, \mu_b, \tilde{\sigma}_b, M_j, \tau_j)_{b \in \{0,1\}}$. If $s_j = s_i$ for both $b \in \{0,1\}$, $\mathcal{B}_i$ outputs $(\mu_b, \tilde{\sigma}_b, M_i, \tau_\star)_{b \in \{0,1\}}$. Otherwise $\mathcal{B}_i$ aborts.

The pair $(\theta_\star, x_\star)$ is uniform, identically distributed to $R(s_i)$ in Hyb_4 . The challenger's oracle accepts $(\mu, \tilde{\sigma}, M_i, \tau_\star)$ if and only if $|E_\mu^{M_i}(\tilde{\sigma})| \leq \eta_y(n - \text{HW}(M_i))$ with respect to $(\theta_\star, x_\star)$ which is precisely the check performed in Hyb_4 as well. Hence the view of $\mathcal{A}$ is distributed exactly as in Hyb_4 . Moreover, $\mathcal{B}_i$ makes at most q queries and wins whenever W_i happens. Therefore, we have

$$\Pr_{\text{Hyb}_4}[W_i] \leq \text{TDSForge}_{\eta_e, \eta_y}(n) = \text{negl}(n)$$

□

This completes the proof of Theorem 5.3. □

Reusable one-time programs. The same modification applies to the construction of $\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ (Section 4). The hardware stores (K, K', k) , where K, K' are PRF keys and $k \in \mathcal{K}$ is the function key. A `TokenSend` session with seed s uses the BB84 secrets $\{(\theta_i, x_i)\}_{i \in [\ell_{\text{in}}]} = F_K(s)$, where now $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^{2n\ell_{\text{in}}}$ and $G : \{0,1\}^n \times \{0,1\}^{n+n\ell_{\text{in}}} \rightarrow \{0,1\}^n$ are pseudorandom functions; the input of G is the pair (s, M) with $M \in \{0,1\}^{n\ell_{\text{in}}}$. On input (s, v, r, M, τ) , the hardware checks $\tau = G_{K'}(s, M)$, recomputes $F_K(s)$, runs the per-group checks of $\mathcal{O}_k$ oracle in $\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$, and outputs $f_k(v)$ if all of them pass. `TokenSend` sends s together with the token and authenticates the pair $\tau = G_{K'}(s, M)$. Correctness follows from that of $\text{NT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ by the same PRF argument as for the reusable signature scheme above. We denote the resulting scheme by $\text{rNT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ and define the many-token simulation security below.

Definition 5.5 (Many-token simulation security). *Let $n \in \mathbb{N}$ and $f_k \in \mathcal{F}_n$ as defined in Definition 2.4. Let $f_{k,N}$ be a stateful oracle with a counter c initialized to 0. On a query $v \in \{0,1\}^{\ell_{\text{in}}}$, $f_{k,N}$ outputs $\perp$ if $c \geq N$. Otherwise it outputs $f_k(v)$ and sets $c = c + 1$. For $k \in \mathcal{K}$, QPT adversary $\mathcal{A}$ with initial state ρ_0 and QPT simulator Sim , consider the following two experiments:*

- $\text{Real}_{\mathcal{A},k}(\rho_0, N)$: *The generator runs $(\text{msk}, \mathcal{O}_k) \leftarrow \text{Generate}(1^n, k)$ and provides $\mathcal{A}$ with classical query access to $\mathcal{O}_k$. Starting from ρ_0 , $\mathcal{A}$ executes at most N sessions of `TokenSend` adaptively, over a noiseless channel, interleaved with queries to $\mathcal{O}_k$. The output of the experiment is the final state of $\mathcal{A}$.*
- $\text{Ideal}_{\text{Sim},k}(\rho_0, N)$: *The simulator Sim gets ρ_0 , number of tokens N and classical query access to the stateful oracle $f_{k,N}(\cdot)$. The output of the experiment is the output of $\text{Sim}^{f_{k,N}(\cdot)}(\rho_0, N)$.*

$\text{rNT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ is simulation secure if for every QPT adversary $\mathcal{A}$ there exists a QPT simulator Sim such that for all $k \in \mathcal{K}$, all initial states ρ_0 , all $N = \text{poly}(n)$ and every QPT distinguisher $\mathcal{D}$,

$$|\Pr[\mathcal{D}(\text{Real}_{\mathcal{A},k}(\rho_0, N)) = 1] - \Pr[\mathcal{D}(\text{Ideal}_{\text{Sim},k}(\rho_0, N)) = 1]| = \text{negl}(n)$$

Theorem 5.6. *Under the assumptions of Theorem 5.3, $\text{rNT-QOTP}(n, \ell_{\text{in}}, \eta_e, \eta_y)$ is many-token simulation secure. That is, for every QPT $\mathcal{A}$ that makes at most $q = \text{poly}(n)$ queries and receives $N = \text{poly}(n)$ tokens, there exists a QPT simulator Sim with oracle access to $f_{k,N}$ such that for all $k \in \mathcal{K}$, all initial states ρ_0 and every QPT distinguisher $\mathcal{D}$,*

$$|\Pr[\mathcal{D}(\text{Real}_{\mathcal{A},k}(\rho_0, N)) = 1] - \Pr[\mathcal{D}(\text{Ideal}_{\text{Sim},k}(\rho_0, N)) = 1]| = \text{negl}(n)$$

Proof Sketch. Fix any $k \in \mathcal{K}$ and QPT $\mathcal{A}$ with initial state ρ_0 . We define Hyb_0 to Hyb_4 as in the proof of Theorem 5.3. Let ρ^{Hyb_j} be the final state of $\mathcal{A}$ in Hyb_j , so we have for all QPT distinguishers $\mathcal{D}$

$$\left| \Pr[\mathcal{D}(\text{Real}_{\mathcal{A},k}(\rho_0, N)) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_4}) = 1] \right| \leq \text{negl}(n)$$

In Hyb_4 , let Bad_i be the event that two queries to $\mathcal{O}_k$ with seed s_i are on distinct inputs and both accepted. Then, same as Lemma 4.4, we can construct a QPT $\mathcal{B}_i$, forging adversary for $\text{NT-TDS}(n, \eta_e, \eta_y)$, such that $\Pr_{\text{Hyb}_4}[\text{Bad}_i] \leq \ell_{\text{in}} \cdot \text{TDSForge}_{\eta_e, \eta_y}^{\mathcal{B}_i}(n)$. Therefore, for $\ell_{\text{in}}, N = \text{poly}(n)$, we have

$$\begin{aligned} \Pr_{\text{Hyb}_4}[\text{Bad}] &\leq \sum_{i \in [N]} \Pr_{\text{Hyb}_4}[\text{Bad}_i] \\ &\leq \sum_{i \in [N]} \ell_{\text{in}} \cdot \text{TDSForge}_{\eta_e, \eta_y}(n) \\ &\leq \text{negl}(n) \end{aligned}$$

Let Hyb_5 be the hybrid that is the same as Hyb_4 but answers queries of $\mathcal{A}$ per seed s_i for $i \in [N]$. The first successful³ query (s_i, v^*, r^*, M, τ) for s_i is answered with $f_k(v^*)$ and $(s_i, v^*, f_k(v^*))$ is stored in cache, later valid queries on (s_i, v, r, M, τ) are answered by $f_k(v^*)$ if $v = v^*$, and answered with $\perp$ if $v \neq v^*$. The hybrids Hyb_4 and Hyb_5 are identical until Bad occurs which happens with at most $\text{negl}(n)$ probability so

$$\left| \Pr[\mathcal{D}(\rho^{\text{Hyb}_4}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_5}) = 1] \right| \leq \text{negl}(n)$$

Now the simulator Sim can simply run $\mathcal{A}$ and simulate Hyb_5 by sampling R, R' and the BB84 states on its own. When it receives a query (s, v, r, M, τ) , it can run the consistency checks of Hyb_5 . If the check passes and it is the first valid query for (s, M) which were authenticated in TokenSend , query $f_{k,N}$ on input v and return whatever $f_{k,N}$ outputs (and store the output in cache). Upon any subsequent query (s, v, r, M, τ) , if (s, v) have been queried before and r passes the verification, Sim returns $f_k(v)$ from its cache without querying the oracle. Otherwise, it returns $\perp$ for any other input (s, v', r, M, τ) or r not passing the verification. Since the oracle in Hyb_5 accepts at most one distinct input per seed s_i , $i \in [N]$, Sim queries $f_{k,N}$ at most N times, so $f_{k,N}$ answers each of these queries. The oracle in Hyb_5 and $\text{Ideal}_{\text{Sim},k}(\rho_0, N)$ are identically distributed and for all QPT distinguisher $\mathcal{D}$,

$$\Pr[\mathcal{D}(\rho^{\text{Hyb}_5}) = 1] = \Pr[\mathcal{D}(\text{Ideal}_{\text{Sim},k}(\rho_0, N)) = 1]$$

This completes the proof of Theorem 5.6.

³We call a query successful if it passes the verification check performed by $\mathcal{O}_k$.

5.2 Security against Compromised Hardware

Our security proofs model the hardware as an ideal classical oracle. Deployed TEEs such as Intel SGX, AMD SEV and ARM TrustZone fall short of this ideal. They have been subject to a long line of attacks that can leak the secrets hardwired in the TEE. In our constructions, this is fatal: an adversary that knows (θ, x) or k can evaluate function f_k everywhere. We discuss these attacks in more detail in Section 1.1.

Therefore, we explore the possibility of distributing secrets across t trusted hardware devices, ideally from independent vendors or architectures. The goal is that the one-time program remains simulation secure even if the evaluator learns the full internal state of any $t - 1$ of them.

Security Model. $\text{Generate}(1^n, k, t)$ takes another parameter $t \in \mathbb{N}$ and outputs msk together with t classical programs $P_1, \dots, P_t$ each of which runs inside its own trusted hardware. An adversary outputs the corrupted set $T \subset [t]$, $|T| \leq t - 1$ and receives the complete description, including all the secrets, of $\{P_i\}_{i \in T}$, and classical query access to the honest hardware $\{\mathcal{O}_i\}_{i \in H}$ where $H = [t] \setminus T$ and $\mathcal{O}_i$ implements program P_i . Knowing the description of P_j is equivalent to fully malicious control over hardware $\mathcal{O}_j$.

