Fast Post-Quantum Ring Signature from Power Residue PRFs with QROM Security

XINYU ZHANG, Monash University, Australia
ZIYI LI, Hong Kong Polytechnic University, China
RON STEINFELD, Monash University, Australia
RAYMOND K. ZHAO, ExeQuantum, Australia
JOSEPH K. LIU, Monash University, Australia
TSZ HON YUEN, Monash University, Australia

Ring signatures enable a user to sign anonymously on behalf of a group, providing a fundamental building block for privacy-preserving applications such as anonymous credentials and private transactions. Existing post-quantum constructions, however, face a persistent tension between efficiency and rigorous quantum security: practical schemes are typically analysed only in the classical random oracle model (ROM), whereas schemes with proofs in the quantum random oracle model (QROM) either incur multi-megabyte signatures, lack implementations, or rely on reductions too loose to support concrete parameter selection.

We present PegaRing, the first post-quantum ring signature to combine practical performance with concretely parameterised security in QROM. At the 128-bit security level and a ring size of 1024, PegaRing produces 28KB signatures, reducing signature size by a factor of 88 relative to the smallest prior concretely parameterised QROM-secure construction. Our implementation signs in 4.325 ms and verifies in 3.074 ms. Compared with the state-of-the-art VOLE-in-the-head ring signature, which is proven secure only in classical ROM, PegaRing reduces signing and verification latency by factors of 2.65 and 3.80, respectively, while providing security in QROM. Finally, when using the first message-dependent Fiat-Shamir challenge as the boundary between offline and online computation, PegaRing's online signing phase takes 0.018 ms, compared with 7.611 ms for VOLE-in-the-head construction, achieving a reduction of more than 400 times. These results substantially narrow the gap between rigorous quantum security and practical performance for post-quantum ring signatures and make QROM-secure anonymous authentication feasible for various anonymity sets.

1 Introduction

Ring signatures allow a user to authenticate a message with respect to an ad hoc set of public keys while hiding which secret key was used [49]. Unlike group signatures [16], they require no group manager, enrolment process, or cooperation from the other members of the ring. This setup-free anonymity has made ring signatures and their variants core building blocks for privacy-preserving systems. The initial motivation was to facilitate anonymous disclosure of secrets (also known as whistleblowing), where a recipient can verify that information came from one member of an authorised group without learning the source [49]. Linkable or unique variants of ring signatures are also widely deployed in anonymous payment [46] and anonymous electronic voting systems [53].

However, many existing ring signatures rely on the hardness of integer factorisation, the discrete logarithm problem, or related assumptions that can be broken by Shor's quantum algorithm [52]. Given the broad range of privacy-oriented applications, it is imperative to develop post-quantum ring signatures that are both concretely efficient and secure against quantum adversaries. Research in this direction has progressed rapidly, with constructions based on quantum-resistant assumptions such as lattice [10, 27, 42, 43, 59], isogeny [10, 13], and symmetric-key-primitives [17, 22, 34, 58].

Authors' Contact Information: Xinyu Zhang, xinyu.zhang2@monash.edu, Monash University, Melbourne, Australia; Ziyi Li, zycal.li@polyu.edu.hk, Hong Kong Polytechnic University, Hong Kong, China; Ron Steinfeld, ron.steinfeld@monash.edu, Monash University, Melbourne, Australia; Raymond K. Zhao, raymond@exequantum.com, ExeQuantum, Melbourne, Australia; Joseph K. Liu, joseph.liu@monash.edu, Monash University, Melbourne, Australia; Tsz Hon Yuen, john.tszhonyuen@monash.edu, Monash University, Melbourne, Australia.

Table 1. Comparison of Post-Quantum Ring Signatures for a Ring of Size 1,024 at the 128-Bit Security Level.

Scheme	Sec. Model	Size (KB)	Sign [Online Time] (ms)	Verify (ms)	Assumption
Concretely Parameterised QROM Constructions					
Derler et al. [22]	QROM	2,463	-	-	LowMC
Feng et al. [29]*	QROM	3,156	-	-	LowMC
PegaRing[a]	QROM	28	4.325 [0.018]	3.074	Power Residue PRF
Classical ROM Constructions					
Faafl [10][a]	ROM	33	13,897	2,752	MSIS, MLWE
DualRing-LB [59][a]	ROM	28	$100\times10^{3\dagger}$	$21\times10^{3\dagger}$	MSIS, MLWE
ATFE Ring [11][b]	ROM	27.7	$\sim$ 21,712	$\sim$ 21,712	ATFE
MPCitH Ring [34][c]	ROM	56	210	$\sim$ 210	LowMC
DualRing-PRF [62][a]	ROM	71	$123\times10^{3\dagger}$	$94\times10^{3\dagger}$	Power Residue PRF
STARK Ring [58][d]	ROM	29	160 [48]	112	Rescue Prime
VOLEitH Ring [17][a]	ROM	5.03	11.47 [7.611]	11.686	AES
PegaRing[a]	ROM	28	3.747 [0.014]	2.452	Power Residue PRF

* Traceable ring signature.
† Python-based running time.
[a] Performance are measured from the same machine with a 3.6GHz Intel Core i7-12700KF CPU.
[b] Performances are estimated from [11, Table 3] with 2.4 GHz Quad-Core Intel Core i5.
[c] Performances are from [34, Table 2], with two Intel XEon CPU E5-2695 2.10GHz server.
[d] Performances are from [58, Table 5] with i9-12900k CPU.

Nevertheless, relying on a quantum-resistant hardness assumption alone is insufficient to establish security against quantum adversaries when the security proof relies on random oracles, as is the case for many post-quantum ring signatures. In practice, random oracles are instantiated by publicly specified hash functions, which a quantum adversary can evaluate on superposition of inputs. Security proofs must therefore remain valid in the quantum random oracle model (QROM), which permits such superposition queries. As shown in [2, 12, 60], security proofs in the classical random oracle model (ROM) do not, in general, automatically extend to the QROM.

The Efficiency-Security Gap. Existing post-quantum ring signature schemes show a persistent separation between practical efficiency and rigorous QROM security. In particular, practical lattice- and symmetric-key-based constructions achieve signatures of a few kilobytes with fast operations, but their security analyses are generally restricted to the classical ROM. Conversely, constructions with concrete QROM-based parameters remain far from practical. The ring signature scheme with QROM instantiation of Derler et al. [22] produces 2,463 KB signatures while the QROM-secure traceable ring signature of Feng et al. [29] produces 3,156 KB signatures for the ring size of 1,024 and neither are implemented. PURSE [41] provides QROM analysis of their unique ring signature, but the authors explicitly state that the losses in their QROM analysis are too large to guide concrete parameter selection. Several other QROM-based constructions [15, 19, 26, 30, 36] provide only asymptotic analyses or conceptual results without concrete parameters or implementations. Consequently, no existing construction simultaneously provides practical communication and computation together with concretely parameterised QROM security.

1.1 Contributions

We bridge the efficiency-security gap by designing the first post-quantum ring signature that achieve both practical performance and concretely parameterised security in QROM. At the core of our approach is the novel observation that power residue PRF relations inherently induce an oblivious linear evaluation (OLE) structure. This structure lets us prove knowledge of a PRF key directly, without relying on a general-purpose proof system. We first use this observation to construct Pegasus, a compact three-move commit-and-open key identification protocol, and then develop the set-membership mechanism to obtain PegaRing. Our contributions are as follows.

- **More efficient signature scheme based on power residue PRFs.** As a stepping stone toward PegaRing, we construct Pegasus (**P**ower r**E**sidue PRF-based si**G**n**A**ture **S**cheme with q**U**antum-ROM **S**ecurity), an efficient signature scheme from power residue PRFs that exploits the OLE structure. We prove the existential unforgeability under chosen message attacks of Pegasus in both the ROM and QROM. At the 128-bit security level, Pegasus achieves 5,610-byte signatures with 1.55 ms signing and verification in QROM. Compared with PorcRoast [9], a prior power residue PRF-based construction whose security was established only in the ROM, Pegasus reduces signature size by a factor of 1.5 and public key size by a factor of eight while provides QROM security. As a standalone QROM-secure signature from power residue PRFs, this result may be of independent interest.
- **PegaRing: a three-move post-quantum ring signature with QROM security.** Building on the DualRing framework [59], we construct a new set-membership for the OLE correlation underlying Pegasus. The prior power residue PRF-based instantiation of DualRing [62] introduces an additional cut-and-choose step, significantly increasing ring signature size and running time. We show how to eliminate this step while preserving the three-move commit-and-open structure for our QROM Fiat-Shamir analysis. To further reduce communication and computation, we generalise the existing β-approximate PRF relation to arithmetic progression lists. Unlike the standard setting, where list elements are sampled independently and uniformly, elements in an arithmetic progression are highly correlated, so the prior analysis does not apply directly. We first bound the distribution of power residue symbols over fixed gap arithmetic progressions and then use Rényi divergence to transfer the bound from ideal distribution to the real one. This analysis may be of independent interest. For the resulting PegaRing scheme, we prove unforgeability against insider corruption and anonymity under full key exposure in both the ROM and the QROM.
- **Implementation and concrete evaluation.** We implement both Pegasus and PegaRing and evaluate their signature sizes, signing and verification latencies. Their underlying three-move protocols admit a straightforward online-offline decomposition: once the public key (resp. the ring) is fixed, the signer can compute the first round message before the message is known. We separately measure the time of offline and online latency in our implementations. For a ring of size 1,024, PegaRing produces 28 KB signatures, 88× smaller than previous QROM-secure post-quantum ring signature with concrete parameters. Furthermore, our scheme signs in 4.325 ms (with 0.018 ms online signing) and verifies in 3.074 ms. We compare against both efficient ROM-only schemes and concretely parameterised QROM schemes in Table 1.[1] Notably, when compared with the state-of-the-art ROM-only ring signature construction based on VOLE-in-the-head [17], while our signature size is approximately five times larger, the online signing time of PegaRing is more than 400× faster while providing QROM security.[2] Together, fast performance and

[1]We benchmark PegaRing, Faafl [10], DualRing-LB [59], DualRing-PRF [62] and VOLEitH Ring [17] on the same platform. For the remaining schemes, we use the results reported in the paper as no complete executable implementation was publicly available to us.
[2]The original evaluation of VOLEitH Ring does not include an online-offline decomposition, we obtain the online latency by assigning operations before the first Fiat-Shamir challenge to offline, while the remaining computations are assigned to online phase. This decomposition is merely for comparison purpose and does not represent a separately optimised online-signing implementation.

concretely parametrised QROM security make PegaRing a practical candidate for ring signature-based real-world applications such as privacy-preserving reporting [28] and anonymous authenticated key exchange [47].

1.2 Technical Overview

We begin by establishing the background of our work: the key identification problem of the Legendre and power residue PRFs.

Key Identification. For an odd prime p, the Legendre PRF (and its generalization named t-th power residue PRF) is conjectured to be a pseudo-random function family [21], indexed by a (secret) key $K \in \mathbb{F}_p$, which on input a field element $a \in \mathbb{F}_p$, outputs a $\log_2(t)$-bit integer indicating the t-th power residuosity of $(K+a)$, denoted as $\mathcal{L}_K^t(a) = \mathcal{L}_0^t(K+a) \in \mathbb{Z}_t$.

To uniquely identify K, the statement (public key) must contain multiple PRF evaluations:

$$pk = (pk_0, \ldots, pk_{L-1}) = (\mathcal{L}_K^t(I_0), \ldots, \mathcal{L}_K^t(I_{L-1})),$$

where $\mathfrak{I} = (I_0, \ldots, I_{L-1}) \in \mathbb{F}_p^L$ is a publicly defined list. The (relaxed) key identification problem of the Legendre and power residue PRFs is then described as follows: given a public key consisting of L PRF outputs, prove the knowledge of a value K such that for $M \leq L$ uniformly random indices $(\bar{\ell}_1, \ldots, \bar{\ell}_M)$, where $\bar{\ell}_e \xleftarrow{\$} [0, L-1]$, it holds that $\mathcal{L}_K^t(I_{\bar{\ell}_e}) = pk_{\bar{\ell}_e}$ for all $e \in [M]$.

Existing works [9, 61] exploit the multiplicative homomorphism of the PRFs to avoid directly evaluating them inside zero-knowledge proofs. Specifically, for values $o_e = (K + I_{\bar{\ell}_e})r_e$, where each $r_e \xleftarrow{\$} \mathbb{F}_p$ is uniformly chosen by the prover, it can be shown that

$$\begin{aligned}\mathcal{L}_0^t(o_e) &= \mathcal{L}_0^t((K + I_{\bar{\ell}_e})r_e) \\ &= \mathcal{L}_0^t(K + I_{\bar{\ell}_e}) + \mathcal{L}_0^t(r_e) \\ &= pk_{\bar{\ell}_e} + \mathcal{L}_0^t(r_e).\end{aligned}$$

Generic zero-knowledge proofs (e.g., MPC-in-the-head) are then employed to ensure that for all $e \in [M]$, o_e is correctly computed using K, $I_{\bar{\ell}_e}$, and r_e. Furthermore, the values K, r_e and $\mathcal{L}_0^t(r_e)$ must be committed before the prover learns the challenge indices $(\bar{\ell}_e)_{e\in[M]}$.

First Observation: OLE Structure of the Relation. Our key insight arises from a close inspection of the structure of $(o_e)_{e\in[M]}$. Specifically, for the verifier's challenge $\bar{\ell}_e \in [0, L-1]$, the honestly computed o_e takes the form of $(K + I_{\bar{\ell}_e})r_e$ for all $e \in [M]$. Expanding this expression yields $o_e = Kr_e + I_{\bar{\ell}_e} r_e$. This observation allows us to map $(o_e, I_{\bar{\ell}_e})$, which are known to the verifier, and (K, r_e), which are known to the prover, to an Oblivious Linear Evaluation (OLE) correlation.

Recall that in an OLE protocol, the sender holds the coefficients (u, v) and the receiver holds the challenge c. Then, at the end of the OLE, the sender learns nothing, while the receiver learns $s = uc + v$. Based on the previous observation, we let the prover be the OLE sender and the verifier be the OLE receiver. Consequently, the key identification problem of the Legendre and power residue PRFs can be solved by M invocations of an OLE protocol, where in each OLE execution, the sender's inputs are $u_e = r_e$ and $v_e = Kr_e$, and the receiver's input is $c_e = I_{\bar{\ell}_e}$. Hence, at the end of the OLE protocol, the receiver obtains $o_e = u_e c_e + v_e = r_e I_{\bar{\ell}_e} + Kr_e$, while the sender learns nothing. However, designing an OLE protocol based on symmetric key primitives is challenging, since OLE typically uses oblivious transfer (OT) as a building block, which requires public key encryption.

Second Observation: Connecting Key Shares with Public List. Our second observation is inspired by the core technique presented in SoftSpokenOT [50] and VOLEitH [5], which shows how to realize vector OLE (VOLE) using a tree-PRG based vector commitment scheme. However, the compiler proposed in [5] *cannot* be directly applied to our setting. In these tree-PRG based VOLE protocols, the sender holds a list $\mathbf{u} = (u_0, \ldots, u_{L-1})$ and the receiver has a challenge c' that is an index in $\{1, 2, \ldots, L\}$. Then, the sender defines

$$u' = \sum_{\ell=0}^{L-1} u_\ell, \quad v' = \sum_{\ell=0}^{L-1} u_\ell \cdot -(\ell + 1).$$

The receiver's VOLE output is :

$$s' = u'c' + v' = \sum_{\ell=0, \ell+1 \neq c'}^{L-1} u_\ell (c' - (\ell + 1)),$$

which can be computed by having the sender reveal $(L-1)$-out-of-L elements in $\mathbf{u}$.

In our case, the prover has inputs $u_e = r_e$ and $v_e = Kr_e$ which obviously do not satisfy the relation required by the compiler. To address this, we first consider split r_e to additive shares such that $r_e = \sum_{\ell=0}^{L-1} r_{e,\ell}$. This unfortunately does not solve our problem since there is still a mismatch between our $v_e = Kr_e = K \sum_{\ell=0}^{L-1} r_{e,\ell}$ and $v'_e = \sum_{\ell=0}^{L-1} u_\ell \cdot -(\ell + 1)$. Moreover, the verifier's challenge $c = I_{\bar{\ell}_e}$ is a random field element in the pre-defined list $\mathfrak{I}$, which does not match with $c' \in \{1, 2, \ldots, L\}$ either.

We solve the problem by a novel idea, which interprets the field elements in $\mathfrak{I}$ as *secret key shares*. It is obvious that the sum of elements in $\mathfrak{I}$ does not equal to the secret key K. Hence, for each $e \in [M]$, the prover needs to send a key adjustment before it learns verifier's challenges. The interpretation allows us to align the prover's inputs and verifier's challenges with those of VOLE sender and receiver in the tree-PRG based protocol. As a result, the prover (signer) can commit to its secret inputs using a tree-PRG-based vector commitment, then open all-but-one shares together with the key adjustment, allowing the verifier to compute o_e and verify the relation. These observations allow us to eliminate the need to prove the correct computation of o_e using a generic zero-knowledge proof (e.g., MPCitH), leading to a standard three-move commit-and-open (C&O) Σ-protocol for the key identification problem of Legendre and power residue PRFs.

PegaRing: Efficient Set Membership Proof without Cut-and-Choose. We develop an efficient set membership proof for Legendre and power residue PRFs using the DualRing framework [59]. DualRing can be seen as a compiler that transforms a special type of Σ-protocols, called the Type-T* canonical identification schemes, into OR-compositions. The key requirement of DualRing is that the verification algorithm V can be decomposed into two algorithms V_1 and V_2 such that $V = V_1 \odot V_2$ for some group operation $\odot$. More importantly, V_1 must be additive or multiplicative homomorphic. The benefit of using the DualRing compiler is that the resulting ring signature consists of a single response and n challenges for a ring of size n. In the post-quantum setting, since the challenge is typically much smaller than the response, applying DualRing compiler can lead to schemes with notably short signatures for small to medium rings.

Although we have a three-move identification scheme as described in the previous subsection, the DualRing compiler cannot be directly applied to obtain a secure set membership proof, as it is not designed for C&O Σ-protocols. The core issue is that the verifier must know the real signer's challenge to recompute the response, which reveals the signer's identity and breaks anonymity. To solve the problem, we modify the signing protocol so that the verifier computes the response using a *mutual challenge*, defined as the sum of all ring member's challenges. However, it is obvious that the response computed from the mutual challenge by the verifier differs from the real signer's response, violating the correctness. The original solution of [62] introduces an additional cut-and-choose round, leading to substantially large

parameters. In contrast, we propose a method that avoids the cut-and-choose and preserves its three-move structure as follows.

Let n be the size of the ring. Then, the challenge of i-th ring member for each $e \in [M]$ is an index $\bar{\ell}_e^{(i)} \in [0, L-1]$, indicating the position of the checked public key symbol $pk_{\bar{\ell}_e^{(i)}}^{(i)}$. The mutual challenge is defined as:

$$\bar{\ell}_e = \sum_{i=1}^{n} \bar{\ell}_e^{(i)} \pmod{L}.$$

Our key insight is that the distance between the mutual challenge $\bar{\ell}_e$ and the real signer's challenge $\bar{\ell}_e^{(\pi)}$ is fixed in the commitment phase. In fact, it is exactly the sum of the challenges from all non-signers:

$$\bar{\ell}'_e = \bar{\ell}_e - \bar{\ell}_e^{(\pi)} = \sum_{i=1, i\neq\pi}^{n} \bar{\ell}_e^{(i)} \pmod{L}.$$

This observation enables the real signer to pre-compute all possible share adjustments by iterating over pairs of the mutual challenge and the real signer's challenge. Since their distance is fixed, there are exactly L such pairs. These share adjustments are committed using a Merkle tree, together with the tree-PRG-based vector commitment in the original Σ-protocol. After receiving the verifier's challenge, the signer opens all-but-one shares according to the mutual challenge and the corresponding share adjustment in the Merkle tree. By adopting generalised arithmetic progression lists, we reduce the size of the Merkle tree, yielding a three-move commit-and-open set membership proof.

Optimising the performance of Pegasus and PegaRing. Tree-PRG-based vector commitment can be improved by combining multiple trees into a single large tree, known as batch all-but-one vector commitment (BAVC)[4]. By sharing the intermediate nodes, BAVC reduces the proof size for opening all-but-M leaves compared to handling trees individually. A recent optimisation of BAVC [14] combines circular correlation robust (CCR) hash with the half-tree technique [35], significantly reduces commitment generation and verification times. However, CCR hashing is typically instantiated with fixed-key AES, which is acceptable in the classical ROM but complicates security proofs in the QROM, since AES is modelled as a random permutation rather than a random function.

To avoid relying on the quantum random permutation model, we separate the correlated BAVC into two components: `BAVC.TreeGen` and `BAVC.LeafCom`. In QROM version of our protocols, we keep the CCR-hash and half-tree techniques for `BAVC.TreeGen`, while instantiating `BAVC.LeafCom` with standard random oracles.[3] On the other hand, in the ROM version, we fully adopt the correlated BAVC in [14]. This modularisation then allows us to follow the the standard technique of [23], resulting in a tight bound on the QROM security of our schemes. Another improvement to the vector commitment is the use of the *Octopus opening*, originally introduced to minimise the number of nodes revealed when opening multiple leaves in a Merkle tree, which can be naturally adapted to the BAVC setting.

2 Related Work

Post-quantum ring Signatures in ROM.. A substantial body of work constructs post-quantum ring signatures and their variants from lattice assumptions [10, 27, 42, 43, 59], isogenies [10, 13], and symmetric key primitives [17, 34, 39, 58]. These constructions provide concrete parameters, compact signatures, and in several cases, implementations. However, although their underlying security assumptions are believed to resist quantum attacks, their ring signature security

[3]This is eventually similar to the tree-PRG-based vector commitment used in [20], but we use batched version (BAVC), while they use the single tree.

proofs apply the Fiat-Shamir transform only in the classical ROM and do not establish security when the random oracle is queried in the superposition.

Among lattice-based constructions, DualRing-LB [59] instantiates the DualRing framework from M-LWE and M-SIS and was reported as the shortest lattice-based ring signature for ring sizes between 4 and 2000. While PegaRing has a larger constant overhead in the communication complexity, our linear term is 8 bytes smaller than that of (optimised) DualRing-LB. Consequently, as the ring size grows, PegaRing achieves better overall efficiency (as shown in Table 1).

MatRiCT+ [27] is primarily a complete Ring Confidential Transaction protocol. The techniques introduced in their work also yield an estimated 18 KB ring signature for ring size of $1,024$, approximately 10 KB smaller than PegaRing. However, MatRiCT+ produces public key of size 3 KB for rings of size+ $1,024$, which is 6 times larger than that of PegaRing. For context, the MatRiCT+ implementation generates and verifies RingCT transaction with 2 input/output accounts in 261 ms and 8 ms at anonymity set of size 50. These figures include functionalities beyond standalone ring signature and are therefore not directly comparable with PegaRing.

Symmetric key primitives-based ring signatures and their variants have been constructed using MPC-in-the-head, STARK, or VOLE-in-the-head proof systems [17, 34, 39, 58, 62]. Among concretely efficient constructions, the VOLE-in-the-head construction [17] achieves the smallest signatures, where its AES-128 instantiation produces 5.03 KB signature for a ring of size $1,024$. The signature size grows sublinearly with the ring size, providing scalable communication, although signing and verification computation remains at least linear in the ring size and their scheme is proven in the classical ROM only.

Post-quantum ring signatures in the QROM.. Prior post-quantum ring signatures with concrete QROM parameters remain expensive. Derler et al. [22] obtain QROM security using the Unruh transform, but their smallest reported parameter set produces a $2,463$ KB signature at ring of size 1,024 and has no reported implementation timings. Feng et al. [29] construct a traceable ring signature in the QROM whose symmetric key instantiation produces a $3,156$ KB signature at the same ring size, again without implementation timings.

Other QROM-secure results primarily address additional security properties for ring signatures or theoretical feasibility. Haque and Scafuro [36] construct threshold ring signatures, Chung et al. [19] construct isogeny-based accountable ring signatures, and Chatterjee et al. [15] develop new post-quantum definition and generic construction. These works do not provide directly comparable benchmarks for ring signatures. PURSE [41] is an implemented lattice-based unique ring signature with a QROM analysis. However, the security losses in their QROM argument are too large to guide parameter selection.

Independent generic QROM analyses. Recent independent work by Beckman and Majenz [6] establishes QROM security bounds for two broad classes of ring signatures, including AOS-based ring signatures such as MayoRS [38] and Erebor [13], and RPSF-based ring signature such as Gandalf [33]. However, for AOS-based ring signatures, the QROM bound is non-tight and incurs a loss exponential in the ring size. For Gandalf, while the resulting QROM bound remains close to its classical security analysis, the scheme itself is explicitly designed for small rings. Their concrete parameters are provided for rings of at most 26 members since the hardness of the underlying R-ISIS problem decreases with the maximum ring size. While an extrapolation gives a signature size of approximately 621 KB at ring of size 1024, which is 22× larger than PegaRing, this does not constitute a valid concrete parameter set at the ring size. Hence, we do not treat their originally reported performance figures as directly comparable benchmarks for PegaRing, since concrete parameters targeting 128-bit QROM security at ring of size $1,024$ are unavailable.

3 Preliminaries

Notations. We denote security parameter by λ and a set of negligible functions by $\mathsf{negl}(\lambda)$ where a function $f(\lambda)$ is negligible if $f(\lambda) = \lambda^{-\omega(1)}$. We use $\mathbb{F}_p$ to denote the prime field for a large prime p where $p = O(2^\lambda)$. For positive integer N and real $q \in [0,1]$, we denote by $\mathfrak{B}(N,q)$ the binomial distribution with N trials and success probability q. We define $[n] = \{1,\ldots,n\}$ and $[0,n-1] = \{0,1,\ldots,n-1\}$. For a bit string h and an integer l, $h[:l]$ denotes the first l-bit of x. We use bold lower case letters (e.g., $\mathbf{v}$) to represent vectors and bold upper case letters (e.g., $\mathbf{V}$) to represent vector of vectors. Finally, 'PPT' stands for probabilistic polynomial time. For better readability, we defer additional notations used in security proofs to Section A.1.

ROM vs. QROM In this work, we prove the security of proposed schemes in both the random oracle model (ROM) and the quantum random oracle model (QROM). We defer the formal definitions and existing lemmas used in our analysis to Section A.2, and present the QROM database property and quantum transition capacity in Section A.3.

3.1 Legendre and Power Residue PRFs

The *Legendre PRF* relies on the conjectured pseudo-randomness of the Legendre symbol with a secret shift [8]. Specifically, on input $a \in \mathbb{F}_p$, the Legendre symbol outputs $\{-1,0,1\}$, depending on whether a is a square modulo p:

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{if } a = b^2 \pmod p \text{ for some } b \in \mathbb{F}_p^* \\ 0, & \text{if } a = 0 \pmod p \\ -1, & \text{otherwise} \end{cases}$$

Then, the Legendre PRF is defined as:

$$\mathcal{L}_0^2(a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{a}{p}\right)\right)\right\rfloor \in \mathbb{Z}_2$$

Namely, the PRF outputs 1 if a is a quadratic non-residue (QNR) modulo p and 0 otherwise.

The *t-th power residue PRF* generalizes the Legendre PRF, which uses higher power residue symbols instead of quadratic residue symbols to increase the PRF throughput [21]. Specifically, for an integer t and a prime p such that $t|(p-1)$, the computation of the t-th power residue symbol $a \mapsto a^{(p-1)/t} \pmod p$ yields an output of $\log_2(t)$-bits, in contrast to the single-bit output obtained by the Legendre PRF. Formally, the t-th power residue PRF is defined as follows:

Definition 1 (t-th Power Residue PRF)**.** Let p be an odd prime and t be an integer such that $p \equiv 1 \pmod t$. Let $g \in \mathbb{F}_p^*$ be the generator. We define the t-th power residue symbol $\mathcal{L}_K^t(\cdot) : \mathbb{F}_p \to \mathbb{Z}_t$ with the hidden shift K as

$$\mathcal{L}_K^t(a) = \begin{cases} i, & \text{if } (a+K)/g^i \equiv z^t \pmod p \text{ for some } z \in \mathbb{F}_p^* \\ 0, & \text{if } (a+K) \equiv 0 \pmod p \end{cases}$$

Both Legendre and power residue PRFs have multiplicative homomorphism, that is $\mathcal{L}_0^t(a \cdot b) = \mathcal{L}_0^t(a) + \mathcal{L}_0^t(b)$ for non-zero $a, b \in \mathbb{F}_p^*$.

