QUASI: Post-Quantum Quorum Attestation of Pre-Validated Statements

Maksym Ioffe
University of California, Santa Barbara
maksym@ucsb.edu

Dahlia Malkhi
University of California, Santa Barbara
dahliamalkhi@gmail.com

***Abstract*—We present QUASI, a newly conceived class of hash-based, post-quantum-secure protocols that enable a group of parties to produce a publicly verifiable attestation about a pre-validated statement. QUASI produces attestations that reduce the size and verification cost to a fraction of a full hash-based signatures collection, without relying on SNARKs or a trusted setup.**

QUASI rests on a surprising discovery: each party need not sign the entire statement. Authenticating only a small subset of symbols suffices for a quorum to attest to the whole. We turn this insight into a randomized construction that achieves reductions of more than 10x in per-party contribution size and hash computations for generation and verification, compared with full hash-based signatures.

QUASI is a generic primitive supporting diverse applications. Operationally, QUASI requires no additional interactive setup or preprocessing beyond independent key generation by parties for the underlying hash-based authentication scheme. Two principal use cases are external attestation for signature-free consensus, enabling clients to verify statements about committed state; and preprocessing for SNARK-based aggregation, compounding QUASI's savings with succinct-proof compression.

1. Introduction

The need for aggregating post-quantum (PQ) signatures has been articulated in several reports [12], [1]. NIST's recent Threshold Call identifies aggregation into sublinear-size results, quantum resistance, and ZKP-friendliness as features of interest [14, Sections 3.2 and 11.1]. Hash-based signatures, such as Lamport-OTS [33] and WOTS [42], are central to post-quantum migration [2]. Such primitives also underpin hash-based signature management schemes such as XMSS [29] and SLH-DSA [44], which play a prominent role in PQ migration. However, hash-based signature schemes lack BLS-style native aggregation: the prevailing approach uses SNARK-based aggregation to compress full signatures contributed by every party [22]. Thus, aggregating 100 hash-based signatures today means either concatenating them ($100\times$ the size and $100\times$ the verification work) or proving all 100 verifications inside a hash-based SNARK, as in Ethereum's plan [24]. Recent constructions extend hash-based signatures to threshold signing, but require trusted-dealer preprocessing and share reconstruction [31], [10].

We present QUASI, a newly conceived class of hash-based attestation protocols. QUASI rests on a surprising discovery: a group of parties can produce a compact, publicly verifiable, PQ-secure attestation without each party contributing a full signature. In particular, each party need not sign the entire statement. The QUASI family is based purely on hashing and incorporates error correction. Its protocols have small per-party contributions compared with full hash-based signatures, have low quorum-attestation generation and verification costs, and do not rely on SNARKs for aggregation.

Core insight and terminology. QUASI is not a signature aggregation scheme, nor does it provide the guarantees of threshold signatures. It addresses a weaker task: honest parties in a quorum-based system have already validated and agreed on a single statement per context, and the goal is to attest to external parties that this statement is valid.

This distinction enables a previously unexplored approach: each party authenticates only a small subset of the symbols that make up the statement, rather than providing a full signature. This is possible because hash-based authentication treats individual input symbols independently. We call a party's contribution a *quasi-signature share* (qShare); a quorum of qShares produces a publicly verifiable *joint quorum attestation* (qAttest). Together, these contributions suffice to verify the entire pre-validated statement validity. The number of symbols that a qShare needs to authenticate decreases as the number of parties increases, reaching as few as one symbols per party.

Use cases. QUASI is a generic primitive supporting diverse applications. The need to make protocol outcomes verifiable by external parties arises across a broad range of quorum-based systems: low-latency payment and transaction certification, as in FastPay, Sui, and Stingray [5], [8], [50]; reliable and atomic broadcast, including Bracha's protocol, the protocols of Cachin et al., and Rampart [13], [16], [47]; and Byzantine consensus, from the foundational Byzantine Generals Problem of Lamport, Shostak, and Pease [34], through the seminal PBFT protocol [17], to modern protocols such as HotStuff [52]. QUASI offers an external attestation layer for these settings when honest parties have pre-validated the statement being attested.

Of particular note, QUASI may provide external attes-

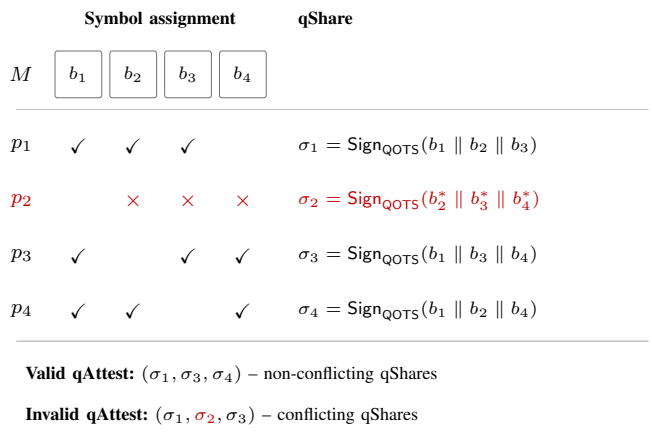

Figure 1. QOTS symbol assignments and qShares for $M = b_1b_2b_3b_4$, with examples of valid and invalid quorum attestations. Red crosses indicate altered symbols at corrupt party p_2.

tation for signature-free Byzantine consensus protocols and blockchains. These protocols, also known as information-theoretic (IT) consensus protocols, achieve agreement using authenticated point-to-point channels. Examples include IT-HotStuff, Forget-IT, TetraBFT, Simple-IT, Quintus, Quantum Cadence [4], [3], [53], [54], [36], [30], and others. Although these protocols provide post-quantum-secure agreement, their internal authentication does not produce publicly verifiable attestations about committed state. QUASI fills this gap by providing an external attestation layer, enabling clients to verify statements about committed state.

Another use case compounds the QUASI advantage with SNARK-based compression. Advances in SNARK proof systems have improved both general-purpose proving and specialized workloads and brought them into reality: PLONK provides succinct verification with a universal, updatable setup [26], while HyperPlonk enables linear-time proving with high-degree custom gates [18]. More recently, Flock accelerates batched Boolean computations, including standard cryptographic hashes, hash chains, and Merkle-path verification [15]. Taking advantage of these developments, Succinct's Telepathy service is a prominent deployed example of SNARK proofs to compress the attestation originating with Ethereum's 512-member "sync committee" [51]. A QUASI-based alternative would improve the attestation layer feeding this pipeline; succinct proving could remain a subsequent compression step.

Technical approach. The key idea is to distribute authentication across parties: each party authenticates a small subset of symbols, and a quorum of their contributions jointly attests to the pre-validated statement. Honest parties 'pin' the values of symbols they authenticate, but the challenge is to prevent corrupt parties from changing any symbols left unauthenticated by honest parties.

To demonstrate this is possible, a simple deterministic construction (depicted in Figure 1) addresses this challenge and provides modest savings: each symbol of the statement M is authenticated by $2f + 1$ parties. Rotating these assignments balances the load, so that each party authenticates approximately two-thirds of M. A qShare by a party consists of a hash-based signature $\mathsf{Sign}_{\mathsf{QOTS}}$ over the symbols assigned to it; any standard hash-based signatures may be employed (Section 3 elaborates on one choice). A qAttest contains $2f + 1$ valid, mutually non-conflicting qShares, ensuring that every symbol is authenticated by at least one honest contributor. While this construction already reduces the per-party contribution by one-third (asymptotically), we are confident that better deterministic parameters are achievable and are currently exploring improved designs.

Our main focus in this paper, however, is on larger systems, where randomization enables more dramatic savings. Here, QUASI authenticates a codeword rather than the statement itself. Provided the number of altered symbols remains within the code's correction capacity, the symbols authenticated by honest parties suffice to detect if the original pre-validated statement has been altered. This creates a tradeoff between qShare size and codeword expansion. Figure 2 illustrates the randomized approach with qShares of two symbols, and a codeword with $1/3$ correcting resilience.

Performance. Compared with a naïve collection of hash-based signatures, QUASI targets savings of more than $10\times$ in (i) per-party contribution size, (ii) generation effort, and (iii) verification effort. These gains arise from reducing the number of hash-chain values and preimages maintained or revealed by each party, as well as the hash computations required for generation and verification. For a 256-bit statement digest M, QUASI represents the input as $m = 32$ symbols over $\mathrm{GF}(2^8)$ and employs an error-correcting code with $3\times$ expansion. It sets the number of symbols authenticated per qShare, denoted d, to ensure that undetected symbol flipping occurs with probability less than 2^{-128}. Table 2 shows a selection of concrete parameters and the resulting savings.

The overall goal is to present the defense against the strongest adversary with the strongest security parameters. However, alongside we present a common case, which may have proportionally fewer amount of Byzantine failures. Hence, we consider possible pragmatic variations which include proportionally reducing number of Byzantine failures by defining $f = b + c$ (b are Byzantine and c crash failures).

Operations. QUASI requires no additional interactive setup or preprocessing beyond key generation for the underlying hash-based authentication scheme. Parties generate keys independently for d symbols rather than m, and publish their selected symbol positions alongside their verification keys. Each party then generates its qShare using $\mathsf{Sign}_{\mathsf{QOTS}}$. Anyone can collect $2f + 1$ valid, mutually non-conflicting qShares on M to form a qAttest.

Across multiple QUASI instances, parties can manage keys using established techniques from stateful hash-based signature schemes, such as XMSS [29], which manages one-time WOTS-family keys. Key management is orthogonal to the QUASI construction and outside the scope of this work.

TABLE 1. GLOSSARY OF NOTATION.