Definition 5.7 (Security against c corrupted hardware devices). *Let $n \in \mathbb{N}$ and $\mathcal{F}_n$ be a function family with key space $\mathcal{K}$ defined in Definition 2.4. A t -hardware noise-tolerant quantum one-time program for $\mathcal{F}_n$ is secure against $c < t$ corrupted hardware devices if for every $T \subset [t]$ with $|T| \leq c$ and every QPT $\mathcal{A}$, there is a QPT simulator Sim such that for all $k \in \mathcal{K}$ and all initial states ρ_0 , the following two experiments are computationally indistinguishable:*

- $\text{Real}_{\mathcal{A}, k, T}(\rho_0)$: *The generator runs $(\text{msk}, \{P_j\}_{j \in [t]}) \leftarrow \text{Generate}(1^n, k, t)$, sends $\{P_j\}_{j \in T}$ and provides classical query access to oracles $\{\mathcal{O}_j\}_{j \in H}$ to $\mathcal{A}$ where $\mathcal{O}_j$ is the oracle implementing program P_j . Starting from ρ_0 , the adversary $\mathcal{A}$ executes TokenSend with the generator over a noiseless channel, possibly deviating from the protocol interleaved with arbitrary queries to $\{\mathcal{O}_j\}_{j \in H}$. The output of the experiment is the final state of $\mathcal{A}$.*
- $\text{Ideal}_{\text{Sim}, k, T}(\rho_0)$: *The simulator Sim gets ρ_0 , $T \subset [t]$ and classical query access to the stateful one-time oracle $f_{k,1}(\cdot)$ defined in Definition 2.5. The output of the experiment is the output of $\text{Sim}^{f_{k,1}(\cdot)}(\rho_0, T)$.*

We first define an MPC protocol Π that satisfies some properties in Definition 5.8 and compile it into a quantum one-time program.

Definition 5.8 (Malicious secure constant round MPC protocol). *Let $n \in \mathbb{N}$ be the security parameter and let $t \in \mathbb{N}$. Let $f_k : \{0, 1\}^{\ell_{\text{in}}} \rightarrow \{0, 1\}^{\ell_{\text{out}}}$ be a function parametrized by $k \in \mathcal{K}$ where $\mathcal{K}$ admits an XOR sharing scheme i.e. any $k \in \mathcal{K}$ can be decomposed into t shares $\{k_1, \dots, k_t\}$ such that $\oplus_{i \in [t]} k_i = k$ and the joint distribution of any $t - 1$ shares is independent of the secret k .*

Let $\mathcal{P} = (P_1, \dots, P_{t+1})$ be a set of $t + 1$ parties where party P_i has input inp_i and randomness rand_i . Let $\mathcal{F}_{\text{par}}$ be a $(t + 1)$ party functionality with public parameters par . Π is a $(t + 1)$ party post-quantum black-box simulation secure MPC protocol for functionality $\mathcal{F}_{\text{par}}$ in the common reference string (CRS) model that satisfies the following properties:

- *The number of rounds of communication is constant R independent of f_k and t .*

- In each round $r \in [R]$, all parties P_i compute message $m_{i,r} \leftarrow \text{NxtMsg}_i(r, \text{crs}, \text{inp}_i, \Gamma_{<r}, \text{rand}_i)$ and broadcast $m_{i,r}$ to all other parties. Γ_r is the concatenation of messages in round r i.e. $\Gamma_r = m_{1,r} \parallel \dots \parallel m_{t+1,r}$, $\Gamma_0 = \varepsilon$ and $\Gamma_{<r} = \Gamma_1 \parallel \dots \parallel \Gamma_{r-1}$. crs is the common reference string.
- After all rounds of communication, P_{t+1} computes $y \leftarrow \text{Output}(\text{crs}, \text{inp}_{t+1}, \Gamma_{<R+1}, \text{rand}_{t+1})$ where $\Gamma_{<R+1}$ is the complete transcript of Π .

Π satisfies correctness: if all parties are honest, $y = \mathcal{F}_{\text{par}}(\text{inp}_1, \dots, \text{inp}_{t+1})$ with at least $1 - \text{negl}(n)$ probability. We say that Π is secure against t corruptions if for all QPT adversaries $\mathcal{A}$ with initial state ρ_0 corrupting up to t parties, there exists a QPT simulator $\mathcal{S}$ with black box access to $\mathcal{A}$ such that the following two experiments are computationally indistinguishable for every choice of public parameters par , inputs $(\text{inp}_1, \dots, \text{inp}_{t+1})$ and initial state ρ_0 .

- $\text{Real}_{\mathcal{A}, T, \text{par}}^\Pi(\rho_0) : \mathcal{A}$ outputs $T \subset [t+1]$, $|T| \leq t$ and receives $\text{par}, \text{crs}, \{\text{inp}_i\}_{i \in T}$. It plays the role of corrupted parties in Π , receiving and sending all the messages on behalf of parties in T , possibly deviating from the protocol (in particular, it chooses the random tapes of the corrupted parties and may use inputs other than $\{\text{inp}_i\}_{i \in T}$). The output of the experiment is the final state of $\mathcal{A}$.
- $\text{Ideal}_{\mathcal{S}, T, \text{par}}^\Pi(\rho_0) : \mathcal{S}$ receives $T, \rho_0, \text{par}, \{\text{inp}_i\}_{i \in T}$ and access to the ideal functionality $\mathcal{F}_{\text{par}}$. $\mathcal{F}_{\text{par}}$ takes inputs from all the parties where $\mathcal{S}$ sends inputs $\{\text{inp}'_i\}_{i \in T}$ of its choice on behalf of the corrupted parties (which may differ from $\{\text{inp}_i\}_{i \in T}$) and the honest parties send $\{\text{inp}'_i\}_{i \notin T}$. Let $\text{inp}'_i = \text{inp}_i$ for $i \notin T$. If any inp'_i is $\perp$, $\mathcal{F}$ outputs $\perp$. Otherwise it outputs $\mathcal{F}(\text{inp}'_1, \dots, \text{inp}'_{t+1})$. The output of the experiment is the output of $\mathcal{S}$.

Remark 5.9. In the rest of this section, we assume the existence of an MPC protocol Π satisfying Definition 5.8. Informally, we expect such a protocol to exist assuming the quantum hardness of learning with errors (LWE). Constant-round MPC protocols in the CRS model secure against a dishonest majority are known, for instance by instantiating the protocol of [GS22] with the maliciously secure OT in the CRS model of [PVW08], which is based on LWE. These protocols follow the broadcast structure of Definition 5.8 and their simulators are straight-line: they use a trapdoor for the crs instead of rewinding the adversary. Since straight-line simulation does not rewind, we expect their security proofs to carry over to quantum adversaries when all underlying primitives are post-quantum secure. We only require the number of rounds to be a constant, not necessarily two.

Construction Sketch. The generator samples crs , secret shares $k = \oplus_{i \in [t]} k_i$ and creates trusted hardware devices $\{\mathcal{O}_j\}_{j \in [t]}$ where $\mathcal{O}_j$ implements $\text{NxtMsg}_j(\cdot, \text{crs}, k_j, \cdot, \text{rand}_j)$. That is, the oracle $\mathcal{O}_j$ implements the next message function for party P_j with $\text{inp}_j = k_j, \text{rand}_j, \text{crs}$ hardwired in it. Given classical query access to $\{\mathcal{O}_j\}_{j \in [t]}$, the evaluator can learn the message $m_{i,r}$ of party P_i in round r by querying $\mathcal{O}_i$ on $(r, \Gamma_{<r})$. The evaluator plays the role of party P_{t+1} with $\text{inp}_{t+1} = v$ and runs MPC protocol Π by generating the transcript round by round using $\{\mathcal{O}_j\}_{j \in [t]}$ allowing it to learn $f_k(v)$. The following issues arise in this naive implementation of the idea:

1. *Resetting attacks:* Because the hardware (oracle) $\mathcal{O}_j$ is stateless, the evaluator could query an honest party's oracle $\mathcal{O}_j$ in round r on multiple different transcripts. This amounts to resetting attacks on honest parties. We prevent it in the same way we enforce one-time-ness. For every oracle $\mathcal{O}_j$ and round $r \in [R]$, the generator issues a BB84 state $B_{j,r}$. To obtain the

message of P_j in round r , the evaluator has to compute a noise-tolerant tokenized signature, with respect to $B_{j,r}$, on the hash of transcript $\Gamma_{<r}$. The hash is computed using a collision resistant hash function sampled and made public in the beginning of the protocol. By the unforgeability of $\text{NT-TDS}(n, \eta_e, \eta_y)$ and collision resistance of hash, the malicious evaluator can obtain the message $m_{j,r}$ on at most one transcript by querying the honest oracle $\mathcal{O}_j$. The next message function of the first round outputs a fixed message based on the input and randomness of the party P_j , for $j \in [t]$, so the generator can simply send the message instead and does not need a signing token for it as the transcript is the fixed empty string $\Gamma_{<1} = \varepsilon$.

2. *Splicing*: Suppose $\mathcal{O}_j$ and $\mathcal{O}_{j'}$ are hardware devices (oracles) corresponding to honest parties. A malicious evaluator can query $\mathcal{O}_j$ in round r to learn message $m_{j,r}$ on a transcript $\Gamma_{<r}$, attach $m_{j,r}$ to a different transcript $\Gamma'_{<r}$ and query $\mathcal{O}_{j'}$ in round $r+1$ creating multiple transcripts for different messages of the corrupted parties. To circumvent this, we use an EUF-CMA (existential unforgeability under chosen-message attack) secure signature scheme Sig . Each oracle $\mathcal{O}_j$ signs the round number, output message and input transcript - $r \| m_{j,r} \| \text{Hash}(\Gamma_{<r})$ in round r with its signing key Sig.sk_j and the new transcript is $\Gamma_{<r+1} = \Gamma_{<r} \| (m_{1,r}, s_{1,r}) \| \dots \| (m_{t,r}, s_{t,r}) \| m_{t+1,r}$ where $s_{j,r}$ is the signature on $r \| m_{j,r} \| \text{Hash}(\Gamma_{<r})$ under Sig.sk_j . Including the round number r ensures that a signature output in one round cannot be used as a signature for any other round. Upon querying any oracle on transcript $\Gamma_{<r}$, it checks that the transcript consists of messages with valid signatures using $\{\text{Sig.vk}_j\}_{j \in [t]}$ and outputs $\perp$ if the check fails.
3. *Committed shares*: Finally, the MPC protocol Π is parameterized by a commitment key ck and commitments $\{\text{com}_i\}_{i \in [t]}$ and securely implements the functionality $\mathcal{G}_{\text{com}}$ which takes as input $\text{inp}_j = (k_j, \delta_j)$ from party $P_{j \in [t]}$, checks if it is a valid opening to com_j under ck . If not, it outputs $\perp$, otherwise it takes as input $\text{inp}_{t+1} = v$ from P_{t+1} and outputs $f_{\oplus_j k_j}(v)$. Thus the functionality prohibits an adversary from changing shares k_j to learn the output of a related function $f_{k'}$. In the one-time program, this is done simply by the generator sampling a commitment key $\text{ck} \leftarrow \text{Com.Setup}(1^n)$, generating $\{\text{com}_i\}_{i \in [t]}$ from the shares $\{k_i\}_{i \in [t]}$ under ck using a statistically binding and computationally hiding commitment scheme Com and generating $\{\mathcal{O}_j\}_j$ such that they implement Π for $\mathcal{G}_{\text{com}}$. $\text{par} = (\text{ck}, \{\text{com}_i\}_{i \in [t]})$ are the public parameters of the functionality and are known to all the parties as defined in Definition 5.8. We stress that ck is sampled by the generator independently of crs : in the security proof, the MPC simulator may generate crs (e.g., together with an extraction trapdoor), but ck is always sampled honestly.