To simplify the notation, we extend the definition of the t-th power residue PRF[4] to be 'list-wise'. Given a list $\mathfrak{I} = \{I_0, \ldots, I_{L-1}\} \in \mathbb{F}_p^L$, we denote the PRF evaluations as

$$\mathcal{L}_K^t(\mathfrak{I}) = (\mathcal{L}_K^t(I_0), \ldots, \mathcal{L}_K^t(I_{L-1})) \in \mathbb{Z}_t^L.$$

We define the NP-relations for the t-th power residue PRF with respect to two different lists. The first list $\mathfrak{I} = (I_0, \ldots, I_{L-1})$ is defined the same as in LegRoast [9], where each element is uniformly random chosen from $\mathbb{F}_p$. The second list $\mathfrak{I}' = (\mathfrak{I}'_0, \ldots, \mathfrak{I}'_{z-1})$ is defined as, for each $j \in [0, z-1]$, $\mathfrak{I}'_j = (a_j, a_j + b, \ldots, a_j + (c-1)b)$, where $b \in \mathbb{F}_p^*$ is a fixed non-zero element, a_j is chosen uniformly random from the field, and $L = z \times c$.

Definition 2 (t-th Power Residue PRF Relation)**.** For an odd prime p, a positive integer t with $t|(p-1)$, and a list $\tilde{\mathfrak{I}} \in \{\mathfrak{I}, \mathfrak{I}'\}$ of L field elements, we define the t-th power residue PRF relation $R_{\mathcal{L}^t}$ as

$$R_{\mathcal{L}^t} = \{(\mathcal{L}_K^t(\tilde{\mathfrak{I}}), K) \in \mathbb{Z}_t^L \times \mathbb{F}_p | K \in \mathbb{F}_p\}.$$

In order to enhance the efficiency of t-th power residue PRFs based signature schemes, Beullens et al. [9] introduced a relaxed PRF relation, which can be extended with respect to the arithmetic progression list $\mathfrak{I}'$ as follows:

Definition 3 (β-approximate PRF relation)**.** Given $\beta \in [0, 1]$, an odd prime p, a positive integer $t|(p-1)$, and a list $\tilde{\mathfrak{I}} \in \{\mathfrak{I}, \mathfrak{I}'\}$ of L field elements, we define the β-approximate PRF relation $R_{\beta\mathcal{L}^t}$ as

$$R_{\beta\mathcal{L}^t} = \{(s, K) \in \mathbb{Z}_t^L \times \mathbb{F}_p | \exists a \in \mathbb{Z}_t : d_H(s + \mathbf{a}, \mathcal{L}_K^t(\tilde{\mathfrak{I}})) \leq \beta L\}$$

where $\mathbf{a} = (a, \ldots, a)$ and $d_H(\cdot, \cdot)$ denotes the Hamming distance.

For uniformly random list $\mathfrak{I}$, the β-approximate PRF relation is as hard as the t-th power residue PRF relation, which is shown in [9, Theorem 1]. We prove a similar result with respect to the arithmetic progression list in Section 5.1. Moreover, we address that the t-th power residue PRFs in our scheme is only used for *key generation*. The security of the signing and verification relies solely on the random oracle assumptions. Currently, no quantum algorithm is known that can recover the secret key from public key symbols derived from the PRFs. For completeness, we present a short survey of existing attacks against Legendre and power residue PRFs in Section B.

3.2 Vector Commitment Schemes

3.2.1 Batch all-but-one vector commitment (BAVC). Batch all-but-one vector commitment (BAVC) is a generalisation of the tree-PRG-based vector commitment [5], which allows all-but-M commitments to be opened efficiently. At a high-level, tree-PRG-based vector commitment expands a single master seed $sd_0 \in \{0,1\}^\lambda$ into $L = 2^d$ seeds using a tree-like structure. Concretely, given a length-doubling pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$, the master seed sd_0 (layer 0) is expanded into two children seeds (layer 1):

$$(sd_{1,0}, sd_{1,1}) \leftarrow \mathsf{PRG}(sd_0).$$

These layer 1 seeds are then used to generate next level of seeds in layer 2:

$$(sd_{1,0,0}, sd_{1,0,1}) \leftarrow \mathsf{PRG}(sd_{1,0}),$$
$$(sd_{1,1,0}, sd_{1,1,1}) \leftarrow \mathsf{PRG}(sd_{1,1}).$$

[4]We will use the term t-th power residue PRF from now on to refer to both the Legendre PRF and higher power residue PRFs. Specifically, as when $t = 2$, it corresponds to the Legendre PRF, and when $t > 2$, it corresponds to higher power residue PRFs.

The process continues until the bottom level with L leaf nodes is reached. Each leaf node is then committed using a PRG and collision-resistant hash function (CRHF) $H_T : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$, which generates λ-bit message and 2λ-bit commitment. Finally, the commitments are aggregated using another CRHF $H_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^{2\lambda}$, where both hashes are modelled as independent random oracles. To avoid cumbersome notation, we use the generic symbol $\mathcal{H}(\cdot)$ when referring to random oracles in BAVC.

In the original tree-PRG based vector commitment, revealing $L-1$ messages requires only $d = \log(L)$ nodes, similar to the authentication path (with respect to the unopened leaf) in a Merkle tree. Since a signature protocol typically needs multiple trees (say M), BAVC combines them into one big tree and interleaves their leaf nodes, hence creating overlaps when opening all-but-M leaves and reducing the overall opening size. In this work, we separate the tree generation and leaf commitment for BAVC to facilitate cleaner security proofs of our scheme in the (Q)ROM. In particular, the BAVC we use contains the following five algorithms:

$$(\mathtt{Setup}^{\mathcal{H}}, \mathtt{TreeGen}^{\mathcal{H}}, \mathtt{LeafCom}^{\mathcal{H}}, \mathtt{Open}^{\mathcal{H}}, \mathtt{Verify}^{\mathcal{H}}),$$

where `Setup` generates public parameters, `TreeGen` expands a master seed into a binary tree using a length-doubling PRG, $\mathtt{LeafCom}^{\mathcal{H}}$ generates random messages and commitments using hash functions (modelled as random oracles), $\mathtt{Open}^{\mathcal{H}}$ opens all-but-M nodes of the tree, and $\mathtt{Verify}^{\mathcal{H}}$ verifies the opened messages and commitments. The syntax of BAVC is defined in Definition 4.

Definition 4 (BAVC Syntax)**.** Let $\mathcal{H}$ denote a set of random oracles. A batch all-but-one vector commitment (BAVC) over message space $\mathcal{M}$ in the random oracle model consists of the following PPT algorithms:

- $\mathsf{pp} \leftarrow \mathtt{Setup}^{\mathcal{H}}(1^\lambda)$: On input the security parameter 1^λ, the setup algorithm outputs public parameters pp. In the following, we omit pp from the input of algorithms when clear from context.
- $(\mathtt{decom}, \mathbf{L}) \leftarrow \mathtt{TreeGen}^{\mathcal{H}}(\mathsf{pp})$: On input the public parameters pp, the algorithm outputs opening information decom together with a set of leaf nodes
$$\mathbf{L} = (l_{e,0}, \ldots, l_{e,L-1})_{e\in[M]}.$$
- $(\mathtt{com}, \mathbf{R}) \leftarrow \mathtt{LeafCom}^{\mathcal{H}}(\mathbf{L}, \mathsf{pp})$: On input $\mathbf{L}$ and pp, the leaf commitment algorithm outputs a commitment com and messages
$$\mathbf{R} = (r_{e,0}, \ldots, r_{e,L-1})_{e\in[M]},$$
where $r_{e,\ell} \in \mathcal{M}$ for all $e \in [M]$ and $\ell \in [0, L-1]$.
- $(\perp \vee \mathtt{decom}_S) \leftarrow \mathtt{Open}^{\mathcal{H}}(\mathtt{decom}, S, \mathsf{pp})$: On input opening information decom, an index set $S = \{\bar{\ell}_1, \ldots, \bar{\ell}_M\}$ with $\bar{\ell}_e \in [0, L-1]$ for all $e \in [M]$ and pp, the opening algorithm outputs either $\perp$ or a partial opening $\mathtt{decom}_S$ corresponding to all-but-M messages.
- $(\perp \vee (\mathtt{com}, \bar{\mathbf{R}})) \leftarrow \mathtt{Verify}^{\mathcal{H}}(\mathtt{decom}_S, S, \mathsf{pp})$: On input $\mathtt{decom}_S$, S and pp, the verification algorithm either rejects by outputting $\perp$, or accepts and outputs a commitment com together with
$$\bar{\mathbf{R}} = ((r_{e,\ell})_{\ell\in[0,L-1],\,\ell\neq\bar{\ell}_e})_{e\in[M]}.$$

Remark 5 (Commitment Generation). In $\mathtt{LeafCom}^{\mathcal{H}}(\mathbf{L}, \mathsf{pp})$, each leaf node is first processed using H_T, producing a pair consisting of a λ-bit message and a 2λ-bit commitment. In this setting, the message space is $\mathcal{M} = \{0,1\}^\lambda$. After interleaving, this yields the tuple
$$(\mathtt{com}_{e,0}, \ldots, \mathtt{com}_{e,L-1})_{e\in[M]}.$$

All such commitments are then aggregated into a single commitment by applying H_{com}:

$$\mathsf{com} \leftarrow H_{\mathsf{com}}\big((\mathsf{com}_{e,0}, \ldots, \mathsf{com}_{e,L-1})_{e\in[M]}\big).$$

In the random oracle model, both H_T and H_{com} are modeled as random oracles, and we collectively denote them by $\mathcal{H}$.

This construction differs slightly from the original BAVC definition, where commitments for each $e \in [M]$ are first compressed via

$$\mathsf{com}_e \leftarrow H^{(e)}_{\mathsf{com}}(\mathsf{com}_{e,0}, \ldots, \mathsf{com}_{e,L-1}),$$

and are then aggregated as

$$\mathsf{com} \leftarrow H_{\mathsf{com}}(\mathsf{com}_1, \ldots, \mathsf{com}_M).$$

While the intermediate hashing step $H^{(e)}_{\mathsf{com}}$ yields a somewhat cleaner modular construction, it complicates the security analysis in the QROM. To facilitate a cleaner proof, we therefore omit this intermediate hashing layer.

Following the definition in [4], correctness allows the sender to abort on certain inputs S during Open (e.g., when the opening size exceeds a threshold). In our scheme, the abort probability remains small so that it does not pose any problem.

Definition 6 (Correctness with Aborts, adapted from [4])**.** BAVC is correct with aborts if, for all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and every challenge set

$$S = \{\bar{\ell}_e\}_{e\in[M]},\ \bar{\ell}_e \in [0, L-1] \text{ for all } e \in [M],$$

the following holds:

$$\Pr\left[\begin{array}{l|l} \mathsf{decom}_S = \bot \vee & (\mathsf{decom}, \mathbf{L}) \leftarrow \mathsf{TreeGen}^{\mathcal{H}}(\mathsf{pp}) \\ (\mathsf{com}' = \mathsf{com} \wedge & (\mathsf{com}, \mathbf{R}) \leftarrow \mathsf{LeafCom}^{\mathcal{H}}(\mathbf{L}) \\ \bar{\mathbf{R}}' = \bar{\mathbf{R}}) & \mathsf{decom}_S \leftarrow \mathsf{Open}^{\mathcal{H}}(\mathsf{decom}, S) \\ & (\mathsf{com}', \bar{\mathbf{R}}') \leftarrow \mathsf{Verify}^{\mathcal{H}}(\mathsf{decom}_S, S) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Here,

$$\bar{\mathbf{R}} = ((r_{e,\ell})_{\ell\in[0,L-1],\,\ell\neq\bar{\ell}_e})_{e\in[M]}$$

denotes the restriction of

$$\mathbf{R} = ((r_{e,\ell})_{\ell\in[0,L-1]})_{e\in[M]}$$

to all coordinates except those indexed by S.

Remark 7 (Handling the Abort of BAVC). In the BAVC-based commitment-and-open protocol, the algorithm may abort, outputting $\mathsf{decom}_S = \bot$, if the opening size induced by the challenge set S exceeds a predefined threshold. To handle this, we let the verifier resample the challenge set until it obtains an S such that $\mathsf{decom}_S \neq \bot$. Formally, this restricts the challenge distribution from the full uniform distribution over $[0, L-1]^M$ to the uniform distribution over the non-aborting subset. The statistical distance between these two distributions is at most the abort probability $\alpha(\lambda)$. For sufficiently large L and M, $\alpha(\lambda)$ is negligible. Thus, in our proofs, we may assume without loss of generality that $\mathsf{decom}_S \neq \bot$, and absorbs $\alpha(\lambda)$ into the negligible error terms of BAVC.

To better accommodate our construction and security analysis, we introduce a notion of online extractability for BAVC. Informally, it requires the existence of an extractor such that, for any PPT adversary that can correctly generate a commitment together with a valid opening for a randomly chosen challenge set S, the extractor, by interacting with

the adversary, can extract the underlying leaf information $\mathbf{L}$ corresponding to the commitment. The formal definition is given below.

Definition 8 (Extractable Binding)**.** Let BAVC be a vector commitment as in Definition 4. We say that BAVC satisfies extractable in the (Q)ROM if, for all $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$, there exists a (quantum) PPT extractor $\mathcal{E}_{\mathsf{BAVC}}$ such that for every (quantum) PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with shared state, the following hold.

- **Simulation Indistinguishability.** There exists a negligible function $\epsilon_{\mathsf{sim}}(\cdot)$ such that for any index set

$$S = \{\bar{\ell}_e\}_{e\in[M]},\ \bar{\ell}_e \in [0, L-1],$$

  we have

$$\mathsf{Dis}(\mathsf{View}_0, \mathsf{View}_1) \le \epsilon_{\mathsf{sim}}(\lambda),$$

  where

$$\begin{aligned}
\mathsf{View}_0:\quad & \mathtt{com}_0 \leftarrow \mathcal{A}_1^{\mathcal{H}}(1^\lambda),\\
& (\bar{\mathbf{R}}_0, \mathtt{decom}_{S,0}) \leftarrow \mathcal{A}_2^{\mathcal{H}}(1^\lambda, S),\\
& (\mathtt{com}_0', \bar{\mathbf{R}}_0') \leftarrow \mathtt{Verify}^{\mathcal{H}}(\mathtt{decom}_{S,0}, S),\\
\mathsf{View}_1:\quad & \mathtt{com}_1 \leftarrow \mathcal{A}_1^{\mathcal{E}_{\mathsf{BAVC}}}(1^\lambda),\\
& (\bar{\mathbf{R}}_1, \mathtt{decom}_{S,1}) \leftarrow \mathcal{A}_2^{\mathcal{E}_{\mathsf{BAVC}}}(1^\lambda, S),\\
& (\mathtt{com}_1', \bar{\mathbf{R}}_1') \leftarrow \mathtt{Verify}^{\mathcal{E}_{\mathsf{BAVC}}}(\mathtt{decom}_{S,1}, S).
\end{aligned}$$

- **Extractability.** There exists a negligible function $\epsilon_{\mathsf{ext}}(\cdot)$ such that

$$\Pr\left[\bar{\mathbf{R}}^* \neq \bar{\mathbf{R}}' \wedge (\mathtt{com}', \bar{\mathbf{R}}') = (\mathtt{com}, \bar{\mathbf{R}})\right] \le \epsilon_{\mathsf{ext}}(\lambda),$$

  where the probability is taken over the following experiment.
  Sample $\bar{\ell}_e \xleftarrow{\$} [0, L-1]$ for all $e \in [M]$, and let $S = \{\bar{\ell}_e\}_{e\in[M]}$. Let

$$\mathtt{com} \leftarrow \mathcal{A}_1^{\mathcal{E}_{\mathsf{BAVC}}}(1^\lambda),\ (\bar{\mathbf{R}}, \mathtt{decom}_S) \leftarrow \mathcal{A}_2^{\mathcal{E}_{\mathsf{BAVC}}}(1^\lambda, S),$$

  and

$$(\mathtt{com}', \bar{\mathbf{R}}') \leftarrow \mathtt{Verify}^{\mathcal{E}_{\mathsf{BAVC}}}(\mathtt{decom}_S, S).$$

  Given com, the extractor outputs

$$\mathbf{R}^* = (r^*_{e,\ell})_{e\in[M],\,\ell\in[0,L-1]}.$$

  We then define

$$\bar{\mathbf{R}}^* = (r^*_{e,\ell})_{e\in[M],\,\ell\in[0,L-1]\setminus\{\bar{\ell}_e\}}.$$

In [4], the notion of selective hiding for BAVC is introduced. We adopt this notion with slight modifications and present our definition below.

Definition 9 (Selective Hiding (adapted from [4]))**.** Let BAVC be a vector commitment scheme as defined in Definition 4. We first define the selective hiding experiment for BAVC, with parameters $M, L = \mathrm{poly}(\lambda)$, as follows:

(1) $\mathsf{pp} \leftarrow \mathtt{Setup}^{\mathcal{H}}(1^\lambda)$, $b \xleftarrow{\$} \{0,1\}$.

(2) $(\mathsf{decom}, \mathbf{L}) \leftarrow \mathsf{TreeGen}^{\mathcal{H}}(\mathsf{pp})$, where

$$\mathbf{L} = (l_{e,0}, \ldots, l_{e,L-1})_{e\in[M]}.$$

(3) $(\mathsf{com}, \mathbf{R}^*) \leftarrow \mathsf{LeafCom}^{\mathcal{H}}(\mathbf{L})$, where

$$\mathbf{R}^* = (r^*_{e,0}, \ldots, r^*_{e,L-1})_{e\in[M]}.$$

(4) $S \leftarrow \mathcal{A}^{\mathcal{H}}(\mathsf{com})$, where $S = \{\bar{\ell}_1, \ldots, \bar{\ell}_M\}$ with $\bar{\ell}_e \in [0, L-1]$ for all $e \in [M]$.
(5) $\mathsf{decom}_S \leftarrow \mathsf{Open}^{\mathcal{H}}(\mathsf{decom}, S)$.
(6) For all $e \in [M]$ and $\ell \in [0, L-1] \setminus \{\bar{\ell}_e\}$, set $r_{e,\ell} \leftarrow r^*_{e,\ell}$.
(7) For each $e \in [M]$, set $r_{e,\bar{\ell}_e} \xleftarrow{\$} \mathcal{M}$ if $b = 0$, and $r_{e,\bar{\ell}_e} \leftarrow r^*_{e,\bar{\ell}_e}$ otherwise.
(8) $b' \leftarrow \mathcal{A}^{\mathcal{H}}\big((r_{e,0}, \ldots, r_{e,L-1})_{e\in[M]}, \mathsf{decom}_S\big)$.
(9) Output 1 if $b' = b$, and 0 otherwise.

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathsf{AdvHD}^{\mathsf{BAVC}}_{\mathcal{A}} := \left|\Pr[b' = b] - \tfrac{1}{2}\right|.$$

We say that BAVC satisfies *selective hiding* if for any (quantum) PPT adversary $\mathcal{A}$, the advantage $\mathsf{AdvHD}^{\mathsf{BAVC}}_{\mathcal{A}}$ is negligible in λ.

3.2.2 Merkle-Tree-based Vector Commitment. We use the Merkle-tree–based vector commitment [45] in our construction of PegaRing. Below, we present a generalised Merkle-tree–based vector commitment that supports the opening of M leaf nodes, rather than just a single leaf node.

Definition 10 (Merkle-Tree Vector Commitment)**.** Let $\mathcal{H}$ be a random oracle. The Merkle-tree-based vector commitment with message space $\mathcal{M}$ is defined by the following PPT algorithms:

- $\mathsf{com}_{\mathsf{MT}} \leftarrow \mathsf{MT.Commit}(\mathbf{V})$: On input a vector

$$\mathbf{V} = (v_{e,0}, \ldots, v_{e,z-1})_{e\in[M]},$$

  where $v_{e,i} \in \mathcal{M}$ for all $i \in [0, z-1]$ and $e \in [M]$, the algorithm outputs a short commitment $\mathsf{com}_{\mathsf{MT}}$.
- $(\mathbf{V}_I, \mathsf{path}_I) \leftarrow \mathsf{MT.Open}(I, \mathbf{V})$: On input a set of indices $I := \{\bar{i}_1, \ldots, \bar{i}_M\}$, where $\bar{i}_e \in [0, z-1]$ and the vector $\mathbf{V}$ that was committed using $\mathsf{MT.Commit}$, the deterministic algorithm outputs M leaf nodes $\mathbf{V}_I = (v_{e,\bar{i}_e})_{e\in[M]}$ and authentication paths path_I.
- $(\perp \vee \mathsf{com}_{\mathsf{MT}}) \leftarrow \mathsf{MT.Verify}(\mathbf{V}_I, \mathsf{path}_I, I)$: On input leaf nodes $\mathbf{V}_I = (v_{e,\bar{i}_e})_{e\in[M]}$, the corresponding authentication paths path_I, and indices $I = \{\bar{i}_1, \ldots, \bar{i}_M\}$, where $\bar{i}_e \in [0, z-1]$, the deterministic algorithm outputs $\mathsf{com}_{\mathsf{MT}}$ (accept the opening) if and only if $\mathbf{V}_I$ contains valid leaf nodes with respect to the path path_I. Otherwise, outputs $\perp$ (reject the opening).

Similar to BAVC, Merkle-tree based vector commitment can be combined to one tree such that open multiple leaves result in shorter proofs. The proof can abort if the opening exceeds a certain threshold. The security (binding and hiding) follows the collision-resistant property of the hash function used to construct the Merkle tree.

3.3 Commit-and-Open Sigma Protocol

We construct a commit-and-open (C&O) Sigma protocol for β-approximate relation $\mathcal{R}_{\beta\mathcal{L}^t}$ as defined in Definition 3 and set membership proof of the relation $\mathcal{R}^{\mathsf{OR}}_{\beta\mathcal{L}^t}$ defined as follows:

$$\mathcal{R}^{\mathsf{OR}}_{\beta\mathcal{L}^t} = \{(pk_1, \ldots, pk_n; sk) : \exists \pi \in [n], (pk_\pi, sk) \in \mathcal{R}_{\beta\mathcal{L}^t}\}.$$

Below is the syntax of C&O Sigma protocol.

Definition 11 (Commit-and-Open Sigma Protocol)**.** Let $\mathcal{R} = \{R_\lambda\}$ be a family of binary relations, and let $\mathcal{H}$ denote a collection of random oracles. A commit-and-open Sigma protocol $\Pi^{\mathcal{H}}$ for $\mathcal{R}$ in the random oracle model is a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{P}, \mathsf{V})$ defined as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, output public parameters pp.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$: On input pp, output a public-secret key pair $(pk, sk) \in R_\lambda$.
- $\mathsf{P}^{\mathcal{H}} = (P_1^{\mathcal{H}}, P_2^{\mathcal{H}})$: The prover is a two-phase, stateful oracle algorithm that may query $\mathcal{H}$.
  - $((A, A_\circ), \mathsf{st}_P) \leftarrow P_1^{\mathcal{H}}(pk, sk)$: On input (pk, sk), output commitments A, auxiliary information $A_\circ$, and prover state st_P.
  - $Z \leftarrow P_2^{\mathcal{H}}(\mathsf{st}_P, (A, A_\circ), S)$: On input $(\mathsf{st}_P, (A, A_\circ), S)$, output a response Z that opens the commitments in A with respect to challenge S.
- $\mathsf{V}^{\mathcal{H}} = (V_1^{\mathcal{H}}, V_2^{\mathcal{H}})$: The verifier is a two-phase, stateful oracle algorithm that may query $\mathcal{H}$.
  - $(S, \mathsf{st}_V) \leftarrow V_1^{\mathcal{H}}(pk, (A, A_\circ))$: On input $(pk, (A, A_\circ))$, output a challenge S and verifier state st_V.
  - $b \leftarrow V_2^{\mathcal{H}}(\mathsf{st}_V, pk, (A, A_\circ), S, Z)$: On input

    $$(\mathsf{st}_V, pk, (A, A_\circ), S, Z),$$

    output $b \in \{0, 1\}$ indicating accept or reject.

As in the classical case, when the challenge S is selected uniformly at random, the verifier can be simplified to a single-stage PPT oracle algorithm $\mathsf{V}^{\mathcal{H}}$. For any $(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, the transcript

$$(b, (A, A_\circ), S, Z) = [\mathsf{P}^{\mathcal{H}}(pk, sk) \leftrightarrow \mathsf{V}^{\mathcal{H}}(pk)]$$

denotes the information exchanged between the prover and verifier throughout the protocol in the presence of the random oracles.

A C&O Sigma protocol satisfies completeness, proof-of-knowledge with online extractability (PoK-OE) and special verifier zero-knowledge (SHVZK), defined as follows.

Definition 12 (Completeness)**.** A C&O Sigma protocol is complete if for all $\lambda \in \mathbb{N}$ and all $(pk, sk) \in R_\lambda$,

$$\Pr\left[b = 0 : (b, (A, A_\circ), S, Z) \leftarrow \langle P^{\mathcal{H}}(pk, sk), V^{\mathcal{H}}(pk)\rangle\right] \leq \mathsf{negl}(\lambda).$$

Definition 13 (PoK-OE)**.** We say that $\Pi^{\mathcal{H}}$ satisfies the *PoK-OE property* in the (Q)ROM if, for any quantum PPT adversary P^*, there exists a quantum PPT extractor $\mathcal{E}$ such that:

- **(Simulation Indistinguishability).** There exists a negligible function $\epsilon_{\mathsf{sim}}(\cdot)$ such that

  $$\mathsf{Dis}(\mathsf{View}_0, \mathsf{View}_1) \leq \epsilon_{\mathsf{sim}}(\lambda),$$

Game $\mathsf{SHVZK}(1^\lambda)$	$\mathsf{getTransCh}(sk, pk, S)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	$((A, A_\circ), \mathsf{st}_P) \leftarrow P_1^{\mathcal{H}}(pk, sk)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$	$Z \leftarrow P_2^{\mathcal{H}}(\mathsf{st}_P, (A, A_\circ), S)$
$S \xleftarrow{\$} \mathcal{CH}, b \xleftarrow{\$} \{0,1\}$	**return** $((A, A_\circ), S, Z)$
$\mathrm{tr}^0 \leftarrow \mathsf{getTransCh}(sk, pk, S)$	
$\mathrm{tr}^1 \leftarrow \mathtt{Sim}^{\mathcal{H}}(pk, S)$	
$b' \leftarrow \mathcal{A}^{\mathcal{H}}(pk, \mathrm{tr}^b)$	
return $\|b' = b\|$	

Fig. 1. SHVZK Game

where

$$\mathsf{View}_0 := (b, (A, A_\circ), S, Z) \leftarrow \left[P^{*\mathcal{H}}(pk) \leftrightarrow V^{\mathcal{H}}(pk)\right],$$
$$\mathsf{View}_1 := (b', (A', A'_\circ), S', Z') \leftarrow \left[P^{*\mathcal{E}}(pk) \leftrightarrow V^{\mathcal{E}}(pk)\right].$$

- **(Online Extractability).** There exists a negligible function $\epsilon_{\mathsf{ext}}(\cdot)$ such that the following holds.
  Let
  $$(b, (A, A_\circ), S, Z) = \left[\mathcal{P}^{*\mathcal{E}}(pk) \leftrightarrow \mathcal{V}^{\mathcal{E}}(pk)\right].$$
  Given the transcript $((A, A_\circ), S, Z)$, the extractor outputs
  $$sk^* \leftarrow \mathcal{E}(pk, (A, A_\circ), S, Z).$$
  Then,
  $$\Pr\left[(pk, sk^*) \notin R_\lambda \wedge b = 1\right] \leq \epsilon_{\mathsf{ext}}(\lambda).$$

Definition 14 (SHVZK)**.** Let $\Pi^{\mathcal{H}} = (\mathtt{Setup}, \mathsf{KeyGen}, P^{\mathcal{H}}, V^{\mathcal{H}})$ be a C&O Sigma protocol with challenge space $\mathcal{CH}$ in the (Q)ROM for $\mathcal{R} = \{R_\lambda\}_{\lambda \in \mathbb{N}}$. $\Pi^{\mathcal{H}}$ satisfies SHVZK property in the (Q)ROM if there exists a (quantum) PPT simulator $\mathtt{Sim}$ such that for any (quantum) PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{SHVZK}}_{\Pi^{\mathcal{H}}, \mathcal{A}}(\lambda) := \left|\Pr[\mathsf{SHVZK}(1^\lambda) \Rightarrow 1] - \tfrac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the SHVZK game is defined in Fig. 1.

3.4 Digital Signature

We recall the syntax and security definition of digital signature scheme.

Definition 15 (Syntax)**.** A signature scheme consists of the following algorithms $(\mathtt{Setup}, \mathsf{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$:

(1) $(\mathsf{pp}) \leftarrow \mathtt{Setup}(1^\lambda)$. On input a security parameter λ in unary, the algorithm generates a set of public parameters denoted as pp.
(2) $(sk, pk) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$: On input public parameters pp generated in the setup phase, the algorithm outputs a secret and public key pair (sk, pk).
(3) $\sigma \leftarrow \mathtt{Sign}(sk, \mathtt{msg}, \mathsf{pp})$: On input a secret key sk, a message $\mathtt{msg}$, and public parameters pp, the (randomized) algorithm outputs a signature σ.

(4) $0/1 \leftarrow \mathtt{Verify}(pk, \mathtt{msg}, \sigma, \mathtt{pp})$: On input a public key pk, a message $\mathtt{msg}$, a signature σ, and public parameters pp, the (deterministic) algorithm outputs 1 (accept) or 0 (reject).