Term	Notation	Meaning
Statement	M, m	Input to be certified, containing m symbols
Encoded statement	$c(M)$, $\hat{m}$	ECC with expansion $\lambda = \hat{m}/m$
Symbol space	$\mathcal{A}$	Alphabet of 8-bit values
Symbol	b_j, c_j, $M\|_j$, $c(M)\|_j$	The j-th symbol of M or $c(M)$
QUASI-OTS	$\mathsf{Sign}_{\mathsf{QOTS}}(w)$	One-time hash-based signature over a word w
qShare map	$\pi_i : [d] \to [\hat{m}]$, d	Party i's symbol assignment
qShare	$\sigma_i(M)$	Party i's quasi signature-share over M
qAttest	$\sigma(M)$	Full attestation over M

2. Preliminaries

This work introduces a newly conceived class of attestation protocols named QUASI. We first give an informal description of QUASI, and follow up more formally in Definition 1. We also present a glossary of notations and terms used throughout the construction in Table 1.

In QUASI, a group of $n = 3f + 1$ parties, up to f of which may be corrupt, can produce an attestation about a pre-validated statement on behalf of quorums of $\geq 2f + 1$ parties, and no such quorum can forge an attestation, which the honest parties in the quorum did not intend to provide, except with negligible probability[1].

Parties in the QUASI primitive each hold an independently generated private key for signing, similar to multi-signature schemes [43]; the public keys of all parties are known to everyone. Each party contributes a quasi-signature share (qShare) on a statement, which does not constitute by itself a valid signature on the statement, akin to classical threshold signature schemes [21]. A quorum of qShares produces a joint quorum attestation (qAttest), which is not required to look like a single signature. Furthermore, there are no secrecy guarantees, and a qAttest may reveal and hold accountable who contributed to an attestation, similar to aggregate signature paradigms [12].

However, while being inspired by signature aggregation schemes, QUASI is not one. It is designed for quorum-based systems to attest to a pre-validated statement, hence it provides a weaker unforgeability guarantee, called *weak-f-unforgeability*, defined below. The goal is to attest to external parties that the statement is valid.

More formally, a QUASI scheme is defined as follows.

Definition 1 (**QUASI**). A *QUASI* scheme is a tuple of probabilistic polynomial-time algorithms $\Pi = (\mathsf{qSetup}, \mathsf{qKeyGen}, \mathsf{qShareSign}, \mathsf{qCombine}, \mathsf{qVerify})$ involving n parties $p_1, \ldots, p_n$ and a threshold f with $1 \leq f \leq n$, defined as follows.

- $\mathsf{qSetup}(1^\kappa, n, f) \to \mathsf{pp}$. On input a security parameter κ, and environment parameters n and f, outputs public parameters pp shared by all parties, including: a description of the statement space $\mathcal{M}$; an error-correcting code $c(\cdot)$ over $\mathcal{M}$ which includes an expansion parameter λ; the quasi signature share (qShare) space Σ_{sh} which includes a qShare size parameter d; and the quorum attestation (qAttest) space Σ.
- $\mathsf{qKeyGen}(\mathsf{pp}) \to (\mathsf{vk}_i, \mathsf{sk}_i)$. A randomized algorithm run *independently* by each party p_i; outputs a verification key vk_i and a signing key sk_i. The combined verification key is $\mathsf{VK} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$.
- $\mathsf{qShareSign}(\mathsf{pp}, \mathsf{sk}_i, M) \to \sigma_i$. On input the public parameters, party p_i's signing key, and a statement $M \in \mathcal{M}$, outputs a *quasi signature share* (qShare) $\sigma_i \in \Sigma_{\mathsf{sh}}$. A qShare does not constitute a valid signature over M on its own.
- $q\mathsf{Combine}(\mathsf{pp}, \mathsf{VK}, S, M, \{\sigma_i\}_{i \in S}) \to \sigma$ or $\perp$. On input the public parameters, the combined verification key, an index set $S \subseteq [n]$ with $|S| \geq 2f+1$, a statement M, and the qShares $\{\sigma_i\}_{i \in S}$, outputs either a qAttest $\sigma \in \Sigma$ or a failure symbol $\perp$.
- $\mathsf{qVerify}(\mathsf{pp}, \mathsf{VK}, M, \sigma) \to \{0, 1\}$. On input the public parameters, the combined verification key, a statement M, and a purported qAttest σ, outputs 1 (accept) or 0 (reject).

We require the following properties.

Correctness. For all $\mathsf{pp} \leftarrow \mathsf{qSetup}(1^\kappa, n, f)$, all $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{qKeyGen}(\mathsf{pp})$, all statements $M \in \mathcal{M}$, all index sets $S \subseteq [n]$ with $|S| \geq 2f + 1$, and all shares $\sigma_i \leftarrow \mathsf{qShareSign}(\mathsf{pp}, \mathsf{sk}_i, M)$ for $i \in S$:

$$\Pr\left[\begin{array}{l} \sigma \leftarrow \mathsf{qCombine}(\mathsf{pp}, \mathsf{VK}, S, M, \{\sigma_i\}_{i \in S}) : \\ \mathsf{qVerify}(\mathsf{pp}, \mathsf{VK}, M, \sigma) = 1 \end{array}\right] = 1$$

Weak-f-Unforgeability. For all $\mathsf{pp} \leftarrow \mathsf{qSetup}(1^\kappa, n, f)$, all $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{qKeyGen}(\mathsf{pp})$, all probabilistic polynomial-time adversaries $\mathcal{A}$ controlling a coalition $\mathcal{C} \subset [n]$ of corrupted parties with $|\mathcal{C}| \leq f$, having access to a signing oracle O^M which is allowed to make $qShareSign(\mathsf{pp}, \mathsf{sk}_i, M)$ requests only on a pre-validated M:

$$\Pr\Big[\mathsf{qVerify}(\mathsf{pp}, \mathsf{VK}, M^*, \sigma^*) = 1 \;\Big|\; M^* \neq M, \\ \sigma^* \leftarrow \mathcal{A}^{\mathsf{O}^M}\big(\mathsf{pp}, \mathsf{VK}, \{\mathsf{sk}_i\}_{i \in \mathcal{C}}\big)\Big] \leq \mathsf{negl}(\kappa).$$

Transparency. The quorum attestation σ output by $\mathsf{qCombine}$ may encode the signing set S and is not required to be indistinguishable from a single-signer signature. In particular, no anonymity guarantee is imposed on σ with respect to the identities of the signers in S.

Remarks. It is worthy of noting how Weak-f-Unforgeability differs from traditional threshold-unforgeability: in the latter, the adversary has access to a signing oracle $O^{\mathcal{Q}}$ which is allowed to make $qShareSign(\mathsf{pp}, \mathsf{sk}_i, M')$ requests on any $M' \in \mathcal{Q}$. That is, it allows the adversary to query honest parties on a set $\mathcal{Q}$ of statements, rather than a unique statement M. It then requires negligible forging success probability for all statements $M^* \notin \mathcal{Q}$, instead of $M^* \neq M$, not queried to any honest party.

1. We set the security guarantee at a conventional level $1 - 2^{-128}$.

QUASI is not designed to withstand this form of attack. To shed insight on this, recall that each party authenticates only d codeword positions. There exist two distinct statements M, M^* whose encodings agree at every position selected by honest party p_h. In other words, both codewords have identical symbols at all positions in π_h, which are authenticated by p_h. If the oracle is allowed to query honest parties on both M and M^*, then the adversary can form a valid qAttest on M by combining qShares from p_h, a set A of f honest parties, and a set B of f corrupt parties; and form another valid qAttest on M^* with qShares from p_h, a distinct set A' of f honest parties, and B.

Operations. QUASI requires no special protocol or preprocessing. For each context, parties independently generate the underlying ($\mathsf{Sign}_{\mathsf{QOTS}}$) keys material, but for far fewer symbols, without coordination or trusted setup. Each public key additionally specifies the d selected symbol positions. To contribute a qShare on M, a party applies the underlying authentication scheme to the symbols at those positions. Any entity may collect contributions from a set $S \subseteq [n]$, $|S| = 2f + 1$, and form $\sigma(M) = \{(i, \sigma_i(M)) : i \in S\}$, provided that the qShares are valid and mutually non-conflicting.

Stateful key management. Across multiple QUASI instances, parties can manage keys using established techniques from stateful hash-based signature schemes, such as XMSS [29], which supports the management of one-time WOTS-family keys. We treat key management as orthogonal to the QUASI construction and leave it outside the scope of this work.

2.1. Adversary models

We consider two possible adversary models: *fully adaptive* and *semi-adaptive*. Both models allow the adversary to corrupt up to f parties, but differ in when it can select which parties to corrupt. In the *fully adaptive* adversary model, the adversary can select f parties to corrupt at any point during the protocol execution. The *semi-adaptive* adversary model requires the adversary to commit to a specific set of at most f parties before the protocol begins. In both models, the adversary can then interact with the remaining $2f + 1$ honest parties through the oracle that provides information about their qShares. The following sections prioritize security against a *fully adaptive* adversary, however also considers the *semi-adaptive* model, which enables stronger efficiency advancements. We will also consider more practical cases, assuming a smaller proportion of Byzantine failures $f = b+c$, where b is the maximal number of Byzantine faults, and c additional crash faults.

2.2. Performance Measure

Our primary performance measure is the number of input bits authenticated per party, or equivalently, the number of symbols when the symbol size is fixed. For a statement M containing m symbols, we measure the reduction from authenticating all m symbols to authenticating only d symbols per qShare, giving a reduction factor of m/d.

The following costs are all derived directly from this measure: the authentication storage required for a qAttest containing $2f + 1$ qShares, the communication cost of collecting those qShares and responding to client queries, and the hash computations required to generate and verify each party's contribution. The per d-symbol authentication costs are inherited directly from the underlying authentication method, such as WOTS (Section 3); the savings arise from reducing the number of input symbols it authenticates.

These reductions also benefit SNARK-based aggregation. The attestation size and the hash computations required for verification contribute directly to the input size and verification work represented within the proof. Reducing these quantities relative to a collection of full hash-based signatures therefore yields proportional savings in the corresponding components of the proving workload.