We denote the primitive by $\text{dNT-QOTP}(n, t, \ell_{\text{in}}, \eta_e, \eta_y)$ and present the construction along with necessary ingredients below.

Ingredients:

- A collision resistant hash family $\{\mathcal{H}_n\}_n$ with output length $h_n \in \mathbb{N}$ secure against QPT adversaries.
- An EUF-CMA secure signature scheme $\text{Sig} = (\text{KeyGen}, \text{Sign}, \text{Verify})$ with deterministic signing secure against QPT adversaries.
- An EUF-1CMA secure MAC scheme $\text{MAC} = (\text{KeyGen}, \text{Tag}, \text{Verify})$ described in Definition 2.1.

- A statistically binding and computationally hiding commitment scheme $\text{Com} = (\text{Setup}, \text{Commit}, \text{Open})$ secure against QPT adversaries, with the following syntax:
 - $\text{Com.Setup}(1^n)$: outputs a commitment key ck .
 - $\text{Com.Commit}(\text{ck}, m; \delta)$: takes as input the commitment key ck , a message m and randomness δ , and outputs a commitment com .
 - $\text{Com.Open}(\text{ck}, \text{com}, m, \delta)$: outputs 1 (accept) or 0 (reject). For all ck, m, δ ,

$$\text{Com.Open}(\text{ck}, \text{Com.Commit}(\text{ck}, m; \delta), m, \delta) = 1$$

Statistical binding requires that, with probability $1 - \text{negl}(n)$ over $\text{ck} \leftarrow \text{Com.Setup}(1^n)$, there exist no com and openings $(m, \delta), (m', \delta')$ with $m \neq m'$ such that $\text{Com.Open}(\text{ck}, \text{com}, m, \delta) = \text{Com.Open}(\text{ck}, \text{com}, m', \delta') = 1$. *Computational hiding* requires that for $\text{ck} \leftarrow \text{Com.Setup}(1^n)$ and any QPT adversary that, given ck , outputs equal-length messages m_0, m_1 , the commitments $\text{Com.Commit}(\text{ck}, m_0; \delta)$ and $\text{Com.Commit}(\text{ck}, m_1; \delta)$ for uniformly random δ are computationally indistinguishable given ck .

- A noise-tolerant tokenized digital signature scheme $\text{NT-TDS}(n, h, \eta_e, \eta_y)$ described in Remark 4.1 denoted by NT-TDS hereafter.
- A $(t + 1)$ party functionality $\mathcal{G}_{\text{com}}$ parametrized by $\text{par} = (\text{ck}, \{\text{com}_i\}_{i \in [t]})$. Parties P_1 to P_t input $\{k_i, \delta_i\}_{i \in [t]}$ and P_{t+1} inputs v . If for any $i \in [t]$, $\text{Com.Open}(\text{ck}, \text{com}_i, k_i, \delta_i) = 0$, $\mathcal{G}_{\text{com}}$ outputs $\perp$. Otherwise it outputs $f_{\oplus_i k_i}(v)$ to P_{t+1} .
- A $(t + 1)$ party MPC protocol Π for $\mathcal{G}_{\text{com}}$ described in Definition 5.8.

Remark 5.10 (On the assumptions). *Theorem 5.11 shows that $\text{dNT-QOTP}(n, t, \ell_{\text{in}}, \eta_e, \eta_y)$ is simulation secure, as long as at least one hardware is uncorrupted, assuming the existence of an MPC protocol Π satisfying Definition 5.8 together with the remaining ingredients above. As discussed in Remark 5.9, we expect all of these ingredients to be instantiable from the quantum hardness of LWE. We stress that this instantiation is only meant as a proof of concept, and some of these ingredients can be instantiated from weaker assumptions, or the construction can be modified to avoid some of them altogether.*

Generate($1^n, k, t$) :

1. Sample $\text{crs}, \text{Hash} \leftarrow \mathcal{H}$ and, independently, a commitment key $\text{ck} \leftarrow \text{Com.Setup}(1^n)$.
2. Sample uniform shares $k_1, \dots, k_t$ with $\bigoplus_j k_j = k$, commitments $\text{com}_j = \text{Com.Commit}(\text{ck}, k_j; \delta_j)$, and signing key pairs $(\text{Sig.sk}_j, \text{Sig.vk}_j) \leftarrow \text{Sig.KeyGen}(1^n, \ell)$, for all $j \in [t]$, where $\ell = \lceil \log R \rceil + \max_{j,r} |m_{j,r}| + h$. The round number $r \in [R]$ is encoded as a $\lceil \log R \rceil$ -bit string in every signed message.
3. For every $j \in [t]$ and every $r \in \{2, \dots, R\}$, compute $(\text{NT-TDS.msk}_{j,r}, \mathcal{O}_{j,r}^{\text{vk}}) \leftarrow \text{NT-TDS.KeyGen}(1^n, h)$.
4. Let $\text{msk} = \{\text{NT-TDS.msk}_{j,r}, \text{Sig.sk}_j\}_{j \in [t], 2 \leq r \leq R}$ and $\text{pp} = (\text{crs}, \text{ck}, \text{Hash}, \{\text{com}_j, \text{Sig.vk}_j, \mathcal{O}_{j,r}^{\text{vk}}\}_{j \in [t], 2 \leq r \leq R})$.

5. Compute trusted hardware $\mathcal{O}_j^{\text{pp}}$ for $j \in [t]$, that implements the program P_j . P_j has hard-wired public parameters pp , the opening (k_j, δ_j) , a random tape rand_j , signing key Sig.sk_j , and $\text{NT-TDS.msk}_{j,r}$. For a transcript $\Gamma_{<r} = \Gamma_1 \parallel \dots \parallel \Gamma_{r-1}$ with $\Gamma_i = (m_{1,i}, s_{1,i}) \parallel \dots \parallel (m_{t,i}, s_{t,i}) \parallel m_{t+1,i}$, we denote by $\bar{\Gamma}_{<r} = \bar{\Gamma}_1 \parallel \dots \parallel \bar{\Gamma}_{r-1}$, where $\bar{\Gamma}_i = m_{1,i} \parallel \dots \parallel m_{t+1,i}$, the transcript of Π obtained by removing all the signatures. P_j answers two kinds of queries.

- (1): Compute the round-1 message $m_{j,1}$ of P_j from its input (k_j, δ_j) , its random tape rand_j and pp , and output $(m_{j,1}, \text{Sig.Sign}(\text{Sig.sk}_j, 1 \parallel m_{j,1} \parallel 0^h))$.
- $(r, \Gamma_{<r}, \sigma_r)$ for $r \geq 2$:
 - (a) Check that $\Gamma_{<r}$ is well formed. That is $\Gamma_{<r} = \Gamma_1 \parallel \dots \parallel \Gamma_{r-1}$ and

$$\Gamma_i = (m_{1,i}, s_{1,i}) \parallel \dots \parallel (m_{t,i}, s_{t,i}) \parallel m_{t+1,i}$$

for all $i \in [r-1]$.

- (b) For each $i \in [t], j \in [r-1]$, check that $\text{Sig.Verify}(\text{Sig.vk}_i, 1 \parallel m_{i,j} \parallel 0^h, s_{i,j}) = 1$ if $j = 1$ or check $\text{Sig.Verify}(\text{Sig.vk}_i, j \parallel m_{i,j} \parallel \text{Hash}(\Gamma_{<j}), s_{i,j}) = 1$ if $j \neq 1$. Output $\perp$ if the check fails.
- (c) If $\text{NT-TDS.Verify}^{\mathcal{O}_{j,r}^{\text{vk}}}(\text{Hash}(\Gamma_{<r}), \sigma_r)$ rejects, output $\perp$. Otherwise continue.
- (d) If all checks pass, compute the round- r message $m_{j,r} \leftarrow \Pi.\text{NxtMsg}_j(r, \text{crs}, (k_j, \delta_j), \bar{\Gamma}_{<r}, \text{rand}_j)$ and output $(m_{j,r}, \text{Sig.Sign}(\text{Sig.sk}_j, r \parallel m_{j,r} \parallel \text{Hash}(\Gamma_{<r})))$.

TokenSend($\text{msk}, \mathcal{N}$):

For each $j \in [t], r \in \{2, \dots, R\}$, the generator and evaluator run $\text{TokenSend}(\text{NT-TDS.msk}_{j,r}, \mathcal{N})$. If TokenSend does not abort, the evaluator learns $\text{state}_{j,r} = (B_{j,r}, M_{j,r}, \tau_{j,r})$ where $B_{j,r}$ is the quantum signing token, $M_{j,r} \in \{0, 1\}^{nh}$ is the erasure mask and $\tau_{j,r}$ is its MAC tag. Let $\text{state} = \{\text{state}_{j,r}\}_{j,r}$.

Evaluate $^{\{\mathcal{O}_j^{\text{pp}}\}_j}(\text{pp}, \text{state}, v)$:

1. Initialize $\Gamma_{<1} = \varepsilon$ i.e. empty string and sample randomness rand_{t+1} .
2. Compute $m_{t+1,1} \leftarrow \Pi.\text{NxtMsg}_{t+1}(1, \text{crs}, v, \Gamma_{<1}, \text{rand}_{t+1})$.
3. For all $j \in [t]$, query $\mathcal{O}_j^{\text{pp}}$ on input (1) and let $(m_{j,1}, s_{j,1})$ be the output.
4. Let $\Gamma_{<2} = (m_{1,1}, s_{1,1}) \parallel \dots \parallel (m_{t,1}, s_{t,1}) \parallel m_{t+1,1}$.
5. For $r \in \{2, \dots, R\}$ do:
 - (a) Compute $m_{t+1,r} \leftarrow \Pi.\text{NxtMsg}_{t+1}(r, \text{crs}, v, \bar{\Gamma}_{<r}, \text{rand}_{t+1})$.
 - (b) For $j \in [t]$ do:
 - i. Compute $h_r = \text{Hash}(\Gamma_{<r})$ and $\sigma_{j,r} \leftarrow \text{NT-TDS.Sign}(\text{state}_{j,r}, h_r)$.
 - ii. Query $\mathcal{O}_j^{\text{pp}}$ on input $(r, \Gamma_{<r}, \sigma_{j,r})$ and let $(m_{j,r}, s_{j,r})$ be the output.
 - (c) Set $\Gamma_r = (m_{1,r}, s_{1,r}) \parallel \dots \parallel (m_{t,r}, s_{t,r}) \parallel m_{t+1,r}$ and $\Gamma_{<r+1} = \Gamma_{<r} \parallel \Gamma_r$.
6. Output $y \leftarrow \Pi.\text{Output}(\text{crs}, v, \bar{\Gamma}_{<R+1}, \text{rand}_{t+1})$

Correctness and communication complexity. By correctness of NT-TDS, TokenSend does not abort and each honestly generated signature $\sigma_{j,r}$ passes verification, except with probability $\text{negl}(n)$. By correctness of Sig, the signatures in an honestly built transcript pass the signature checks of every oracle. Hence, by a union bound over the $t(R-1)$ signatures, where $t = \text{poly}(n)$ and R is constant, every oracle returns the correct next message of Π except with $\text{negl}(n)$ probability. The evaluator then holds an honest transcript of Π , and by correctness of Π it outputs $\mathcal{G}_{\text{com}}((k_1, \delta_1), \dots, (k_t, \delta_t), v) = f_k(v)$ except with $\text{negl}(n)$ probability. The quantum token $B_{j,r}$ is a BB84 state over nh qubits (as $|\text{Hash}(\Gamma_{<r})| = h$). So the total token size is $t(R-1)nh = \text{poly}(n)$.