We define the perfect correctness of the signature scheme.

Definition 16 (Perfect Correctness)**.** For any message $\mathtt{msg}$ and any $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$,

$$\Pr\left[\mathtt{Verify}(pk, \mathtt{msg}, \sigma, \mathtt{pp}) = 1 \;\middle|\; \begin{array}{l}(pk, sk) \leftarrow \mathtt{KeyGen}(pp)\\ \sigma \leftarrow \mathtt{Sign}(sk, \mathtt{msg}, \mathtt{pp})\end{array}\right] = 1$$

In our security proof of Pegasus, we first show that our signature scheme achieves existential unforgeability in the key only model (EUF-KO), where the adversary is limited to using only the public key to attempt a forgery.

Definition 17 (EUF-KO)**.** Let λ be the security parameter and let $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$ be the public parameters. A signature scheme is EUF-KO-secure if, for any PPT adversary $\mathcal{A}$, its advantage $\mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathcal{A}}$ in the EUF-KO game is negligible, where

$$\mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathcal{A}} := \Pr\left[b = 1 \;\middle|\; \begin{array}{l}(sk, pk) \leftarrow \mathtt{KeyGen}(\mathtt{pp}),\\ (\mathtt{msg}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{H}}(pk)\\ b \leftarrow \mathtt{Verify}(pk, \mathtt{msg}^*, \sigma^*)\end{array}\right].$$

We then extend our proof to existential unforgeability against chosen message attack (EUF-CMA) by allowing the adversary to make signing oracle queries, and showing the reduction from EUF-CMA to EUF-KO.

Definition 18 (EUF-CMA)**.** Let λ be the security parameter and let $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$ the public parameters. A signature scheme is EUF-CMA-secure if, for any PPT adversary $\mathcal{A}$, its advantage $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}$ in the EUF-CMA game is negligible, where

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}} := \Pr\left[\begin{array}{l}\mathtt{msg}^* \notin Q_s\\ \wedge\, b = 1\end{array} \;\middle|\; \begin{array}{l}(sk, pk) \leftarrow \mathtt{KeyGen}(\mathtt{pp}),\\ (\mathtt{msg}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathtt{Sign}}(pk)\\ b \leftarrow \mathtt{Verify}(pk, \mathtt{msg}^*, \sigma^*)\end{array}\right].$$

Here, $\mathcal{A}^{\mathtt{Sign}}$ denotes that $\mathcal{A}$ is given oracle access to the signing oracle, and Q_s denotes the set of messages queried by $\mathcal{A}$ to the signing oracle.

3.5 Ring Signature

Finally, we recall the definition of ring signatures. Let $\mathbb{PK} = (pk_1, pk_2, pk_3, \dots)$ denote the set of all public keys added to the system and a ring of size n is denoted as $\mathcal{PK} \subseteq \mathbb{PK}$ such that $|\mathcal{PK}| = n$.

Definition 19 (Syntax)**.** A *ring signature* scheme is a tuple of PPT algorithms $(\mathtt{R\text{-}Setup}, \mathtt{R\text{-}KeyGen}, \mathtt{R\text{-}Sign}, \mathtt{R\text{-}Verify})$ that are described as follows:

- $\mathtt{R\text{-}pp} \leftarrow \mathtt{R\text{-}Setup}(1^\lambda)$: On input a security parameter λ in unary, outputs a set of public parameters denoted as R-pp.
- $(pk, sk) \leftarrow \mathtt{R\text{-}KeyGen}(\mathtt{R\text{-}pp})$: On input the public parameters generated from Setup, outputs a public key pk and the corresponding secret key sk.
- $\sigma \leftarrow \mathtt{R\text{-}Sign}(\mathcal{PK}, \mathtt{msg}, sk_\pi, \mathtt{R\text{-}pp})$: On input a list $\mathcal{PK} \subseteq \mathbb{PK}$ of n public keys, a message $\mathtt{msg}$, and a secret key sk_π corresponds to pk_π where $pk_\pi \in \mathcal{PK}$, the (randomized) algorithm outputs a signature σ.
- $0/1 := \mathtt{R\text{-}Verify}(\mathcal{PK}, m, \sigma)$: On input a list $\mathcal{PK}$ of n public keys, a message $\mathtt{msg}$ and a ring signature σ, the (deterministic) algorithm outputs 0 (reject) if the ring signature is invalid with respect to the message $\mathtt{msg}$ and the ring $\mathcal{PK}$. Otherwise, output 1 (accept).

We require the standard correctness condition, where for any security parameter λ, public parameters $\mathsf{R\text{-}pp} \leftarrow \mathsf{R\text{-}Setup}(1^\lambda)$, a list of secret and public keys $\{(sk_i, pk_i)\}_{i\in[n]}$ output by $\mathsf{R\text{-}KeyGen}(\mathsf{R\text{-}pp})$, signer index $\pi \in [n]$, and message msg, we have

$$1 = \mathsf{R\text{-}Verify}(\mathcal{PK}, \mathsf{msg}, \mathsf{R\text{-}Sign}(\mathcal{PK}, \mathsf{msg}, sk_\pi, \mathsf{R\text{-}pp}), \mathsf{R\text{-}pp}),$$

where $\mathcal{PK} = (pk_1, \ldots, pk_n)$.

We assume the adversary has (classical) access to corruption and singing oracles, following the assumption in [36]. We leave the quantum access to signing and corruption oracles (such as those formalized in [15]) as future works.

Definition 20 (Unforgeability w.r.t. Insider Corruption)**.** For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, a ring signature scheme is said to satisfy unforgeability with respect to insider corruption if $\mathcal{A}$ has at most negligible advantage ($\mathsf{Adv}^{\mathsf{EUF}}_{\mathcal{A}}(1^\lambda)$) in winning the following game:

- **Setup.** The challenger runs $\mathsf{R\text{-}pp} \leftarrow \mathsf{R\text{-}Setup}(1^\lambda)$ and generates key pairs $(sk_i, pk_i) \leftarrow \mathsf{R\text{-}KeyGen}(\mathsf{R\text{-}pp})$ for all $i \in [N]$. It sends the set of public keys $\mathbb{PK} = (pk_1, \ldots, pk_N)$ to $\mathcal{A}$ and initializes two empty lists, Q_{SO} and Q_{CO}.
- **Oracle queries.** The adversary $\mathcal{A}$ may adaptively issue polynomially many queries to the signing oracle O_{sig} and the corruption oracle O_{cor}:
  - For each signing query $(\mathsf{msg}, \mathcal{PK}, pk_i)$, if $pk_i \notin \mathcal{PK}$ or $\mathcal{PK} \not\subseteq \mathbb{PK}$, the oracle O_{sig} returns $\perp$. Otherwise, it computes $\sigma \leftarrow \mathsf{R\text{-}Sign}(\mathcal{PK}, \mathsf{msg}, sk_i, \mathsf{R\text{-}pp})$ and returns σ. The challenger then records $(pk_i, \mathcal{PK}, \mathsf{msg})$ in Q_{SO}.
  - For each corruption query i, if $pk_i \notin \mathbb{PK}$, the oracle O_{cor} returns $\perp$. Otherwise, it returns sk_i. The challenger then records pk_i in Q_{CO}.
- **Forgery.** Finally, $\mathcal{A}$ outputs $(\mathcal{PK}^*, \mathsf{msg}^*, \sigma^*)$. The adversary wins if the following hold simultaneously:
  - $\mathcal{PK}^* \subseteq \mathbb{PK} \backslash Q_{CO}$ (the ring only contains uncorrupted public keys),
  - $(\cdot, \mathcal{PK}^*, \mathsf{msg}^*) \notin Q_{SO}$ (the message-ring pair was never queried to the signing oracle), and
  - $1 = \mathsf{R\text{-}Verify}(\mathcal{PK}^*, \mathsf{msg}^*, \sigma^*, \mathsf{R\text{-}pp})$.

Definition 21 (Anonymity against Full Key Exposure)**.** For any polynomial time adversary $\mathcal{A}$, a ring signature is anonymous against full key exposure if any PPT adversary $\mathcal{A}$ has at most negligible advantage ($\mathsf{Adv}^{\mathsf{EUF}}_{\mathcal{A}}$) to win the following game:

- The challenger runs $\mathsf{R\text{-}pp} \leftarrow \mathsf{R\text{-}Setup}(1^\lambda)$ and generates key pairs $(sk_i, pk_i) \leftarrow \mathsf{R\text{-}KeyGen}(\mathsf{R\text{-}pp})$ for all $i \in [N]$. The challenger samples a challenge bit $b^* \xleftarrow{\$} \{0, 1\}$ and sends $\mathsf{R\text{-}pp}$, $\mathbb{PK} = \{pk_i\}_{i\in[N]}$, and $\{sk_i\}_{i\in[N]}$ to $\mathcal{A}$.
- $\mathcal{A}$ computes a challenge $(\mathcal{PK}, \mathsf{msg}, i_0, i_1)$, where $\mathcal{PK}$ contains pk_{i_0}, pk_{i_1} and sends it to the challenger. Note that we do not assume that $\mathcal{PK} \subseteq \mathbb{PK}$.
- The challenge runs $\sigma \leftarrow \mathsf{R\text{-}Sign}(\mathcal{PK}, \mathsf{msg}, sk_{i_b}, \mathsf{R\text{-}pp})$ and sends σ to $\mathcal{A}$.
- $\mathcal{A}$ outputs b, and we say $\mathcal{A}$ wins the anonymity game if $b = b^*$.

4 C&O Sigma Protocol for Legendre and Power Residue PRFs

4.1 Protocol Description

As the core building block of PegaRing, we construct a C&O Σ-protocol for the Legendre and power residue PRF relations, following the approach outlined in Section 1.2. We first specify the setup and key generation procedures in Algorithms 1 and 2, respectively, and then present the complete protocol in Algorithm 3.

Algorithm 1: Setup

$\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$

On input a security parameter λ, the algorithm computes the following public parameters:

- A large odd prime p,
- An integer $t \geq 2$ such that $t|(p-1)$,
- An integer L (the number of PRF symbols in the public key),
- A list $\mathfrak{I} = (I_0, \ldots, I_{L-1})$ where I_ℓ are randomly chosen from $\mathbb{F}_p$ for all $\ell \in [0, L-1]$,
- An integer M (the number of parallel executions).
- Parameters BAVC.pp generated by executing $\mathtt{BAVC.Setup}(1^\lambda)$.

Output $\mathsf{pp} = (p, t, L, M, \mathfrak{I}, \mathtt{BAVC.pp})$

Algorithm 2: Key Generation

$(sk, pk) \leftarrow \mathtt{KeyGen}(\mathsf{pp})$

On input public parameters pp, the algorithm computes the secret key and the public key as follows:

- Randomly select K from $\mathbb{F}_p^*$ and set $sk = K$,
- Compute $pk = \mathcal{L}_K^t(\mathfrak{I}) = (\mathcal{L}_K^t(I_0), \ldots, \mathcal{L}_K^t(I_{L-1}))$.

Output (pk, sk).

4.2 Security Analysis of the Sigma Protocol

Let $\mathcal{R}_{\beta\mathcal{L}^t}$ denote the β-approximate PRF relation from Definition 3 and $L_\mathcal{R}$ be the language induced by $\mathcal{R}_{\beta\mathcal{L}^t}$ (i.e., $L_\mathcal{R} = \{pk : \exists sk \text{ such that } (pk, sk) \in \mathcal{R}_{\beta\mathcal{L}^t}\}$). The construction in Algorithm 3 satisfies completeness, proof-of-knowledge with online extraction (PoK-OE) and special honest verifier zero-knowledge (SHVZK) in the (quantum) random oracle model.

Theorem 22 (Completeness)**.** *Assume the completeness (with negligible abort probability) of BAVC, if the witness-statement pair $(pk, sk) \in \mathcal{R}_{\beta\mathcal{L}^t}$ and both parties are honest, then the verifier always accepts the transcript.*

Proof Sketch. The key to the completeness proof is that by revealing all-but-one shares of r_e together with the key share adjustment, the verifier can reconstruct $o_e = (K + I_{\bar{\ell}_e})r_e$ for all $e \in [M]$. We defer the proof to Section C.3.

Theorem 23 (PoK-OE)**.** *Let $\beta \in [0, 1]$ be fixed and let $\mathcal{H}$ be modelled as a set of (quantum) random oracles. If the BAVC protocol is extractable binding in the (Q)ROM, then the commit-and-open Sigma protocol $\Pi^\mathcal{H}$ is a PoK-OE in the (Q)ROM.*

Proof Sketch. For any PPT adversary $\mathcal{P}^*$, we construct a PPT extractor $\mathcal{E}$ that relies on the BAVC extractor. In the simulation of $\mathcal{H}$, if $\mathcal{P}^*$ produces a valid transcript through interaction with $\mathcal{E}$, then $\mathcal{E}$ extracts a valid secret key sk. In particular, $\mathcal{E}$ invokes the BAVC extractor to get all committed messages $\mathbf{R} = (r_{e,0}, \ldots, r_{e,L-1})_{e\in[M]}$, which allows it to compute the candidate witness K^* from the key share adjustments $\Delta K_e = Kr_e + \sum_{\ell=0}^{L-1} I_\ell r_{e,\ell}$. For an accepting transcript, the value o_e must satisfy $\mathcal{L}_0^t(o_e) = T_e + pk_{\bar{\ell}_e}$, where $(T_e)_{e\in[M]}$ were fixed and sent in the first round. Thus, the check can be seen as verifying PRF outputs with a fixed public shift. According to the definition of the β-approximate PRF relation (Definition 3), for any incorrect $K^* \neq K$, there are at most $(1-\beta)\cdot L$ values in $\mathfrak{I}$ that can lead to matching t-th power-residue symbols. Hence, the probability that all M challenges fall in these "bad" values is at most $(1-\beta)^M$. Together with the BAVC extraction error $\epsilon_{\mathtt{BAVC}}$ in the (Q)ROM, we can bound the extraction error ϵ_{ext} of $\mathcal{E}$ by $\epsilon_{\mathtt{BAVC}} + (1-\beta)^M$. We defer the formal proof to Section C.4.

Algorithm 3: Σ-Protocol based on Legendre and Power Residue PRFs

Procedure `Commitment`(sk, pp)**:**
  The prover $\mathcal{P}$ generates the BAVC tree by running the tree generation algorithm: $(\mathsf{decom}, \mathbf{L}) \leftarrow \mathsf{BAVC.TreeGen}(\mathsf{BAVC.pp})$.
  Next, $\mathcal{P}$ derives the commitment and the message by running the leaf commitment algorithm: $(\mathsf{com}, \mathbf{R}) \leftarrow \mathsf{BAVC.LeafCom}(\mathbf{L}, \mathsf{BAVC.pp})$, where $\mathbf{R} = (r_{e,0}, \ldots, r_{e,L-1})_{e\in[M]}$.
  for $e = 1$ **to** M **do**
    Compute $r_e = \sum_{\ell=0}^{L-1} r_{e,\ell}$ and $T_e = \mathcal{L}_0^t(r_e)$.
    Compute key adjustment $\Delta K_e = Kr_e + \sum_{\ell=0}^{L-1} I_\ell r_{e,\ell}$, where $K = sk$.
  $\mathcal{P}$ sends $A = \mathsf{com}$ and $A_\circ = (T_e, \Delta K_e)_{e\in[M]}$ to $\mathcal{V}$.
Procedure `Challenge`$(A, A_\circ)$**:**
  On receive the commitment A and the auxiliary information $A_\circ$, the verifier $\mathcal{V}$ randomly samples the challenge set $S := \{\bar{\ell}_e\}_{e\in[M]}$ where $\bar{\ell}_e \xleftarrow{\$} [0, L-1]$, and sends S to $\mathcal{P}$.
Procedure `Response`(decom, S)**:**
  On input the challenge S, $\mathcal{P}$ computes $\mathsf{decom}_S \leftarrow \mathsf{BAVC.Open}(\mathsf{decom}, S)$ and sends the response $Z = \mathsf{decom}_S$ to $\mathcal{V}$.
Procedure `Verify`$(pk, ((A, A_\circ), S, Z), \mathsf{pp})$**:**
  $\mathcal{V}$ parses $A = \mathsf{com}$, $A_\circ = (T_e, \Delta K_e)_{e\in[M]}$, $S = \{\bar{\ell}_e\}_{e\in[M]}$, and $Z = \mathsf{decom}_S$.
  $\mathcal{V}$ re-computes commitment and messages by running $(\mathsf{com}', \bar{\mathbf{R}})_{e\in[M]} \leftarrow \mathsf{BAVC.Verify}(\mathsf{decom}_S, S)$, where $\bar{\mathbf{R}} = (r_{e,\ell})_{\ell\in[0,L-1]\setminus\{\bar{\ell}_e\},e\in[M]}$.
  $\mathcal{V}$ checks if $\mathsf{com}' \neq \mathsf{com}$ and runs the following predicate function $V(pk, S, \bar{\mathbf{R}}, A_\circ)$, which is defined as:
  for $e = 1$ **to** M **do**
    Compute $o_e = I_{\bar{\ell}_e} \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} - \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + \Delta K_e$.
    Check if $T_e \stackrel{?}{=} \mathcal{L}_0^t(o_e) - pk_{\bar{\ell}_e}$.
  If any previous check fails, $\mathcal{V}$ outputs 0. Otherwise, $\mathcal{V}$ accepts the signature and outputs 1.

Theorem 24 (SHVZK). *Let $\mathcal{H}$ be modelled as a set of (quantum) random oracles. If the BAVC protocol satisfies selective hiding in the (Q)ROM, then the commit-and-open Σ-protocol $\Pi^{\mathcal{H}}$ has SHVZK in the (Q)ROM.*

Proof Sketch. We construct the simulator which samples the key share adjustments $(\Delta K_e^*)_{e\in[M]}$ uniformly from $\mathbb{F}_p$. For every $e \in [M]$, given the known challenge $\bar{\ell}_e$, it computes o_e^* exactly as the honest verifier would, and sets $T_e^* = \mathcal{L}_0^t(o_e^*) - pk_{\bar{\ell}_e}$. Since ΔK_e^* is uniform, the input o_e^* is also uniform in $\mathbb{F}_p$. Therefore, T_e^* is the PRF evaluation on a uniform field element. Indistinguishability between the real and simulated transcripts is shown via a simple hybrid argument. In the real protocol, ΔK_e and T_e are derived from the committed messages and the witness K. In the first hybrid, we replace unopened $r_{e,\bar{\ell}_e}$ by an independent uniform sample. Then, by hiding property of BAVC, this change is indistinguishable to the verifier. Since $r_{e,\bar{\ell}_e}$ is uniform, the derived $\Delta K_e'$ is uniformly distributed (like ΔK_e^*) and T_e' is the PRF evaluation of a uniform value (like T_e^*), which proves that indistinguishability between the simulated and real transcripts. The formal proof is shown in Section C.5

4.3 Pegasus: Post-Quantum Signature based on Legendre and Power Residue PRFs

By applying the Fiat-Shamir transform [31] to the Σ-protocol described in Section 4.1, we obtain a post-quantum signature scheme for t-th power residue PRFs. The construction is shown in Algorithm 4. In addition to the public

Algorithm 4: Pegasus Signing and Verification

Procedure Sign(msg, sk, pp):
- Sample a random salt: $\mathtt{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$.
- Compute the BAVC commitment: $(\mathtt{decom}, \mathbf{L}) \leftarrow \mathtt{BAVC.TreeGen}(\mathtt{BAVC.pp})$ and $(\mathtt{com}, \mathbf{R}) \leftarrow \mathtt{BAVC.LeafCom}(\mathtt{BAVC.pp}, \mathbf{L})$, where $\mathbf{R} = (r_{e,0}, \ldots, r_{e,L-1})_{e\in[M]}$.
- **for** $e = 1$ **to** M **do**
  - Compute $r_e = \sum_{\ell=0}^{L-1} r_{e,\ell}$ and $T_e = \mathcal{L}_0^t(r_e)$.
  - Compute key share adjustment $\Delta K_e = Kr_e + \sum_{\ell=0}^{L-1} I_\ell r_{e,\ell}$.
- Define $\sigma_1 = (\mathtt{com}, T_1, \ldots, T_M, \Delta K_1, \ldots, \Delta K_M)$.
- Initiate the counter: $\mathtt{ctr} = 0$.
- Compute the challenge $h = \mathcal{H}_{\mathsf{FS}}(\mathtt{salt}, \sigma_1, \mathtt{msg}, \mathtt{ctr})$.
- Expand h: $S \leftarrow \mathtt{expand}(h[:128])$ where $S := \{\bar{\ell}_e\}_{e\in[M]}$ and for every $e \in [M]$, $\bar{\ell}_e \in [0, L-1]$
- **if** $\perp \leftarrow \mathtt{BAVC.Open}(\mathtt{decom}, S)$ **then**
  - Return to Step 9 with $\mathtt{ctr} = \mathtt{ctr} + 1$.
- **else**
  - $\mathtt{decom}_S \leftarrow \mathtt{BAVC.Open}(\mathtt{decom}, S)$
- Output the signature $\sigma = (\mathtt{salt}, h, \mathtt{ctr}, \mathtt{decom}_S, (\Delta K_e)_{e\in[M]})$.

Procedure Verify(msg, pk, σ, pp):
- Parse $\sigma = (\mathtt{salt}, h, \mathtt{ctr}, \mathtt{decom}_S, (\Delta K_e)_{e\in[M]})$.
- Expand the challenge h: $S \leftarrow \mathtt{expand}(h[:128])$ where $S := \{\bar{\ell}_e\}_{e\in[M]}$ and for every $e \in [M]$, $\bar{\ell}_e \in [0, L-1]$
- Recompute $(\mathtt{com}', \bar{\mathbf{R}}) \leftarrow \mathtt{BAVC.Verify}(\mathtt{decom}_S, S)$, where $\bar{\mathbf{R}} = (r_{e,\ell})_{\ell\in[0,L-1]/\{\bar{\ell}_e\},e\in[M]}$.
- **for** $e = 1$ **to** M **do**
  - Compute $o_e = I_{\bar{\ell}_e} \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} - \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + \Delta K_e$.
  - Recompute residuosity symbols: $T'_e = \mathcal{L}_0^t(o_e) - pk_{\bar{\ell}_e}$.
- Check if $h \stackrel{?}{=} \mathcal{H}(\mathtt{salt}, \sigma'_1, \mathtt{msg}, \mathtt{ctr})$ where $\sigma'_1 = (\mathtt{com}', T'_1, \ldots, T'_M, \Delta K_1, \ldots, \Delta K_M)$.

parameters generated in Algorithm 1, we define a hash function $\mathcal{H}_{\mathsf{FS}} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ for the Fiat-Shamir transform, and an expend function $\mathtt{expand} : \{0,1\}^\lambda \to \{0,1\}^*$ to derive verifier challenges.

4.4 Security Analysis of Pegasus in (Q)ROM

4.4.1 EUF-KO Security in (Q)ROM. We first show existential unforgeability of Pegasus in the key-only model. The argument builds on the PoK-OE property of the underlying Σ-protocol, and follows the general structure of the Fiat–Shamir security proof. However, since our construction of Pegasus introduces slight modifications to the FS transform, the proof must additionally account for these changes.

Theorem 25 (EUF-KO of Pegasus in the ROM)**.** *Let $\beta \in [0,1]$ be fixed, and let $\mathcal{H}$ and $\mathcal{H}_{\mathsf{FS}}$ be modelled as random oracles. If the commit-and-open protocol $\Pi^{\mathcal{H}}$ is a PoK-OE in the ROM, and the β-approximate PRF relation $\mathcal{R}_{\beta\mathcal{L}^t}$ is hard, then the signature scheme Pegasus$^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}} := \mathsf{FS}^{\mathcal{H}_{\mathsf{FS}}}[\Pi^{\mathcal{H}}]$ achieves EUF-KO security in the ROM.*

Proof Sketch. If $\mathcal{A}$ outputs a valid message–signature pair (m, σ), then, with overwhelming probability, it must have queried $\mathcal{H}_{\mathsf{FS}}$ on some input related to the random oracle output used in the forgery. Consequently, the PPT extractor $\mathcal{E}$ (as defined in the PoK-OE notion) can leverage the forged signature together with $\mathcal{A}$'s queries to extract the secret key

sk from the public key pk. This extraction would contradict the hardness of the underlying NP relation $\mathcal{R}_{\beta\mathcal{L}^t}$ associated with (pk, sk). We defer the full proof of Theorem 25 to Section C.6.

We further analyse the EUF-KO security of Pegasus in the QROM. Unlike the classical ROM, in the QROM an adversary can issue quantum queries to the QRO and receive superposition responses. In this setting, the classical lazy-sampling technique is inapplicable, as we cannot maintain a classical query list to extract information. Fortunately, several proof techniques have been developed for the QROM. In particular, the CStO technique [60] enables both the simulation of the QRO and the extraction of classical information, which we employ in our analysis.

Theorem 26 (EUF-KO of Pegasus in the QROM)**.** *Let $\beta \in [0,1]$ be fixed, and let $\mathcal{H}$ and $\mathcal{H}_{\mathsf{FS}}$ be modelled as quantum random oracles. If the commit-and-open protocol $\Pi^{\mathcal{H}}$ is a PoK-OE in the QROM, and the β-approximate PRF relation $\mathcal{R}_{\beta\mathcal{L}^t}$ is hard, then the signature scheme $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}} := \mathsf{FS}^{\mathcal{H}_{\mathsf{FS}}}[\Pi^{\mathcal{H}}]$ achieves EUF-KO security in the QROM.*

Proof Sketch. The proof strategy is analogous to that in the classical ROM, where we construct a quantum PPT algorithm $\mathcal{B}$ that simulates $\mathcal{H}_{\mathsf{FS}}$ via CStO and extracts transcripts of the underlying C&O Σ-protocol. We define two database properties: SUC, which captures the case where the adversary queries the QRO and outputs a valid signature while the extracted Σ-protocol transcript is invalid, and CL, which corresponds to database collisions. Our analysis bounds the probability of quantum transition capacity $[\![\bot \xrightarrow{q} \mathsf{SUC} \cup \mathsf{CL}]\!]$, where q is the number of quantum queries the adversary makes to the $\mathcal{H}_{\mathsf{FS}}$ (modelled as QRO). We present the full security proof of EUF-KO in the QROM in Section C.7

4.4.2 EUF-CMA Security in the (Q)ROM. We now reduce the existential unforgeability of Pegasus under chosen-message attacks (EUF-CMA) to EUF-KO.

Theorem 27 (EUF-CMA of Pegasus in the ROM)**.** *Let $\mathcal{H}$ and $\mathcal{H}_{\mathsf{FS}}$ be random oracles. If the commit-and-open protocol $\Pi^{\mathcal{H}}$ is a PoK-OE and has SHVZK in the ROM, and the β-approximate PRF relation $\mathcal{R}_{\beta\mathcal{L}^t}$ is hard, then the signature scheme $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}} := \mathsf{FS}^{\mathcal{H}_{\mathsf{FS}}}[\Pi^{\mathcal{H}}]$ achieves EUF-CMA security in the ROM.*

Proof Sketch. In the EUF-CMA game, the adversary is given access to a signing oracle, which is simulated by the reduction algorithm $\mathcal{B}$. The simulation leverages the SHVZK property of the underlying protocol and the programmability of the random oracle. To embed the challenge, $\mathcal{B}$ programs the random oracle at a specific query and the simulation aborts if the adversary has queried this point before. We defer the full proof to Section C.8

Theorem 28 (EUF-CMA of Pegasus Signature in the QROM)**.** *Let $\mathcal{H}$ and $\mathcal{H}_{\mathsf{FS}}$ be quantum random oracles. If the commit-and-open protocol $\Pi^{\mathcal{H}}$ is a PoK-OE and has SHVZK in the QROM, and the β-approximate PRF relation $\mathcal{R}_{\beta\mathcal{L}^t}$ is hard, then the signature scheme $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}} := \mathsf{FS}^{\mathcal{H}_{\mathsf{FS}}}[\Pi^{\mathcal{H}}]$ achieves EUF-CMA security in the QROM.*

Proof Sketch. The proof idea is similar to that of Theorem 27. In particular, $\mathcal{B}$ simulates $\mathcal{H}_{\mathsf{FS}}$ using the CStO technique [60] in the RO Simulation Phase. Additionally, in the signing simulation phase, $\mathcal{B}$ reprograms the QRO rather than aborts. The probability that $\mathcal{A}$ detects such reprogramming is bounded using Lemma 49. The full security proof is shown in Section C.9

5 PegaRing: Post-Quantum Ring Signature

5.1 Approximate PRF Relation with Arithmetic Progression List

We begin by recalling a generalised list structure using arithmetic progressions, which is applied to improve the efficiency of PegaRing. Let $\mathfrak{I}'$ be a list which includes z sub-lists $\mathfrak{I}'_0, \ldots, \mathfrak{I}'_{z-1}$, where each sub-list $\mathfrak{I}'_j = (a_j, a_j+b, \ldots, a_j+b(c-1))$

for $j \in [0, z-1]$. Each a_j is sampled uniformly from $\mathbb{F}_p$, and the fixed gap $b \in \mathbb{F}_p^*$ is shared across all sub-lists. By setting $L = c \times z$, we can "stretch" the sub-lists and get a list of L field elements. In the set membership proof, we interpret each index $\ell \in [0, L-1]$ as $\ell = z'c + c'$ where $z' = \left\lfloor \frac{\ell}{c} \right\rfloor$ and $c' = \ell \pmod{c}$. Hence, $z' \in [0, z-1]$ denotes the index of the sub-list in $\mathfrak{I}'$, while $c' \in [0, c-1]$ denotes the position of the element in the sub-list $\mathfrak{I}'_{z'}$. Note that $I_\ell = I_{z'c+c'} = a_{z'} + bc'$.