3. Building Blocks

3.1. QUASI-OTS

We employ an underlying hash-based signature scheme for d-symbol authentication. Each key pair is used for at most one signing invocation.

Let $\mathcal{A} = \{0, 1\}^8$ be the symbol alphabet. QUASI-OTS is a hash-based one-time signature construction for words of d symbols, using WOTS to authenticate the concatenated $8d$-bit word, including its WOTS checksum. Its standard signing interface is

$$\Pi_{\mathsf{QOTS}} = (\mathsf{Setup}_{\mathsf{QOTS}}, \mathsf{KeyGen}_{\mathsf{QOTS}}, \mathsf{Sign}_{\mathsf{QOTS}}, \mathsf{Verify}_{\mathsf{QOTS}}).$$

- $\mathsf{Setup}_{\mathsf{QOTS}}(1^\kappa, d) \to \mathsf{pp}_{\mathsf{QOTS}}$. Output public parameters specifying the security parameter κ, the word space $\mathcal{A}^d$, the WOTS instantiation and its parameters, and the signing-key, verification-key, and signature spaces.
- $\mathsf{KeyGen}_{\mathsf{QOTS}}(\mathsf{pp}_{\mathsf{QOTS}}) \to (\mathsf{vk}^{\mathsf{QOTS}}, \mathsf{sk}^{\mathsf{QOTS}})$. Generate a WOTS key pair for authenticating words in $\mathcal{A}^d$.
- $\mathsf{Sign}_{\mathsf{QOTS}}(\mathsf{pp}_{\mathsf{QOTS}}, \mathsf{sk}^{\mathsf{QOTS}}, a) \to \tau$. For $a = a_1 \cdots a_d \in \mathcal{A}^d$, concatenate the symbols in their given order and apply WOTS encoding and signing, including the checksum. Output the resulting signature τ.
- $\mathsf{Verify}_{\mathsf{QOTS}}(\mathsf{pp}_{\mathsf{QOTS}}, \mathsf{vk}^{\mathsf{QOTS}}, a, \tau) \to \{0, 1\}$. Reject malformed inputs. Otherwise, apply WOTS verification to the same ordered word and its checksum.

QUASI-OTS inherits correctness and one-time unforgeability from the underlying WOTS instantiation. For completeness, we elaborate on the details of the underlying WOTS signing scheme in Appendix A.

3.2. Expanding Input to a Codeword

We employ a Reed-Solomon (RS) error-correcting code (ECC), denoted $c(\cdot)$, over statements $M \in \mathcal{M}$ (other codes may be substituted if needed).

A statement M is represented as a vector of $m = |M|/w$ symbols over an alphabet of w-bit symbols. We need to make sure that symbol width w is sufficiently large. The reason is that RS employs polynomials over $GF(2^w)$. In order to be able to encode m interpolation points, the field GF(2^w) must have size at least m, i.e., $2^w \geq m$. In our case, usually the statement to sign is a cryptographic digest of length 256 bits. The smallest w that satisfies $2^w \geq 256$ is $w = 8$. Therefore, henceforth M is represented by $m = 32$ symbols, each of 8 bits.

With a code of size $\hat{m} = |c(M)| = \lambda m$, an RS code can correct e corruptions in unknown symbols as well as o omissions of known symbols, such that $2e + o \leq (\lambda - 1)m$. Put differently, every pair of codewords differ in at least $D_{min} \coloneqq (\lambda - 1)m + 1$ symbols. The factor $\lambda = \hat{m}/m$ is called the expansion factor.

4. The QUASI Scheme

In this section, we develop the foundations of a QUASI construction with an ambitious goal: each party authenticates only d symbols, for a small constant d chosen as a function of n.

It is easy to see that unless each symbol of M is authenticated by $2f+1$ parties, there exists a collection of $2f+1$ parties and a symbol $b^* = M|_*$ such that b^* is not authenticated by any honest party. We refer to such a b^* as *flipped*: an adversary could produce a qAttest mixing authenticators over valid M symbols by honest parties with authenticators only by corrupt parties over b^* flipped.

In order to nevertheless get below $2f+1$ authenticators per M symbol, QUASI authenticates a codeword $c(M)$ over M, rather than M itself. For example, Figure 2 illustrates the QUASI approach with qShares of $d = 2$ symbols over an encoding $c(M)$ with expansion $\lambda = 3$.

Next, we describe the recipe for parties to authenticate subsets of $c(M)$.

4.1. Generic QUASI Construction

The family of QUASI constructions separates three interchangeable components: the encoding c, the symbol-assignment regime, and the number d of symbols authenticated per party. Generally, the assignment may be deterministic or randomized; this paper focuses on the latter. Their common requirement is a coverage condition: every set of $f + 1$ honest contributors must leave at most $2e$ *'holes'*, for the code that has minimum distance $D_{min} = 2e + 1$.

Parameters and Coverage. Let $n = 3f + 1$, with at most f corrupt parties, and let a qAttest contain $2f + 1$ qShares. Choose an injective encoding $c : \mathcal{A}^m \to \mathcal{A}^{\hat{m}}$ as described in Section 3, with minimum Hamming distance at least $2e + 1$. Its expansion factor is $\lambda = \hat{m}/m$.

An assignment regime specifies a public mapping $\pi_i : [d] \to [\hat{m}]$ for each party p_i. The mappings are fixed during set up time. For $H \subseteq [n]$, define its covered positions and holes by:

$$U(H) \coloneqq \bigcup_{i \in H} \pi_i([d]), \qquad E(H) \coloneqq [\hat{m}] \setminus U(H).$$

The required coverage condition is:

$$\forall H \subseteq [n], \quad |H| = f + 1 \implies |E(H)| \leq 2e.$$

For deterministic assignments, this condition holds unconditionally. For randomized assignments, the analysis bounds the probability that it fails. For an MDS code, including Reed–Solomon, $\hat{m} = m + 2e$, so the condition is equivalent to $|U(H)| \geq m$.

API. The construction uses QUASI-OTS to authenticate an ordered word of d symbols. Define the word selected by party p_i as:

$$a_i(M) \coloneqq c(M) \mid_{\pi_i(1)} \| \cdots \| \, c(M) \mid_{\pi_i(d)} .$$

$\mathsf{qSetup}(1^\kappa, n, f) \to \mathsf{pp}$. Check that $n \geq 3f + 1$; otherwise, output $\perp$. Select the statement space $\mathcal{M} = \mathcal{A}^m$, the encoding c, the parameter d, and an assignment regime specifying $(\pi_1, \ldots, \pi_n)$. Run $\mathsf{pp}_{\mathsf{QOTS}} \leftarrow \mathsf{Setup}_{\mathsf{QOTS}}(1^\kappa, d)$. The choices of c, d, and the assignment regime must satisfy (4.1), or provide a bound on its failure probability.

Output public parameters specifying n, f, $\mathcal{M}$, c, $\hat{m}$, e, d, the public mappings, and $\mathsf{pp}_{\mathsf{QOTS}}$. Let Σ_{sh} be the QUASI-OTS signature space, and define the qAttest space as:

$$\Sigma \coloneqq \{\{(i, \sigma_i) : i \in S\} \mid S \subseteq [n], \ |S| = 2f + 1, \ \sigma_i \in \Sigma_{\mathsf{sh}}\}.$$

$\mathsf{qKeyGen}(\mathsf{pp}) \to (\mathsf{vk}_i, \mathsf{sk}_i)$. Party p_i independently generates:

$$(\mathsf{vk}_i^{\mathsf{QOTS}}, \mathsf{sk}_i^{\mathsf{QOTS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{QOTS}}(\mathsf{pp}_{\mathsf{QOTS}}).$$

Set $\mathsf{vk}_i = (\pi_i, \mathsf{vk}_i^{\mathsf{QOTS}})$ and $\mathsf{sk}_i = (\pi_i, \mathsf{sk}_i^{\mathsf{QOTS}})$. The registered verification keys form $\mathsf{VK} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$.

$\mathsf{qShareSign}(\mathsf{pp}, \mathsf{sk}_i, M) \to \sigma_i$. For a pre-validated statement $M \in \mathcal{M}$, party p_i authenticates its selected word:

$$\sigma_i \leftarrow \mathsf{Sign}_{\mathsf{QOTS}}\big(\mathsf{pp}_{\mathsf{QOTS}}, \mathsf{sk}_i^{\mathsf{QOTS}}, a_i(M)\big).$$

Each QUASI-OTS key pair is used for at most one $\mathsf{qShareSign}$ invocation.

$\mathsf{qCombine}(\mathsf{pp}, \mathsf{VK}, S, M, \{\sigma_i\}_{i \in S}) \to \sigma$ or $\perp$. Require $S \subseteq [n]$ and $|S| = 2f + 1$. Form $\sigma = \{(i, \sigma_i) : i \in S\}$ and output it if $\mathsf{qVerify}(\mathsf{pp}, \mathsf{VK}, M, \sigma) = 1$; otherwise, output $\perp$.

$\mathsf{qVerify}(\mathsf{pp}, \mathsf{VK}, M, \sigma) \to \{0, 1\}$. Reject malformed inputs, statements outside $\mathcal{M}$, and attestations not containing exactly $2f+1$ distinct registered parties. For every $(i, \sigma_i) \in \sigma$, compute $a_i(M)$ and check:

$$\mathsf{Verify}_{\mathsf{QOTS}}\big(\mathsf{pp}_{\mathsf{QOTS}}, \mathsf{vk}_i^{\mathsf{QOTS}}, a_i(M), \sigma_i\big) = 1.$$

Accept if and only if all checks pass. Thus, every accepted qShare authenticates symbols of the same claimed codeword $c(M)$, and the qShares are mutually non-conflicting. Verification requires exact agreement with the claimed codeword; it does not correct conflicting authenticated values.