Theorem 5.11. *Assume the existence of the ingredients listed above, in particular an MPC protocol Π satisfying Definition 5.8. Then $\text{dNT-QOTP}(n, t, \ell_{\text{in}}, \eta_e, \eta_y)$ is simulation secure against a malicious evaluator corrupting all but one hardware. Concretely, let $T \subset [t]$, $|T| \leq t-1$. For all QPT $\mathcal{A}$, there exists QPT simulator Sim such that for all QPT distinguisher $\mathcal{D}$ and $k \in \mathcal{K}$,*

$$|\Pr[\mathcal{D}(\text{Real}_{\mathcal{A},k,T}(\rho_0)) = 1] - \Pr[\mathcal{D}(\text{Ideal}_{\text{Sim},k,T}(\rho_0)) = 1]| \leq \text{negl}(n)$$

where $\text{Real}_{\mathcal{A},k,T}(\rho_0)$, $\text{Ideal}_{\text{Sim},k,T}(\rho_0)$ are defined in Definition 5.7.

Proof. Fix $k \in \mathcal{K}$, $T \subset [t]$, $|T| = t-1$ and QPT adversary $\mathcal{A}$ with initial state ρ_0 . Let $H = [t] \setminus T$ be the set of honest hardware devices (oracles). If any QPT $\mathcal{A}$ corrupting $t-1$ hardware devices is simulatable, then any adversary corrupting $T' \subset T$ is also simulatable. Therefore, we only consider maximal adversaries $\mathcal{A}$ that corrupt $t-1$ hardware devices. WLOG we assume that $T = [t-1]$ is the set of corrupted hardware devices and $\mathcal{O}_t^{\text{pp}}$ is the only honest hardware for which $\mathcal{A}$ has classical query access.

Consider the following sequence of hybrids. Denote the final state of $\mathcal{A}$ in Hyb_j by ρ^{Hyb_j} :

- **Hyb₀:** This is the real experiment $\text{Real}_{\mathcal{A},k,T}(\rho_0)$. $\mathcal{A}$ gets $\text{pp}, \{k_i, \delta_i, \text{rand}_i\}_{i \in [t-1]}$ in clear and classical query access to $\mathcal{O}_t^{\text{pp}}$. We have $\rho^{\text{Hyb}_0} = \text{Real}_{\mathcal{A},k,T}(\rho_0)$.
- **Hyb₁:** The only difference from Hyb_0 is that $\mathcal{A}$ gets classical query access to a stateful oracle $\mathcal{O}_t^{(1)}$. For every $r \in \{2, \dots, R\}$, $\mathcal{O}_t^{(1)}$ outputs $(m_{t,r}, s_{t,r}) \leftarrow \mathcal{O}_t^{\text{pp}}(r, \Gamma_{<r}, \sigma_r)$ on first valid query that passes all checks and stores $(\Gamma_{<r}, m_{t,r}, s_{t,r})$. On every subsequent query $(r, \Gamma'_{<r}, \sigma'_r)$, $\mathcal{O}_t^{(1)}$ outputs $(m_{t,r}, s_{t,r})$ if $\Gamma'_{<r} = \Gamma_{<r}$ and $\text{NT-TDS}^{\mathcal{O}_{t,r}^{\text{yk}}}(\text{Hash}(\Gamma'_{<r}), \sigma'_r) = 1$ and $\perp$ on every other query.
- **Hyb₂:** The only difference from the previous hybrid is that $\mathcal{A}$ gets query access to $\mathcal{O}_t^{(2)}$. For any $2 \leq r \leq R$, $\mathcal{O}_t^{(2)}$ outputs $\perp$ if $\mathcal{A}$ queries $(r, \Gamma_{<r}, \sigma_r)$ where $\Gamma_{<r}$ contains a message $m_{t,r'}$ that was never output by $\mathcal{O}_t^{(2)}$. That is for every round r' , $\mathcal{O}_t^{(2)}$ records the first valid query that passes all the checks and on a query $(r, \Gamma_{<r}, \sigma_r)$ for $r > r'$, it checks that the messages $m_{t,1}, m_{t,2}, \dots, m_{t,r-1}$ recorded by $\mathcal{O}_t^{(2)}$ match with the messages in transcript $\Gamma_{<r}$.

If this check passes, $\mathcal{O}_t^{(2)}$ outputs whatever $\mathcal{O}_t^{(1)}$ outputs. Otherwise it outputs $\perp$.

Lemma 5.12. *For all QPT distinguisher $\mathcal{D}$,*

$$\left| \Pr[\mathcal{D}(\rho^{\text{Hyb}_0}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1] \right| \leq \text{negl}(n)$$

Proof. Let Bad_r be the event that $\mathcal{A}$ makes two successful queries $(r, \Gamma_{<r}, \sigma_r)$ and $(r, \Gamma'_{<r}, \sigma'_r)$ such that

$$\Gamma_{<r} \neq \Gamma'_{<r} \quad \text{and} \quad \text{NT-TDS}^{\mathcal{O}_{t,r}^{\text{vk}}}(\text{Hash}(\Gamma'_{<r}), \sigma'_r) = 1$$

By union bound, we have $\Pr[\text{Bad}] = \Pr[\cup_r \text{Bad}_r] \leq \sum_{r=2}^R \Pr[\text{Bad}_r]$.

Conditioned on $\neg \text{Bad}$, every valid query of round r carries the same transcript $\Gamma_{<r}$ as the first valid query of round r . On every such query, $\mathcal{O}_t^{\text{pp}}$ outputs the same pair $(m_{t,r}, s_{t,r})$ as on the first one, because $\Pi.\text{NxtMsg}_t$ is deterministic given the hard-wired random tape rand_t and Sig.Sign is deterministic. This is exactly the stored answer that $\mathcal{O}_t^{(1)}$ returns. Hence $\mathcal{O}_t^{\text{pp}}$ and $\mathcal{O}_t^{(1)}$ answer every query identically, and the view of $\mathcal{A}$ is identical in both hybrids. The probability that Bad_r , for any r , occurs in either Hyb_0 or Hyb_1 is at most $\text{negl}(n)$ by the unforgeability of NT-TDS (Definition 2.2). Suppose Bad_r happens in Hyb_0 or Hyb_1 , then we have following two possibilities:

- $\text{Hash}(\Gamma_{<r}) = \text{Hash}(\Gamma'_{<r})$. This will happen with $\text{negl}(n)$ probability as $\Gamma_{<r} \neq \Gamma'_{<r}$ and Hash is collision resistant.
- $\text{Hash}(\Gamma_{<r}) \neq \text{Hash}(\Gamma'_{<r})$. This will also happen with at most $\text{negl}(n)$ probability because NT-TDS is one-time unforgeable. If not, then we can construct a forging adversary $\mathcal{B}$ for NT-TDS that runs $\mathcal{A}$ and outputs $(\text{Hash}(\Gamma_{<r}), \sigma_r)$ and $(\text{Hash}(\Gamma'_{<r}), \sigma'_r)$ i.e. valid signatures for two different messages with just one token $B_{t,r}$.

Hence Bad_r , for all r , occurs with at most $\text{negl}(n)$ probability and $\Pr[\text{Bad}] \leq \text{negl}(n)$. Therefore for all QPT $\mathcal{D}$,

$$\left| \Pr[\mathcal{D}(\rho^{\text{Hyb}_0}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1] \right| \leq \text{negl}(n)$$

□

Lemma 5.13. *For all QPT distinguisher $\mathcal{D}$,*

$$\left| \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_2}) = 1] \right| \leq \text{negl}(n)$$

Proof. In both hybrids 1 and 2, for each $2 \leq r \leq R$, all valid queries (for which the oracle does not output $\perp$) of round r carry the same transcript $\Gamma_{<r}$ as the first valid query of round r and receive the same answer $(m_{t,r}, s_{t,r})$ recorded by the oracle. In particular, the oracle signs at most one message per round under Sig.sk_t , namely $1 \| m_{t,1} \| 0^h$ in round 1 and $r \| m_{t,r} \| \text{Hash}(\Gamma_{<r})$ in round $r \geq 2$ for the recorded $\Gamma_{<r}$. Let Bad_r be the event that $\mathcal{A}$ makes a query $(r, \Gamma_{<r}, \sigma_r)$ that passes all the checks of $\mathcal{O}_t^{(1)}$ and, for some $r' < r$, $\Gamma_{<r}$ contains a pair $(m'_{t,r'}, s'_{t,r'})$ such that either no message has been recorded by the oracle for round r' so far, or $m'_{t,r'} \neq m_{t,r'}$ where $m_{t,r'}$ is the recorded message. Since the query passes the signature checks, $\text{Sig.Verify}(\text{Sig.vk}_t, r' \| m'_{t,r'} \| \text{Hash}(\Gamma_{<r'}), s'_{t,r'}) = 1$ (with $\text{Hash}(\Gamma_{<r'})$ replaced by 0^h if $r' = 1$). However, the message $r' \| m'_{t,r'} \| \text{Hash}(\Gamma_{<r'})$ was never signed by the oracle: messages signed in any other round start with a different round number, and the only message signed in round r' (if any) contains $m_{t,r'} \neq m'_{t,r'}$. Hence whenever Bad_r occurs, $\mathcal{A}$ has produced a valid signature under Sig.vk_t on a message that was never signed. Therefore if Bad_r happens with non-negligible probability in Hyb_1 or Hyb_2 , we can construct an EUF-CMA forger $\mathcal{B}$ for Sig that receives Sig.vk_t , samples all the other secrets itself, simulates the hybrid for $\mathcal{A}$ using its signing oracle to compute signatures under Sig.sk_t , and outputs $(r' \| m'_{t,r'} \| \text{Hash}(\Gamma_{<r'}), s'_{t,r'})$ when Bad_r occurs for the first time. So $\Pr_{\text{Hyb}_1}[\text{Bad}_r] = \Pr_{\text{Hyb}_2}[\text{Bad}_r] \leq \text{negl}(n)$ by EUF-CMA security of Sig and $\Pr_{\text{Hyb}_1}[\text{Bad}] = \Pr_{\text{Hyb}_2}[\text{Bad}] \leq \sum_{r=2}^R \Pr_{\text{Hyb}_2}[\text{Bad}_r] \leq \text{negl}(n)$.