To establish the β-approximate PRF relation with respect to the arithmetic progression list $\mathfrak{I}'$ defined above, we first prove a lemma showing that the distribution of t-th power residue symbols evaluated over the arithmetic progression is close to uniform.

Lemma 29. *For c integers $s_0, \ldots, s_{c-1} \in \mathbb{Z}_t$, a fixed $b \in \mathbb{F}_p^*$, and an odd prime $p > c$ with $t|p-1$,*

$$\left| N_{p,b}(s_0, \ldots, s_{c-1}) - \frac{p}{t^c} \right| < A_{c,t}\sqrt{p} + \frac{c}{t}$$

where $N_{p,b}(s_0, \ldots, s_{c-1}) := \{\#a \in \mathbb{F}_p : \mathcal{L}_0^t(a + \ell b) = s_\ell \text{ for } \ell = 0, \ldots, c-1\}$ and $A_{c,t} = \frac{c(t-1)}{t} - 1 + \frac{1}{t^c}$.

The lemma shows that for a fixed non-zero $b \in \mathbb{F}_p^*$ and a uniformly random chosen $a \in \mathbb{F}_p$, the probability of

$$\mathcal{L}_0^t(a) = s_0, \mathcal{L}_0^t(a+b) = s_1, \ldots, \mathcal{L}_0^t(a + (L-1)b) = s_{L-1}$$

for any $(s_0, \ldots, s_{c-1}) \in \mathbb{Z}_t^c$ is at most $\frac{1}{t^c} + O(\frac{1}{\sqrt{p}})$.

Theorem 30. *Let $\mathfrak{B}(n, q)$ denote the binomial distribution with n samples and the success probability q. Take $K \in \mathbb{F}_p$ and $s = \mathcal{L}_K^t(\mathfrak{I}')$, with probability at least*

$$1 - 2.72 \cdot t(p-1) \cdot \Pr[\mathfrak{B}(L, 1/t) > (1-\beta)L]$$

over the choice of $(a_0, \ldots, a_{z-1})$, there exists only one witness for $s \in R_{\beta\mathcal{L}^t}$, namely K, which is also a witness for the exact relation $R_{\mathcal{L}^t}$.

Following the result of Lemma 29, which establishes the *real* distribution of the PRF sequence over arithmetic progression lists, we prove Theorem 30 by analysing the probability under the *ideal* distribution of the sequences and switch from the ideal case to the real case using the Rényi divergence [48]. We defer full proofs for Lemma 29 and Theorem 30 to Section C.2.

5.2 Protocol Description

We define the setup for PegaRing, which outputs R-pp that includes parameters pp generated by Algorithm 1, replacing $\mathfrak{I}$ with $\mathfrak{I}'$ and adding $z, c \in \mathbb{Z}$ such that $z \times c = L$. Furthermore, R-pp also includes descriptions of hash functions $\mathcal{H}_a : \{0,1\}^* \to \{0,1\}^{2\lambda}$ for $a \in \{\mathsf{FS}, \mathsf{MT}, \mathcal{PK}\}$ and an expand function $\mathsf{expand} : \{0,1\}^\lambda \to \{0,1\}^*$. For $\bar{\ell} \in [0, L-1]$, let $\mathtt{split}(\bar{\ell}) = (\bar{z}, \bar{c})$), where $\bar{z}_e = \left\lfloor \frac{\bar{\ell}}{c} \right\rfloor$, and $\bar{c} = \bar{\ell} \bmod c$. The key generation is defined the same as that in Algorithm 2 and the PegaRing protocol is presented in Algorithm 5.

Following the discussion in Section 1.2, the mutual challenge is defined as the sum of challenges for all ring members $\bar{\ell}_e = \sum_{i=1}^n \bar{\ell}_e^{(i)}$. Consider the simpler case when there is no arithmetic progression. Since $\bar{\ell}'_e = \sum_{i \neq \pi} \bar{\ell}_e^{(i)}$ is fixed before the challenge derivation, this means that the "distance" between the mutual challenge and real signer's challenge is fixed. Hence, if the mutual challenge is I_0, real signer's challenge must be $I_{0-\bar{\ell}'_e \bmod L}$. Therefore, by iterating mutual challenge being $I_0, I_1, \ldots, I_{L-1}$, we get L pairs of possible mutual challenge and real signer's challenge, which leads to L different key share adjustments.

Algorithm 5: PegaRing Signing and Verification

Procedure $\mathtt{R\text{-}Sign}(\mathtt{msg}, sk, \mathcal{PK}, \mathtt{R\text{-}pp})$:

Sample a random salt: $\mathtt{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$.

for $i \in [n], i \neq \pi$ **do**

Compute $h_i \leftarrow \{0,1\}^{\lambda}$ and $\{\bar{\ell}_e^{(i)}\}_{e\in[M]} \leftarrow \mathtt{expand}(h_i)$, where each $\bar{\ell}_e^{(i)} \in [0, L-1]$. Define $(\bar{z}_e^{(i)}, \bar{c}_e^{(i)}) = \mathtt{split}(\bar{\ell}_e)$.

Compute $(\mathtt{decom}, \mathbf{L}) \leftarrow \mathtt{BAVC.TreeGen}(\mathtt{BAVC.pp})$, $(\mathtt{com}, \mathbf{R}) \leftarrow \mathtt{BAVC.LeafCom}(\mathbf{L})$, where $\mathbf{R} = (r_{e,\ell})_{e\in[M],\ell\in[0,L-1]}$.

for $e = 1$ **to** M **do**

Compute $r_e = \sum_{\ell=0}^{L-1} r_{e,\ell}$, $\bar{z}'_e = \sum_{i\neq\pi} \bar{z}_e^{(i)} \pmod z$, $\bar{c}'_e = \sum_{i\neq\pi} \bar{c}_e^{(i)} \pmod c$ and $T_e = \mathcal{L}_0^t(r_e) - \sum_{i=1,i\neq\pi}^{n} pk_{\bar{\ell}_e^{(i)}}^{(i)}$.

for $j = 0$ *to* $z-1$ **do**

Let $\bar{j} = j - \bar{z}'_e \pmod z$, compute $\Delta K_{e,j}^0 = (K - (a_j + b\bar{c}'_e) + a_{\bar{j}})r_e + \sum_{\ell=0}^{L-1} I_\ell r_{e,\ell}$ and $\Delta K_{e,j}^1 = \Delta K_{e,j}^0 + bcr_e$.

Let $b_e \xleftarrow{\$} \{0,1\}$, compute the leaf node as $v_{e,j}^0 = \mathcal{H}_{\mathsf{MT}}(\mathtt{salt}, e, j, 0, \Delta K_{e,j}^{b_e})$, $v_{e,j}^1 = \mathcal{H}_{\mathsf{MT}}(\mathtt{salt}, e, j, 1, \Delta K_{e,j}^{1-b_e})$, and $v_{e,j} = \mathcal{H}_{\mathsf{MT}}(\mathtt{salt}, e, j, v_{e,j}^0, v_{e,j}^1)$.

Compute $\mathtt{com}_{\mathsf{MT}} = \mathtt{MT.Commit}(\mathbf{V})$, where $\mathbf{V} = ((v_{1,0}, \ldots, v_{M,0}), \ldots, (v_{1,z-1}, \ldots, v_{M,z-1}))$.

Compute $h_{\mathcal{PK}} = \mathcal{H}_{\mathcal{PK}}(\mathcal{PK})$ for $\mathcal{PK} = (pk_1, \ldots, pk_n)$ and define $\sigma_1 = (\mathtt{com}, \mathtt{com}_{\mathsf{MT}}, (T_e)_{e\in[M]})$.

Initiate the counter $\mathtt{ctr} = 0$, compute $h = \mathcal{H}_{\mathsf{FS}}(\mathtt{salt}, \sigma_1, \mathtt{msg}, \mathtt{ctr}, h_{\mathcal{PK}})$ and $h_\pi = h[:\lambda] \oplus \bigoplus_{i=1,i\neq\pi}^{n} h_i$.

Expand h_π: $(\bar{\ell}_e^{(\pi)})_{e\in[M]} \leftarrow \mathtt{expand}(h_\pi)$ where for every $e \in [M]$, $\bar{\ell}_e^{(\pi)} \in [0, L-1]$.

for $e = 1$ **to** M **do**

Compute $(\bar{z}_e^{(\pi)}, \bar{c}_e^{(\pi)})$ from $\bar{\ell}_e^{(\pi)}$, define $\bar{z}_e = \sum_{i=1}^{n} \bar{z}_e^{(i)} \pmod z$, $\bar{c}_e = \sum_{i=1}^{n} \bar{c}_e^{(i)} \pmod c$, and $\bar{\ell}_e = \bar{z}_e c + \bar{c}_e$.

If $\bar{c}_e \geq \bar{c}'_e$, set $\Delta K_e = \Delta K_{e,\bar{z}_e}^0$ and v_e to the hash of $\Delta K_{e,\bar{z}_e}^1$. Otherwise, set $\Delta K_e = \Delta K_{e,\bar{z}_e}^1$ and v_e to the hash of $\Delta K_{e,\bar{z}_e}^0$.

Let $d_e \in \{0,1\}$ indicates the hash sequence of ΔK_e and v_e.

Define $S = \{\bar{\ell}_e\}_{e\in[M]}$ and $I = \{\bar{z}_e\}_{e\in[M]}$.

If $\perp \leftarrow \mathtt{BAVC.Open}(\mathtt{decom}, S)$ or $\perp \leftarrow \mathtt{MT.Open}(I, \mathbf{V})$, return to Step 14 with $\mathtt{ctr} = \mathtt{ctr} + 1$. Otherwise, compute $\mathtt{decom}_S \leftarrow \mathtt{BAVC.Open}(\mathtt{decom}, S)$ and $(\mathbf{V}_I, \mathtt{path}_I) \leftarrow \mathtt{MT.Open}(I, \mathbf{V})$.

Let $\sigma = (\mathtt{salt}, \mathtt{ctr}, \mathtt{decom}_S, \mathtt{path}_I, (\Delta K_e, v_e, d_e)_{e\in[M]}), (h_i)_{i\in[n]})$.

Procedure $\mathtt{R\text{-}Verify}(\mathtt{msg}, \mathcal{PK}, \sigma, \mathtt{R\text{-}pp})$:

Parse $\sigma = (\mathtt{salt}, \mathtt{ctr}, \mathtt{decom}_S, \mathtt{path}_I, (\Delta K_e, v_e, d_e)_{e\in[M]}), (h_i)_{i\in[n]})$.

for $i = 1$ **to** n **do**

Expand h_i: $(\bar{\ell}_e^{(i)})_{e\in[M]} \leftarrow \mathtt{expand}(h_i)$ where for every $e \in [M]$, $\bar{\ell}_e^{(i)} \in [0, L-1]$. Define $(\bar{z}_e^{(i)}, \bar{c}_e^{(i)}) = \mathtt{split}(\bar{\ell}_e^{(i)})$

for $e = 1$ **to** M **do**

Compute $\bar{z}_e = \sum_{i=1}^{n} \bar{z}_e^{(i)} \pmod z$, $\bar{c}_e = \sum_{i=1}^{n} \bar{c}_e^{(i)} \pmod c$, and $\bar{\ell}_e = \bar{z}_e c + \bar{c}_e$.

Define $S := \{\bar{\ell}_e\}_{e\in[M]}$ and $I = \{\bar{z}_e\}_{e\in[M]}$.

Recompute Merkle tree leaf based on d_e: If $d_e = 0$, compute $v_{e,\bar{z}_e} = \mathcal{H}_{\mathsf{MT}}(\mathtt{salt}, e, j, \mathcal{H}_{\mathsf{MT}}(\mathtt{salt}, e, j, 0, \Delta K_e), v_e)$. Otherwise ($d_e = 1$), compute $v_{e,\bar{z}_e} = \mathcal{H}_{\mathsf{MT}}(\mathtt{salt}, e, j, v_e, \mathcal{H}_{\mathsf{MT}}(\mathtt{salt}, e, j, 1, \Delta K_e))$.

Define $\mathbf{V}_I = (v_{e,\bar{z}_e})_{e\in[M]}$ and recompute $(\mathtt{com}', \bar{\mathbf{R}})_{e\in[M]}) \leftarrow \mathtt{BAVC.verify}(\mathtt{decom}_S, S)$, where $\bar{\mathbf{R}} = ((r_{e,\ell})_{\ell\in[0,L-1]\setminus\{\bar{\ell}_e\}}$ and $\mathtt{com}'_{\mathsf{MT}} \leftarrow \mathtt{MT.Verify}(\mathbf{V}_I, \mathtt{path}_I, I)$.

for $e \in [M]$ **do**

Compute $o_e = (a_{\bar{z}_e} + b\bar{c}_e) \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} - \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + \Delta K_e$ and $T'_e = \mathcal{L}_0^t(o_e) - \sum_{i=1}^{n} pk_{\bar{\ell}_e^{(i)}}^{(i)}$.

Check if $\bigoplus_{i=1}^{n} h_i \stackrel{?}{=} \mathcal{H}_{\mathsf{FS}}(\mathtt{salt}, \sigma'_1, \mathtt{msg}, \mathtt{ctr}, \mathcal{H}_{\mathcal{PK}}(\mathcal{PK}))[:\lambda]$ where $\sigma'_1 = (\mathtt{com}', \mathtt{com}'_{\mathsf{MT}}, (T'_e)_{e\in[M]})$.

We now explain how the arithmetic progression structure reduces the number of challenge pairs. Recall that every index $\ell \in [0, L-1]$ can be written as $\ell = z'c + c'$, where $z' = \left\lfloor \frac{\ell}{c} \right\rfloor$ and $c' = \ell \bmod c$, and the corresponding list element is $I_\ell = a_{z'} + bc'$. Accordingly, for each non-signer $i \neq \pi$, we write $\bar{\ell}^{(i)} = \bar{z}^{(i)} + b\bar{c}^{(i)}$ and $I_{\bar{\ell}^{(i)}} = a_{\bar{z}^{(i)}} + b\bar{c}^{(i)}$. Since c-coordinate uses the same gap b, the difference between mutual challenge $\bar{c}$ and the real signer's challenge $\bar{c}^{(\pi)}$ is a fixed value $\bar{c}' = \sum_{i \neq \pi} \bar{c}^{(i)}$. However, the elements $a_0, \ldots, a_{z-1}$ are sampled independently, so the difference between z-coordinates of the mutual challenge $a_{\bar{z}}$ and real signer's challenge $a_{\bar{z}^{(\pi)}}$ must be considered separately for every $\bar{z} \in [0, z-1]$. This reduces the number of possible mutual challenge and real signer's challenge pairs by a factor of c. However, since the real signer does not know $\bar{c}$ in advance, and $\bar{c}^{(\pi)} = \bar{c} - \bar{c}'$ or $\bar{c}^{(\pi)} = \bar{c} - \bar{c}' + c$ (i.e., the wrap around), we must consider two possibilities for key share adjustments, corresponding to ΔK^0 and ΔK^1 in Algorithm 5.

5.3 Security Analysis of PegaRing in (Q)ROM

We defer the correctness of PegaRing to Section C.10. Below, we present PegaRing's unforgeability with respect to insider corruption and anonymity against full key exposure in the (Q)ROM.

Theorem 31. *Let $\beta \in [0,1]$ be fixed, $\mathcal{H}_{\mathsf{FS}}$ and $\mathcal{H}_{\mathsf{MT}}$ be modelled as random oracles. If the commit-and-open protocol $\Pi^{\mathcal{H}}$ is a PoK-OE in the ROM and β-approximate PRF relation $\mathcal{R}_{\beta\mathcal{L}^t}$ is hard, then the ring signature scheme PegaRing is unforgeable with respect to the insider corruption in the ROM.*

Proof Sketch. Similar to the reduction from Pegasus to PoK-OE, if an adversary $\mathcal{A}$ produces a valid forge (m^*, σ^*) targeting the challenge public key $pk^* \in \mathcal{PK}$, then it must have queried the random oracles $\mathcal{H}_{\mathsf{FS}}$ and $\mathcal{H}_{\mathsf{MT}}$. Hence, we can extract a transcript for the underlying Σ-protocol (Section 4.1). Then, by the PPT extractor from PoK-OE, we can recover the secret key sk^*, thereby breaking the underlying NP relation $\mathcal{R}_{\beta\mathcal{L}^t}$ associated with (pk^*, sk^*). The full proof is shown in Section C.11.

Theorem 32. *Let $\beta \in [0,1]$ be fixed, $\mathcal{H}_{\mathsf{FS}}$ and $\mathcal{H}_{\mathsf{MT}}$ be modelled as quantum random oracles. If the commit-and-open protocol $\Pi^{\mathcal{H}}$ is a PoK-OE in the QROM and β-approximate PRF relation $\mathcal{R}_{\beta\mathcal{L}^t}$ is hard, then the ring signature scheme PegaRing is unforgeable with respect to the insider corruption in the QROM.*

Proof Sketch. The overall proof strategy is similar to that of Theorem 31, except that the random oracles are simulated using the CStO technique [60]. In the proof, we assume classical access to the signing and corruption oracles, following the assumption in [36]. This is justified since the adversary is modelled as quantum-capable, whereas honest users implement the schemes on classical machines. We again define the database properties SUC and CL similarly to that in the proof of Theorem 26 and bound the quantum transition capacity. We defer the full proof to Section C.12.

Theorem 33. *Our ring signature scheme (PegaRing) is anonymous against full key exposure in the random oracle model if BAVC is hiding. Concretely, an adversary makes q_h random oracle queries and q_s signing oracle queries can win the anonymity game with advantage $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A}}(\lambda) \leq \frac{q_s(q_s+q_h)}{2^{2\lambda}}$.*

Proof Sketch. We prove the anonymity by simulating the signature without using any secret key. The simulation is basically similar to that of Pegasus, which is done by programming the Fiat-Shamir random oracle to a pre-defined challenge. The indistinguishability between the simulated ring signature and the real ring signature is then argued based on the hiding property of BAVC. We move the full proof to Section C.13.

Theorem 34. *Our ring signature scheme (PegaRing) is anonymous against full key exposure in the quantum random oracle model if BAVC is hiding. Concretely, an adversary makes q_h quantum random oracle queries and q_s signing oracle queries can win the anonymity game with advantage* $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A}}(\lambda) \leq \frac{q_s(q_s+q_h)}{2^\lambda}$.

Proof. The proof is similar to that of Theorem 33, except that $\mathcal{B}$ needs to program the QRO. By the OW2H lemma, the probability that any quantum PPT adversary distinguishes this programming is bounded above by $\frac{q_s(q_s+q_h)}{2^\lambda}$. □

6 Implementation and Benchmarking

We present concrete parameter choices for Pegasus and PegaRing. Targeting $\lambda = 128$-bit security, we choose t as a 30-bit square free integer with small prime factors to speed up the computation of t-th power residue PRF. We select a 138-bit prime p satisfying $t|(p-1)$. We set $L = 128$, resulting in a public key of less than 0.5 KB. Given t, p, and L, we compute $\beta = 0.914$ following [9, Theorem 1] for Pegasus. According to Theorem 23, we determine M such that $(1-\beta)^M \leq 2^{-\lambda}$, which yields $M = 37$. For BAVC abort parameter, we set $T_{\mathsf{open}} = \log_2(L) \times M \times 0.9$. Overall, these parameters yield 5,610-byte Pegasus signatures. The QROM instantiation signs in 1.579 ms, including 0.007 ms of online computation, and verifies in 1.599 ms. The corresponding ROM instantiation signs in 0.878 ms, requires 0.003 ms online, and verifies in 0.876 ms. Compared to prior PRF-based signature [9], Pegasus reduces the signature size by up to 1.5×, and the public key size by more than 8×.

For PegaRing with same parameters as above, the condition in Theorem 30 yields $c = 2$ and $z = L/c = 64$. We report signature size, signing and verification time with respect to more ring sizes in Table 2.[5] Our implementation is written in C/C++ with AVX2 optimisations applied to hash functions. All experiments were conducted on a machine running Ubuntu 24.04.4 LTS (Linux kernel 6.16.0) equipped with an Intel Core i7-12700KF CPU (base frequency 3.6GHz) and 32 GB of RAM. All benchmarks were executed on a single core.

Table 2. Performance of PegaRing (QROM) with Different Ring Sizes.

Ring Size	16	64	256	1024
Size (KB)	12.64	13.39	16.39	28.39
Ring Prep. (ms)	0.017	0.067	0.272	1.073
Sign (ms)	3.224	3.160	3.223	3.252
Online (ms)	0.017	0.017	0.018	0.018
Verify (ms)	1.936	1.907	1.945	2.001

7 Conclusion

We presented PegaRing, the first post-quantum ring signature to combine practical performance with concretely parameterised security in the QROM. Our construction begins with a three-move commit-and-open Σ-protocol that exploits the OLE structure inherent in Legendre and power residue PRF evaluations. Applying Fiat-Shamir transform to the Σ-protocol, we design Pegasus, an efficient standalone signature scheme that improves substantially upon the previous power residue PRF-based construction [9] and serves as the foundation of PegaRing.

Building on the DualRing framework, we developed a three-move set-membership protocol for the power residue PRFs. Our construction eliminates the additional cut-and-choose step required by the previous PRF-enabled instantiation

[5]We separate the time for ring preparation (i.e., hash the public keys in the ring) from signing and verification. However, Table 1 report total signing and verification time that includes the preparation time.

[62] while preserving the commit-and-open structure needed by our QROM analysis. We further generalised the β-approximate PRF relation to arithmetic progression lists and established its hardness through a new analysis based on Rényi divergence. We prove unforgeability against insider corruption and anonymity under full key exposure of our ring signature in both the ROM and QROM.

Targeting 128-bit security and a ring of 1, 024 members, PegaRing produces 28.39 KB signatures, signs in 3.334 ms, and verifies in 2.033 ms, while having only 0.018 ms of message-dependent online signing. Its signature is approximately 87× smaller than the smallest previously reported concretely parameterised QROM-secure construction [22]. When compared with state-of-the-art ring signature based on VOLE-in-the-head [17], PegaRing reduces signing and verification latencies by a factor of 2.65 and 3.80, respectively, for a ring of size 1, 024. Furthermore, under the same online-offline boundary, PegaRing achieves more than 400× faster online signing, while additional providing QROM security.

Future Works. The current PegaRing scheme has a signature size that grows linearly with the number of ring members. Constructing a variant with logarithmic-size signatures is non-trivial. The main challenge lies in the lack of an efficient, tailored sum argument (based on symmetric key primitives) that can simultaneously prove the relation between users' challenges and the mutual challenge, while also demonstrating that the public key components used in verification correspond to the claimed challenge indices.

Acknowledgments

We sincerely thank Muhammed Esgin, Dongxi Liu and Sushmita Ruj for their help with the early draft of this paper

References

[1] Andris Ambainis, Mike Hamburg, and Dominique Unruh. 2019. Quantum Security Proofs Using Semi-classical Oracles. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*. 269–295.

[2] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. 2014. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 474–483.

[3] Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. 2024. One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures. In *ASIACRYPT (1) (Lecture Notes in Computer Science, Vol. 15484)*. Springer, 463–493.

[4] Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. 2025. One tree to rule them all: Optimizing ggm trees and owfs for post-quantum signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 463–493.

[5] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. 2023. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In *Annual International Cryptology Conference*. Springer, 581–615.

[6] Marvin Beckmann and Christian Majenz. 2026. Quantum Oracle Distribution Switching and its Applications to Fully Anonymous Ring Signatures. *arXiv preprint arXiv:2602.16268* (2026).

[7] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *CCS*. ACM, 62–73.

[8] Ward Beullens, Tim Beyne, Aleksei Udovenko, and Giuseppe Vitto. 2020. Cryptanalysis of the Legendre PRF and Generalizations. *IACR Transactions on Symmetric Cryptology* (2020), 313–330.

[9] Ward Beullens and Cyprien Delpech de Saint Guilhem. 2020. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In *PQCrypto*. Springer, 130–150.

[10] Ward Beullens, Shuichi Katsumata, and Federico Pintore. 2020. Calamari and Falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In *ASIACRYPT*. Springer, 464–492.

[11] Markus Bläser, Zhili Chen, Dung Hoang Duong, Antoine Joux, Tuong Nguyen, Thomas Plantard, Youming Qiao, Willy Susilo, and Gang Tang. 2024. On digital signatures based on group actions: QROM security and ring signatures. In *International Conference on Post-Quantum Cryptography*. Springer, 227–261.

[12] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. 2011. Random oracles in a quantum world. In *International conference on the theory and application of cryptology and information security*. Springer, 41–69.

[13] Giacomo Borin, Yi-Fu Lai, and Antonin Leroux. 2024. Erebor and durian: Full anonymous ring signatures from quaternions and isogenies. *IACR Communications in Cryptology* (2024).

[14] Dung Bui, Kelong Cong, and Cyprien Delpech de Saint Guilhem. 2025. Faster VOLEitH Signatures from All-But-One Vector Commitment and Half-Tree. In *Australasian Conference on Information Security and Privacy*. Springer, 205–223.

[15] Rohit Chatterjee, Kai-Min Chung, Xiao Liang, and Giulio Malavolta. 2022. A note on the post-quantum security of (ring) signatures. In *IACR International Conference on Public-Key Cryptography*. Springer, 407–436.

[16] David Chaum and Eugène Van Heyst. 1991. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 257–265.

[17] James Hsin-Yu Chiang, Ivan Damgård, William R Duro, Sunniva Engan, Sebastian Kolby, and Peter Scholl. 2025. Post-Quantum Threshold Ring Signature Applications from VOLE-in-the-Head. *Cryptology ePrint Archive* (2025).

[18] Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. 2021. On the Compressed-Oracle Technique, and Post-Quantum Security of Proofs of Sequential Work. In *EUROCRYPT (2) (Lecture Notes in Computer Science, Vol. 12697)*. Springer, 598–629.

[19] Kai-Min Chung, Yao-Ching Hsieh, Mi-Ying Huang, Yu-Hsuan Huang, Tanja Lange, and Bo-Yin Yang. 2021. Isogeny-based group signatures and accountable ring signatures in QROM. *arXiv preprint arXiv:2110.04795* (2021).

[20] Hongrui Cui, Hanlin Liu, Di Yan, Kang Yang, Yu Yu, and Kaiyi Zhang. 2024. Resolved: Shorter signatures from regular syndrome decoding and vole-in-the-head. In *IACR International Conference on Public-Key Cryptography*. Springer, 229–258.

[21] Ivan Bjerre Damgård. 1988. On the randomness of Legendre and Jacobi sequences. In *ASIACRYPT*. Springer, 163–172.

[22] David Derler, Sebastian Ramacher, and Daniel Slamanig. 2018. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *PQCrypto*. Springer, 419–440.

[23] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. 2022. Efficient NIZKs and signatures from commit-and-open protocols in the QROM. In *Annual International Cryptology Conference*. Springer, 729–757.

[24] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. 2022. Efficient NIZKs and Signatures from Commit-and-Open Protocols in the QROM. In *CRYPTO (2) (Lecture Notes in Computer Science, Vol. 13508)*. Springer, 729–757.

[25] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. 2022. Online-Extractability in the Quantum Random-Oracle Model. In *EUROCRYPT (3) (Lecture Notes in Computer Science, Vol. 13277)*. Springer, 677–706.

[26] Priyanka Dutta, Willy Susilo, Fuchun Guo, and Dung Hoang Duong. 2026. Lattice-based logarithmic-size forward-secure ring signatures in QROM. *Theoretical Computer Science* 1068 (2026), 115787.

[27] Muhammed F Esgin, Ron Steinfeld, and Raymond K Zhao. 2022. MatRiCT+: More efficient post-quantum private blockchain payments. In *IEEE S&P*. IEEE, 1281–1298.

[28] Saeed Banaeian Far and Maryam Rajabzadeh Asaar. 2024. A blockchain-based anonymous reporting system with no central authority: Architecture and protocol. *Cyber Security and Applications* 2 (2024), 100032.

[29] Hanwen Feng, Jianwei Liu, Dawei Li, Ya-Nan Li, and Qianhong Wu. 2021. Traceable ring signatures: general framework and post-quantum security. *Designs, Codes and Cryptography* 89, 6 (2021), 1111–1145.