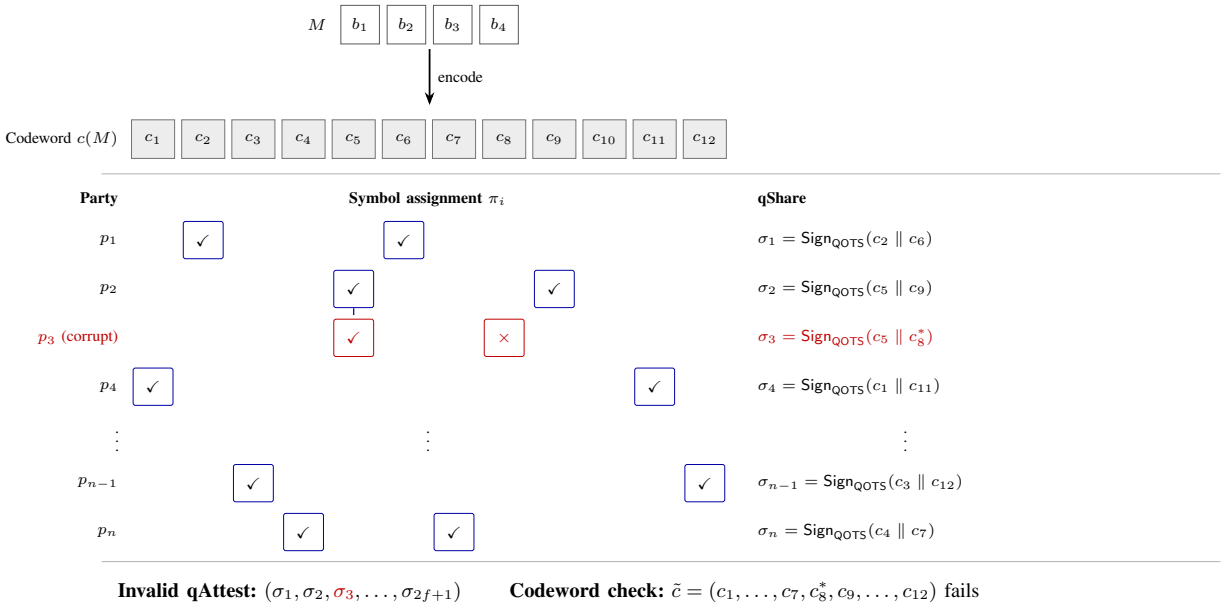

Figure 2. Randomized symbol assignments and QOTS shares for the codeword $c(M)$ of $M = b_1b_2b_3b_4$. Each party authenticates two symbols. The honest party p_2 and corrupt party p_3 both authenticate c_5, whereas only p_3 authenticates c_8. The alteration c_8^* creates no pairwise authentication conflict but is detected by the codeword check.

4.2. Correctness and Security

Theorem 1 (Validity)**.** *For every $M \in \mathcal{M}$ and every set $S \subseteq [n]$ of $2f+1$ honest parties, let $\sigma_i \leftarrow$ qShareSign$(\mathsf{pp}, \mathsf{sk}_i, M)$ for each $i \in S$. Then qCombine outputs a qAttest σ satisfying:*

$$\mathit{qVerify}(\mathsf{pp}, \mathsf{VK}, M, \sigma) = 1$$

with probability 1*, independently of the coverage condition.*

Proof. By correctness of QUASI-OTS, every σ_i passes verification on the selected word $a_i(M)$. All contributions authenticate symbols of the same codeword $c(M)$ and are therefore mutually non-conflicting. Since S contains $2f+1$ distinct registered parties, qCombine and qVerify accept. □

Theorem 2 (Weak-f-unforgeability)**.** *Suppose the coverage condition in Section 4.1 holds and the code has minimum distance $D_{min} = 2e+1$. In a context where honest parties contribute only on the pre-validated statement M, an adversary controlling at most f parties cannot produce an accepted qAttest for $M' \neq M$, except with the probability of breaking the underlying QUASI-OTS authentication.*

Proof. An accepted qAttest contains contributions from $2f+1$ distinct parties, including at least $f+1$ honest parties. Choose a set H of $f+1$ such honest contributors.

Unless QUASI-OTS authentication is broken, each accepted honest contribution authenticates the same selected word as its contribution on M. An accepted contribution from an honest party that never contributed in this context would itself constitute an authentication forgery. Consequently, $c(M') \mid_j = c(M) \mid_j$ for every $j \in U(H)$.

The two codewords can therefore differ only at holes:

$$\mathrm{dist}_{\mathrm{H}}\big(c(M), c(M')\big) \leq |E(H)| \leq 2e.$$

Distinct codewords have distance at least $D_{min} = 2e+1$. Hence $c(M') = c(M)$, and injectivity of c gives $M' = M$. □

If a randomized assignment violates the coverage condition with probability at most $\epsilon_{\mathsf{assign}}$, the overall forgery probability is at most $\epsilon_{\mathsf{assign}} + \epsilon_{\mathsf{auth}}$, where ϵ_{auth} bounds the probability of a successful attack on the underlying authentication. Deterministic constructions eliminate the assignment-failure term.

5. The Randomized Construction

We now present the randomized QUASI variant instantiating the scheme in Section 4. The main benefit of randomizing symbol selection is a graceful degradation against varying degrees of actual corrupt parties.

Fix $m = 32$ eight-bit statement symbols and a Reed–Solomon encoding of length $\hat{m} = 96$ and minimum distance $D_{min} = 65 = 2e+1$, where $e = 32$. During setup, each party independently samples a mapping $\pi_i : [d] \to [96]$, choosing each $\pi_i(j)$ independently and uniformly at random, without repetition.

Choosing the number d of symbols each party authenticates from $c(M)$ creates a tradeoff between the size of per-party qShare contribution and amount of potentially flipped symbols. Provided the number of flipped symbols is bounded

accordingly, the valid symbols which are authenticated by honest parties suffice to detect and prevent these flips. The larger the system, the smaller is a sufficient d. Section 5.1 provides an analysis proving that flipped symbols remain within the code's correction capacity with probability $1-2^{-128}$ for suitable choices of d, both against a fully-adaptive adversary and a semi-adaptive one; a sample of concrete values is then calculated in Section 6.

5.1. Forgery Probability Bounds

We proceed with the analysis of the relationship between the QUASI parameters that are needed to achieve negligible forgery probability, under different adversarial models and security settings. The analysis below links between $n = 3f+1$, the environment; and d, the choice of qShare size.

The adversary game we analyze is modeled as follows: honest parties choose their mappings $[d] \to [\hat{m}]$ uniformly at random and publish them along with their public QUASI-OTS keys.

The adversary can select any subset S' of $f+1$ honest parties and query for their qShares over M. It can use f corrupt parties under its control to generate their qShares over $c(M)^*$ and form a qAttest, where $c(M)^*$ has certain symbols flipped at positions left unauthenticated by S'. The adversary wins if the fraction of the symbols in $c(M)$ that S' leaves unauthenticated matches the code distance D_{min}, allowing the adversary to undetectably flip symbols.

For each subset S' of $f+1$, denote $E(S')$ the total number of symbols left unauthenticated by S'. The game's outcome is the maximum size of $E(S')$ over all selections of sets S': $e_{actual} := \max\{E(S') \mid |S'| = f+1\}$. The goal is to link d, n, and f, such that with high probability $e_{actual} < D_{min}$.

Adaptive adversary. We proceed with a probabilistic analysis with a fully adaptive adversary, who may select f parties to corrupt after seeing the honest parties' mappings. Abstractly, this game consists of a set of $3f+1$ *throws* $\{\pi_i\}_{i=1..(3f+1)}$, representing the mappings π_i selected by parties before the adversary chooses f to corrupt. Every π_i consists of d balls. Each throw places d balls into $\hat{m}$ bins uniformly at random, such that no two balls in a throw can land in the same bin.

Consider a subset T of D_{min} bins. We estimate the probability that there exists a subset of $f+1$ throws entirely avoiding T. The probability that π_i avoids the specific set T entirely is $\binom{\hat{m}-D_{min}}{d} \cdot \binom{\hat{m}}{d}^{-1}$. We can bound this probability from above by assuming the selection of d bins is fully independent (i.e., allow repetitions):

$$q = \binom{\hat{m}-D_{min}}{d}\binom{\hat{m}}{d}^{-1} \leq \left(\frac{\hat{m}-(2\hat{m}/3+1)}{\hat{m}}\right)^d < \left(\frac{1}{3}\right)^d .$$

Since the set of $3f+1$ throws are independent, this experiment forms a Bernoulli trial with $3f+1$ repetitions, each with success probability q.

Let $B_f(q) \coloneqq \Pr_{X\sim\mathrm{Bin}(3f+1,q)}[X \geq f+1]$ denote the binomial upper-tail probability of $f+1$ successes. Since $q < 1/3$, applying a Chernoff bound yields:

$$\begin{aligned} B_f(q) &\leq \Pr[\mathrm{Bin}(3f+1,q) > (3f+1)/3] \\ &\leq \exp\left(-(3f+1)\cdot D_{\mathrm{KL}}\left(\frac{1}{3}\middle\|q\right)\right) \end{aligned}$$

where D_{KL} is KL-divergence with parameters $a = \frac{1}{3}$ and q, i.e., $D_{\mathrm{KL}}\left(\frac{1}{3}\|q\right) = -\frac{1}{3}\ln(\frac{27}{4}q(1-q)^2)$, which leads to:

$$B_f(q) \leq \left[\frac{27}{4}q(1-q)^2)\right]^{\frac{(3f+1)}{3}} .$$

The preceding argument applies to a fixed set T of bins. Taking a union bound over all subsets of D_{min} bins bounds the probability that the maximal number e_{actual} of holes left by any subset S' reaches the code distance as follows:

$$\Pr[e_{actual} \geq D_{min}] \leq \binom{\hat{m}}{D_{min}} B_f(q)$$

For $\hat{m} = 3m$, $D_{min} = 2m+1$, Stirling's approximation gives $\binom{3m}{2m+1} < \binom{3m}{m} \approx (27/4)^m\sqrt{3/(4\pi m)}$. We use the upper bound $\binom{3m}{2m+1} \leq (27/4)^m$ to obtain:

$$\Pr[e_{actual} \geq D_{min}] \leq \left(\frac{27}{4}\right)^m \left[\frac{27}{4}q\,(1-q)^2\right]^{\frac{(3f+1)}{3}}$$

The probability above now depends on three parameters: d, the number of symbols per throw, n, the number of overall parties, and f, the fraction of Byzantine ones. This allows to fix n and determine d accordingly; or conversely, fix d-size qShares and determine a required system size n; Section 6 derives concrete numbers.