So $\Pr[\text{Bad}] \leq \text{negl}(n)$ and conditioned on $\neg \text{Bad}$, the output distribution of oracles in $\text{Hyb}_1, \text{Hyb}_2$ is identical and the view of $\mathcal{A}$ is also identical. Therefore, for all QPT $\mathcal{D}$,

$$\left| \Pr[\mathcal{D}(\rho^{\text{Hyb}_1}) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_2}) = 1] \right| \leq \text{negl}(n)$$

□

Lemma 5.14. *In Hyb_2 , the following holds except with negligible probability:*

1. *For each round $2 \leq r \leq R$, all valid queries of round r , i.e., queries $(r, \Gamma_{<r}, \sigma_r)$ on which $\mathcal{O}_t^{(2)}$ does not output $\perp$, carry the same transcript $\Gamma_{<r}$ and receive the same answer $(m_{t,r}, s_{t,r})$. Let $q_r = (r, \Gamma_{<r}, \sigma_r)$ denote the first valid query of round r , if one exists, and let $Q = \{q_r\}_r$ be the set of these first valid queries.*
2. *Let $q_r, q_{r'} \in Q$ with $r < r'$. Then q_r is queried before $q_{r'}$.*
3. *There is a single transcript $\Gamma_{<R+1}$ such that for all $2 \leq r \leq R$ and $(r, \Gamma_{<r}, \sigma_r) \in Q$, $\Gamma_{<r}$ is a prefix of $\Gamma_{<R+1}$.*

Proof. (1) follows from the description of $\mathcal{O}_t^{(2)}$ in Hyb_2 . It either outputs $\perp$ or forwards the query to $\mathcal{O}_t^{(1)}$. For each $2 \leq r \leq R$, $\mathcal{O}_t^{(1)}$ answers a query of round r other than $\perp$ only if its transcript equals the transcript $\Gamma_{<r}$ stored at the first valid query of round r , and it then returns the stored answer $(m_{t,r}, s_{t,r})$. Hence all valid queries of round r carry $\Gamma_{<r}$ and receive $(m_{t,r}, s_{t,r})$, and Q contains at most one query per round. In particular, the oracle signs at most one message under Sig.sk_t in each round.

For (2), let $q_{r'} \in Q$ and $r < r'$. Since $q_{r'}$ is valid, it passes the check of $\mathcal{O}_t^{(2)}$, so the round- r message of P_t in its transcript equals the message $m_{t,r}$ recorded by $\mathcal{O}_t^{(2)}$. A round- r message is recorded only at the first valid query of round r . Hence q_r exists and was made before $q_{r'}$.

For (3) suppose that it is not true and there exists $(r_1, \Gamma_{<r_1}, \sigma_{r_1}), (r_2, \Gamma_{<r_2}, \sigma_{r_2}) \in Q$ for $r_1 > r_2$ such that $\Gamma_{<r_2}$ is not a prefix of $\Gamma_{<r_1}$. Then, we can always find a r such that $r_2 \leq r \leq r_1$ and $\Gamma_{<r}$ is not a prefix of $\Gamma_{<r+1}$. That is $\Gamma_{<r+1} = \Gamma'_{<r} \parallel \Gamma_r$ for $\Gamma'_{<r} \neq \Gamma_{<r}$. This means that Γ_r contains a pair $(m_{t,r}, s'_{t,r})$, where $m_{t,r}$ is the message recorded by the oracle in round r (by the check of $\mathcal{O}_t^{(2)}$), such that $\text{Sig.Verify}(\text{Sig.vk}_t, r \parallel m_{t,r} \parallel \text{Hash}(\Gamma'_{<r}), s'_{t,r}) = 1$, whereas the only message signed by the oracle in round r is $r \parallel m_{t,r} \parallel \text{Hash}(\Gamma_{<r})$. Now we have two following possibilities:

- $\text{Hash}(\Gamma_{<r}) = \text{Hash}(\Gamma'_{<r})$. In this case, $\mathcal{A}$ has produced a collision of Hash . By the collision resistant property of Hash , this will happen with at most $\text{negl}(n)$ probability.
- $\text{Hash}(\Gamma'_{<r}) \neq \text{Hash}(\Gamma_{<r})$. In this case, $\mathcal{A}$ has forged a signature $s'_{t,r}$ on $r \parallel m_{t,r} \parallel \text{Hash}(\Gamma'_{<r})$ under Sig.sk_t . The message $r \parallel m_{t,r} \parallel \text{Hash}(\Gamma'_{<r})$ was never signed by the oracle: messages signed in other rounds start with a different round number, and the only message signed in round r is $r \parallel m_{t,r} \parallel \text{Hash}(\Gamma_{<r})$ by (1). Then by the EUF-CMA security of Sig , this case will also happen with at most $\text{negl}(n)$ probability.

Hence (3) always holds in Hyb_2 except with at most $\text{negl}(n)$ probability. □

Thus, there is a single unique transcript for Π in Hyb_2 that builds round by round. This allows us to construct a corresponding adversary $\mathcal{A}_\Pi$ for the MPC protocol that runs $\mathcal{A}$.

Description of $\mathcal{A}_\Pi$:

1. $\mathcal{A}_\Pi$ corrupts parties $\{P_i\}_{i \in [t-1]}$ and P_{t+1} .
2. $\mathcal{A}_\Pi$ receives crs , $\text{par} = (\text{ck}, \{\text{com}_i\}_{i \in [t]})$ and inputs of corrupted parties $\text{inp}_i = (k_i, \delta_i)$ for $i \in [t-1]$ (the input of P_{t+1} is arbitrary and ignored by $\mathcal{A}_\Pi$). It samples the random tapes rand_i , for $i \in [t-1]$, uniformly at random exactly as in **Generate**.
3. $\mathcal{A}_\Pi$ samples $(\text{NT-TDS.msk}_{i,r}, \mathcal{O}_{i,r}^{\text{vk}}) \leftarrow \text{NT-TDS.KeyGen}(1^n, h)$ and $(\text{Sig.sk}_i, \text{Sig.vk}_i) \leftarrow \text{Sig.KeyGen}(1^n, \ell)$ for all $i \in [t], 2 \leq r \leq R$.
4. $\mathcal{A}_\Pi$ samples $\text{Hash} \leftarrow \mathcal{H}$ and sends pp to $\mathcal{A}$ where

$$\text{pp} = \left(\text{crs}, \text{ck}, \text{Hash}, \{\text{com}_j, \text{Sig.vk}_j, \mathcal{O}_{j,r}^{\text{vk}}\}_{j \in [t], 2 \leq r \leq R} \right)$$

5. $\mathcal{A}_\Pi$ sends the descriptions of the programs $\{P_i\}_{i \in [t-1]}$ to $\mathcal{A}$. Note that program P_i of one-time program is implementing the computation of party P_i in Π .
6. $\mathcal{A}_\Pi$ runs $\text{TokenSend}(\text{NT-TDS.msk}_{i,r}, \cdot)$ with $\mathcal{A}$ for all $i \in [t]$ and $2 \leq r \leq R$, playing the role of the honest generator. In particular, $\mathcal{A}$ receives the signing tokens $B_{i,r}$ and the tags on its erasure masks exactly as in **Hyb₂**.
7. $\mathcal{A}_\Pi$ keeps a cache to store all queries on which $\mathcal{A}_\Pi$ returns anything other than $\perp$, along with the responses.
8. Whenever $\mathcal{A}$ queries on (1), $\mathcal{A}_\Pi$ returns the cached response if there is one. Otherwise, $\mathcal{A}_\Pi$ receives $m_{t,1}$ from honest party P_t , computes

$$s_{t,1} \leftarrow \text{Sig.Sign}(\text{Sig.sk}_t, 1 \| m_{t,1} \| 0^h)$$

and sends $(m_{t,1}, s_{t,1})$ to $\mathcal{A}$. $\mathcal{A}_\Pi$ records the response in cache.

9. Whenever $\mathcal{A}$ queries on $(r, \Gamma_{<r}, \sigma_r)$, $\mathcal{A}_\Pi$ runs the checks of $\mathcal{O}_t^{(2)}$ from **Hyb₂** (including the check of σ_r using $\text{NT-TDS.msk}_{t,r}$) using its prior recorded responses, and returns $\perp$ if any check fails. If the checks pass and the cache already contains a response for round r , it returns the cached response if $\Gamma_{<r}$ is the cached transcript and $\perp$ otherwise. Otherwise, it parses $m_{1,r-1}, \dots, m_{t-1,r-1}$ and $m_{t+1,r-1}$ from $\Gamma_{<r}$ and sends them to P_t as the round- $(r-1)$ messages of the corrupted parties. It receives $m_{t,r}$ from P_t , computes $s_{t,r} \leftarrow \text{Sig.Sign}(\text{Sig.sk}_t, r \| m_{t,r} \| \text{Hash}(\Gamma_{<r}))$, sends $(m_{t,r}, s_{t,r})$ to $\mathcal{A}$ and records $(r, \Gamma_{<r}, m_{t,r}, s_{t,r})$ in cache.
10. Finally it outputs whatever $\mathcal{A}$ outputs.

$\mathcal{A}_\Pi$ simulates $\mathcal{A}$ exactly as in **Hyb₂** and is a valid adversary of Π corrupting t out of $t+1$ parties whenever the conditions in Lemma 5.14 hold, that is $\mathcal{A}$ queries in a manner which forms a single transcript $\Gamma_{<R+1}$ for Π . Since these conditions hold except with $\text{negl}(n)$ probability, for all QPT $\mathcal{D}$,

$$\left| \Pr[\mathcal{D}(\text{Real}_{\mathcal{A}_\Pi, \text{par}, T}^\Pi(\rho_0)) = 1] - \Pr[\mathcal{D}(\rho^{\text{Hyb}_2}) = 1] \right| \leq \text{negl}(n)$$

where $\text{Real}_{\mathcal{A}_\Pi, \text{par}, T}^\Pi(\rho_0)$ is the output of $\mathcal{A}_\Pi$ as defined in Definition 5.8 with $\text{par} = (\text{ck}, \{\text{com}_i\}_{i \in [t]})$ generated as in **Generate**, i.e., $\text{ck} \leftarrow \text{Com.Setup}(1^n)$ and $\text{com}_i = \text{Com.Commit}(\text{ck}, k_i; \delta_i)$ for all $i \in [t]$.