[30] Hanwen Feng, Jianwei Liu, Qianhong Wu, and Ya-Nan Li. 2020. Traceable ring signatures with post-quantum security. In *Cryptographers' Track at the RSA Conference*. Springer, 442–468.

[31] Amos Fiat and Adi Shamir. 1986. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*. Springer, 186–194.

[32] Paul Frixons and André Schrottenloher. 2021. Quantum security of the legendre prf. *Mathematical Cryptology* 1, 2 (2021), 52–69.

[33] Phillip Gajland, Jonas Janneck, and Eike Kiltz. 2024. Ring signatures for deniable AKEM: Gandalf's fellowship. In *Annual International Cryptology Conference*. Springer, 305–338.

[34] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. 2022. Efficient Set Membership Proofs using MPC-in-the-Head. *Proceedings on Privacy Enhancing Technologies* 2 (2022), 304–324.

[35] Xiaojie Guo, Kang Yang, Xiao Wang, Wenhao Zhang, Xiang Xie, Jiang Zhang, and Zheli Liu. 2023. Half-tree: Halving the cost of tree expansion in COT and DPF. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 330–362.

[36] Abida Haque and Alessandra Scafuro. 2020. Threshold ring signatures: new definitions and post-quantum security. In *IACR International Conference on Public-Key Cryptography*. Springer, 423–452.

[37] Novak Kaludjerović, Thorsten Kleinjung, and Dusan Kostic. 2020. Improved key recovery on the Legendre PRF. *Cryptology ePrint Archive* (2020).

[38] Shuichi Katsumata, Guilhem Niot, Ida Tucker, and Thom Wiggers. 2025. Comprehensive Deniability Analysis of Signal Handshake Protocols:{X3DH},{PQXDH} to Fully {Post-Quantum} with Deniable Ring Signatures. In *34th USENIX Security Symposium (USENIX Security 25)*. 6797–6816.

[39] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. 2018. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *Proceedings of the 2018 ACM CCS*. 525–537.

[40] Dmitry Khovratovich. 2019. Key recovery attacks on the Legendre PRFs within the birthday bound. *Cryptology ePrint Archive* (2019).

[41] Guangyu Liao, Zengpeng Li, Guangsheng Feng, Mei Wang, and Hongwu Lv. 2025. Purse: post-quantum unique ring signature for anonymous transactions. *IEEE Internet of Things Journal* (2025).

[42] Vadim Lyubashevsky and Ngoc Khanh Nguyen. 2022. BLOOM: Bimodal Lattice One-out-of-Many Proofs and Applications. In *ASIACRYPT*. Springer, 95–125.

[43] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. 2021. SMILE: set membership from ideal lattices with applications to ring signatures and confidential transactions. In *CRYPTO*. Springer, 611–640.
[44] Alexander May and Floyd Zweydinger. 2022. Legendre PRF (multiple) key attacks and the power of preprocessing. In *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*. IEEE, 428–438.
[45] Ralph C Merkle. 1989. A certified digital signature. In *Asiacrypt*. Springer, 218–238.
[46] Shen Noether, Adam Mackenzie, et al. 2016. Ring confidential transactions. *Ledger* 1 (2016), 1–18.
[47] José Ignacio Escribano Pablos, María Isabel González Vasco, Ángel Pérez del Pozo, and Claudio Soriente. 2025. Anonymous authenticated key exchange. In *International Conference on Applied Cryptography and Network Security*. Springer, 516–546.
[48] Alfréd Rényi. 1961. On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*, Vol. 4. University of California Press, 547–562.
[49] Ronald L Rivest, Adi Shamir, and Yael Tauman. 2001. How to leak a secret. In *ASIACRYPT*. Springer, 552–565.
[50] Lawrence Roy. 2022. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In *Annual International Cryptology Conference*. Springer, 657–687.
[51] István András Seres, Máté Horváth, and Péter Burcsi. 2023. The Legendre pseudorandom function as a multivariate quadratic cryptosystem: security and applications. *Applicable Algebra in Engineering, Communication and Computing* (2023), 1–31.
[52] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.
[53] Patrick P Tsang and Victor K Wei. 2005. Short linkable ring signatures for e-voting, e-cash and attestation. In *International Conference on Information Security Practice and Experience*. Springer, 48–60.
[54] Dominique Unruh. 2015. Revocable quantum timed-release encryption. *Journal of the ACM (JACM)* 62, 6 (2015), 1–76.
[55] Dominique Unruh. 2015. Revocable Quantum Timed-Release Encryption. *J. ACM* 62, 6 (2015), 49:1–49:76.
[56] Wim van Dam and Sean Hallgren. 2000. Efficient quantum algorithms for shifted quadratic character problems. *arXiv preprint quant-ph/0011067* (2000).
[57] Wim Van Dam, Sean Hallgren, and Lawrence Ip. 2006. Quantum algorithms for some hidden shift problems. *SIAM J. Comput.* 36, 3 (2006), 763–778.
[58] Yuxi Xue, Xingye Lu, Man Ho Au, and Chengru Zhang. 2024. Efficient linkable ring signatures: new framework and post-quantum instantiations. In *European Symposium on Research in Computer Security*. Springer, 435–456.
[59] Tsz Hon Yuen, Muhammed F Esgin, Joseph K Liu, Man Ho Au, and Zhimin Ding. 2021. DualRing: generic construction of ring signatures with efficient instantiations. In *Annual International Cryptology Conference*. Springer, 251–281.
[60] Mark Zhandry. 2019. How to Record Quantum Queries, and Applications to Quantum Indifferentiability. In *CRYPTO (2) (Lecture Notes in Computer Science, Vol. 11693)*. Springer, 239–268.
[61] Xinyu Zhang, Ron Steinfeld, Muhammed F Esgin, Joseph K Liu, Dongxi Liu, and Sushmita Ruj. 2024. Loquat: A SNARK-Friendly Post-quantum Signature Based on the Legendre PRF with Applications in Ring and Aggregate Signatures. In *Annual International Cryptology Conference*. Springer, 3–38.
[62] Xinyu Zhang, Ron Steinfeld, Joseph K Liu, Muhammed F Esgin, Dongxi Liu, and Sushmita Ruj. 2024. DualRing-PRF: Post-quantum (Linkable) Ring Signatures from Legendre and Power Residue PRFs. In *Australasian Conference on Information Security and Privacy*. Springer, 124–143.

Appendix

A Additional Definitions

A.1 Additional Notations

Let B be a Boolean statement, $\|B\|$ denotes a bit, which is 1 if B is true, and 0 otherwise. We use $\otimes$ to represent tensor product. For a finite set S, the cardinality of S is denoted by $|S|$, and $x \xleftarrow{\$} S$ means that x is uniformly random sampled from S. For two sets S_1 and S_2, $S_1 \times S_2$ denotes the Cartesian product of them. Let D be a distribution, and let $x \leftarrow D$ denote sampling x from D. We use $\mathsf{Dis}(D_1, D_2)$ to denote the statistical distance between two distributions D_1 and D_2. For an algorithm $\mathcal{A}$, given an input string inp, we write $\mathsf{out} \leftarrow \mathcal{A}(\mathsf{inp})$ (or equivalently $\mathcal{A}(\mathsf{out}) \rightarrow \mathsf{inp}$) to denote that out is the output of $\mathcal{A}$ on input inp. If $\mathcal{A}$ is deterministic, then out is fixed and denoted by $\mathsf{out} := \mathcal{A}(\mathsf{inp})$. We may also use the notation $\mathsf{out} \leftarrow \mathcal{A}(\mathsf{inp}; r)$ to make the randomness r explicit for a probabilistic algorithm $\mathcal{A}$. When $\mathcal{A}$ has oracle access to O, we write $\mathcal{A}^O$.

A.2 Security Models

The Random Oracle Model (ROM), introduced by Bellare and Rogaway [7], idealizes a hash function as a truly random function accessible only through black-box queries. Depending on how the adversary interacts with the oracle, the model is classified into the classical ROM and the quantum ROM.

A.2.1 Random Oracle Model. In the classical Random Oracle Model (ROM), algorithms can only access the random oracle through classical queries. The oracle is typically simulated via lazy sampling, where an adversary cannot distinguish between interactions with the true random oracle and with the lazy sampling simulation (Lemma 35).

Lemma 35. *There exists a PPT algorithm $\mathcal{S}$ that simulates the classical random oracle for a hash function $H : \{0,1\}^* \to \{0,1\}^n$ by maintaining a hash list L_H, generated via lazy sampling. For any PPT adversary $\mathcal{A}$, we have*

$$\Pr[\mathcal{A}^{O_H}(1^\lambda) \to 1] = \Pr[\mathcal{A}^{\mathcal{S}}(1^\lambda) \to 1],$$

where $\mathcal{S}$ is formally described as follows:

- *Initialize $L_H = \emptyset$;*
- *If $\exists\ (x, y) \in L_H$:* **return** *y;*
- *Otherwise, sample $y \xleftarrow{\$} Y$, update $L_H \leftarrow L_H \cup (x, y)$, and* **return** *$y$.*

During the simulation, the simulator records query-response pairs in a transcript, which it can later inspect to extract useful information (e.g., witness).

A.2.2 Quantum Random Oracle Model. In the Quantum Random Oracle Model (QROM), the hash function is idealized as a publicly and quantum-accessible random oracle. Generally, the quantum computation corresponding to any function $H : X \to Y$ can be represented as a unitary transformation U_H:

$$|x\rangle_{\mathsf{X}} |y\rangle_{\mathsf{Y}} |H\rangle_{\mathsf{H}} \to |x\rangle_{\mathsf{X}} |y \oplus H(x)\rangle_{\mathsf{Y}} |H\rangle_{\mathsf{H}},$$

where $x \in X$, $y \in Y$ and X, Y, and H denote the registers that store x, y, and H, respectively.

Let $\mathcal{H} := \{H | H : X \to Y\}$. A QRO can be defined analogously to the classical RO by selecting a function $H \xleftarrow{\$} \mathcal{H}$ uniformly at random from the family of functions. For each query $\sum_x \alpha_x |x, y_x\rangle$, the QRO performs the following operation:

$$\mathsf{O}_H : \sum_x \alpha_x |x, y_x\rangle \to \sum_x \alpha_x |x, y_x \oplus H(x)\rangle .$$

and returns $\sum_x \alpha_x |x, y_x \oplus H(x)\rangle$.

In addition, we may consider another type of QRO, referred to as the Phase Oracle (PhO). For each basis $|x\rangle$ and $|y\rangle$, the PhO performs the following operation:

$$\mathsf{PhO}_H : |x\rangle |y\rangle \to (-1)^{y \cdot H(x)} |x\rangle |y\rangle .$$

Lemma 36 ($\mathsf{O}_H \Leftrightarrow \mathsf{PhO}_H$ (Lemma 3. in [60]))**.** *For any adversary $\mathcal{A}$ makes queries to O_H, let $\mathcal{B}$ be the adversary that is identical to $\mathcal{A}$, except it performs the Hadamard transformation $\mathsf{H}^{\otimes n}$ to the response registers before and after each query. Then $\Pr[\mathcal{A}^{O_H}() \to 1] = \Pr[\mathcal{B}^{PhO_H}() \to 1]$.*

Remark 37. We introduce the Hadamard transform, which maps computational basis states to the Fourier basis. Let $Y = \{0,1\}^n$. For each $y \in Y$, the n-bit Hadamard transform is defined as

$$\mathsf{H}^{\otimes n} : |y\rangle \to |\widehat{y}\rangle,$$

where $|\widehat{y}\rangle = \frac{1}{\sqrt{2^n}} \sum_{y' \in \{0,1\}^n} (-1)^{y \cdot y'} |y'\rangle$. It is obvious that $\mathsf{H}^{\otimes n}(|0\rangle) = |\tau\rangle$, where τ is the uniform superposition state.

From an equivalent perspective, the QRO can be initialized as a uniform superposition over all functions $\sum_H \frac{1}{\sqrt{|\mathcal{H}|}} |H\rangle$, where $\mathcal{H} := \{H : X \to Y\}$. This type of QRO is referred to as the Standard Oracle (StO), as formalized in Definition 38.

Definition 38 (StO [60])**.** The StO oracle is initialized in a uniform superposition over all functions $\mathcal{H}$ with domain X and range Y. For any basis state $|x\rangle$ and $|y\rangle$, the oracle performs the following unitary operation:

$$|x\rangle |y\rangle \otimes \sum_H \frac{1}{\sqrt{|\mathcal{H}|}} |H\rangle \to \sum_H \frac{1}{\sqrt{|\mathcal{H}|}} |x\rangle |y \oplus H(x)\rangle |H\rangle,$$

where $x \in X$, $y \in Y$, and $\mathcal{H}$ denotes the set of all functions with domain X and range Y.

For a function $H : X \to Y$, it can be represented as

$$H(x_0) \| H(x_1) \| \cdots \| H(x_{2^m-1}),$$

where $X := \{0,1\}^m$ and $x_i \in X$ for $i \in \{0, 1, \cdots, 2^m - 1\}$.

Based on this perspective, the internal state of the quantum oracle corresponding to such a function can be expressed as $\otimes_{x \in X} |H(x)\rangle$. In particular, the internal state of the StO can be represented by $\otimes_{x \in X} |\tau\rangle$. We emphasize that this representation is purely conceptual and is only used for analysis; it does not correspond to a physically realizable oracle. We redefine the unitary operation $\mathsf{StO} := \mathsf{I} \otimes \mathsf{CNOT}_{\mathsf{Y},\mathsf{D}_x} \otimes \mathsf{I}^{\otimes 2^m}$:

$$\mathsf{StO} : |x\rangle |y\rangle \otimes |\tau\rangle^{\otimes 2^m} \to \frac{1}{\sqrt{2^n}} \sum_{y_x \in \{0,1\}^n} |x\rangle |y \oplus y_x\rangle |y_x\rangle_{\mathsf{D}_x} \otimes |\tau\rangle^{\otimes 2^m - 1},$$

where D_x the register that stores the output corresponding to x, for each $x \in X$.

Remark 39. The CNOT gate is a two-register unitary operation acting on a control register C and a target register T. Its action on computational basis states is defined as

$$\mathsf{CNOT}_{\mathsf{C},\mathsf{T}} : |c\rangle |t\rangle \mapsto |c\rangle |t \oplus c\rangle, \quad c, t \in \{0,1\}.$$

For n-bit registers Y (control) and D_x (target), the n-bit CNOT is defined as

$$\mathsf{CNOT}_{\mathsf{D}_x,\mathsf{Y}} : |y\rangle |y_x\rangle \mapsto |y\rangle |y_x \oplus y\rangle, \quad y, y_x \in \{0,1\}^n.$$

A.2.3 Compressed Random Oracle. It can be observed that a query on $|x\rangle$ affects the register D_x, thereby entangling D_x with the query registers X and Y into a joint quantum system. For simplicity, we therefore focus on the action of the unitary

$$\mathsf{U}_1 := \mathsf{I}_\mathsf{X} \otimes \mathsf{CNOT}_{\mathsf{D}_x,\mathsf{Y}}$$

which acts as

$$\mathsf{U}_1 : |x\rangle |y\rangle |y_x\rangle \mapsto |x\rangle |y \oplus y_x\rangle |y_x\rangle$$

in the computational basis.

We now analyze the effect of U_1 in the Fourier basis. For each $y \in \{0,1\}^n$, define the Fourier basis state

$$|\widehat{y}\rangle := \mathsf{H}^{\otimes n} |y\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{y \cdot z} |z\rangle.$$

It is a well-known property of the Hadamard transform and the CNOT gate that conjugating $\mathsf{CNOT}_{\mathrm{D}_x,\mathrm{Y}}$ by Hadamard gates on both registers swaps the direction of control and target. That is,

$$(\mathsf{H}^{\otimes n} \otimes \mathsf{H}^{\otimes n})\, \mathsf{CNOT}_{\mathrm{D}_x,\mathrm{Y}}\, (\mathsf{H}^{\otimes n} \otimes \mathsf{H}^{\otimes n}) = \mathsf{CNOT}_{\mathrm{Y},\mathrm{D}_x}.$$

Consequently, in the Fourier basis, the unitary U_1 acts as

$$|x\rangle\,|\widehat{y}\rangle\,|\widehat{y_x}\rangle \;\longmapsto\; |x\rangle\,|\widehat{y}\rangle\,|\widehat{y_x \oplus y}\rangle\,.$$

Extending this action to all output registers of the oracle, we define the Fourier-basis standard oracle FStO as

$$\mathsf{FStO} : |x\rangle\,|\widehat{y}\rangle\,|\widehat{y_0}\rangle \cdots |\widehat{y_{2^m-1}}\rangle \longmapsto |x\rangle\,|\widehat{y}\rangle\,|\widehat{y_0}\rangle \cdots |\widehat{y_x \oplus y}\rangle \cdots |\widehat{y_{2^m-1}}\rangle\,.$$

The above transformation shows that quantum queries necessarily entangle the query registers with the internal oracle state, thereby recording the queried input and its corresponding response in the oracle's memory. This observation underlies Zhandry's compressed standard random oracle (CStO) framework [60], which provides a powerful tool for simulating quantum random oracles.

Let us define the compression operator acting on the Fourier basis registers of D_x as

$$\mathsf{Comp}_x := |\perp\rangle\langle\widehat{0}| + \sum_{\widehat{z}\neq\widehat{0}} |\widehat{z}\rangle\langle\widehat{z}|\,, \tag{1}$$

where $|\widehat{y}\rangle = \mathsf{H}^{\otimes n}|y\rangle$ is the Fourier basis state corresponding to the computational basis state $|y\rangle$. That is, Comp_x acts as

$$\mathsf{Comp}_x : |\widehat{y}\rangle \mapsto \begin{cases} |\perp\rangle\,, & \widehat{y} = \widehat{0}, \\ |\widehat{y}\rangle\,, & \widehat{y} \neq \widehat{0}. \end{cases}$$

Define the global compression operator as

$$\mathsf{Comp} := \bigotimes_{x\in X} \mathsf{Comp}_x.$$

Definition 40 (Compressed StO on the Fourier basis (CFStO) [18, 60])**.** The Compressed StO on the Fourier basis (CFStO) is defined as

$$\mathsf{CFStO} := \mathsf{Comp} \circ \mathsf{FStO} \circ \mathsf{Comp}, \tag{2}$$

Due to the exponential size of the internal registers, the above representation is inefficient. The StO can be efficiently simulated using the techniques of [60].

Definition 41 (CStO [18, 25, 60])**.** Let $Y' := Y \cup \{\perp\}$. Consider a sparse classical encoding function Encode, which compresses a database $\mathbf{y} = \{y_x\}_{x\in X}$ with at most q non-$\perp$ entries into

$$L_q := \{(x_1, y_1), \ldots, (x_l, y_l), \perp\}.$$

The entries (x_i, y_i) are sorted according to the order of x_i, and $\perp$ represents the absence of a value. Then, for any basis state $|\mathbf{y}\rangle$, we define

$$\mathsf{Encode} : |\mathbf{y}\rangle \mapsto |x_1, y_1\rangle \cdots |x_l, y_l\rangle\,|\perp\rangle\,,$$

and its inverse is Decode.

Let $D = \bigotimes_{x\in X} D_x$ denote the oracle's internal state, where each D_x lives in $\mathcal{H}_{D_x} = \mathbb{C}[\{0,1\}^n \cup \{\perp\}]$. The CStO is defined as

$$\mathsf{CStO} := \mathsf{Encode} \circ \mathsf{Comp} \circ \mathsf{F} \circ \mathsf{Comp} \circ \mathsf{Decode}, \tag{3}$$

where F is the unitary acting on X, Y, and D defined as

$$\mathsf{F} : |x, y\rangle\,|D\rangle \mapsto |x, y \oplus y_x\rangle\,|D\rangle\,, \quad (x, y_x) \in D.$$

The CStO technique provides a perfect simulation of the quantum random oracle (QRO).

Lemma 42 (CStO $\Leftrightarrow$ StO [60])**.** *The compressed standard oracle* CStO *and the standard oracle* StO *are perfectly indistinguishable.*

Lemma 43 (Adapted from Lemma 2.6 in [24])**.** *Let $\mathcal{A}$ be a quantum oracle algorithm that outputs $\mathbf{x} = (x_1, \ldots, x_n) \in X^n$ and some auxiliary register $z \in Z$. Define $\tilde{\mathcal{A}}$ as the algorithm that runs $\mathcal{A}$, then makes classical queries on each x_i to obtain $\mathbf{y} = (y_1, \ldots, y_n) = H(\mathbf{x})$, and finally outputs $(\mathbf{x}, \mathbf{y}, z)$.*

Suppose that instead of the full standard oracle, $\tilde{\mathcal{A}}$ interacts with a compressed standard oracle (CStO), and let D denote the internal oracle state obtained by measuring the compressed oracle at the end of the execution. Then, conditioned on the output $(\mathbf{x}, \mathbf{y}, z)$ of $\tilde{\mathcal{A}}$,

$$\Pr\left[\mathbf{y} = D(\mathbf{x}) \,\middle|\, (\mathbf{x}, \mathbf{y}, z)\right] \geq 1 - \frac{2n}{|Y|},$$

where $|Y|$ is the size of the oracle output space.

A.3 Database Property for QROM

We now formalize the database properties used in the analysis of the Compressed Standard Oracle (CStO).

Let D denote the set of internal states of the random oracle during the query process, which we refer to as the *database*. Formally, we define $\mathcal{D} := \{D : X \to Y \cup \{\perp\}\}$.

Definition 44 (Database Property (Def. 5.1 in [18]))**.** A *database property* on $\mathcal{D}$ is a subset $\mathsf{P} \subseteq \mathcal{D}$.

We introduce the following notations for manipulating database properties:

- For any database $D \in \mathcal{D}$, let $D[x \to y] \in \mathcal{D}$ denote the database obtained by updating D at x, i.e.,
$$D[x \to y](x') := \begin{cases} y & \text{if } x' = x, \\ D(x') & \text{otherwise.} \end{cases}$$
- The set of databases that coincide with D outside of x is denoted by
$$D|^x := \{D[x \to y] \mid y \in Y \cup \{\perp\}\}.$$
- Furthermore, for a database property P, the notation $\mathsf{P}|_{D|^x}$ denotes the set of values y such that the updated database $D[x \to y]$ satisfies P.
$$\mathsf{P}|_{D|^x} := \{y \in Y \cup \{\perp\} | D[x \to y] \subseteq \mathsf{P}\}.$$

Definition 45 (Quantum Transition Capacity (Def. 5.5 [18]))**.** Let $\mathsf{P}, \mathsf{P}' \subseteq \mathcal{D}$ be two database properties. The quantum transition capacity of a single query from P to P' is defined as

$$[\![\mathsf{P} \to \mathsf{P}']\!] := \max_{x,y,D} \left\| \mathsf{P}'|_{D|^x}\, \mathsf{CStO}_{x,y}\, \mathsf{P}|_{D|^x} \right\|,$$

where the maximization is over all $x \in X$, $y \in Y$, and $D \in \mathcal{D}$.

Furthermore, for q sequential queries, we define the *quantum transition capacity over q steps* as

$$[\![\mathsf{P} \xrightarrow{q} \mathsf{P}']\!] := \sup_{U_1,\ldots,U_{q-1}} \left\| \mathsf{P}' \,\mathrm{CStO}\, U_{q-1}\, \mathrm{CStO} \cdots \mathrm{CStO}\, U_1\, \mathrm{CStO}\, \mathsf{P} \right\|,$$

where the supremum is taken over all unitaries $U_1, \ldots, U_{q-1}$ acting on $\mathbb{C}[X] \otimes \mathbb{C}[Y] \otimes \mathcal{H}_D$.

Lemma 46 (Database Properties Sequence (Lemma 5.6 [24]))**.** *For any sequence of database properties* $\mathsf{P}_0, \mathsf{P}_1, \ldots, \mathsf{P}_q$*, the following holds:*

$$[\![\neg\mathsf{P}_0 \xrightarrow{\$} \mathsf{P}_q]\!] \le \sum_{s=1}^{q} [\![\neg\mathsf{P}_{s-1} \xrightarrow{\$} \mathsf{P}_s]\!].$$

We have the following theorem to bound quantum transition capacities.

Theorem 47 ([18, 24])**.** *Let* P *and* P' *be database properties with trivial intersection, i.e.,* $\mathsf{P} \cap \mathsf{P}' = \emptyset$*. For every* $D \in \mathcal{D}$ *and* $x \in X$*, define*

$$\mathsf{L}^{x,D} := \begin{cases} \mathsf{P}|_{D|^x}, & \text{if } \perp \in \mathsf{P}'|_{D|^x}, \\ \mathsf{P}'|_{D|^x}, & \text{if } \perp \in \mathsf{P}|_{D|^x}, \end{cases}$$

and let $\mathsf{L}^{x,D}$ *be either of the two if* $\perp \notin \mathsf{P}|_{D|^x} \cup \mathsf{P}'|_{D|^x}$*. Then,*

$$[\![\mathsf{P} \xrightarrow{\$} \mathsf{P}']\!] \le \max_{x,D} \sqrt{10 \Pr[U \in \mathsf{L}^{x,D}]},$$

where U *is uniform over* Y*. The maximization can be restricted to* $D \in \mathcal{D}$ *and* $x \in X$ *for which both* $\mathsf{P}|_{D|^x}$ *and* $\mathsf{P}'|_{D|^x}$ *are non-empty.*

Lemma 48 ([24])**.** *Denote the collision database by*

$$\mathsf{CL} := \{D \in \mathcal{D} : \exists\, x \neq x' \text{ such that } D(x) = D(x')\},$$

where $\mathcal{D} : X \to Y$*. Then, for the set of databases with at most* i *non-*$\perp$ *entries outside of* CL*, we have*

$$[\![\mathsf{SZ}_{\le i} \setminus \mathsf{CL} \to \mathsf{CL}]\!] \le 2e\sqrt{\frac{i+1}{|Y|}},$$

where $\mathsf{SZ}_{\le i} := \{D : \#\{z : D(z) \neq \perp\} \le s\}$.

A.3.1 Programming of QRO. Based on the original OW2H lemma proposed by Unruh [54], several refined versions have been introduced to reduce the reduction loss. In this subsection, we present the Semi-Classical OW2H (SC-OW2H) lemma [1].

Lemma 49 (SC-OW2H, [1])**.** *Let* $S \subseteq X$ *be random. Let* $G, H : X \to Y$ *be random functions satisfying* $\forall x \notin S$*, G(x)=H(x). Let* z *be a random bitstring. (*S, G, H, z *may have arbitrary joint distribution.) Let* $\mathcal{A}$ *be an oracle algorithm of query depth* d*. Let* $\mathcal{B}^H$ *be an oracle algorithm that on input* z *does the following: pick* $i \xleftarrow{\$} \{1, \cdots, d\}$*, run* $\mathcal{A}^H(z)$ *untill (just before) the* i*-th query, measure all query input registers in the computations basis, output the set* T *of measurement outcomes. Let*

$$P_{\text{left}} := \Pr[b \leftarrow \mathcal{A}^H(z) : b = 1],$$

$$P_{\text{right}} := \Pr[b \leftarrow \mathcal{A}^G(z) : b = 1],$$

and

$$P_{\text{guess}} := \Pr[S \cap T \neq \emptyset : T \leftarrow \mathcal{B}^H(z)].$$

Then

$$|P_{\text{left}} - P_{\text{right}}| \leq 2d\sqrt{P_{\text{guess}}},$$

and

$$|\sqrt{P_{\text{left}}} - \sqrt{P_{\text{right}}}| \leq 2d\sqrt{P_{\text{guess}}}.$$

The same result holds with $\mathcal{B}^G$ instead of $\mathcal{B}^H$ in the definition of P_{guess}.

B Attacks against Legendre and Power Residue PRFs

The security of Legendre and power residue PRFs has been extensively studied over the past several decades.

Attacks on Shifted Legendre Symbol Problem (SLSP). Let p be an odd prime and K be a uniformly random value in $\mathbb{F}_p$. The SLSP requires an adversary to find the hidden shift K with *oracle access* to the PRF, which on the adversarial input $a \in \mathbb{F}_p$, outputs the Legendre symbol $\left(\frac{K+a}{p}\right)$.

It is known that if the adversary is allowed to query the PRF in a quantum mechanism superposition, then a polynomial-time quantum algorithm can determine K and break the pseudo-randomness of the PRF, using just a single query [56, 57]. However, if the oracle can only be queried classically, Khovratovich [40] showed a memoryless collision search algorithm with $O(\sqrt{p}\log p)$ Legendre symbol evaluations when $\sqrt{p}\log p$ queries are available. Recent works concurrently improved the attack presented in [40], either by exploiting the multiplicative property of the Legendre function [8, 37] or by allowing input-independent pre-computations [44].