Semi-adaptive adversary. The bound above assumes a fully adaptive adversary. The semi-adaptive model forces the adversary to commit to specific f corrupt parties before all parties reveal their π_i selections. For this setting, we are interested in bounding a bad event e'_{actual} occurring within the $2f+1$ honest throws only, since the adversary will have already designated f corruptions. Thus, the bad event e'_{actual} becomes: $f+1$ out of $2f+1$ honest throws avoid D_{min} bins, and the probability bound on $\Pr[e'_{actual} \geq D_{min}]$ can be tightened to:

$$\Pr[e'_{actual} \geq D_{min}] \leq \left(\frac{27}{4}\right)^m [4q\,(1-q)]^{\frac{(2f+1)}{2}} .$$

Byzantine–crash adversary. So far, we have considered the case where all f tolerated failures are Byzantine. Practically, we can consider more pragmatic settings and see how we can scale the problem to weaker settings. Thus, we can expand the model to $f = b+c$, where b corresponds to Byzantine failures and c to crash failures; $b = f$ corresponds to the usual adversary. Under this model, the adversary can contribute at most b corrupted qShares. A valid qAttest remains consisting of $2f+1$ qShares, and it is possible to collect these from parties which do not suffer either type of fault.

In the fully-adaptive adversary model, the bad event e''_{actual} occurs here when $2f+1-b$ throws out of $3f+1$ avoid D_{min} bins, resulting in the following probability bound on $\Pr[e''_{actual} \geq D_{min}]$:

$$\min\left\{1, \left(\frac{27}{4}\right)^m e^{-(3f+1)\, D_{\mathrm{KL}}(a \,\|\, q)}\right\}$$

where $a = \frac{2f+1-b}{3f+1}$.

A very similar analysis for a semi-adaptive adversary results in the following probability bound:

$$\min\left\{1, \left(\frac{27}{4}\right)^m e^{-(2f+1)\, D_{\mathrm{KL}}(a' \,\|\, q)}\right\}$$

where $a' = \frac{2f+1-b}{2f+1}$.

6. Concrete Numbers

We evaluate the tradeoff in concrete numbers between the system size $n = 3f+1$, the number d of symbols authenticated per qShare, and the assignment-failure probability. Throughout, we consider statements of $m = 32$ symbols and codewords of length $\hat{m} = 3m$ and minimum code distance $D_{min} := 2m + 1$. Let $\epsilon_{\mathsf{assign}}$ denote the probability that the assignment leaves at least D_{min} holes. Our default target is $\epsilon_{\mathsf{assign}} \leq 2^{-128}$. The reduction in authenticated symbols per party, relative to authenticating the full statement, is m/d.

6.1. Required system size for a given d

We first fix d and determine a sufficient system size for the default assignment-failure target. We consider fully adaptive and semi-adaptive adversaries, followed by mixed Byzantine–crash faults. Table 2 summarizes the sufficient thresholds and the corresponding reductions m/d for all three models.

Fully adaptive adversary. The bounds in Section 5.1 are practical when $d \geq 2$. We evaluate $d \in \{2, 3, 5, 10, 21\}$ using the general analysis for arbitrary $d \geq 2$.

For example, with $d = 2$ and $m = 32$, the numerical expression for the bound is approximately

$$\epsilon_{\mathsf{assign}} \leq 2^{\,2.754888m - 0.251629(3f+1)}.$$

This gives the sufficient threshold $f \geq 287$. Increasing d to 3 substantially reduces the required system size: the corresponding threshold is $f \geq 103$.

Semi-adaptive adversary. Under the semi-adaptive model, the analysis also permits $d = 1$. The default target is achieved with $f = 1272$ for $d = 1$, and with $f = 161$ for $d = 2$.

Mixed Byzantine–crash faults. Keeping the total fault bound $f = b + c$ fixed while reducing the Byzantine-fault bound b permits more favorable parameters. As an illustrative case, consider $b = \lfloor 3f/4 \rfloor$.

For the fully adaptive model, the analysis in Section 5.1 yields the following KL-divergence expression, where q is the avoidance probability defined there:

$$\epsilon_{\mathsf{assign}} \leq \binom{\hat{m}}{\hat{m}/3} \exp\left(-(3f+1) D_{\mathrm{KL}}\left(\frac{5}{12}\middle\| q\right)\right).$$

TABLE 2. CONCRETE PARAMETERS AND SAVINGS UNDER FULLY ADAPTIVE AND SEMI-ADAPTIVE ADVERSARY MODELS. HERE $n = 3f + 1$ AND $f = b + c$.

Symbols d	**1**	**2**	**3**	**5**	**10**	**21**
Reduction m/d	$\times 32$	$\times 16$	$\times 10.7$	$\times 6.4$	$\times 3.2$	$\times 1.5$
Fully adaptive						
$b = f$						
Parties n	–	862	310	127	52	22
Corrupt f	–	287	103	42	17	7
$b = 3f/4$						
Parties n	–	493	211	94	40	19
Corrupt f	–	164	70	31	13	6
$b = f/2$						
Parties n	–	325	154	73	34	16
Corrupt f	–	108	51	24	11	5
Semi-adaptive						
$b = f$						
Parties n	3817	484	232	109	49	22
Corrupt f	1272	161	77	36	16	7
$b = 3f/4$						
Parties n	1270	298	160	82	37	16
Corrupt f	423	99	53	27	12	5

Using $\hat{m} = 3m$, this becomes

$$\epsilon_{\mathsf{assign}} \leq \left(\frac{27}{4}\right)^m \left[\frac{12^{12}}{5^5 7^7} q^5 (1-q)^7\right]^{\frac{(3f+1)}{12}}.$$

The corresponding expression for the semi-adaptive model is

$$\epsilon_{\mathsf{assign}} \leq \left(\frac{27}{4}\right)^m \left[\frac{8^8}{5^5 3^3} q^5 (1-q)^3\right]^{\frac{(2f+1)}{8}}.$$

6.2. Varying the assignment-failure target

A uniform assignment-failure target of 2^{-128} may be unnecessarily conservative. Applications could instead require this bound in the common case while allowing graceful degradation as the number of Byzantine faults increases, tolerating a higher assignment-failure probability in extreme but rare fault configurations. This flexibility can enable smaller qShares for a fixed system size and fault bound, or smaller systems for a fixed d.

Figure 3 shows how the proportion of possible Byzantine failures for target security varies with d for fixed system sizes. Figure 4 shows the effect of varying the Byzantine-fault bound.

6.3. Selecting parameters

The general bounds allow parameter selection beyond the examples above. Given an adversary model, system size, fault bounds, and assignment-failure target, we select the smallest d satisfying the corresponding bound.

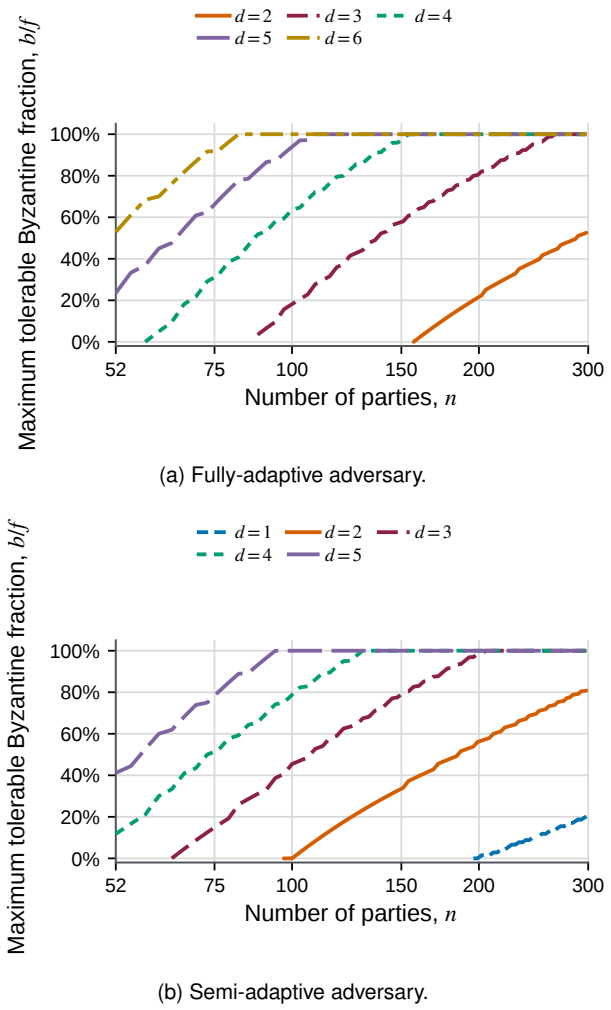

Figure 3. Maximum tolerable Byzantine fraction b/f as a function of the number of parties n, under (a) fully adaptive and (b) semi-adaptive adversaries, for varying d, with $n = 3f + 1$ and $m = 32$. For each configuration, the maximum Byzantine tolerance is determined using the refined binomial-tail union bound with a security target of 2^{-128}. Curves connect discrete $n = 3f + 1$ configurations by straight segments as visual guides. Values of $b/f = 100\%$ indicate that the bound is satisfied for the full Byzantine fault budget $b = f$.