By the simulation security of Π from Definition 5.8, there exists a simulator $\mathcal{S}$ that gets ρ_0 , $\text{par} = (\text{ck}, \{\text{com}_i\}_{i \in [t]})$, the corrupted parties' inputs $\{k_i, \delta_i\}_{i \in [t-1]}$, a single classical query to $\mathcal{G}_{\text{com}}$ and black box access to $\mathcal{A}_{\Pi}^{\Pi}$. $\text{Real}_{\mathcal{A}_{\Pi}, \text{par}, T}^{\Pi}(\rho_0)$ is computationally indistinguishable from $\text{Ideal}_{\mathcal{S}, \text{par}, T}^{\Pi}(\rho_0)$. Note that $\mathcal{S}$ may generate crs itself (e.g., together with a trapdoor that allows it to extract the inputs of the corrupted parties), whereas ck is part of par and is sampled honestly by $\text{Com.Setup}(1^n)$, independently of crs and of $\mathcal{S}$.

Consider the following sequence of hybrids. Let ρ^{Hyb_j} be the output of $\mathcal{S}$:

- **Hyb₃**: This is the same experiment as $\text{Ideal}_{\mathcal{S}, \text{par}, T}^{\Pi}(\rho_0)$ where $\mathcal{S}$ has oracle access to $\mathcal{G}_{\text{com}}$ and $\text{par} = (\text{ck}, \text{com})$ with $\text{ck} \leftarrow \text{Com.Setup}(1^n)$, $\text{com} = (\text{com}_1, \dots, \text{com}_t)$ such that $\text{com}_i = \text{Com.Commit}(\text{ck}, k_i; \delta_i)$, $k = \oplus_{i \in [t]} k_i$. We have $\rho^{\text{Hyb}_3} = \text{Ideal}_{\mathcal{S}, \text{par}, T}^{\Pi}(\rho_0)$.
- **Hyb₄**: The only difference from the previous hybrid is that $\mathcal{S}$ gets single classical query access to $\mathcal{G}_{\text{com}, k}$ which is defined as follows:
 1. $\mathcal{G}_{\text{com}, k}$ is parameterized by $\{k_1, \dots, k_t\}$, ck and $\{\text{com}_1, \dots, \text{com}_t\}$, where $\text{ck} \leftarrow \text{Com.Setup}(1^n)$ and $\text{com}_i = \text{Com.Commit}(\text{ck}, k_i; \delta_i)$ for all $i \in [t]$.
 2. Upon receiving input (k'_i, δ'_i) from P_i , for $i \in [t-1]$, it outputs $\perp$ if $\text{Com.Open}(\text{ck}, \text{com}_i, k'_i, \delta'_i) = 0$.
 3. Otherwise it receives input v from P_{t+1} , and outputs $f_k(v)$ to P_{t+1} where $k = \oplus_{i \in [t]} k_i$.
- **Hyb₅**: The only difference from the previous hybrid is that $\mathcal{S}$ gets query access to $\mathcal{G}_{\text{com}, k-t}$ which is defined as follows:
 1. $\mathcal{G}_{\text{com}, k-t}$ is parameterized by $\{k, k_1, \dots, k_{t-1}\}$, ck and $\{\text{com}_1, \dots, \text{com}_t\}$ where $\text{ck} \leftarrow \text{Com.Setup}(1^n)$, $\text{com}_i = \text{Com.Commit}(\text{ck}, k_i; \delta_i)$ for all $i \in [t-1]$ and $\text{com}_t = \text{Com.Commit}(\text{ck}, 0; \delta_t)$.
 2. Upon receiving input (k'_i, δ'_i) from P_i , for $i \in [t-1]$, it outputs $\perp$ if $\text{Com.Open}(\text{ck}, \text{com}_i, k'_i, \delta'_i) = 0$.
 3. Otherwise it receives input v from P_{t+1} , and outputs $f_k(v)$ to P_{t+1} .

Lemma 5.15. *For all QPT distinguisher $\mathcal{D}$,*

$$\left| \Pr \left[\mathcal{D} \left(\rho^{\text{Hyb}_3} \right) = 1 \right] - \Pr \left[\mathcal{D} \left(\rho^{\text{Hyb}_4} \right) = 1 \right] \right| \leq \text{negl}(n)$$

Proof. The checks performed by $\mathcal{G}_{\text{com}}$ and $\mathcal{G}_{\text{com}, k}$ are identical and the only difference is that $\mathcal{G}_{\text{com}}$ outputs $f_{\oplus k'_i}(v)$ on valid inputs (k'_i, δ'_i) and v whereas $\mathcal{G}_{\text{com}, k}$ discards all k'_i and uses hardcoded $\{k_1, \dots, k_t\}$ and outputs $f_{\oplus k_i}(v)$. Let Bad_i be the event that $\mathcal{S}$ sends (k'_i, δ'_i) as input of party P_i to the functionality such that $k'_i \neq k_i$ and $\text{Com.Open}(\text{ck}, \text{com}_i, k'_i, \delta'_i) = 1$. Define $\text{Bad} = \cup_{i \in [t-1]} \text{Bad}_i$. Conditioned on $\neg \text{Bad}$, the output of functionality $\mathcal{G}_{\text{com}}$ in **Hyb₃** is identical to the output of functionality $\mathcal{G}_{\text{com}, k}$ in **Hyb₄** and the view of $\mathcal{S}$ is identical.

If Bad_i happens, for any $i \in [t-1]$, then (k_i, δ_i) and (k'_i, δ'_i) are two valid openings of com_i under ck to different messages $k_i \neq k'_i$. In both hybrids, $\text{ck} \leftarrow \text{Com.Setup}(1^n)$ is sampled honestly, independently of crs and of $\mathcal{S}$. By the statistical binding property of Com , with probability $1 - \text{negl}(n)$ over the choice of ck , no commitment under ck has valid openings to two different messages. Hence for all $i \in [t-1]$, $\Pr_{\text{Hyb}_3}[\text{Bad}_i] = \Pr_{\text{Hyb}_4}[\text{Bad}_i] \leq \text{negl}(n)$. Therefore, we have for all QPT $\mathcal{D}$,

$$\left| \Pr \left[\mathcal{D} \left(\rho^{\text{Hyb}_3} \right) = 1 \right] - \Pr \left[\mathcal{D} \left(\rho^{\text{Hyb}_4} \right) = 1 \right] \right| \leq \text{negl}(n)$$

□

Lemma 5.16. *For all QPT distinguisher $\mathcal{D}$,*

$$\left| \Pr \left[\mathcal{D} \left(\rho^{\text{Hyb}_4} \right) = 1 \right] - \Pr \left[\mathcal{D} \left(\rho^{\text{Hyb}_5} \right) = 1 \right] \right| \leq \text{negl}(n)$$

Proof. The only difference between Hyb_4 and Hyb_5 is that com_t is a commitment to 0 and independent of k_t . Suppose a QPT distinguisher $\mathcal{D}$ distinguishes ρ^{Hyb_4} and ρ^{Hyb_5} with non-negligible advantage. We construct a QPT adversary $\mathcal{B}$ against the computational hiding of Com . $\mathcal{B}$ receives $\text{ck} \leftarrow \text{Com.Setup}(1^n)$ from its challenger, samples uniform shares $k_1, \dots, k_t$ with $\oplus_{i \in [t]} k_i = k$, outputs the messages $(m_0, m_1) = (k_t, 0)$ and receives a challenge commitment com^* to m_b . It computes $\text{com}_i = \text{Com.Commit}(\text{ck}, k_i; \delta_i)$ for all $i \in [t-1]$, sets $\text{com}_t = \text{com}^*$, and runs $\mathcal{S}$ on $\rho_0, \{k_i, \delta_i\}_{i \in [t-1]}$ and $\text{par} = (\text{ck}, (\text{com}_1, \dots, \text{com}_t))$. It answers the single query of $\mathcal{S}$ to the functionality exactly as $\mathcal{G}_{\text{com},k}$ does, which only requires ck , $\{\text{com}_i\}_{i \in [t-1]}$ and k (the opening of com_t is never checked). Finally, $\mathcal{B}$ outputs the output of $\mathcal{D}$ on the output of $\mathcal{S}$. If $b = 0$, the view of $\mathcal{S}$ is distributed exactly as in Hyb_4 , and if $b = 1$, exactly as in Hyb_5 . Hence $\mathcal{B}$ breaks the computational hiding of Com with the same advantage as $\mathcal{D}$, a contradiction. $\square$

$\mathcal{S}$ receives ck , $\{k_i\}_{i \in [t-1]}$ and $\text{com} = \{\text{com}_i\}_{i \in [t]}$ distribution of which is independent of the key k since k_t is uniformly random and unknown to $\mathcal{S}$ in Hyb_5 . The only component in view of $\mathcal{S}$ that depends on k is the output of $\mathcal{G}_{\text{com},k-t}$ functionality. Since $\mathcal{S}$ receives at most one output from this functionality, we can construct a simulator Sim , with just oracle access to $f_{k,1}$, that runs $\mathcal{S}$ and provides access to the functionality $\mathcal{G}_{\text{com},k-t}$ such that the view of $\mathcal{S}$ is identical to its view in Hyb_5 . This is the simulator for adversary $\mathcal{A}$ of the one-time program. We describe simulator $\text{Sim}^{f_{k,1}}(\rho_0, T)$ that has query access to stateful one-time oracle $f_{k,1}(\cdot)$.

Description of simulator $\text{Sim}^{f_{k,1}}(\rho_0, T)$

1. Sim samples $\text{ck} \leftarrow \text{Com.Setup}(1^n)$ and $k_i \xleftarrow{\$} \mathcal{K}$ for all $i \in [t-1]$, and computes $\text{com}_i = \text{Com.Commit}(\text{ck}, k_i; \delta_i)$. It also computes $\text{com}_t = \text{Com.Commit}(\text{ck}, 0; \delta_t)$.
2. Sim sends $\rho_0, \{k_i, \delta_i\}_{i \in [t-1]}$ and $\text{par} = (\text{ck}, \text{com})$ with $\text{com} = \{\text{com}_i\}_{i \in [t]}$ to $\mathcal{S}$.
3. Upon input $\{k'_i, \delta'_i\}_{i \in [t-1]}$ and v from $\mathcal{S}$, Sim checks that $\text{Com.Open}(\text{ck}, \text{com}_i, k'_i, \delta'_i) = 1$ for all $i \in [t-1]$. If not, it returns $\perp$ to $\mathcal{S}$.
4. If the checks pass, Sim queries $f_{k,1}$ on v and returns the output to $\mathcal{S}$.
5. Sim outputs whatever $\mathcal{S}$ outputs.