In particular, if the number of oracle queries is bounded by $L \leq \sqrt[4]{p}$, [8] presented a table-based collision search algorithm that recovers the hidden shift with a time complexity of $O\left(\frac{p\log^2 p}{L^2}\right)$ Legendre symbol evaluations and a memory cost of $O(L^2)$. [37] shows a similar table-based collision attack that further reduces the run time complexity to $O(\sqrt{p\log\log p})$ on a $\Theta(\log p)$-bit machine by using $\sqrt[4]{p\log^2 p\log\log p}$ queries and $O(\sqrt{p\log\log p}\log p)$ memory. By allowing pre-computation [44], the query complexity can be brought down (from $\sqrt{p}\log p$) to $O(\sqrt[3]{p})$, with $O\left(\sqrt[3]{p^2}\right)$ (offline) run time and $O(\sqrt[3]{p})$ memory. There also exists a multiple-key attack against the Legendre PRF with $O(\sqrt{mp})$ time/query complexity to recover m hidden shifts [44]. Another work [51] shows that recovering the hidden shift of the Legendre PRF is equivalent to solving a specific family of multivariate quadratic (MQ) equation systems, and algebraic cryptanalysis does not improve previous results.

Furthermore, [37] shows that if we allow the classical memory with quantum random access (QRACM) model, the (classical) query complexity, computation complexity and memory complexity (in QRACM) can be improved to $O(\sqrt[6]{p})$, $O(\sqrt[3]{p})$, and $O(\sqrt[3]{p})$, respectively. As demonstrated in [32], there exists a quantum attack on the SLSP with $2^{O(\sqrt{\log_2 p})}\sqrt[3]{p}$ time complexity, while using $\sqrt[3]{p}$ classical queries and a subexponential number of qubits memory. The key advantage of the attack in [32] is that they removed the requirement of QRACM, as there is no practical and scalable implementation of QRACM at the moment.

Attacks on Key Recovery Let us first define the adversary's advantage in breaking the NP relation $\mathcal{R}$.

Definition 50 (Advantage in Breaking NP relation $\mathcal{R}$)**.** Let $\mathcal{A}$ be a PPT adversary. We define the advantage of $\mathcal{A}$ in breaking the relation $\mathcal{R} := \{R_\lambda\}_{\lambda\in\mathbb{N}}$ under distribution D as

$$\mathsf{Adv}^{\mathcal{R}}_{\mathcal{A}}(1^\lambda) := \Pr\left[(x,w') \in R_\lambda : (x,w) \leftarrow D(R_\lambda); w' \leftarrow \mathcal{A}(x)\right].$$

We say that the relation $\mathcal{R}$ is hard with respect to distribution D if, for any PPT adversary $\mathcal{A}$, the advantage satisfies $\mathsf{Adv}^{\mathcal{R}}_{\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.

Let $\mathcal{R} = \mathcal{R}_{\mathcal{L}^t}$, we have $\mathsf{Adv}^{\mathcal{R}^t_{\mathcal{L}}}_{\mathcal{A}}(1^\lambda)$ denote the advantage of $\mathcal{A}$ in the key recovery problem for the Legendre and power residue PRFs. Unlike the SLSP, in the key recovery problem, the adversary loses the ability to query the PRF. Namely, the adversary is required to find the hidden shift K with only a small amount of random, uncontrolled PRF evaluations (i.e., public key).

Notably, the polynomial-time quantum attacks introduced in [56, 57] become infeasible in the context of the key recovery since the adversary is not able to perform quantum superposition queries to the PRF. Moreover, if the number of oracle calls is bounded by L, the complexities of key recovery attacks in [8, 37, 40] become $O\left(\frac{p\log p}{L}\right)$, $O\left(\frac{p\log^2 p}{L^2}\right)$ and $O\left(\frac{p\log p\log\log p}{L^2}\right)$, respectively. These complexities are exponential in relation to the security parameter, especially when L is sufficiently small. It is crucial to emphasize that the enhancements proposed in [8, 37] are reliant on the sequential characteristics of Legendre PRF queries. Consequently, it remains an open question whether these attacks can be extended to random point queries. In addition, it is worth noting that the attacks described in [32, 44] are not applicable, as both of them rely on a substantial number of oracle queries.

For **power residue symbols**, the best known attack is due to [8], in which the number of queries is bounded by $L \leq \sqrt[4]{p}$, and the time complexity is $O\left(\frac{p\log^2 p}{L^2\log^2 t}\right)$ power residue symbol evaluations with $O(L^2\log t)$ memory, especially for small t. A better attack when t is a large value has a run time of $O\left(\frac{p\log^2 p}{Lt\log^2 t}\right)$ with $O(L\log t)$ memory. Overall, for both Legendre and power residue symbols, we have $\mathsf{Adv}^{\mathcal{R}^t_{\mathcal{L}}}_{\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.

C Proofs

C.1 Security Proofs of BAVC

In this subsection, we establish the online extractability and selective hiding properties of BAVC in both the classical random oracle model (ROM) and the quantum random oracle model (QROM).

For notational simplicity, throughout the proofs we model the PRG, H_T, and H_{com} as domain-separated instantiations of random oracles. Let $H_T(x) := (G(x), H(0\|x))$, and $H_{\mathsf{com}}(x) := H(1\|x)$, where

$$G : \{0,1\}^* \to \{0,1\}^\lambda \text{ and } H : \{0,1\}^* \to \{0,1\}^{2\lambda}$$

are modeled as random oracles.

C.1.1 Extractable Binding. We first prove the extractable binding property of BAVC in the classical random oracle model.

Lemma 51 (Extractable Binding in the ROM)**.** *The BAVC protocol satisfies online extractability in the random oracle model with* $\epsilon_{\mathsf{ext}}(\lambda) \leq \frac{q^2+M(L-1)+1}{2^{2\lambda}}$*, where* q *denotes the total number of oracle queries made by* $\mathcal{A}$*.*

Proof. Let $\mathcal{A}$ be any PPT adversary. We construct a PPT extractor $\mathcal{E}_{\mathsf{BAVC}}$ that simulates the random oracles G and H for $\mathcal{A}$ via lazy sampling while maintaining the query lists L_G and L_H. After interacting with $\mathcal{A}$, the extractor takes com as input and extracts $\mathbf{L}^*$ by inspecting L_H. For the honest verification procedure, $\mathcal{E}_{\mathsf{BAVC}}$ continues to answer all oracle queries using the same lazy-sampled query lists L_G and L_H, thereby preserving a consistent oracle state throughout. By the definition of the classical random oracle model, this simulation is perfect, and hence $\epsilon_{\mathsf{sim}} = 0$.

Next, we bound ϵ_{ext}. Fix $S = \{\bar{\ell}_e\}_{e\in[M]}$. The extractor parses the query list L_H to extract

$$\mathbf{L}^* = (l^*_{e,\ell_e})_{e\in[M],\ell_e\in[0,L-1]}$$

such that there exists

$$\mathbf{H} = (\mathsf{com}_{e,\ell})_{e\in[M],\,\ell\in[0,L-1]}$$

satisfying

$$(1\|\mathbf{H}, \mathsf{com}) \in L_H,$$

and

$$(0\|l^*_{e,\ell_e}, \mathsf{com}_{e,\ell}) \in L_H \text{ for all } e \in [M],\ \ell_e \in [0, L-1] \setminus \{\bar{\ell}_e\}.$$

If no such values exist, the extractor aborts; otherwise, it computes $r^*_{e,\ell_e} = H(0\|l^*_{e,\ell_e})$ for all $e \in [M],\ \ell_e \in [0, L-1]\setminus\{\bar{\ell}_e\}$. Let $\bar{\mathbf{R}}^* := (r^*_{e,\ell_e})_{e\in[M],\ell_e\in[0,L-1]\setminus\{\bar{\ell}_e\}}$.

Let Coll denote the event that a collision occurs in the random oracle H among all oracle queries made during the execution of $\mathcal{A}$. By the birthday bound,

$$\Pr[\mathsf{Coll}] \le \frac{(q+ML)^2}{2^{2\lambda}},$$

where q denotes the total number of oracle queries made by $\mathcal{A}$.

In the following, we condition on the event ¬Coll.

Let

$$\mathsf{com} \leftarrow \mathcal{A}_1^{\mathcal{E}_{\mathsf{BAVC}}}(1^\lambda),\ (\bar{\mathbf{R}}, \mathsf{decom}_S) \leftarrow \mathcal{A}_2^{\mathcal{E}_{\mathsf{BAVC}}}(1^\lambda, S),$$

and let

$$(\mathsf{com}', \bar{\mathbf{R}}') \leftarrow \mathtt{Verify}^{\mathcal{E}_{\mathsf{BAVC}}}(\mathsf{decom}_S, S),$$

where

$$\bar{\mathbf{R}}' = (r'_{e,\ell_e})_{e\in[M],\,\ell_e\in[0,L-1]\setminus\{\bar{\ell}_e\}}.$$

By the construction of BAVC, conditioned on the event ¬Coll together with $\mathsf{com}' = \mathsf{com}$ and $\bar{\mathbf{R}}' = \bar{\mathbf{R}}$, it follows that

$$\mathbf{H} = \left(H(0\|l^*_{e,\ell_e})\right)_{e\in[M],\ell\in[0,L-1]},\ \mathsf{com}' = H(1\|\mathbf{H}),$$

and for all $e \in [M]$ and $\ell_e \in [0, L-1] \setminus \{\bar{\ell}_e\}$, it holds that

$$r'_{e,\ell_e} = G(l^*_{e,\ell_e}).$$

Then, according to the definition of LeafCom, the extracted $\mathbf{L}^*$ yields $(\mathsf{com}^*, \mathbf{R}^*)$ that is consistent with the adversary's opening, and hence

$$\bar{\mathbf{R}}^* = \bar{\mathbf{R}}'.$$

There extraction fails only if the extractor aborts. It follows that the extraction error $\epsilon_{\mathsf{ext}}(\lambda)$ is upper bounded by

$$\Pr\left[\mathcal{E}_{\mathsf{BAVC}} \text{ aborts} \wedge (\mathsf{com}', \bar{\mathbf{R}}') = (\mathsf{com}, \bar{\mathbf{R}}) | \neg\mathsf{Coll}\right] + \Pr[\mathsf{Coll}].$$

We bound the probability by cases.

- **Case 1: Missing top-level query.** Suppose that there exists no $\mathbf{H}$ such that $(1\|\mathbf{H}, \mathsf{com}) \in L_H$. Then the adversary $\mathcal{A}$ never queried the random oracle on input $(1\|\mathbf{H})$ corresponding to com. Hence, when Verify computes com′ via a fresh oracle query, the output is uniformly random. Therefore,

$$\Pr\left[\mathsf{com}' = \mathsf{com} \wedge \bar{\mathbf{R}} = \bar{\mathbf{R}}'\right] \le \Pr[\mathsf{com}' = \mathsf{com}] \le \frac{1}{2^{2\lambda}}.$$

- **Case 2: Missing leaf queries outside the challenge set.** Suppose that there exists $\mathbf{H}$ such that $(1\|\mathbf{H}, \mathtt{com}) \in L_H$, but there exists some $(e, \ell) \in [M] \times [0, L-1] \setminus \{\bar{\ell}_e\}$ such that

$$(0\|l_{e,\ell}, \mathtt{com}_{e,\ell}) \notin L_H.$$

  Then the value $H(0\|l_{e,\ell})$ is not fixed by lazy sampling and will be sampled uniformly at random upon its first query during verification. Conditioned on $\mathtt{com} = \mathtt{com}'$ and $\bar{\mathbf{R}} = \bar{\mathbf{R}}'$, we must have

$$H(0\|l_{e,\ell}) = \mathtt{com}_{e,\ell},$$

  which happens with probability at most $2^{-2\lambda}$ for each such (e, ℓ). Taking a union bound over all $M(L-1)$ possible pairs, we obtain

$$\Pr\left[\mathtt{com}' = \mathtt{com} \wedge \bar{\mathbf{R}} = \bar{\mathbf{R}}'\right] \leq \frac{M(L-1)}{2^{2\lambda}}.$$

Combining the above cases, we obtain

$$\epsilon_{\mathsf{ext}}(\lambda) \leq \frac{q^2 + M(L-1) + 1}{2^{2\lambda}}.$$

□

Lemma 52 (Extractable Binding in the QROM)**.** *The BAVC protocol extractable binding in the QROM with extraction error*

$$\epsilon_{\mathsf{ext}}(\lambda) \leq \frac{60q^3 + 6\sqrt{ML}q^{\frac{3}{2}} + 10MLq^2}{2^{2\lambda}} + \frac{2ML}{2^{\lambda}} + \frac{2ML}{2^{2\lambda}},$$

where q denote the total number of $\mathcal{A}$'s queries to random oracles.

Proof. The proof idea follows the same structure as in the classical ROM, with the main difference that the random oracle is simulated in the quantum-accessible setting.

Specifically, for a quantum adversary $\mathcal{A}$, the PPT extractor $\mathcal{E}_{\mathrm{BAVC}}$ simulates the random oracle G and H using the compressed state oracle (CStO) technique. During the interaction with $\mathcal{A}$, no measurement is performed. After $\mathcal{A}$ outputs $(\mathtt{com}, \bar{\mathbf{R}}, \mathtt{decom}, S)$ and terminates, the extractor invokes the honest verifier $\mathtt{Verify}$, which is evaluated consistently within the same oracle state. Only after completing this step does the extractor measure the internal state of the compressed oracles to obtain databases D_G and D_H. This ordering ensures that the measurement does not affect either the adversary's or the verifier's execution. By the properties of CStO, the simulation is perfect. Therefore, $\epsilon_{\mathsf{sim}} = 0$.

Fix $S = \{\bar{\ell}_e\}_{e\in[M]}$. Since $\mathtt{Verify}$ is an honest classical algorithm, it queries the inputs l_{e,ℓ_e} for all $e \in [M]$ and $\ell_e \in [0, L-1] \setminus \{\bar{\ell}_e\}$ to the oracles G and H. Consider

$$(\mathtt{com}', \bar{\mathbf{R}}') \leftarrow \mathtt{Verify}^{\mathcal{E}_{\mathrm{BAVC}}}(\mathtt{decom}_S, S),$$

where

$$\bar{\mathbf{R}}' := \{r'_{e,\ell}\}_{e\in[M],\, \ell\in[0,L-1]\setminus\{\bar{\ell}_e\}}.$$

$\mathcal{E}_{\mathrm{BAVC}}$ then extracts $\mathbf{L}^* := \{l_{e,\ell_e}\}_{e\in[M],\, \ell\in[0,L-1]}$ from the database D_H such that, for all $e \in [M]$ and $\ell_e \in [0, L-1]\setminus\{\bar{\ell}_e\}$,

$$\mathtt{com}'_{e,\ell} = H(0\|l_{e,\ell_e}) \text{ and } r'_{e,\ell_e} = G(l_{e,\ell_e}),$$

and moreover,

$$\mathtt{com}' = H(1\|\mathbf{H}), \text{ where } \mathbf{H} = \{\mathtt{com}'_{e,\ell_e}\}_{e\in[M],\, \ell_e\in[0,L-1]}.$$

Finally, the extractor computes $r^*_{e,\ell_e} = G(l^*_{e,\ell})$ for all $e \in [M]$ and $\ell_e \in [0, L-1]\setminus\{\bar{\ell}_e\}$ and outputs $\bar{\mathbf{R}}^* := (r^*_{e,\ell_e})_{e\in[M],\ell_e\in[0,L-1]\setminus\{\bar{\ell}_e\}}$.

According to Lemma 43, the upper bound of

$$\Pr\begin{bmatrix}\exists\, e\in[M],\ \ell_e\in[0,L-1]\setminus\{\bar{\ell}_e\}\text{ s.t. } r'_{e,\ell_e}\neq D_G(l_{e,\ell_e})\vee\\ \exists\, e\in[M],\ \ell_e\in[0,L-1]\setminus\{\bar{\ell}_e\}\text{ s.t. } com'_{e,\ell_e}\neq D_H(0\|l_{e,\ell_e})\vee\\ \mathsf{com}'\neq D_H(1\|\mathbf{H})\end{bmatrix}$$

is

$$\frac{2ML}{2^{\lambda}}+\frac{2ML}{2^{2\lambda}}.$$

Let q denote the total number of $\mathcal{A}$'s queries and

$$\mathsf{com}\leftarrow\mathcal{A}_1^{\mathcal{E}_{\mathsf{BAVC}}}(1^{\lambda})\text{ and }(\bar{\mathbf{R}},\mathsf{decom}_S)\leftarrow\mathcal{A}_2^{\mathcal{E}_{\mathsf{BAVC}}}(S),$$

where

$$\bar{\mathbf{R}}:=\{r_{e,\ell_e}\}_{e\in[M],\ell_e\in[0,L-1]\setminus\{\bar{\ell}_e\}}.$$

To upper bound the extraction failure probability, we define the following database properties SUC and CL:

$$\mathsf{SUC}:=\left\{D:\begin{array}{l}\exists\,\mathbf{L}^*\text{ such that }\forall e\in[M],\ \ell_e\in[0,L-1]\setminus\{\bar{\ell}_e\},\\ (0\|l_{e,\ell},com_{e,\ell})\in D_H;(1\|\mathbf{H},\mathsf{com})\in D_H;\\ \text{but }\bar{\mathbf{R}}^*\neq\bar{\mathbf{R}}'\end{array}\right\}$$

, and

$$\mathsf{CL}:=\left\{D:\exists\, x\neq x'\text{ such that }D(x)=D(x')\neq\perp\right\}.$$

Similar to the analysis in the classical ROM, the extraction error $\epsilon_{\mathsf{ext}}(\lambda)$ satisfies

$$\kappa(\lambda)=\max_{\mathcal{A}}\Pr[D_H\in\mathsf{SUC}\cup\mathsf{CL}]+\frac{2ML}{2^{\lambda}}+\frac{2ML}{2^{2\lambda}}.$$

According to the basic properties of quantum transition capacity and Lemma 46 (Lemma 5.6 in [25]), we have

$$[\![\perp\xrightarrow{q}\mathsf{CL}]\!]\leq\sum_{i=0}^{q-1}[\![\mathsf{SZ}_{\leq i}\setminus\mathsf{CL}\rightarrow\mathsf{CL}]\!]\leq 2eq\sqrt{\frac{q}{2^{2\lambda}}},$$

and

$$[\![\perp\xrightarrow{q}\mathsf{SUC}]\!]\leq\sum_{i=0}^{q-1}[\![\mathsf{SZ}_{\leq i}\setminus\mathsf{SUC}\rightarrow\mathsf{SUC}]\!].$$

Finally, applying Theorem 47, we bound each term $[\![\mathsf{SZ}_{\leq i}\setminus\mathsf{SUC}\rightarrow\mathsf{SUC}]\!]$ as follows. Fix an arbitrary database D and input x, and assume that x is fresh, i.e., $D(x)=\perp$. Let $\mathsf{P}:=\mathsf{SZ}_{\leq i}\setminus\mathsf{SUC}$ and $\mathsf{P}':=\mathsf{SUC}$. We proceed by a case analysis.

- Case 1: $D\notin\mathsf{SUC}$, the prefix of x is 0. Because $D(x)=\perp$, $\perp\notin\mathsf{P}'|_{D|^x}$. Set $\mathsf{L}:=\mathsf{P}'_{D|^x}$. Furthermore, the following implications hold: u belongs to L if and only if $D[x\rightarrow u]\in\mathsf{P}'$. If $D[x\rightarrow u]\in\mathsf{P}'$, then $D[x\rightarrow u]\in\mathsf{SUC}$. Finally, if $D[x\rightarrow u]\in\mathsf{SUC}$, then $u\in\{\mathsf{com}_{e,\ell}\}_{e\in[M],\ell\in[0,L-1]}$.
  Thus, we obtain
  $$\Pr[u\in\mathsf{L}:u\xleftarrow{\$}\{0,1\}^{2\lambda}]\leq\frac{qML}{2^{2\lambda}}.$$
- Case 2: $D\notin\mathsf{SUC}$ and the prefix of x is 1. Because $D(x)=\perp$, $\perp\notin\mathsf{P}'|_{D|^x}$. Set $\mathsf{L}:=\mathsf{P}'_{D|^x}$. Furthermore, the following implications hold: u belongs to L if and only if $D[x\rightarrow u]\in\mathsf{P}'$. If $D[x\rightarrow u]\in\mathsf{P}'$, then $D[x\rightarrow u]\in\mathsf{SUC}$.

Finally, if $D[x \to u] \in \mathsf{SUC}$, then $u = \mathtt{com}$. Therefore, we can bound

$$\Pr[u \in \mathsf{L} : u \xleftarrow{\$} \{0,1\}^{2\lambda}] \leq \frac{q}{2^{2\lambda}}.$$

We have

$$[\![\mathsf{SZ}_{\leq i} \backslash \mathsf{SUC} \to \mathsf{SUC}]\!] \leq \sqrt{\frac{10qML}{2^{2\lambda}}}.$$

Overall, we obtain

$$\epsilon_{\mathsf{ext}}(\lambda) \leq \frac{20ML q^3 + 8e^2 q^3}{2^{2\lambda}} + \frac{2ML}{2^{\lambda}} + \frac{2ML}{2^{2\lambda}},$$

where q denote the total number of $\mathcal{A}$'s queries to random oracles. □

C.1.2 Selective Hiding. We first recall the following selective hiding result for BAVC in the classical ROM, established in [3].

Lemma 53 (Selective Hiding in the ROM [3])**.** *The BAVC protocol satisfies selective hiding in the ROM.*

We next observe that the same property extends to the quantum random oracle model.

Lemma 54 (Selective Hiding in the QROM)**.** *The BAVC protocol satisfies selective hiding in the QROM.*

Proof. The proof is nearly identical to the classical ROM case. The only distinction arises in handling the adversary's queries to the QRO: when reprogramming the QRO's responses at $x \in \{l_{e,\bar{l}_e}\}_{e\in[M]}$ to random values, we must employ programming techniques specific to the QROM. In particular, we rely on the OW2H lemma [55] to bound the adversary's probability of detecting such reprogramming. □

C.2 Proofs for Approximate PRF relation with Arithmetic Progression List

C.2.1 Proof of Lemma 29.

Proof. Let $\xi = e^{2\pi i/t}$ be a primitive t-th root of unity, and let $\mu_t := \{\xi^k | 0 \leq k < t\} \subset \mathbb{C}$ denote the set of all complex t-th roots of unity. It is obvious that μ_t is isomorphic to $\mathbb{Z}_t$. That is, there exists a bijection $\tau : \mathbb{Z}_t \to \mu_t$, defined by $\tau(k) = \xi^k$ for $k \in \mathbb{Z}_t$.

Let $\chi : \mathbb{F}_p^* \to \mu_t$ be a multiplicative character of order t derived from the t-th power residue symbol. We can define $\chi(x) = \xi^{\mathcal{L}_0^t(x)}$ for $x \in \mathbb{F}_p^*$, which is indeed a group homomorphism since $\chi(xy) = \xi^{\mathcal{L}_0^t(xy)} = \xi^{\mathcal{L}_0^t(x)+\mathcal{L}_0^t(y)} = \xi^{\mathcal{L}_0^t(x)} \cdot \xi^{\mathcal{L}_0^t(y)} = \chi(x)\chi(y)$. We extend χ to $\mathbb{F}_p$ by defining $\chi(0) = 0$.

The target vector $(s_0, \ldots, s_{c-1}) \in \mathbb{Z}_t^c$ is mapped to $(\omega_0, \ldots, \omega_{c-1}) \in \mu_t^c$ via the isomorphism τ (i.e., $\omega_\ell = \tau(s_\ell)$ for all $\ell \in [0, c-1]$). Hence, it is equivalent to prove the lemma for $N_{p,b}(\omega_0, \ldots, \omega_{c-1}) := \{\#a \in \mathbb{F}_p : \chi(a + b\ell) = \omega_\ell \text{ for all } 0 \leq \ell \leq c-1\}$.

The key fact we use is, for some $\omega, \eta \in \mu_t$, if $\eta = \omega$, then $(\omega^{-1}\eta) = \xi^0 = 1$, hence $\sum_{j=0}^{t-1}(\omega^{-1}\eta) = \sum_{j=0}^{t-1} 1^t = t$. On the other hand, if $\eta \neq \omega$, $(\omega^{-1}\eta) = \xi^k$ for some $0 < k < t$. Let $r = \xi^k \neq 1$, then we have the finite geometric sum $\sum_{j=0}^{t-1} r^j = \frac{1-r^t}{1-r}$. Furthermore, since $r \in \mu_t$, we have $r^t = 1$. Hence, when $\eta \neq \omega$, we have $\sum_{j=0}^{t-1}(\omega^{-1}\eta) = 0$.

We define the following equation, where for $\omega \in \mu_t$ and $x \in \mathbb{F}_p^*$, we have

$$\frac{1}{t}\left(\sum_{j=0}^{t-1}(\omega^{-1}\chi(x))^j\right) = \begin{cases} 1 & \text{if } \omega = \chi(x) \\ 0 & \text{if } \omega \neq \chi(x) \end{cases}$$

By replacing x with $(a + b\ell)$ under the condition that $a + b\ell \neq 0$ for $\ell \in [0, c-1]$, we define the function $F_b(a)$ as

$$F_b(a) = \prod_{\ell=0}^{c-1} \frac{1}{t} \left(\sum_{j=0}^{t-1} \left(\omega_\ell^{-1} \chi(a + b\ell) \right)^j \right)$$

Note that $F_b(a) = 1$ if and only if for all $\ell \in [0, c-1]$, $\omega_\ell = \chi(a + b\ell)$. Otherwise (i.e., there exists some $\ell \in [0, L-1]$ such that $\omega_\ell \neq \chi(a + b\ell)$), $F_b(a) = 0$.

Hence, we have

$$\begin{aligned} N_{p,b}(\omega_0, \ldots, \omega_{c-1}) &= \sum_{\substack{a \in \mathbb{F}_p \\ a, a+b, \ldots, a+(c-1)b \neq 0}} F_b(a) \\ &= \sum_{\substack{a \in \mathbb{F}_p \\ a, a+b, \ldots, a+(c-1)b \neq 0}} \prod_{\ell=0}^{c-1} \frac{1}{t} \left(\sum_{j=0}^{t-1} \left(\omega_\ell^{-1} \chi(a + b\ell) \right)^j \right) \end{aligned}$$

Let us first consider the case when $(a + b\ell) = 0$ for some $\ell \in [0, c-1]$. By convention, let us define $0^0 = 1$. Observe that if $(a + b\ell) = 0$, we have $\frac{1}{t} \sum_{j=0}^{t-1} (\omega_\ell^{-1} \cdot 0)^j = \frac{1}{t}$. Furthermore, there exists at most one $(a + b\ell) = 0$ in each sequence (i.e., when $a = -b\ell$). Hence, for any sequence with $(a + b\ell) = 0$ we have

$$\left| \prod_{\ell=0}^{c-1} \frac{1}{t} \left(\sum_{j=0}^{t-1} \left(\omega_\ell^{-1} \chi(a + b\ell) \right)^j \right) \right| \leq \frac{1}{t}.$$

For a fixed $b \in \mathbb{F}_p^*$, there are exactly c such numbers $a = 0, a = -b, a = -2b, \ldots, a = -(c-1)b$ in $\mathbb{F}_p$. Let $|e_c| \leq c$, then we have

$$\begin{aligned} N_{p,b}(\omega_0, \ldots, \omega_{L-1}) &= \sum_{a \in \mathbb{F}_p} \prod_{\ell=0}^{c-1} \frac{1}{t} \left(\sum_{j=0}^{t-1} \left(\omega_\ell^{-1} \chi(a + b\ell) \right)^j \right) + \frac{e_c}{t} \\ &= \frac{1}{t^c} \sum_{a \in \mathbb{F}_p} \left(\prod_{\ell=0}^{c-1} \left(\sum_{j=0}^{t-1} \left(\omega_\ell^{-1} \chi(a + b\ell) \right)^j \right) \right) + \frac{e_c}{t} \end{aligned}$$

We now expand the product inside the sum: for each $a \in \mathbb{F}_p$, let $F'_b(a) = \prod_{\ell=0}^{c-1} \left(\sum_{j=0}^{t-1} (\omega_\ell^{-1} \chi(a + b\ell))^j \right)$

$$\begin{aligned} F'_b(a) &= \prod_{\ell=0}^{c-1} \left(1 + \sum_{j=1}^{t-1} (\omega_\ell^{-1} \chi(a + b\ell))^j \right) \\ &= 1 + \sum_{S \subset \{0, \ldots, c-1\}/\emptyset} \left(\prod_{\ell \in S} \left(\sum_{j=1}^{t-1} (\omega_\ell^{-1} \chi(a + b\ell))^j \right) \right) \\ &= 1 + \sum_{S \subset \{0, \ldots, c-1\}/\emptyset} \left(\sum_{\substack{(j_\ell), \ell \in S \\ 1 \leq j_\ell \leq t-1}} \left(\prod_{\ell \in S} \omega_\ell^{-j_\ell} \right) \left(\prod_{\ell \in S} \chi(a + b\ell)^{j_\ell} \right) \right) \end{aligned}$$

We define $[c] = \{0, \ldots, c-1\}$, $S_c = 2^{[c]} \backslash \emptyset = \{S \subseteq [c] : S \neq \emptyset\}$, and for every $S \in S_c$, $J_S := \{\mathbf{j} = (j_\ell)_{\ell \in S} : 1 \leq j_\ell \leq t-1 \text{ for every } \ell \in S\}$. Finally, define $\omega_{S,\mathbf{j}} = \prod_{\ell \in S} \omega_\ell^{-j_\ell} \in \mu_t$. Now, we have

$$\begin{aligned} N_{p,b}(\omega_0, \ldots, \omega_{c-1}) &= \frac{1}{t^c} \sum_{a \in \mathbb{F}_p} F'_b(a) + \frac{e_c}{t} \\ &= \frac{p}{t^c} + \frac{1}{t^c} \sum_{a \in \mathbb{F}_p} \left(\sum_{S \in S_c} \left(\sum_{\mathbf{j} \in J_S} \omega_{S,\mathbf{j}} \left(\prod_{\ell \in S} \chi(a + b\ell)^{j_\ell} \right) \right) \right) + \frac{e_c}{t} \\ &= \frac{p}{t^c} + \frac{1}{t^c} \sum_{S \in S_c} \left(\sum_{\mathbf{j} \in J_S} \omega_{S,\mathbf{j}} \left(\sum_{a \in \mathbb{F}_p} \chi \left(\prod_{\ell \in S} (a + b\ell)^{j_\ell} \right) \right) \right) + \frac{e_c}{t}, \end{aligned}$$

which is the desired sum over $a \in \mathbb{F}_p$.