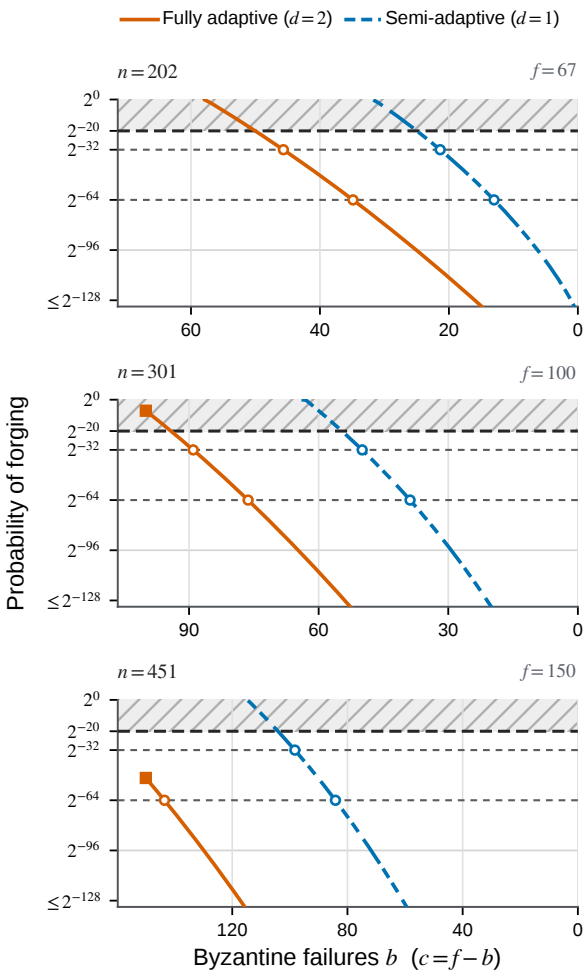

Figure 4. Probability of forging as a function of Byzantine failures considering a large-scale number of parties n under fully adaptive and semi-adaptive adversaries for smallest corresponding d, with $m = 32$. Curves terminate when the bound reaches 1 or 2^{-128}.

For small systems, e.g., around $n \leq 70$, the simple deterministic construction from the introduction provides a useful point of comparison: each party authenticates approximately two-thirds of the statement, giving an asymptotic reduction of $1.5\times$. The randomized construction offers progressively larger reductions as the system grows, exceeding $10\times$ in suitable parameter regimes.

Parameter selection should therefore compare the randomized bounds with the deterministic construction and use whichever authenticates fewer symbols per party.

7. Related Works

Since QUASI lies at the intersection of (1) threshold and aggregate signatures, (2) hash-based one-time signatures, and (3) quorum protocols, we review related work in these three dimensions.

Threshold signatures. Desmedt et al. [21] introduced the concepts of *threshold signatures* which relies on parties holding shares of a single signing key associated with a common public key. The key idea is that each party produces a partial signature, where any collection of t valid shares can be combined to create a standard signature, without revealing the signers. The cost of threshold signature schemes is that they necessitate pre-coordination and sharing of the signing key among parties. Examples of threshold signature schemes include threshold RSA [49], threshold BLS [9], threshold ECDSA [27], and FROST (threshold Schnorr) [32]. Micali et al. extended this to the notion of *multisignatures* in [43] where each party holds an independent signing key and uses it to contribute a signature. These signatures are then collected into a multisignature certifying that a specified set of parties

signed the same message. Verification generally requires public keys for specific participants. Hence signers are known and no shared secret key or threshold secret-sharing structure is used. Examples of multisignature schemes include Bellare–Neven (BN) [6], MuSig [40], MuSig2 [45], and BLS multisignatures [11]. Extending to signatures on potentially different messages, Boneh et al. introduced *aggregate signatures* in [12]. When all parties sign the same message, aggregate signatures closely resemble multisignatures. Examples of aggregate signature schemes include BGLS [12], LMRS sequential aggregate signatures [38], and LOSSW sequential aggregate signatures [37].

While not being a threshold-signature scheme, QUASI is a newly conceived hybrid primitive resting on intersection of the above mentioned schemes: it supports quorum-based aggregation, partial signatures are bound to their respective signers, and it allows any threshold of valid, non-conflicting shares satisfying the quorum requirement to create a finalized attestation.

More recent work extends threshold signing to post-quantum hash-based signatures. Kelsey, Lang, and Lucks [31] present an early construction in which a trusted dealer generates all one-time signing secrets in advance, distributes shares, and publishes auxiliary data. An aggregator then combines the parties' contributions to reconstruct the signature values selected by the message. PRAWNS [10] follows a related preprocessing-and-reconstruction approach based on threshold pseudorandom functions (PRFs). Its lattice-based instantiation supports large groups, while its purely hash-based instantiation targets smaller committees.

QUASI takes a fundamentally different and simpler approach than both: independently generated keys and sparse symbol authentication enable dealer-free attestation for large groups of parties. QUASI's weak-f-unforgeability shares with PRAWNS's context-aware construction and Haystack's message-binding mechanism the goal of preventing malicious recombination of contributions into an unauthorized result.

Hash-based Signatures. The notion of one-time signatures (OTS) was first introduced by Lamport in [33]. The key idea is that signatures are constructed by revealing secret values whose hashes form the public key. Merkle subsequently developed hash-based signature constructions[42], allowing multiple one-time public keys to be authenticated under a single Merkle-tree root. Further improvements of Winternitz one-time signatures (WOTS) and their extension WOTS+ [28] use hash chains to reduce signature size at the expense of additional hash computations. These constructions provide a foundation for post-quantum signatures based on cryptographic hash functions, where examples include XMSS [29] and LMS/HSS [41]. Both are stateful schemes that require tracking used one-time keys to prevent reuse, and are standardized in NIST SP 800-208 [19]. In contrast, SPHINCS+, standardized as SLH-DSA in FIPS 205 [44], provides stateless hash-based signatures by combining FORS, WOTS+, and a hypertree, eliminating explicit signing-state management at the cost of larger signatures. More recent advancements include leanSig and leanXMSS have been proposed for post-quantum blockchain consensus [23], [20], with leanVM [35] supporting proof-based aggregation of signature verification. However, none of these hash-based signatures inherently support compact aggregation of signatures produced by independent signers.

A significant body of work uses hash-based zero-knowledge proofs, such as STARKs [7], to prove the correctness of signature verification without revealing the underlying signature or public key. By committing to an execution trace using Merkle trees and verifying algebraic constraints, these systems can certify the validity of classical signatures, such as ECDSA [48], while keeping their public keys hidden. This approach has been explored for post-quantum migration [25], [48]. More recent advances in succinct proof systems, including PLONK [26], its hash-based extensions such as Plonky2 [46], and Flock [15], have substantially improved proving efficiency. In particular, Flock demonstrates efficient batch proving of cryptographic hash computations.

QUASI targets shares of parties that can be combined in the compact attestation, without requiring a shared threshold signing key or general-purpose zero-knowledge proof. However, QUASI is orthogonal to, and compatible with, proof-based compression. Putting the two together compounds the compression benefits of both.

Quorum-based protocols. Byzantine quorum systems [39] provide a fundamental building block for fault-tolerant distributed systems. In the classical setting of $n = 3f+1$ parties, quorums of size $2f+1$ intersect in at least $f + 1$ parties, guaranteeing at least one honest party in their intersection. Beyond consensus, quorum-based constructions underpin Byzantine reliable broadcast [13], fault-tolerant storage and register emulation [39], and payment systems such as FastPay [5], Stingray [50], and the Sui blockchain [8].

Quorum intersection is also central to Byzantine fault-tolerant consensus. The pioneering Practical Byzantine Fault Tolerance (PBFT) protocol of Castro and Liskov [17] established a practical quorum-based approach to Byzantine consensus. Numerous subsequent protocols have built upon this foundation, and a comprehensive survey is beyond our scope. Notably, HotStuff [52] uses threshold signatures to aggregate quorum votes into compact quorum certificates (QCs), enabling linear communication complexity in the optimistic case. It remains an open question whether sub-linear aggregation of hash-based signatures is possible in HotStuff-like protocols.

More recently, information-theoretic BFT protocols, including IT-HotStuff [4], TetraBFT [53], Forget-IT [3], Simple-IT [54], Quintus [36], Quantum Cadence [30], and others, solved consensus without digital signatures, relying instead on authenticated point-to-point communication. While these protocols avoid public-key signature operations during consensus, they do not inherently produce transferable, publicly verifiable certificates that external clients can independently verify.

These developments highlight the importance of efficient external attestation for quorum-based systems. QUASI fills this role by enabling parties to produce publicly verifiable post-quantum attestations of statements already validated by

a quorum, without modifying the underlying consensus protocol. Thus, QUASI provides a complementary mechanism for externally certifying the outcomes of both traditional and signature-free quorum protocols.

8. Discussion

QUASI opens many directions for further research, from tighter analysis and improved constructions to practical optimizations and new applications.

The construction and analysis of Section 5.1 leave substantial room to improve the constants. In particular, the union bound used to link d and f is fairly crude; a tighter analysis may yield smaller qShares or support smaller systems. Applications may also tolerate an assignment-failure probability higher than 2^{-128}, allowing more favorable parameters. Likewise, corruption thresholds other than $n/3$ merit exploration.

We explored a randomized construction that expands the statement into a codeword to tolerate flipped symbols. We are confident that deterministic constructions which tolerate symbol flips can also achieve significant benefits beyond the simple assignment of each symbol to $2f + 1$ parties, particularly for medium-size systems. We are currently exploring improved deterministic symbol assignments.