Sim is QPT and makes at most 1 query to $f_{k,1}$. Moreover, Sim perfectly simulates functionality $\mathcal{G}_{\text{com},k-t}$ from Hyb_5 and the view of $\mathcal{S}$ is identical to its view in Hyb_5 . Therefore, we have for all QPT $\mathcal{D}$,

$$\Pr[\mathcal{D}(\text{Ideal}_{\text{Sim},k,T}(\rho_0)) = 1] = \Pr[\mathcal{D}(\rho^{\text{Hyb}_5}) = 1]$$

This completes the proof of Theorem 5.11. $\square$

6 AI Disclosure

Claude Opus 5.5 and ChatGPT 6.1 Astra and Sol Pro were used only for editorial assistance, including finding references, polishing the exposition, and generating figures and tables. They also assisted with checking mathematical arguments and tightening the numerical calculations in Appendix A. The authors are responsible for the manuscript’s scientific claims and conclusions.

7 Acknowledgements

We are grateful to Kaushik Chakraborty and Fatih Kaleoglu for several helpful discussions and valuable feedback.

Prabhanjan Ananth and Divyanshu Bhardwaj are supported by the National Science Foundation under grants FET-2329938, CAREER-2341004, and FET-2530160.

References

- [AB26] Prabhanjan Ananth and Divyanshu Bhardwaj. *Quantum One Time Programs: Less Assumptions, More Feasibility and One Message 2PC*. Cryptology ePrint Archive, Paper 2026/300. 2026. URL: <https://eprint.iacr.org/2026/300> (cit. on p. 3).
- [BGS13] Anne Broadbent, Gus Gutoski, and Douglas Stebila. “Quantum One-Time Programs”. In: *Advances in Cryptology – CRYPTO 2013*. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 344–360. DOI: 10.1007/978-3-642-40084-1_20. URL: <https://arxiv.org/abs/1211.1080> (cit. on pp. 3, 4).
- [BGZ21] Anne Broadbent, Sevag Gharibian, and Hong-Sheng Zhou. “Towards Quantum One-Time Memories from Stateless Hardware”. In: *Quantum* 5 (2021), p. 429. DOI: 10.22331/q-2021-04-08-429. URL: <https://quantum-journal.org/papers/q-2021-04-08-429/> (cit. on p. 3).
- [BJKS21] Robert Buhren, Hans Niklas Jacob, Thilo Krachenfels, and Jean-Pierre Seifert. “One Glitch to Rule Them All: Fault Injection Attacks Against AMD’s Secure Encrypted Virtualization”. In: *ACM Conference on Computer and Communications Security (CCS)*. 2021 (cit. on p. 10).
- [BKS+22] Pietro Borrello, Andreas Kogler, Martin Schwarzl, Moritz Lipp, Daniel Gruss, and Michael Schwarz. “ÆPIC Leak: Architecturally Leaking Uninitialized Data from the Microarchitecture”. In: *USENIX Security Symposium*. 2022 (cit. on p. 10).
- [BMS+20] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, and Frank Piessens. “LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection”. In: *IEEE Symposium on Security and Privacy (S&P)*. 2020 (cit. on p. 10).
- [BMW+18] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. “Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution”. In: *USENIX Security Symposium*. 2018 (cit. on p. 10).

- [BS23] Shalev Ben-David and Or Sattath. “Quantum Tokens for Digital Signatures”. In: *Quantum* 7 (2023), p. 901. DOI: 10.22331/q-2023-01-19-901. URL: <https://doi.org/10.22331/q-2023-01-19-901> (cit. on p. 3).
- [BSS21] Amit Behera, Or Sattath, and Uriel Shinar. *Noise-Tolerant Quantum Tokens for MAC*. Cryptology ePrint Archive, Paper 2021/1353. 2021. URL: <https://eprint.iacr.org/2021/1353> (cit. on pp. 3–9, 14, 17, 20, 22, 50, 52, 53).
- [CCX+19] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H. Lai. “SgxPectre: Stealing Intel Secrets from SGX Enclaves via Speculative Execution”. In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. 2019 (cit. on p. 10).
- [CGLZ19] Kai-Min Chung, Marios Georgiou, Ching-Yi Lai, and Vassilis Zikas. “Cryptography with Disposable Backdoors”. In: *Cryptography* 3.3 (2019), p. 22. DOI: 10.3390/cryptography3030022. URL: <https://doi.org/10.3390/cryptography3030022> (cit. on p. 3).
- [CSFP20] David Cerdeira, Nuno Santos, Pedro Fonseca, and Sandro Pinto. “SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems”. In: *IEEE Symposium on Security and Privacy (S&P)*. 2020 (cit. on p. 10).
- [DKMN16] Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade, and Tobias Nilges. *From Stateful Hardware to Resettable Hardware Using Symmetric Assumptions*. Cryptology ePrint Archive, Paper 2016/201. 2016. URL: <https://eprint.iacr.org/2016/201> (cit. on p. 3).
- [DOVV26] Jesse De Meulemeester, David Oswald, Ingrid Verbauwhede, and Jo Van Bulck. “Battering RAM: Low-Cost Interposer Attacks on Confidential Computing via Dynamic Memory Aliasing”. In: *2026 IEEE Symposium on Security and Privacy (SP)*. 2026, pp. 4391–4407. DOI: 10.1109/SP63933.2026.00052 (cit. on p. 10).
- [GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. “One-Time Programs”. In: *Proceedings of the 28th Annual Conference on Cryptology: Advances in Cryptology*. CRYPTO 2008. Santa Barbara, CA, USA: Springer-Verlag, 2008, pp. 39–56. ISBN: 9783540851738. DOI: 10.1007/978-3-540-85174-5_3. URL: https://doi.org/10.1007/978-3-540-85174-5_3 (cit. on p. 3).
- [GLR+25] Aparna Gupte, Jiahui Liu, Justin Raizes, Bhaskar Roberts, and Vinod Vaikuntanathan. “Quantum One-Time Programs, Revisited”. In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*. STOC 2025. Association for Computing Machinery, 2025, pp. 213–221. DOI: 10.1145/3717823.3718204. URL: <https://doi.org/10.1145/3717823.3718204> (cit. on p. 3).
- [GS22] Sanjam Garg and Akshayaram Srinivasan. “Two-round Multiparty Secure Computation from Minimal Assumptions”. In: *J. ACM* 69.5 (Oct. 2022). ISSN: 0004-5411. DOI: 10.1145/3566048. URL: <https://doi.org/10.1145/3566048> (cit. on p. 37).
- [Liu23] Qipeng Liu. “Depth-Bounded Quantum Cryptography with Applications to One-Time Memory and More”. In: *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*. Vol. 251. Leibniz International Proceedings in Informatics (LIPIcs). 2023, 82:1–82:18. DOI: 10.4230/LIPIcs.ITCS.2023.82 (cit. on p. 3).

- [LZW+21] Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. “Cipher-Leaks: Breaking Constant-time Cryptography on AMD SEV via the Ciphertext Side Channel”. In: *USENIX Security Symposium*. 2021 (cit. on p. 10).
- [MOG+20] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. “Plundervolt: Software-based Fault Injection Attacks against Intel SGX”. In: *IEEE Symposium on Security and Privacy (S&P)*. 2020 (cit. on p. 10).
- [NBB20] Alexander Nilsson, Pegah Nikbakht Bideh, and Joakim Brorsson. *A Survey of Published Attacks on Intel SGX*. 2020. arXiv: 2006.13598 (cit. on p. 10).
- [PLOB17] Stefano Pirandola, Riccardo Laurenza, Carlo Ottaviani, and Leonardo Banchi. “Fundamental limits of repeaterless quantum communications”. In: *Nature Communications* 8.1 (Apr. 2017). ISSN: 2041-1723. DOI: 10.1038/ncomms15043. URL: <http://dx.doi.org/10.1038/ncomms15043> (cit. on p. 7).
- [PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. “A Framework for Efficient and Composable Oblivious Transfer”. In: *Proceedings of the 28th Annual Conference on Cryptology: Advances in Cryptology*. CRYPTO 2008. Santa Barbara, CA, USA: Springer-Verlag, 2008, pp. 554–571. ISBN: 9783540851738. DOI: 10.1007/978-3-540-85174-5_31. URL: https://doi.org/10.1007/978-3-540-85174-5_31 (cit. on p. 37).
- [RKB+18] Marie-Christine Roehsner, Joshua A. Kettlewell, Tiago B. Batalhão, Joseph F. Fitzsimons, and Philip Walther. “Quantum advantage for probabilistic one-time programs”. In: *Nature Communications* 9 (2018), p. 5225. DOI: 10.1038/s41467-018-07591-2 (cit. on p. 3).
- [Shm22] Omri Shmueli. “Semi-quantum Tokenized Signatures”. In: *Advances in Cryptology – CRYPTO 2022*. 2022 (cit. on p. 3).
- [SKGY20] Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom. *SGAxe: How SGX Fails in Practice*. <https://sgaxe.com/>. 2020 (cit. on p. 10).
- [Sta25] Lev Stambler. *Cryptography without Long-Term Quantum Memory and Global Entanglement: Classical Setups for One-Time Programs, Copy Protection, and Stateful Obfuscation*. 2025. arXiv: 2504.21842 [quant-ph]. URL: <https://arxiv.org/abs/2504.21842> (cit. on p. 3).
- [XMZ+20] Feihu Xu, Xiongfeng Ma, Qiang Zhang, Hoi-Kwong Lo, and Jian-Wei Pan. “Secure quantum key distribution with realistic devices”. In: *Rev. Mod. Phys.* 92 (2 May 2020), p. 025002. DOI: 10.1103/RevModPhys.92.025002. URL: <https://link.aps.org/doi/10.1103/RevModPhys.92.025002> (cit. on p. 7).
- [Zha21] Mark Zhandry. “How to Construct Quantum Random Functions”. In: *J. ACM* 68.5 (Aug. 2021). ISSN: 0004-5411. DOI: 10.1145/3450745. URL: <https://doi.org/10.1145/3450745> (cit. on pp. 11, 31).

A Optimized Noise-Tolerant Scheme

We refine the concrete single-token unforgeability analysis of the noise-tolerant conjugate tokenized MAC of Behera, Sattath, and Shinar [BSS21]. The changes relative to their proof are marked in red.

Set

$$\alpha = \frac{2 + \sqrt{2}}{4} = \cos^2(\pi/8), \quad \delta = 1 - \alpha,$$

and, for $a \in [0, 1]$, $b \in (0, 1)$, recall the definition of binary relative entropy:

$$D(a\|b) = a \ln \frac{a}{b} + (1 - a) \ln \frac{1 - a}{1 - b}, \quad 0 \ln 0 = 0.$$

Equivalently,

$$e^{D(a\|b)} = \left(\frac{a}{b}\right)^a \left(\frac{1 - a}{1 - b}\right)^{1 - a}.$$

Logarithms are natural, and the factors at $a = 0, 1$ are interpreted by continuity.