Let us define $f_{S,\mathbf{j}}(x) = \prod_{\ell \in S} (x + b\ell)^{j_\ell}$. We first determine the number of distinct roots of $f_{S,\mathbf{j}}$. For every $\ell \in S$, the corresponding root is $x = -b\ell$, and these roots are pairwise distinct. Hence, $f_{S,\mathbf{j}}$ has exactly $|S|$ distinct roots. Furthermore, $f_{S,\mathbf{j}}$ is not a perfect t-th power, since $1 \leq j_\ell \leq t-1$. Hence, we can apply the Weil's bound, which says for the non-trivial multiplicative character χ of order $t > 1$, if $f \in \mathbb{F}_p[X]$ has d distinct roots and is not a perfect t-th power, then

$$\left| \sum_{x \in \mathbb{F}_p} \chi(f(x)) \right| \leq (d-1)\sqrt{p}.$$

This inequality is an equality if $\deg(f) = 1$, and generally is a strict inequality if $\deg(f) \geq 2$. Applying the Weil's bound to the polynomial $f_{S,\mathbf{j}}(x)$, which is not a perfect t-th power and has at most $|S|$ distinct roots, we get

$$\left| \sum_{a \in \mathbb{F}_p} \chi(f_{S,\mathbf{j}}(a)) \right| \leq (|S| - 1)\sqrt{p}$$

Recall that the character expansion gives

$$N_{p,b}(\omega_0, \ldots, \omega_{c-1}) = \frac{p}{t^c} + \frac{1}{t^c} \sum_{S \in S_c} \sum_{\mathbf{j} \in J_S} \omega_{S,\mathbf{j}} \sum_{a \in \mathbb{F}_p} \chi(f_{S,\mathbf{j}}(a)) + \frac{e_c}{t}$$

where $|e_c| \leq c$ accounts for the exceptional values of a for which $a + b\ell = 0$ for some $\ell \in [c]$, and $|\omega_{S,\mathbf{j}}| = 1$. Taking absolute values and applying the triangle inequity gives

$$\begin{aligned} \left| N_{p,b}(\omega_0, \ldots, \omega_{c-1}) - \frac{p}{t^c} \right| &\leq \frac{1}{t^c} \sum_{S \in S_c} \sum_{\mathbf{j} \in J_S} \left| \sum_{a \in \mathbb{F}_p} \chi(f_{S,\mathbf{j}}(a)) \right| + \frac{e_c}{t} \\ &\leq \frac{\sqrt{p}}{t^c} \sum_{S \in S_c} \sum_{\mathbf{j} \in \mathbf{J_S}} (|S| - 1) + \frac{c}{t} \end{aligned}$$

We now evaluate the combinatorial sum. Fix $k \in \{1, \ldots, c\}$. There are $\binom{c}{k}$ subsets $S \subseteq \{0, \ldots, c-1\}$ of size k. For each such S, every exponent j_ℓ can be chosen independently from $\{1, \ldots, t-1\}$, so there are $(t-1)^k$ possible exponent

vectors and each corresponding character sum is bounded by $(k-1)\sqrt{p}$. Consequently,

$$\left|N_{p,b}(\omega_0,\ldots,\omega_{c-1}) - \frac{p}{t^c}\right| \le \frac{\sqrt{p}}{t^c}\sum_{k=1}^{c}\binom{c}{k}(t-1)^k(k-1) + \frac{c}{t}.$$

The combinatorial sum satisfies

$$\begin{aligned}\sum_{k=1}^{c}\binom{c}{k}(t-1)^k(k-1) &= \sum_{k=1}^{c}k\binom{c}{k}(t-1)^k - \sum_{k=1}^{c}\binom{c}{k}(t-1)^k \\ &= c(t-1)t^{c-1} - (t^c-1) \\ &= t^{c-1}(c(t-1)-t)+1.\end{aligned}$$

Define $B_{c,t} = t^{c-1}(c(t-1)-t)+1$, we have

$$\left|N_{p,b}(\omega_0,\ldots,\omega_{c-1}) - \frac{p}{t^c}\right| \le \frac{B_{c,t}}{t^c}\sqrt{p} + \frac{c}{t}.$$

Define $A_{c,t} = \frac{B_{c,t}}{t^c} = \frac{c(t-1)}{t} - 1 + \frac{1}{t^c}$, then we have the bound

$$\left|N_{p,b}(\omega_0,\ldots,\omega_{c-1}) - \frac{p}{t^c}\right| \le A_{c,t}\sqrt{p} + \frac{c}{t}.$$

□

C.2.2 Proof of Theorem 30.

Proof. Let X be all possible events under distribution D and U is the uniform distribution on X. For $x \in X$, Lemma 29 shows $D(x) = \frac{1}{t^c} + \frac{A_{c,t}}{\sqrt{p}} + \frac{c}{pt}$ and the ideal probability is $U(x) = \frac{1}{t^c}$. Hence, $|D(x) - U(x)| \le \delta_{c,t,p}$ for each $x \in X$ with $\delta_{c,t,p} = \frac{A_{c,t}}{\sqrt{p}} + \frac{c}{tp}$, where $A_{c,t} = \frac{c(t-1)}{t} - 1 + \frac{1}{t^c}$.

For the maximum Rényi divergence denoted as $R_\infty(D \parallel U) = \max_x \frac{D(x)}{U(x)}$, since $U(x) = \frac{1}{t^c}$ is the same for all x, we get

$$\begin{aligned}\frac{D(x)}{U(x)} &= t^c D(x) \\ &= (U(x) + (D(x) - U(x)))t^c \\ &= 1 + (D(x) - U(x))t^c \\ &\le 1 + t^c|D(x) - U(x)| \\ &\le 1 + t^c\delta_{c,t,p}\end{aligned}$$

Therefore, $R_\infty(D \parallel U) \le 1 + t_c\delta_{c,t,p}$. Substituting $\delta_{c,t,p}$, we get

$$t^c\delta = t^c\left(\frac{A_{c,t}}{\sqrt{p}} + \frac{c}{tp}\right) = \frac{A_{c,t}t^c}{\sqrt{p}} + \frac{ct^{c-1}}{p}.$$

Define $B_{c,t} = A_{c,t}t^c = t^{c-1}(c(t-1)-t)+1$ and $\rho_{c,t,p} = \frac{B_{c,t}}{\sqrt{p}} + \frac{ct^{c-1}}{p}$.

Since $(a_0,\ldots,a_{z-1}) \in \mathbb{F}_p^z$ are chosen uniformly at random from $\mathbb{F}_p$, by the multiplicative property of maximum Rényi divergence, we get

$$R_\infty(D^z \parallel U^z) = (R_\infty(D \parallel U))^z \le (1+\rho_{c,t,p})^z.$$

Using $1 + x \le e^x$ for $x \ge 0$, we obtain

$$R_\infty(D^z \parallel U^z) \le \exp(z\rho_{c,t,p}).$$

That is,

$$\Pr_{D^z}[x] \le \exp\left(z\left(\frac{B_{c,t}}{\sqrt{p}} + \frac{ct^{c-1}}{p}\right)\right)\Pr_{U^z}[E].$$

We now compute the ideal probability $\Pr_{U^z}[x]$. Let $L = z \times c$ and $b \in \mathbb{F}_p^*$ be a fixed non-zero field element. The setup algorithm uniformly random chooses $a_0, \ldots, a_{z-1} \in \mathbb{F}_p^z$ and defines $\mathfrak{I}' = (\mathfrak{I}'_0, \ldots, \mathfrak{I}'_{z-1}) = ((a_j + b\ell)_{\ell\in[0,c-1]})_{j\in[0,z-1]}$. For a fixed key $K \in \mathbb{F}_p$, let us define for each $j \in [0, z-1]$, $\mathbf{s}_j = (\mathcal{L}_0^t(K + a_j), \ldots, \mathcal{L}_0^t(K + a_j + b(\ell-1)))$, where $a_j \xleftarrow{\$} \mathbb{F}_p$. Then, for a fixed pair $(K', \alpha) \in \mathbb{F}_p/\{K\} \times \mathbb{Z}_t$, we define $\mathbf{s}'_j = (\mathcal{L}_0^t(K' + a_j) + \alpha, \ldots, \mathcal{L}_0^t(K' + a_j + b(\ell-1)) + \alpha)$. Let $d_H(\mathbf{s}'_j, \mathbf{s}_j) = \#\{0 \le \ell \le c : \mathbf{s}'_j[\ell] \ne \mathbf{s}_j[\ell]\}$ be the Hamming distance between two vectors $\mathbf{s}_j$ and $\mathbf{s}'_j$.

Assume the ideal distribution of each element in $\mathbf{s}_j$ and $\mathbf{s}'_j$, the distance $d_H(\mathbf{s}_j, \mathbf{s}'_j)$ is distributed as $\mathfrak{B}(c, 1/t)$. Furthermore, since each a_j is chosen independently from $\mathbb{F}_p$, then for all $j \in [0, z-1]$, we have the probability that $\sum_{j=0}^{z-1} d_H(\mathbf{s}_j, \mathbf{s}'_j) \le \beta zc$ is at most

$$\Pr[\mathfrak{B}(zc, 1/t) > (1-\beta)zc].$$

Recall that $L = zc$ and since there exists $(p-1)t$ possible values for (K', α), the probability that there exists a non-trivial witness for the β-relaxed relation is at most $\Pr[\mathfrak{B}(L, 1/t) > (1-\beta)L](p-1)t$, under the assumption of an ideal distribution of $\mathbf{s}_j$ and $\mathbf{s}'_j$. Then, we obtain the bound on the real probability on the distribution of $\sum_{j=0}^{z-1} d_H(\mathbf{s}_j, \mathbf{s}'_j) \ge \beta L$:

$$\Pr_{D^z}[x] \le \gamma_{z,c,t,p} \cdot \Pr[\mathfrak{B}(L, 1/t) > (1-\beta)L](p-1)t,$$

by applying the argument based on Rényi divergence, where $\gamma_{z,c,t,p} = \exp\left(z\left(\frac{B_{c,t}}{\sqrt{p}} + \frac{ct^{c-1}}{p}\right)\right)$ is the upper bound on the maximum Rényi divergence between the real and ideal distributions over z independent blocks. Note that when $z\left(\frac{B_{c,t}}{\sqrt{p}} + \frac{ct^{c-1}}{p}\right) \le 1$, we have $\gamma_{z,c,t,p} < 2.72$. □

C.3 Proof of Theorem 22

Proof. Let $\bar{\mathbf{R}} = (r_{e,\ell})_{\ell\in[0,L-1]\setminus\{\bar{\ell}_e\},e\in[M]}$ be the opening of BAVC. We show that for every $e \in [M]$, the verifier computes $o_e = (K + I_{\bar{\ell}_e})r_e$, where $r_e = \sum_{\ell=0}^{L-1} r_{e,\ell}$:

$$\begin{aligned}
o_e &= I_{\bar{\ell}_e} \sum_{\ell=0,\ell\ne\bar{\ell}_e}^{L-1} r_{e,\ell} - \sum_{\ell=0,\ell\ne\bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + \Delta K_e \\
&= I_{\bar{\ell}_e} \sum_{\ell=0,\ell\ne\bar{\ell}_e}^{L-1} r_{e,\ell} - \sum_{\ell=0,\ell\ne\bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + Kr_e + \sum_{\ell=0}^{L-1} I_\ell r_{e,\ell} \\
&= Kr_e + I_{\bar{\ell}_e} \sum_{\ell=0,\ell\ne\bar{\ell}_e}^{L-1} r_{e,\ell} + I_{\bar{\ell}_e} r_{e,\bar{\ell}_e} \\
&= Kr_e + I_{\bar{\ell}_e} \sum_{\ell=0}^{L-1} r_{e,\ell} \\
&= Kr_e + I_{\bar{\ell}_e} r_e \\
&= (K + I_{\bar{\ell}_e})r_e
\end{aligned}$$

Due to the multiplicative homomorphism of the Legendre and power residue PRFs, we have $\mathcal{L}_0^t(o_e) = \mathcal{L}_0^t(K + I_{\bar{\ell}_e}) + \mathcal{L}_0^t(r_e) = pk_{\bar{\ell}_e} + T_e$. □ □

C.4 Proof of Theorem 23

Proof. We begin by describing the extractor $\mathcal{E}$, which invokes the BAVC extractor to recover a witness $sk^* = K$ for the β-approxim-ate PRF relation.

- **Extractor Description.** For any (quantum) PPT adversary P^*, whenever P^* queries the oracles G or H, the (quantum) PPT extractor $\mathcal{E}$ forwards these queries to $\mathcal{E}_{\mathsf{BAVC}}$ and returns the corresponding responses to P^*. After P^* outputs a transcript $((A, A_\circ), S, Z)$, the extractor $\mathcal{E}$ proceeds as follows:
  (1) $\mathcal{E}$ parses $A = \mathtt{com}$, $A_\circ = (T_e, \Delta K_e)_{e\in[M]}$, $S = \{\bar{\ell}_e\}_{e\in[M]}$, and $Z = \mathtt{decom}_S$.
  (2) $\mathcal{E}$ runs the BAVC extractor $\mathcal{E}_{\mathsf{BAVC}}$ on input $\mathtt{com}$ to obtain the leaf nodes
  $$\mathbf{L}^* = (l^*_{e,0}, \ldots, l^*_{e,L-1})_{e\in[M]}.$$
  It then computes the corresponding committed vector
  $$\mathbf{R}^* = (r^*_{e,0}, \ldots, r^*_{e,L-1})_{e\in[M]}, \text{ where } r^*_{e,\ell} = G(l^*_{e,\ell}).$$
  (3) For each $e \in [M]$, $\mathcal{E}$ computes a candidate witness
  $$K^*_e = \left(\Delta K_e - \sum_{\ell=0}^{L-1} I_\ell\, r^*_{e,\ell}\right) r_e^{-1}, \text{ where } r_e = \sum_{\ell=0}^{L-1} r^*_{e,\ell}.$$
- **Probability Analysis of ϵ_{sim}.** Since $\mathcal{E}_{\mathsf{BAVC}}$ simulates G and H via lazy sampling in the ROM (or using CStO in the QROM), it follows that $\epsilon_{sim} = 0$.
- **Probability Analysis of κ.** For $e \in [M]$, $\mathcal{E}$ computes the β-approximate witness candidate $K^*_e = (\Delta K_e - \sum_{\ell=0}^{L-1} I_\ell r^*_{e,\ell}) r_e^{-1}$, where $r_e = \sum_{\ell=0}^{L-1} r^*_{e,\ell}$. Suppose the adversary outputs $A^*_\circ = (\Delta K^*_e, T^*_e)_{e\in[M]}$, we define $I(\Delta K^*_e)$ be the subset of indices in $[0, L-1]$ such that for $j = 0, 1, \ldots, L-1$, the equation 4 holds
  $$\mathcal{L}_0^t(I_j \sum_{\ell=0,\ell\neq j}^{L-1} r_{e,\ell} - \sum_{\ell=0,\ell\neq j}^{L-1} r_{e,\ell} I_\ell + \Delta K^*_e) = pk_j + T^*_e \tag{4}$$
  Therefore, in an accepting transcript, we must have $\bar{\ell}_e \in I(\Delta K^*_e)$ for all $e \in [M]$. Since the challenge indices $(\bar{\ell}_e)_{e\in[M]}$ are sampled uniformly from $[0, L-1]$, the probability of this event is $P = \prod_{e\in[M]} \frac{|I(\Delta K^*_e)|}{L}$.
  For each extracted candidate witness K^*_e, let us define $\beta_{K^*_e}$ be the Hamming distance of two sequences $\mathcal{L}^t_{K^*_e}(\mathfrak{I}) + \mathcal{L}^t_0(r_e) \in \mathbb{Z}_t^L$ and $pk + T^*_e \in \mathbb{Z}_t^L$:
  $$\beta_{K^*_e}(Q) = d_H((\mathcal{L}^t_{K^*_e}(\mathfrak{I}) + (\mathcal{L}^t_0(r_e)), (T^*_e + pk))),$$
  Notably, $\beta_{K^*_e}$ represents the number of $\ell \in [0, L-1]$ such that equation 4 does *not* hold, in other words, $\beta_{K^*_e} = L - |I(\Delta K^*_e)|$. If none of K^*_e is not a β-approximate witness for the statement pk, it follows that $\beta_{K^*_e} > \beta L$ for all $e \in [M]$, according to the Definition 3. Therefore, we can bound for each $e \in [M]$ $|I(\Delta K^*_e)|$ as $|I(\Delta K^*_e)| < (1-\beta)L$. Combined with the previous argument, we have a bound on the successful cheating probability:
  $$P = \prod_{e\in[M]} \frac{|I(\Delta K^*_e)|}{L} < \prod_{e\in[M]} \frac{(1-\beta)L}{L} = (1-\beta)^M.$$

From the definitions of PoK-OE, it follows directly that

$$\epsilon_{\mathsf{ext}}(\lambda) \le \epsilon_{\mathsf{BAVC}}(\lambda) + P,$$

where $\epsilon_{\mathsf{BAVC}}(\lambda)$ denotes the BAVC extraction error (Definition 8), and P is the bound derived above.

Therefore, if the BAVC protocol satisfies extractable binding in the (Q)ROM, then the commit-and-open Σ-protocol $\Pi^{\mathcal{H}}$ satisfies PoK-OE in the (Q)ROM, assuming that P is negligible. □

C.5 Proof of Theorem 24

Proof. We begin by describe the simulator which generates a transcript on input the challenge S and the public key pk.

- **Simulator Construction**. On input the challenge public key pk and any verifier challenge $S = \{\bar{\ell}_1, \ldots, \bar{\ell}_M\}$, the simulator $\mathsf{Sim}^{\mathcal{H}}$ does the following:
  (1) Sample public parameters $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$.
  (2) Invoke BAVC algorithms $(\mathtt{decom}, \mathbf{L}) \leftarrow \mathtt{TreeGen}(\mathtt{BAVC.pp})$ and $(\mathtt{com}, \mathbf{R}) \leftarrow \mathtt{LeafCom}^{\mathcal{H}}(\mathbf{L})$ to get random messages $\mathbf{R} = (r_{e,0}, \ldots, r_{e,L-1})_{e\in[M]}$, where $\mathtt{BAVC.pp}$ is included in the public parameters pp.
  (3) For each $e \in [M]$, sample $\Delta K_e^* \xleftarrow{\$} \mathbb{F}_p$ and compute
  $$T_e^* = \mathcal{L}_0^t\left(I_{\bar{\ell}_e} \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} - \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + \Delta K_e^*\right) - pk_{\bar{\ell}_e}.$$
  (4) Run $\mathtt{decom}_S \leftarrow \mathtt{Open}^{\mathcal{H}}(\mathtt{decom}, S)$.
  (5) $\mathtt{Sim}$ outputs the simulated transcript $((A, A_\circ^*), S, Z)$ such that $A = \mathtt{com}$, $A_\circ^* = (T_e^*, \Delta K_e^*)_{e\in[M]}$, and $Z = \mathtt{decom}_S$.

  It is immediate that $((A, A_\circ^*), S, Z)$ is an accepting transcript, as the BAVC commitment and openings are honestly generated, and for each $e \in [M]$,
  $$o_e = I_{\bar{\ell}_e} \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} - \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + \Delta K_e^*,$$
  it holds that $T_e^* = \mathcal{L}_0^t(o_e) - pk_{\bar{\ell}_e}$.

- **Hybrid Argument.** We show that the distribution of simulated transcripts $((A, A_\circ^*), S, Z)$ is indistinguishable from the real distribution of $((A, A_\circ), S, Z)$ in a hybrid argument, assuming BAVC satisfies selective hiding.
  – H0 **(Real)**. Honest prover (knows the witness K) follows Algorithm 3 and outputs a real transcript $((A, A_\circ), S, Z)$, where
  $$A = \mathtt{com},\ A_\circ = (T_e, \Delta K_e)_{e\in[M]},\ Z = \mathtt{decom}_S.$$
  Here, the prover runs BAVC algorithms honestly, generating messages $\mathbf{R} = (r_{e,0}, \ldots, r_{e,L-1})_{e\in[M]}$. It then computes, for each $e \in [M]$,
  $$r_e = \sum_{\ell=0}^{L-1} r_{e,\ell},\ T_e = \mathcal{L}_0^t(r_e),\ \Delta K_e = K r_e + \sum_{\ell=0}^{L-1} I_\ell r_{e,\ell}.$$

– H1 **(Randomizing unopened leaves)**. For each $e \in [M]$, replace the unopened message $r_{e,\bar{\ell}_e}$ with an independent uniform sample $r'_{e,\bar{\ell}_e} \xleftarrow{\$} \mathbb{F}_p$. Because the BAVC is hiding, replacing unopened messages with fresh random values yields a distribution for the new transcript $((A, A'_\circ), S, Z)$ which is indistinguishable from $((A, A_\circ), S, Z)$, with distinguishing advantage at most $\mathsf{AdvHD}^{\mathsf{BAVC}}_{\mathcal{A}}$. Concretely, after this substitution, we have

$$r'_e = \sum_{\ell \neq \bar{\ell}_e} r_{e,\ell} + r'_{e,\bar{\ell}_e},\ T'_e = \mathcal{L}_0^t(r'_e),$$

and

$$\begin{aligned}\Delta K'_e &= K r'_e + \sum_{\ell \neq \bar{\ell}_e} I_\ell r_{e,\ell} + I_{\bar{\ell}_e} r'_{e,\bar{\ell}_e} \\ &= (K + I_{\bar{\ell}_e}) r_{e,\bar{\ell}_e} + \sum_{\ell \neq \bar{\ell}_e} (K + I_\ell) r_{e,\ell}\end{aligned}$$

Since $r'_{e,\bar{\ell}_e}$ is uniform and independent, r'_e is uniform over $\mathbb{F}_p$. Hence, T'_e is the Legendre or t-th power residue PRF of a uniform input, and $\Delta K'_e$ is also uniform over $\mathbb{F}_p$, provided that $K + I_{\bar{\ell}_e} \neq 0$. Since K is sampled uniformly from $\mathbb{F}_p^*$, $K + I_{\bar{\ell}_e} = 0$ happens with probability at most $\frac{1}{p}$, and across M independent challenges, the probability is at most M/p, which is negligible for an exponentially large p.

– H2 **(Simulator)**. Fix $(r_{e,\ell})_{\ell \in [0,L-1] \setminus \{\bar{\ell}_e\}, e \in [M]}$, $(I_\ell)_{\ell \in [0,L-1]}$ and $pk_{\bar{\ell}_e}$, for each $e \in [M]$, replace ΔK_e by an independent uniform $\Delta K^*_e \xleftarrow{\$} \mathbb{F}_p$ and set

$$T^*_e = \mathcal{L}_0^t\left(I_{\bar{\ell}_e} \sum_{\ell=0, \ell \neq \bar{\ell}_e}^{L-1} r_{e,\ell} - \sum_{\ell=0, \ell \neq \bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + \Delta K^*_e \right) - pk_{\bar{\ell}_e}.$$

Keep the commitment com and the opening $Z = \mathsf{decom}_S$ unchanged, let $((A, A^*_\circ), S, Z)$ denote the resulting transcript. Since ΔK^*_e is uniform and independent, the masked input inside $\mathcal{L}_0^t(\cdot)$ is uniform over $\mathbb{F}_p$. Hence, $(T^*_e, \Delta K^*_e)$ has exactly the same distribution as $(T'_e, \Delta K'_e)$ from H1 for every $e \in [M]$. Therefore, the overall transcript distribution in H2 is identical to that of H1.

□

C.6 Proof of Theorem 25

Proof. Let $\mathcal{A}$ be a PPT adversary against EUF-KO security that issues at most q queries to $\mathcal{H}_{\mathsf{FS}}$. We first construct a PPT adversary $\mathcal{B}$ that, by invoking both $\mathcal{A}$ and the extractor $\mathcal{E}$, recovers the secret key sk corresponding to the public key pk. This guarantees that $(pk, sk) \in \mathcal{R}_{\beta\text{-leg}^t}$.

- **Construction.** The algorithm $\mathcal{B}$ proceeds as follows:
  (1) **Setup Phase.** Upon receiving the public parameters

$$(p, t, L, \mathcal{I}, \mathcal{I}')$$

and a statement $pk = \mathcal{L}_K^t(\mathcal{I})$, $\mathcal{B}$ selects an integer M and runs $\mathsf{pp} \leftarrow \mathsf{BAVC.Setup}(1^\lambda)$.
Let

$$\mathsf{pp} := (p, t, L, M, \mathcal{I}, \mathsf{BAVC.pp}).$$

It then runs adversary $\mathcal{A}$ and extractor $\mathcal{E}$, forwarding pp and pk to them.

(2) **RO Simulation Phase.** $\mathcal{B}$ simulates $\mathcal{H}_{\mathsf{FS}}$ via lazy sampling and maintains a query list L_H. When $\mathcal{A}$ makes a query to $\mathcal{H}$, $\mathcal{B}$ forwards the query to $\mathcal{E}$ and returns the corresponding response to $\mathcal{A}$.

(3) **Challenge Phase.** Once $\mathcal{A}$ outputs a forgery (m, σ), $\mathcal{B}$ parses $\sigma = (\mathsf{salt}, h, \mathtt{ctr}, \mathtt{decom}_S, (\Delta K_e)_{e\in[M]})$ and computes $S \leftarrow \mathsf{Expand}(h[:128])$.It then searches the query list L_H for a record

$$((\mathsf{salt}, \mathsf{com}, (T_e)_{e\in[M]}, (\Delta K_e)_{e\in[M]}, m, \mathsf{ctr}), h) \in L_H.$$

- If no such entry exists, $\mathcal{B}$ aborts.
- Otherwise, $\mathcal{B}$ extracts $(T_e)_{e\in[M]}$ and sends $((A, A_\circ), S, Z)$ to $\mathcal{E}$, where $A = \mathsf{com}$, $A_\circ = (T_e, \Delta K_e)_{e\in[M]}$, and $Z = \mathtt{decom}_S$.

(4) **Output Phase.** Finally, when $\mathcal{E}$ outputs sk, $\mathcal{B}$ outputs the same sk.

- **Probability Analysis.** We begin by analyzing the simulation of the random oracles. By the properties of random oracles, the simulations of $\mathcal{H}_{\mathsf{FS}}$ and $\mathcal{H}$ are perfect. Next, we turn to the analysis of $\mathcal{B}$'s success probability. To facilitate the analysis, we define the following events Event, Event_1, Event_2, and Event_3:
  - Event: $\mathcal{B}$ succeeds.
  $$\Pr[\mathsf{Event}] = \Pr[\mathcal{B} \text{ succeeds}].$$
  - Event_1: (m, σ) is a valid forgery.
  $$\Pr[\mathsf{Event}_1] = \mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathsf{SIG},\mathcal{A}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}}(1^\lambda).$$
  - Event_2: $\mathcal{B}$ aborts.
  $$\Pr[\mathsf{Event}_2] = \Pr[\mathcal{B} \text{ aborts}].$$
  - Event_3: $((A, A_\circ), S, Z)$ is a valid transcript of $\Pi^{\mathcal{H}}$.