A practical implementation would provide end-to-end measurements and may reveal further opportunities for improvement by leveraging the underlying QUASI one-time hash-based authentication and its reduced size. These opportunities include optimizing stateful and stateless key management and amortizing Winternitz checksum symbols across qShares. The smaller per-party contributions also motivate tuning and optimizing SNARK-based aggregation, with end-to-end comparisons against proving the verification of collections of full hash-based signatures.

QUASI may also be useful within conventional signature-based consensus protocols, beyond serving as an external attestation layer. The reason is that hash-based authentication is simple and computationally efficient compared with conventional digital signatures, while their main hurdle, size, is removed by QUASI. Such integration would require examining which protocol steps can use QUASI's weaker guarantees while preserving the consensus security.

Finally, QUASI introduces a new foundation for group-based attestation. Its lower bounds, security properties, and relationship to threshold signatures, aggregate signatures, and multisignatures merit further study. Exploring these connections may reveal applications in other cryptographic settings and clarify the tradeoffs between attestation guarantees and efficiency.

9. Acknowledgment

We thank Benjamin Marsh and Nibesh Shrestha for their valuable feedback and suggestions that helped improve this paper.

References

[1] AARONSON, S., BONEH, D., DRAKE, J., KANNAN, S., LINDELL, Y., AND MALKHI, D. Quantum computing & blockchain. Position paper, Coinbase Independent Advisory Board on Quantum Computing and Blockchain, Apr. 2026.

[2] AARONSON, S., AND SHI, Y. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM 51*, 4 (2004), 595–605.

[3] ABRAHAM, I., DAS, S., EFRON, Y., AND KOMATOVIC, J. Forget-IT: Optimal good-case latency for information-theoretic BFT. Cryptology ePrint Archive, Paper 2026/355, 2026. Also published at PODC 2026.

[4] ABRAHAM, I., AND STERN, G. Information theoretic HotStuff. arXiv:2009.12828, 2020.

[5] BAUDET, M., DANEZIS, G., AND SONNINO, A. FastPay: High-performance byzantine fault tolerant settlement. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies* (2020), AFT '20, ACM, pp. 163–177.

[6] BELLARE, M., AND NEVEN, G. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (2006), ACM, pp. 390–399.

[7] BEN-SASSON, E., BENTOV, I., HORESH, Y., AND RIABZEV, M. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018.

[8] BLACKSHEAR, S., CHURSIN, A., DANEZIS, G., KICHIDIS, A., KOKORIS-KOGIAS, L., LI, X., LOGAN, M., MENON, A., NOWACKI, T., SONNINO, A., WILLIAMS, B., AND ZHANG, L. Sui Lutris: A blockchain combining broadcast and consensus. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security* (2024), CCS '24, ACM, pp. 2606–2620.

[9] BOLDYREVA, A. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *Public Key Cryptography—PKC 2003* (2003), vol. 2567 of *Lecture Notes in Computer Science*, Springer, pp. 31–46.

[10] BONEH, D., BÜNZ, B., JUSTIN D., LITCHEV, Y., AND NAZIRKHANOVA, K. PRAWNS: Threshold hash-based signatures from threshold PRFs. Cryptology ePrint Archive, Paper 2026/2142, 2026.

[11] BONEH, D., DRIJVERS, M., AND NEVEN, G. Compact multi-signatures for smaller blockchains. In *Advances in Cryptology—ASIACRYPT 2018* (2018), Lecture Notes in Computer Science, Springer, pp. 435–464.

[12] BONEH, D., GENTRY, C., LYNN, B., AND SHACHAM, H. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003* (2003), E. Biham, Ed., vol. 2656 of *Lecture Notes in Computer Science*, Springer, pp. 416–432.

[13] BRACHA, G. Asynchronous byzantine agreement protocols. *Information and Computation 75*, 2 (1987), 130–143.

[14] BRANDÃO, L. T. A. N., AND PERALTA, R. NIST first call for multi-party threshold schemes. NIST Interagency/Internal Report 8214C, National Institute of Standards and Technology, Jan. 2026.

[15] BÜNZ, B., ROTHBLUM, R., AND WANG, W. Flock: Fast proving for batch boolean computations. Cryptology ePrint Archive, Paper 2026/1329, 2026.

[16] CACHIN, C., KURSAWE, K., PETZOLD, F., AND SHOUP, V. Secure and efficient asynchronous broadcast protocols. In *Advances in Cryptology—CRYPTO 2001* (2001), J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*, Springer, pp. 524–541.

[17] CASTRO, M., AND LISKOV, B. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (1999), OSDI '99, USENIX Association, pp. 173–186.

[18] CHEN, B., BÜNZ, B., BONEH, D., AND ZHANG, Z. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In *Advances in Cryptology—EUROCRYPT 2023, Proceedings, Part II* (2023), C. Hazay and M. Stam, Eds., vol. 14005 of *Lecture Notes in Computer Science*, Springer, pp. 499–530.

[19] COOPER, D. A., APON, D. C., DANG, Q. H., DAVIDSON, M. S., DWORKIN, M. J., AND MILLER, C. A. Recommendation for stateful hash-based signature schemes. NIST Special Publication 800-208, National Institute of Standards and Technology, Oct. 2020.

[20] CORATGER, T., WEDDERBURN, K., WAMBSGANS, T., AND DRAKE, J. Eip-8450: Post-quantum consensus key registry. Ethereum Improvement Proposals, 2026. Draft.

[21] DESMEDT, Y., AND FRANKEL, Y. Shared generation of authenticators and signatures. In *Advances in Cryptology – CRYPTO '91* (1992), vol. 576 of *Lecture Notes in Computer Science*, Springer, pp. 457–469.

[22] DRAKE, J., KHOVRATOVICH, D., KUDINOV, M., AND WAGNER, B. Hash-based multi-signatures for post-quantum Ethereum. *IACR Communications in Cryptology 2*, 1 (2025). Article 13.

[23] DRAKE, J., KHOVRATOVICH, D., KUDINOV, M., AND WAGNER, B. Technical note: LeanSig for post-quantum ethereum. Cryptology ePrint Archive, Paper 2025/1332, 2025.

[24] ETHEREUM FOUNDATION POST-QUANTUM TEAM. Post-quantum Ethereum. Project website and roadmap. Accessed November 3, 2026.

[25] FAN, X., TAN, T. G., HO, N., AND CHOY, S. H. Enabling a smooth migration towards post-quantum security for Ethereum. In *International Conference on Blockchain—ICBC 2024* (2024), Springer, pp. 3–15.

[26] GABIZON, A., WILLIAMSON, Z. J., AND CIOBOTARU, O. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019.

[27] GENNARO, R., AND GOLDFEDER, S. One round threshold ECDSA with identifiable abort. Cryptology ePrint Archive, Paper 2020/540, 2020. Revised version; the authors note that this report is superseded by later joint work.

[28] HÜLSING, A. W-OTS+ – shorter signatures for hash-based signature schemes. In *Progress in Cryptology – AFRICACRYPT 2013* (2013), A. Youssef, A. Nitaj, and A. E. Hassanien, Eds., vol. 7918 of *Lecture Notes in Computer Science*, Springer, pp. 173–188.

[29] HÜLSING, A., BUTIN, D., GAZDAG, S., RIJNEVELD, J., AND MOHAISEN, A. XMSS: eXtended merkle signature scheme. RFC 8391, Internet Research Task Force, May 2018.

[30] JALALZAI, M. M., KLENZE, T., AND KOMATOVIC, J. Quantum cadence: Post-quantum consensus with multiple concurrent proposers. arXiv:2610.32803, 2026.

[31] KELSEY, J., LANG, N., AND LUCKS, S. Turning hash-based signatures into distributed signatures and threshold signatures: Delegate your signing capability, and distribute it among trustees. *IACR Communications in Cryptology 2*, 2 (2025).

[32] KOMLO, C., AND GOLDBERG, I. FROST: Flexible round-optimized schnorr threshold signatures. Cryptology ePrint Archive, Paper 2020/852, 2020.

[33] LAMPORT, L. Constructing digital signatures from a one way function. Tech. Rep. CSL-98, SRI International, Computer Science Laboratory, Oct. 1979.

[34] LAMPORT, L., SHOSTAK, R., AND PEASE, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems 4*, 3 (1982), 382–401.

[35] LEANETHEREUM. leanVM: Minimal zkvm for ethereum. GitHub repository, 2026. Accessed November 10, 2026.

[36] LIU, C., MALKHI, D., NAYAK, K., AND SHRESTHA, N. Quintus: Two-round good-case information theoretic BFT for $n = 5f + 1$. Cryptology ePrint Archive, Paper 2026/1520, 2026.

[37] LU, S., OSTROVSKY, R., SAHAI, A., SHACHAM, H., AND WATERS, B. Sequential aggregate signatures and multisignatures without random oracles. In *Advances in Cryptology—EUROCRYPT 2006* (2006), vol. 4004 of *Lecture Notes in Computer Science*, Springer, pp. 465–485.

[38] LYSYANSKAYA, A., MICALI, S., REYZIN, L., AND SHACHAM, H. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology—EUROCRYPT 2004* (2004), vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 74–90.

[39] MALKHI, D., AND REITER, M. K. Byzantine quorum systems. *Distributed Computing 11*, 4 (1998), 203–213.

[40] MAXWELL, G., POELSTRA, A., SEURIN, Y., AND WUILLE, P. Simple schnorr multi-signatures with applications to Bitcoin. *Designs, Codes and Cryptography 87* (2019), 2139–2164.

[41] MCGREW, D., CURCIO, M., AND FLUHRER, S. Leighton-micali hash-based signatures. RFC 8554, Internet Research Task Force, Apr. 2019.

[42] MERKLE, R. C. A certified digital signature. In *Advances in Cryptology—CRYPTO '89* (1990), G. Brassard, Ed., vol. 435 of *Lecture Notes in Computer Science*, Springer, pp. 218–238.