Construction. Fix a token length $L \geq 1$ and a verification threshold $0 \leq \eta_y < 1/2$. Write $[L] = \{1, \dots, L\}$, let H be the Hadamard gate, and define

$$C_b(K) = \{i \in [L] : \theta_i = b\}, \quad E_b(K, \sigma) = \{i \in C_b(K) : \sigma_i \neq x_i\}, \quad t = \lfloor \eta_y L \rfloor$$

for $K = (x, \theta) \in \{0, 1\}^{2L}$, $b \in \{0, 1\}$, and $\sigma \in \{0, 1\}^L$. The parameter t is the maximum number of checked mismatches allowed in each signature.

Key generation. The algorithm **KeyGen**(1^L) does the following:

- Sample $x, \theta \leftarrow \{0, 1\}^L$ independently and uniformly.
- Output the classical secret key $K = (x, \theta)$.

Token generation. The algorithm **TokenGen**(K) does the following:

- For each $i \in [L]$, prepare one qubit $H^{\theta_i}|x_i\rangle$.
- Output the L -qubit token $|x^\theta\rangle = \bigotimes_{i=1}^L H^{\theta_i}|x_i\rangle$.

Signing. The algorithm **Sign**(b, ρ), on a bit $b \in \{0, 1\}$ and an L -qubit token register ρ , does the following:

- Apply H^b to each qubit and measure every qubit in the computational basis. Denote the measurement outcomes by $\sigma_1, \dots, \sigma_L$.
- Output the classical signature $\sigma = (\sigma_1, \dots, \sigma_L)$.

Verification. The algorithm **Verify** $_{\eta_y}(K, b, \sigma)$ does the following:

- Compute $E_b(K, \sigma)$ and output $\mathbf{1}[|E_b(K, \sigma)| \leq t]$.

The threshold is relative to the full length L , rather than $|C_b(K)|$.

Single-token unforgeability experiment. Let $q \in \mathbb{N}_0$, let $\mathcal{A}$ be a quantum adversary, and let ρ_{aux} be its initial auxiliary state, independent of the key subsequently sampled by the challenger. Define $\text{Forge}_{L,\eta_y,q}^{\mathcal{A}}(\rho_{\text{aux}})$ as follows:

1. The challenger samples $K = (x, \theta) \leftarrow \text{KeyGen}(1^L)$.
2. The challenger prepares $|x^\theta\rangle \leftarrow \text{TokenGen}(K)$ and sends exactly one copy of this token to $\mathcal{A}$. The adversary is initialized with ρ_{aux} and the public parameters (L, η_y, q) .
3. The adversary makes at most q adaptive classical verification queries. For a query (b, σ) , the challenger returns $\text{Verify}_{\eta_y}(K, b, \sigma)$.
4. The adversary outputs two classical strings (σ_0, σ_1) .
5. The experiment outputs 1 if $\sigma_0, \sigma_1 \in \{0, 1\}^L$ and

$$\text{Verify}_{\eta_y}(K, 0, \sigma_0) = \text{Verify}_{\eta_y}(K, 1, \sigma_1) = 1;$$

otherwise, it outputs 0. The experiment also outputs 0 if the adversary aborts or exceeds its query budget. The two final checks are performed by the challenger and are not counted among the q queries.

Theorem A.1 (Refined single-token unforgeability). *Let $L \geq 1$, $q \in \mathbb{N}_0$, and $0 \leq 2\eta_y < \delta$. For every quantum adversary $\mathcal{A}$ making at most q classical verification queries in the above experiment, and every initial auxiliary state ρ_{aux} independent of the sampled key, put*

$$p = \Pr[\text{Forge}_{L,\eta_y,q}^{\mathcal{A}}(\rho_{\text{aux}}) = 1].$$

Define

$$a_L = \frac{2\lfloor \eta_y L \rfloor}{L}, \quad C_q = 2 \binom{q+4}{2}.$$

Then

$$p \leq \min\{1, C_q e^{-LD(a_L \|\delta)}\} \leq \min\{1, C_q e^{-LD(2\eta_y \|\delta)}\}. \quad (4)$$

Proof. Fix $\mathcal{A}$ and ρ_{aux} as in the theorem. We construct, for each $m \in \{1, \dots, L\}$, a noiseless m -qubit forger with success probability s_m satisfying

$$p\Delta_m \leq s_m \leq C_q \alpha^m, \quad \Delta_m = \frac{\binom{L-2\lfloor \eta_y L \rfloor}{m}}{\binom{L}{m}}.$$

Step I establishes the lower bound, Step II applies noiseless unforgeability, and Step III combines these inequalities to obtain (4).

Step I: Reduction from noise-tolerant to noiseless forgery. *Noiseless verify-and-reveal forging experiment.* For $m, Q \in \mathbb{N}_0$, a quantum adversary $\mathcal{B}$, and an initial auxiliary state ρ_{aux} independent of the challenge key, define $\text{Forge}_{m,0,Q}^{\text{rev},\mathcal{B}}(\rho_{\text{aux}})$ as follows:

1. The challenger independently samples $x^*, \theta^* \leftarrow \{0,1\}^m$ uniformly, sets $K^* = (x^*, \theta^*)$, and gives $\mathcal{B}$ exactly one copy of $\bigotimes_{i=1}^m H^{\theta_i^*} |x_i^*\rangle$. The adversary is initialized with ρ_{aux} and public parameters (m, Q) .
2. The adversary makes at most Q adaptive classical queries to V_{0,K^*}^{rev} . On a query $(b, s) \in \{0,1\} \times \{0,1\}^m$, the oracle returns 0 if $E_b(K^*, s) \neq \emptyset$, and $(1, C_b(K^*), \emptyset)$ otherwise. Malformed queries are rejected. The checked sets are defined as above at length m .
3. The adversary outputs two strings (s_0, s_1) . The experiment outputs 1 exactly when $s_0, s_1 \in \{0,1\}^m$ and $E_0(K^*, s_0) = E_1(K^*, s_1) = \emptyset$, and 0 otherwise. An abort or exceeding the query budget also gives 0. The two final checks are not counted among the Q queries.

Claim A.2 (BSS21 sampling reduction). *Fix $\mathcal{A}$, ρ_{aux} , and p as in Theorem A.1, and put $t = \lfloor \eta_y L \rfloor$. For every $m \in \{0, \dots, L\}$, the reduction of [BSS21, Proposition 3 and proof of Lemma 1] supplies an adversary $\mathcal{R}_m$ satisfying*

$$\begin{aligned} s_m &:= \Pr[\text{Forge}_{m,0,q+2}^{\text{rev},\mathcal{R}_m}(\rho_{\text{aux}}) = 1], \\ s_m &\geq p\Delta_m, \quad \Delta_m = \frac{\binom{L-2t}{m}}{\binom{L}{m}}. \end{aligned} \tag{5}$$

For $m = 0$, choose two empty signatures, giving $s_0 = \Delta_0 = 1$; set $\binom{L-2t}{m} = 0$ when $m > L - 2t$.

Step II: Noiseless unforgeability. We use the noiseless verify-and-reveal query bound of [BSS21, Sections 5.1–5.2, Propositions 3–6] as a black box, stated below in our notation. Proposition 6 is phrased asymptotically; the query-index calculation in the proof of Proposition 5 gives the following concrete estimate for each fixed m, Q . Proposition 3 handles repeated successful queries without increasing the query budget. The $Q + 2$ includes the two final-candidate checks, and a key-independent auxiliary state can be included in the adversary's initialization.

Claim A.3 (Noiseless query bound (BSS21)). *For every $m \geq 1$, $Q \in \mathbb{N}_0$, quantum adversary $\mathcal{B}$ making at most Q classical verify-and-reveal queries, and initial auxiliary state ρ_{aux} independent of the sampled challenge key,*

$$\Pr[\text{Forge}_{m,0,Q}^{\text{rev},\mathcal{B}}(\rho_{\text{aux}}) = 1] \leq 2 \binom{Q+2}{2} \alpha^m. \tag{6}$$

The adversary is computationally unbounded and may perform arbitrary joint quantum operations on the token and its auxiliary registers.

Application to the sampling reduction. Apply Claim A.3 to the adversary $\mathcal{R}_m$ supplied by Claim A.2, with $Q = q + 2$. Its initial auxiliary state is independent of the challenge key, as required. This gives

$$s_m \leq 2 \binom{q+4}{2} \alpha^m = C_q \alpha^m \quad (m \geq 1). \tag{7}$$

The same inequality holds for $m = 0$, since $s_0 = 1 \leq C_q$. Combining (5) and (7),

$$\boxed{p\Delta_m \leq s_m \leq C_q \alpha^m \text{ for every } m = 0, \dots, L.} \quad (8)$$

Thus Δ_m is BSS21's exact sampling loss, and $C_q = 2^{\binom{q+4}{2}}$ is the noiseless query factor from Claim A.3 with $Q = q + 2$.

Step III: Exact averaging and optimization. Set $k = 2\lfloor \eta_y L \rfloor$ and $a = k/L$, so $0 \leq a \leq 2\eta_y < \delta$. For $0 \leq u < 1$, multiply (8) by $w_m = \binom{L}{m} u^m (1-u)^{L-m}$ and sum over m . The forging probability p is the same in every inequality, and the binomial theorem gives

$$\begin{aligned} \sum_{m=0}^L w_m \Delta_m &= \sum_{m=0}^{L-k} \binom{L-k}{m} u^m (1-u)^{L-m} = (1-u)^k, \\ \sum_{m=0}^L w_m \alpha^m &= (1-u + u\alpha)^L = (1-\delta u)^L. \end{aligned}$$

Therefore

$$p \leq C_q \frac{(1-\delta u)^L}{(1-u)^k}. \quad (9)$$

For $0 < a < \delta$, choose $u = (\delta - a)/(\delta(1-a)) \in (0, 1)$. Substitution gives

$$\frac{(1-\delta u)^L}{(1-u)^k} = \left[\left(\frac{\delta}{a} \right)^a \left(\frac{1-\delta}{1-a} \right)^{1-a} \right]^L = e^{-LD(a\|\delta)}.$$

For $a = 0$, taking $u \uparrow 1$ gives $\alpha^L = e^{-LD(0\|\delta)}$. Hence

$$p \leq C_q e^{-LD(k/L\|\delta)}. \quad (10)$$

Since $D(a\|\delta)$ is decreasing on $[0, \delta]$, with derivative $\ln \frac{a(1-\delta)}{\delta(1-a)} < 0$ for $0 < a < \delta$, the inequality $k/L \leq 2\eta_y$ yields

$$\boxed{p \leq C_q e^{-LD(2\eta_y\|\delta)}}. \quad (11)$$

Together with $p \leq 1$, (10) and (11) prove (4).

Comparison with BSS21. The exact sampling factor Δ_m already appears in [BSS21, proof of Lemma 1]. Their final estimate lower-bounds it by $(1 - 2\eta_y L / (L - m))^m$ and chooses one prescribed admissible length [BSS21, Proposition 7]. Here we retain the exact factors for all lengths and average the resulting inequalities to obtain the explicit KL bound, refining the concrete analysis of the same reduction. $\square$