[43] MICALI, S., OHTA, K., AND REYZIN, L. Accountable-subgroup multisignatures. In *Proceedings of the 8th ACM Conference on Computer and Communications Security* (2001), ACM, pp. 245–254.

[44] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Stateless hash-based digital signature standard. Federal Information Processing Standards Publication 205, National Institute of Standards and Technology, Aug. 2024.

[45] NICK, J., RUFFING, T., AND SEURIN, Y. MuSig2: Simple two-round schnorr multi-signatures. In *Advances in Cryptology—CRYPTO 2021, Part I* (2021), vol. 12825 of *Lecture Notes in Computer Science*, Springer, pp. 189–221.

[46] POLYGON ZERO TEAM. Plonky2: Fast recursive arguments with PLONK and FRI. Technical report, Polygon Zero, Sept. 2022. Draft dated September 7, 2022.

[47] REITER, M. K. Secure agreement protocols: Reliable and atomic group multicast in Rampart. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security* (1994), CCS '94, ACM, pp. 68–80.

[48] RISC ZERO. RISC Zero developer documentation: Use cases. Developer documentation. Includes proving the validity of ECDSA signature verification; accessed November 14, 2026.

[49] SHOUP, V. Practical threshold signatures. In *Advances in Cryptology—EUROCRYPT 2000* (2000), vol. 1807 of *Lecture Notes in Computer Science*, Springer, pp. 207–220.

[50] SRIDHAR, S., SONNINO, A., AND KOKORIS-KOGIAS, L. Stingray: Fast concurrent transactions without consensus. arXiv:2501.06531, 2025.

[51] SUCCINCT LABS. Introducing Telepathy: The first decentralized and secure zkSNARK interoperability protocol for Ethereum. Blog post. Accessed November 5, 2026.

[52] YIN, M., MALKHI, D., REITER, M. K., GUETA, G. G., AND ABRAHAM, I. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing* (2019), PODC '19, ACM, pp. 347–356.

[53] YU, Q., LOSA, G., AND WANG, X. TetraBFT: Reducing latency of unauthenticated, responsive BFT consensus. arXiv:2405.02615, 2024. Full version of the PODC 2024 paper.

[54] YU, Q., VILLACIS, J., LOSA, G., XIANG, Z., AND WANG, X. Simple-IT: Practical low-latency signature-free BFT consensus. arXiv:2606.14404, 2026.

Appendix A.

To authenticate an ordered subset of d codeword symbols, QUASI concatenates them into a word and applies standard WOTS signing and verification. We refer to this adaptation as *Quasi One-Time Authentication Scheme* (QOTS), which inherits the one-time unforgeability guarantees of WOTS.

Quasi One-Time Authentication Scheme (QOTS). Let $\mathcal{A} = \{0, \dots, 2^\omega - 1\}$ be the symbol alphabet. Let $a = (a_1, \dots, a_d) \in \mathcal{A}^d$ denote the ordered word authenticated by QOTS. In the outer QUASI construction, party p_i obtains this word by selecting positions from the codeword $c(M)$ according to the mapping $\pi_i : [d] \to [\hat{m}]$:

$$a := c(M)_{\pi_i(1)} \cdots c(M)_{\pi_i(d)} \in \mathcal{A}^d.$$

Let $\Pi_{\mathsf{WOTS}} = (\mathsf{Setup}_{\mathsf{WOTS}}, \mathsf{KeyGen}_{\mathsf{WOTS}}, \mathsf{Sign}_{\mathsf{WOTS}}, \mathsf{Verify}_{\mathsf{WOTS}})$ be a one-time signature scheme for words in $\mathcal{A}^d$.

WOTS encoding. For a word $a = (a_1, \dots, a_d)$, WOTS appends the fixed-length base-2^ω encoding of the checksum $C(a) = \sum_{j=1}^{d}(2^\omega - 1 - a_j)$. The checksum requires $t = \left\lfloor \frac{\log_2(d(2^\omega - 1))}{\omega} \right\rfloor + 1$ symbols. We can then write the resulting encoding as $E(a) = (a_1, \dots, a_d, s_1, \dots s_t) = (e_1, \dots, e_{d+t})$, where $s_1, \dots, s_t$ are base-2^ω checksum digits. Following the basic WOTS construction, the scheme samples an independent secret starting value x_i for each encoded symbol. The scheme constructs hash chains of length 2^ω, with each public endpoint computed as $y_i = H^{2^\omega - 1}(x_i)$. The signature will be defined as $\tau_i = H^{e_i}(x_i)$. To verify the signature, it is sufficient to check:

$$\forall i \in [d + t] : \qquad H^{2^\omega - 1 - e_i}(\tau_i) = y_i.$$

Here, H^r denotes r successive hash applications of the hash function H. Alternatively, a concrete instantiation may instead use the addressed, randomized chains of WOTS+, with its corresponding security assumptions and verification algorithm.

QOTS interface. The scheme $\Pi_{\mathsf{QOTS}} = (\mathsf{Setup}_{\mathsf{QOTS}}, \mathsf{KeyGen}_{\mathsf{QOTS}}, \mathsf{Sign}_{\mathsf{QOTS}}, \mathsf{Verify}_{\mathsf{QOTS}})$ is defined as follows. In the terminology of QUASI, its authentication output is a *qShare*.

- $\mathsf{Setup}_{\mathsf{QOTS}}(1^\kappa, d) \to \mathsf{pp}_{\mathsf{QOTS}}$. Run $\mathsf{pp}_{\mathsf{WOTS}} \leftarrow \mathsf{Setup}_{\mathsf{WOTS}}(1^\kappa)$ for words in $\mathcal{A}^d$. Output $\mathsf{pp}_{\mathsf{QOTS}} = (\mathsf{pp}_{\mathsf{WOTS}}, 2^\omega, d)$.
- $\mathsf{KeyGen}_{\mathsf{QOTS}}(\mathsf{pp}_{\mathsf{QOTS}}) \to (\mathsf{vk}^{\mathsf{QOTS}}, \mathsf{sk}^{\mathsf{QOTS}})$. Generate $(\mathsf{vk}_{\mathsf{WOTS}}, \mathsf{sk}_{\mathsf{WOTS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{WOTS}}(\mathsf{pp}_{\mathsf{WOTS}})$. Output $\mathsf{vk}^{\mathsf{QOTS}} = \mathsf{vk}_{\mathsf{WOTS}}$ and $\mathsf{sk}^{\mathsf{QOTS}} = \mathsf{sk}_{\mathsf{WOTS}}$.
- $\mathsf{Sign}_{\mathsf{QOTS}}(\mathsf{pp}_{\mathsf{QOTS}}, \mathsf{sk}^{\mathsf{QOTS}}, a) \to \tau$. Check that $a \in \mathcal{A}^d$ and output

$$\tau \leftarrow \mathsf{Sign}_{\mathsf{WOTS}}(\mathsf{pp}_{\mathsf{WOTS}}, \mathsf{sk}_{\mathsf{WOTS}}, a).$$

- $\mathsf{Verify}_{\mathsf{QOTS}}(\mathsf{pp}_{\mathsf{QOTS}}, \mathsf{vk}^{\mathsf{QOTS}}, a, \tau) \to \{0, 1\}$. Reject malformed inputs. Otherwise, output

$$\mathsf{Verify}_{\mathsf{WOTS}}(\mathsf{pp}_{\mathsf{WOTS}}, \mathsf{vk}_{\mathsf{WOTS}}, a, \tau).$$

Correctness. For every valid $a \in \mathcal{A}^d$ and every honestly generated key pair, an honestly generated QOTS signature τ verifies with probability 1. Indeed, signing and verification apply WOTS to the same word a. For the basic hash-chain construction, each component satisfies

$$H^{2^\omega - 1 - e_i}\big(H^{e_i}(x_i)\big) = H^{2^\omega - 1}(x_i) = y_i.$$

Thus, correctness follows directly from WOTS correctness, as completing the hash chain from each signature component yields its corresponding public endpoint.

One-time unforgeability on selected symbols. An adversary receives $\mathsf{vk}^{\mathsf{QOTS}}$ and may request a qShare on one chosen word a. It wins by producing (a^*, τ^*) such that it passes verification $\mathsf{Verify}_{\mathsf{QOTS}}(\mathsf{pp}_{\mathsf{QOTS}}, \mathsf{vk}^{\mathsf{QOTS}}, a^*, \tau^*) = 1$ and $a^* \neq a$. QOTS is one-time unforgeable on selected symbols if every probabilistic polynomial-time adversary wins with probability negligible in κ.

Security argument. Any successful QOTS forgery immediately yields a WOTS forgery. A reduction embeds its WOTS verification key as $\mathsf{vk}^{\mathsf{QOTS}} = \mathsf{vk}_{\mathsf{WOTS}}$, answers the sole QOTS signing query by requesting a WOTS signature on a, and outputs (a^*, τ^*). This is a valid signature on a different word since $a^* \neq a$. Consequently, QOTS inherits one-time unforgeability from WOTS, with the same success probability.

The role of the checksum parallels that of the unrevealed preimages in Lamport-OTS. An adversary can advance a revealed hash-chain value, thereby increasing its encoded symbol. However, any distinct word either decreases a data symbol or, if all data symbols only increase, decreases the checksum. In either case, some encoded symbol decreases, requiring a chain value that cannot be obtained by merely advancing the revealed values. The underlying WOTS security theorem formalizes this argument under its hash-function assumptions.

Scope and one-time restriction. A qShare authenticates only $a \in \mathcal{A}^d$. In the outer QUASI construction, this word consists of the codeword symbols selected by mapping π_i. Certification of the complete statement is provided by the quorum construction. Each QOTS key pair must be used for at most one signing query. Post-quantum security follows from the same reduction when the underlying WOTS instantiation is secure against quantum polynomial-time adversaries.