  Define
  $$P_1 := \Pr[\mathsf{Event}_2|\mathsf{Event}_1],$$
  $$P_2 := \Pr[\neg\mathsf{Event}_3 \mid \neg\mathsf{Event}_2 \cap \mathsf{Event}_1],$$
  and
  $$P_3 := \Pr[\mathsf{Event} \mid \neg\mathsf{Event}_2 \cap \mathsf{Event}_1 \cap \mathsf{Event}_3].$$
  By standard probabilistic arguments, we have
  $$\Pr[\mathsf{Event}] \geq P_3 \cdot \Pr[\mathsf{Event}_1] - P_1 - P_2.$$
  We now analyze P_1, P_2 and P_3 as follows:
  - $P_1 = \Pr[\mathsf{Event}_2|\mathsf{Event}_1]$. This case corresponds to $\mathcal{A}$ producing a valid forgery (m, σ) without querying $\mathcal{H}_{\mathsf{FS}}$ on
  $$x = (\mathsf{salt}, \mathsf{com}, (T_e)_{e\in[M]}, (\Delta K_e)_{e\in[M]}, m, \mathsf{ctr}).$$
  By the randomness of $\mathcal{H}_{\mathsf{FS}}$, the probability of this event is bounded by the guessing probability of $\mathcal{A}$. Hence,
  $$P_1 \leq \frac{1}{2^{2\lambda}}.$$
  - $P_2 := \Pr[\neg\mathsf{Event}_3|\mathsf{Event}_1 \cap \neg\mathsf{Event}_2]$. This case corresponds to $\mathcal{B}$ not aborting and (m, σ) being a valid forgery, yet $((A, A_\circ), S, Z)$ failing to be a valid proof for $\Pi^{\mathcal{H}}$. Again, by the randomness of $\mathcal{H}_{\mathsf{FS}}$ and the

completeness of $\Pi^{\mathcal{H}}$, we obtain:

$$P_2 \leq \frac{q^2}{2^{2\lambda}}.$$

– $P_3 := \Pr[\mathsf{Event} \mid \neg\mathsf{Event}_2 \cap \mathsf{Event}_1 \cap \mathsf{Event}_3]$. We analyze to obatin

$$P_3 \geq \Pr[\mathsf{Event}|\mathsf{Event}_3] - \frac{q^2}{2^{2\lambda}}.$$

According to the PoK-OE of the protocol $\Pi^{\mathcal{H}}$, we have

$$\Pr[\mathsf{Event}|\mathsf{Event}_3] \geq 1 - \epsilon_{\mathsf{ext}}(\lambda).$$

Finally, we conclude that

$$\mathsf{Adv}^{\mathcal{R}_{\beta\mathcal{L}^t}}_{\mathcal{B}}(1^\lambda) \geq \mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathcal{A}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}}(1^\lambda) - \epsilon_{\mathsf{ext}}(\lambda) - \frac{2q^2+1}{2^{2\lambda}}.$$

This shows that the EUF-KO advantage of $\mathcal{A}$ is bounded by the advantage of $\mathcal{B}$ in breaking the hardness of the relation $\mathcal{R}_{\beta\mathcal{L}^t}$, up to negligible terms. Therefore, under the assumed hardness of $\mathcal{R}_{\beta\mathcal{L}^t}$, the signature scheme SIG (i.e., $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}$) is EUF-KO secure in the ROM. □

C.7 Proof of Theorem 26

Proof. Let $\mathcal{A}$ be a quantum PPT adversary against the EUF-KO security, making at most q quantum queries to $\mathcal{H}_{\mathsf{FS}}$, and let $\mathcal{E}$ be a quantum PPT extractor as defined in the PoK-OE notion.

- **Construction.** We construct a quantum PPT algorithm $\mathcal{B}$. The construction of $\mathcal{B}$ is similar to that in the classical ROM proof, except for **RO Simulation Phase** and **Challenge Phase**.
  (1) **RO Simulation Phase.** $\mathcal{B}$ simulates $\mathcal{H}_{\mathsf{FS}}$ via CStO [60]. If $\mathcal{A}$ queries $\mathcal{H}$, then $\mathcal{B}$ forwards the query to $\mathcal{E}$ and returns the response to $\mathcal{A}$.
  (2) **Challenge Phase.** Once $\mathcal{A}$ outputs a forgery (m, σ), $\mathcal{B}$ parses

  $$\sigma = (\mathsf{salt}, h, \mathtt{ctr}, \mathtt{decom}_S, (\Delta K_e)_{e\in[M]}),$$

  and computes $S \leftarrow \mathsf{Expand}(h[:128])$. It then runs

  $$\mathsf{Verify}(pk, m, \sigma)$$

  and proceeds only if the output is 1.
  Next, $\mathcal{B}$ measures the internal state of the compressed oracle to obtain the database D_H. From D_H, $\mathcal{B}$ extracts $(T_e)_{e\in[M]}$. It then sends

  $$((A, A_\circ), S, Z)$$

  to $\mathcal{E}$, where $A = \mathtt{com}$, $A_\circ = (T_e, \Delta K_e)_{e\in[M]}$, and $Z = \mathtt{decom}_S$.

- **Probability Analysis.** According to Lemma 42, the simulation of the QROs by $\mathcal{B}$ is perfect. We now analyze the success probability of $\mathcal{B}$. By Lemma 43, we have

  $$\Pr\left[D_H(\mathsf{salt}, \mathtt{com}, (T_e)_{e\in[M]}, (\Delta K_e)_{e\in[M]}, m, \mathtt{ctr}) \neq h\right] \leq \frac{2}{2^{2\lambda}}.$$

To this end, we define the following database properties SUC and CL:

$$\mathsf{SUC} := \left\{ \begin{array}{c} \exists\, x = (\mathsf{salt}, \mathsf{com}, \{T_e\}_{e\in[M]}, (\Delta K_e)_{e\in[M]}, m, \mathtt{ctr}) \\ D : \text{s.t. } (x,h) \in D \ \wedge\ \mathsf{Verify}(pk, m, \sigma) = 1; \\ \text{but } \Pi.\mathsf{Verify}(pk, ((A, A_\circ), S, Z), \mathsf{pp}) \neq 1 \end{array} \right\},$$

and

$$\mathsf{CL} := \{D : \exists\, x \neq x' \text{ s.t. } D(x) = D(x') \neq \perp\}.$$

The condition $D_{H_{\mathsf{FS}}} \in \mathsf{SUC}$ means that $\mathcal{A}$ produces a valid forgery while $\Pi.\mathsf{Verify}^D(pk, ((\mathsf{com}, A_\circ), S, Z), \mathsf{pp}) \neq 1$. Similarly, $D_{H_{\mathsf{FS}}} \in \mathsf{CL}$ indicates that there exists a collision in $D_{H_{\mathsf{FS}}}$. We have

$$\begin{aligned} \mathsf{Adv}_{\mathcal{B}}^{\mathcal{R}_{\beta\mathcal{L}^t}}(1^\lambda) \geq & \mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathsf{SIG},\mathcal{A}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}}(1^\lambda) - \epsilon_{\mathsf{ext}}(\lambda) - \frac{2}{2^{2\lambda}} \\ & - \max_{\mathcal{A}} \Pr[D_{H_{\mathsf{FS}}} \in \mathsf{SUC} \cup \mathsf{CL}]. \end{aligned}$$

We denote

$$[\![\perp \xrightarrow{q} \mathsf{SUC} \cup \mathsf{CL}]\!] := \sqrt{\max_{\mathcal{A}} \Pr[D_{H_{\mathsf{FS}}} \in \mathsf{SUC} \cup \mathsf{CL}]}.$$

and introduce the database property $\mathsf{SZ}_{\leq i} := \{D : \#\{z : D(z) \neq \perp\} \leq i\}$, where q is the number of $\mathcal{A}$'s quantum queries to $\mathcal{H}_{\mathsf{FS}}$. We then analyze upper bounds for $[\![\perp \xrightarrow{q} \mathsf{SUC} \cup \mathsf{CL}]\!]$. By basic properties of quantum transition capacity and Lemma 46 (Lemma 5.6 in [25]), the upper bound of

$$[\![\perp \xrightarrow{q} \mathsf{SUC} \cup \mathsf{CL}]\!]$$

is

$$\sum_{i=0}^{q-1} [\![\mathsf{SZ}_{\leq i} \backslash \mathsf{SUC} \to \mathsf{SUC}]\!] + \sum_{i=0}^{q-1} [\![\mathsf{SZ}_{\leq i} \backslash \mathsf{CL} \to \mathsf{CL}]\!].$$

- $[\![\mathsf{SZ}_{\leq i} \backslash \mathsf{CL} \to \mathsf{CL}]\!]$. By Lemma 48 (Equation (8) in [25]), the collision probability of the database can be bounded as
$$[\![\mathsf{SZ}_{\leq i} \backslash \mathsf{CL} \to \mathsf{CL}]\!] \leq 2e\sqrt{\frac{(i+1)}{2^{2\lambda}}} \leq 2e\sqrt{\frac{q}{2^{2\lambda}}}.$$
- $[\![\mathsf{SZ}_{\leq i} \backslash \mathsf{SUC} \to \mathsf{SUC}]\!]$. Consider an arbitrary but fixed $D \in \mathcal{D}$ and input $x \in X$, assuming $D(x) = \perp$ (i.e., x is fresh). Define $\mathsf{P} := \mathsf{SZ}_{\leq s} \backslash \mathsf{SUC}$ and $\mathsf{P}' := \mathsf{SUC}$. If $D \notin \mathsf{SUC}$, then $D(x) = \perp$ and $\perp \notin \mathsf{P}'|_{D|^x}$. We set $\mathsf{L} := \mathsf{P}'_{D|^x}$. Therefore Hence,
$$\Pr[u \in \mathsf{L} | u \xleftarrow{\$} \{0,1\}^{2\lambda}] \leq \Pr[u = h | u \xleftarrow{\$} \{0,1\}^{2\lambda}] = \frac{q}{2^{2\lambda}}.$$
By Theorem 47, we obtain the bound
$$[\![\mathsf{SZ}_{\leq i} \backslash \mathsf{SUC} \to \mathsf{SUC}]\!] \leq \sqrt{\frac{10q}{2^{2\lambda}}}.$$

In conclusion, we derive the following bound:

$$\mathsf{Adv}_{\mathcal{B}}^{\mathcal{R}_{\beta\mathcal{L}^t}}(1^\lambda) \geq \mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathsf{SIG},\mathcal{A}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}}(1^\lambda) - \epsilon_{\mathsf{ext}}(\lambda) - \frac{8e^2q^3 + 20q^3}{2^{2\lambda}}.$$

This inequality shows that, under the assumed hardness of $\mathcal{R}_{\beta\mathcal{L}^t}$, the signature scheme $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}$ is EUF-KO secure in the QROM. □

C.8 Proof of Theorem 27

Proof. Let $\mathcal{A}$ be a PPT adversary against the EUF-CMA security of the signature scheme. We construct a reduction $\mathcal{B}$ against the EUF-KO security by leveraging the SHVZK property of $\Pi^{\mathcal{H}}$.

- **Construction.** Let $\mathtt{Sim}$ be the PPT simulator as defined in the SHVZK property. We construct a PPT adversary $\mathcal{B}$ as follows:
  (1) **Setup Phase.** On input of the public parameters pp and a public key pk, $\mathcal{B}$ runs $\mathcal{A}$ on the same inputs pp and pk.
  (2) **RO Simulation Phase.** $\mathcal{B}$ simulates the random oracle $\mathcal{H}_{\mathsf{FS}}$ via lazy sampling and maintains the query list $L_{H_{\mathsf{FS}}}$.
  (3) **Signing Oracle Simulation Phase.** Upon receiving a signing query on a message m, $\mathcal{B}$ invokes Sim. Concretely, it samples $\mathtt{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$ and $h^* \xleftarrow{\$} \{0,1\}^{2\lambda}$ uniformly at random. Then, it derives a challenge set $S \leftarrow \mathtt{expand}(h[:128])$, and invokes the simulator Sim to produce
  $$((\mathtt{com}, A_\circ), Z),$$
  where $A_\circ = (T_e, \Delta K_e)_{e\in[M]}$ and $Z = \mathtt{decom}_S$. Next, it sets
  $$\sigma := (\mathtt{salt}, h^*, \mathtt{ctr}, \mathtt{decom}_S, (\Delta K_e)_{e\in[M]})$$
  with $\sigma_1 = (\mathtt{salt}, \mathtt{com}, A_\circ)$. If $x^* = (\mathtt{salt}, \sigma_1, m, \mathtt{ctr})$ has not been queried before, $\mathcal{B}$ programs the oracle so that $(x^*, h^*) \in L_{H_{\mathsf{FS}}}$ and then returns σ; otherwise, it aborts.
  (4) **Forgery.** When $\mathcal{A}$ outputs a forgery (m, σ) in the EUF-CMA game, $\mathcal{B}$ forwards (m, σ) to the EUF-KO game.
- **Probability Analysis.** The simulation of $\mathcal{H}_{\mathsf{FS}}$ is perfect. For the signing oracle simulation, $\mathcal{B}$ aborts if x has already been queried by $\mathcal{A}$. Since $\mathtt{salt}$ is chosen uniformly at random, the probability that $\mathcal{B}$ aborts is at most $\frac{q_s(q_s+q_H)}{2^{2\lambda}}$, where q_H and q_s denote the number of queries made by $\mathcal{A}$ to $\mathcal{H}_{\mathsf{FS}}$ and the signing oracle, respectively. When $\mathcal{B}$ does not abort, by the definition of SHVZK, the simulation of the signing oracle can be distinguished from the real one by $\mathcal{A}$ with probability at most $\mathsf{Adv}^{\mathsf{SHVZK}}_{\Pi^{\mathcal{H}},\mathcal{A}^{\mathcal{H}}}(1^\lambda)$. Hence, we obtain
  $$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(1^\lambda) \le \mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathcal{B}}(1^\lambda) + \mathsf{Adv}^{\mathsf{SHVZK}}_{\Pi^{\mathcal{H}},\mathcal{A}}(1^\lambda) + \frac{q_s(q_s+q_H)}{2^{2\lambda}}.$$

This inequality shows that if the underlying protocol $\Pi^{\mathcal{H}}$ satisfies SHVZK and $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}$ is EUF-KO secure, then the signature scheme $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}$ is also EUF-CMA secure in the ROM. By combining this with Theorem 25, we obtain the claimed result. □

C.9 Proof of Theorem 28

Proof. The proof idea is similar to that of Theorem 27. In particular, $\mathcal{B}$ simulates $\mathcal{H}_{\mathsf{FS}}$ using the CStO technique [60] in the **RO Simulation Phase**. Additionally, in the **Signing Simulation Phase**, $\mathcal{B}$ reprograms the QRO rather than aborts. The probability that $\mathcal{A}$ detects such reprogramming is bounded using Lemma 49.

Define $\mathcal{H}'_{\mathsf{FS}}$ as follows: if $x \neq x^*$, then $\mathcal{H}'_{\mathsf{FS}}(x) = \mathcal{H}_{\mathsf{FS}}(x)$; if $x = x^*$, then $\mathcal{H}'_{\mathsf{FS}}(x) = h^*$. For any quantum PPT adversary $\mathcal{D}$ and quantum PPT extractor $\mathcal{E}$ defined as in Lemma 49, denote
$$P_{\mathsf{left}} := \Pr[b \leftarrow \mathcal{D}^{\mathcal{H}'_{\mathsf{FS}}}(z) : b = 1],$$

$$P_{\text{right}} := \Pr[b \leftarrow \mathcal{D}^{\mathcal{H}_{\mathsf{FS}}}(z) : b = 1],$$

and

$$P_{\text{guess}} := \Pr[x = x^* \neq \varnothing : x \leftarrow \mathcal{E}^{\mathcal{H}_{\mathsf{FS}}}(z)].$$

According to Lemma 49, we have

$$|P_{\text{left}} - P_{\text{right}}| \leq 2q\sqrt{P_{\text{guess}}},$$

where q denotes the number of the distinguisher's queries to the QRO. Since the `salt` in x^* is chosen uniformly at random, it follows that $P_{\text{guess}} \leq \frac{1}{2^{2\lambda}}$. Putting this together, we obtain

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathcal{B}}(1^\lambda) + \mathsf{Adv}^{\mathsf{SHVZK}}_{\Pi^{\mathcal{H}},\mathcal{A}}(1^\lambda) + \frac{2q_s \cdot (q_s + q_H)}{2^\lambda},$$

where q_H and q_S denote the number of queries made by $\mathcal{A}$ to $\mathcal{H}_{\mathsf{FS}}$ and the signing oracle.

This inequality shows that if the underlying protocol $\Pi^{\mathcal{H}}$ satisfies SHVZK and $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}$ is EUF-KO secure in the QROM, then the signature scheme $\textsc{Pegasus}^{\mathcal{H}_{\mathsf{FS}},\mathcal{H}}$ is also EUF-CMA secure in the QROM. Together with Theorem 26, this completes the proof. □

C.10 Correctness of PegaRing

In the verification, to make sure

$$T'_e = \mathcal{L}^t_0(o_e) - \sum_{i=1}^{n} pk^{(i)}_{\bar{\ell}^{(i)}_e} = \mathcal{L}^t_0(r_e) - \sum_{i\neq\pi} pk^{(i)}_{\bar{\ell}^{(i)}_e},$$

the value o_e must be reconstructed as

$$o_e = (K + I_{\bar{\ell}^{(\pi)}_e})r_e.$$

However, opening BAVC at position $\bar{\ell}_e$ only allows the verifier to recover $(K + I_{\bar{\ell}_e})r_e$ (see the correctness of Pegasus). To bridge this gap, we adjust the key share by setting

$$\Delta K_e = (K - I_{\bar{\ell}_e} + I_{\bar{\ell}^{(\pi)}_e})r_e + \sum_{\ell=0}^{L-1} I_\ell r_{e,\ell},$$

which cancels the term $I_{\bar{\ell}_e} r_e$ obtained from the opening and injects the required $I_{\bar{\ell}^{(\pi)}_e} r_e$, so the verifier recovers o_e with respect to the real signer's challenge.

Another subtlety for correctness is the modular reduction hidden in the key share adjustment. Recall that we have

$$I_{\bar{\ell}_e} = a_{\bar{z}_e} + b\bar{c}_e \;\text{ and }\; I_{\bar{\ell}^{(\pi)}_e} = a_{\bar{z}^{(\pi)}_e} + b\bar{c}^{(\pi)}_e.$$

In the formation of key share adjustment, we let the real signer iterate the pair of values $(a_{\bar{z}_e}, a_{\bar{z}^{(\pi)}_e})$ with

$$\bar{z}_e - \bar{z}^{(\pi)}_e = \sum_{i\neq\pi} \bar{z}^{(i)}_e$$

for every $\bar{z}_e = 0, 1, \ldots, z-1$ (note that $\sum_{i\neq\pi} \bar{z}^{(i)}_e$ is a fixed value since all non-signers' challenges are generated in the first round). For the pair of $\bar{c}_e$ and $\bar{c}^{(\pi)}_e$, we use the fixed gap $\bar{c}'_e = \sum_{i\neq\pi} \bar{c}^{(\pi)}_e$ rather than enumerating since these values are in the same additively cyclic group $\mathbb{Z}_c$. However, the gap is defined over modulo c. Concretely, we have

$$\bar{c}^{(\pi)}_e = \bar{c}_e - \bar{c}'_e \pmod{c} = \begin{cases} \bar{c}_e - \bar{c}'_e & \text{if } \bar{c}_e \geq \bar{c}'_e, \\ \bar{c}_e - \bar{c}'_e + c & \text{otherwise.} \end{cases}$$

To hide which case holds, we prepare two key share adjustments, $\Delta K^0_{e,j}$ (without wraparound) and $\Delta K^1_{e,j}$ (with wraparound, including the extra $bc \cdot r_e$), and commit to them in random order. With this issue handled, the remaining correctness follows directly from PEGASUS and DualRing.

C.11 Proof of Theorem 31

PROOF. Let $\mathcal{A}$ be a PPT adversary against the insider-corruption unforgeability of PEGARING, making q_h^{FS} and q_h^{MT} queries to $\mathcal{G}_{\mathsf{FS}}$ and $\mathcal{G}_{\mathsf{MT}}$, respectively.

- **Construction.** The construction of $\mathcal{B}$ proceeds as follows:
  (1) **Steup Phase.** Upon receiving the public parameters and a statement $pk^* = \mathcal{L}_K^t(\mathcal{I})$, $\mathcal{B}$ selects correlated parameters and constructs $\mathsf{R\text{-}pp}$. It then chooses an index $i^* \in [N]$ uniformly at random. For every $j \in [N]$ with $j \neq i^*$, $\mathcal{B}$ generates a key pair $(pk_j, sk_j) \leftarrow \mathsf{R\text{-}KeyGen}(\mathsf{R\text{-}pp})$, and sets $pk_{i^*} = pk^*$. Let $\mathbb{PK} = \{pk_1, \ldots, pk_N\}$ denote the resulting public key list. Finally, $\mathcal{B}$ forwards $(\mathsf{R\text{-}pp}, \mathbb{PK})$ to $\mathcal{A}$ and $(\mathsf{R\text{-}pp}, pk^*)$ to the extractor $\mathcal{E}$ of PoK-OE.
  (2) **RO Simulation Phase.** $\mathcal{B}$ simulates the random oracles $\mathcal{G}_{\mathsf{FS}}$ and $\mathcal{G}_{\mathsf{MT}}$ for $\mathcal{A}$ using lazy sampling, while maintaining the corresponding query lists L_G^{FS} and L_G^{MT}. If $\mathcal{A}$ makes a query to $\mathcal{H}$, then $\mathcal{B}$ forwards this query to $\mathcal{E}$ and returns the response back to $\mathcal{A}$.
  (3) **Corruption Oracle Simulation Phase.** For each query i, if $pk_i \notin \mathbb{PK}$ or $pk_i = pk^*$, then $\mathcal{B}$ aborts. Otherwise, it returns the corresponding sk_i.
  (4) **Signing Oracle Simulation Phase.** For any query
  $$(\mathsf{msg}, \mathcal{PK}, pk_\pi),$$
  $\mathcal{B}$ aborts if $pk_\pi \notin \mathbb{PK}$. Furthermore, if $pk_\pi \in \mathcal{PK} \backslash \{pk^*\}$ then $\mathcal{B}$ returns $\sigma \leftarrow \mathsf{R\text{-}Sign}(\mathsf{msg}, sk_\pi, \mathcal{PK}, \mathsf{R\text{-}pp})$. Otherwise, $\mathcal{B}$ computes the following:
    (a) $\mathcal{B}$ randomly selects $h_1, \ldots, h_n \xleftarrow{\$} \{0,1\}^\lambda$ and expands each seed to $(\bar{\ell}_e^{(i)})_{e \in [M]} \leftarrow \mathsf{expand}(h_i)$, where $\bar{\ell}_e^{(i)} \in [0, L-1]$.
    (b) $\mathcal{B}$ computes $(\bar{z}_e, \bar{c}_e) = \left(\left\lfloor \frac{\bar{\ell}_e^{(i)}}{c} \right\rfloor, \bar{\ell}_e^{(i)} \pmod c \right)$. Then it computes $\bar{z}_e = \sum_{i=1}^n \bar{z}_e^{(i)} \pmod z$, $\bar{c}_e = \sum_{i=1}^n \bar{c}_e^{(i)} \pmod c$, and $\bar{\ell}_e = \bar{z}_e c + \bar{c}_e$.
    (c) $\mathcal{B}$ uniformly randomly selects $\Delta K_e \xleftarrow{\$} \mathbb{F}_p$ and computes
    $$o_e = I_{\bar{\ell}_e} \sum_{\ell=0, \ell \neq \bar{\ell}_e}^{L-1} r_{e,\ell} + \sum_{\ell=0, \ell \neq \bar{\ell}_e}^{L-1} r_{e,\ell} I_\ell + \Delta K_e.$$
    (d) $\mathcal{B}$ randomly choose $b_e \xleftarrow{\$} \{0,1\}$ and set
    $$\Delta K^{b_e}_{e,\bar{z}_e} = \Delta K_e, \Delta K^{1-b_e}_{e,\bar{z}_e} \xleftarrow{\$} \mathbb{F}_p.$$
    For other $j \in [0, z-1] \backslash \{\bar{z}_e\}$, $(\Delta K^0_{e,j}, \Delta K^1_{e,j}) \xleftarrow{\$} \mathbb{F}_p^2$.
    (e) $\mathcal{B}$ computes $T_e = \mathcal{L}_0^t(o_e) - \sum_{i=1}^n pk^{(i)}_{\bar{\ell}_e^{(i)}}$.
    (f) $\mathcal{B}$ computes all other first round components following the protocol.
    (g) $\mathcal{B}$ checks whether there exists an h such that
    $$((\mathsf{com}, \mathsf{com}_{\mathsf{MT}}, (T_e)_{e \in [M]}), h) \in L_G^{\mathsf{FS}}.$$

If such an h exists, $\mathcal{B}$ aborts. Otherwise, it computes $h[:128] = \bigoplus_{i=1}^{n} h_i$. This programs the random oracle $\mathcal{G}_{\mathsf{FS}}$ so that, on input $(\mathtt{com}, \mathtt{com}_{\mathsf{MT}}, (T_e)_{e\in[M]})$, it outputs h.

(h) $\mathcal{B}$ follows the remaining steps to generate σ honestly.

(5) **Challenge Phase.** Once $\mathcal{A}$ outputs a forgery

$$(\mathtt{msg}^*, \sigma^*, \mathcal{PK}^*),$$

$\mathcal{B}$ parses σ^* to obtain

$$(\mathtt{salt}, \mathtt{ctr}, \mathtt{decom}_S, \mathtt{path}_I, (\Delta K_e, v_e, d_e)_{e\in[M]}), (h_i)_{i\in[n]}).$$

(a) For all $i \in [n]$, $\mathcal{B}$ computes the following
- $(\bar{\ell}_e^{(i)}) \leftarrow \mathsf{expand}(h_i)$,
- $(\bar{z}_e^{(i)}, \bar{c}_e^{(i)}) = \left(\lfloor \frac{\bar{\ell}_e^{(i)}}{c} \rfloor, \bar{\ell}_e^{(i)} \pmod c\right)$, note that $I_{\bar{\ell}_e^{(i)}} = a_{\bar{z}_e} + b\bar{c}_e$.

(b) $\mathcal{B}$ computes the mutual challenge

$$\bar{z}_e = \sum_{i=1}^{n} \bar{z}_e^{(i)} \pmod z, \ \bar{c}_e = \sum_{i=1}^{n} \bar{c}_e^{(i)} \pmod c.$$

(c) $\mathcal{B}$ checks if $pk^* \in \mathcal{PK}$,
- If $pk^* \notin \mathcal{PK}$, $\mathcal{B}$ aborts.
- Otherwise, $\mathcal{B}$ finds the index of pk^* in $\mathcal{PK}$ and set the index be π.

(d) $\mathcal{B}$ checks the query list L_G^{MT} for any entry of the form

$$((\mathtt{salt}, e, \bar{z}_e, b, \Delta K'_e), v_e),$$

where $b \in \{0, 1\}$.
- If no such entry exists, $\mathcal{B}$ aborts.
- Otherwise, $\mathcal{B}$ extracts $\Delta K'_e$ and computes $r_e^* = \frac{\Delta K_e - \Delta K'_e}{bc}$.

(e) $\mathcal{B}$ checks the query list L_G^{FS} for any entry of the form

$$((\mathtt{salt}, \sigma_1, \mathtt{msg}, \mathtt{ctr}, \mathcal{PK}), h) \in L_G^{\mathsf{FS}},$$

where $\sigma_1 = (\mathtt{com}, \mathtt{com}_{\mathsf{MT}}, (T_e)_{e\in[M]})$ and $h[:128] = \bigoplus_{i=1}^{n} h_i$.
- If no such entry exists, $\mathcal{B}$ aborts.
- Otherwise, $\mathcal{B}$ extracts $(T_e)_{e\in[M]}$ and computes $T_e^* = T_e + \sum_{i=0, i\neq\pi}^{n} pk_{\bar{\ell}_e^{(i)}}^{(i)}$.

(f) $\mathcal{B}$ computes $\Delta K_e^* = \Delta K_e - I_{\bar{\ell}_e^{(\pi)}} r_e + I_{\bar{\ell}_e} r_e$ and sends $((A, A_\circ), S, Z)$ to $\mathcal{E}$, where $A = \mathtt{com}$, $A_\circ = (T_e^*, \Delta K_e^*)_{e\in[M]}$, $S = (\bar{\ell}_e)_{e\in[M]}$ and $Z = \mathtt{decom}_S$

- **Output Phase.** Finally, when $\mathcal{E}$ outputs sk^*, $\mathcal{B}$ outputs sk^*.
- **Probability Analysis.** By the properties of random oracles, the simulation of $\mathcal{G}_{\mathsf{FS}}$ and $\mathcal{G}_{\mathsf{MT}}$ is perfect. To facilitate the analysis of $\mathcal{B}$'s success probability, we define the following probabilistic events:
  - Event: $\mathcal{B}$ succeeds.

$$\Pr[\mathsf{Event}] = \Pr[\mathcal{B} \text{ succeeds}] = \mathsf{Adv}_{\mathcal{B}}^{\mathcal{R}_{\beta\mathcal{L}^t}}(1^\lambda).$$

  - Event_1: $\mathcal{B}$ wins the unforgeability game.

$$\Pr[\mathsf{Event}_1] = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{EUF}}(1^\lambda).$$

- Event_2: $\mathcal{B}$ aborts during the **Corruption Oracle Simulation Phase**.
- Event_3: $\mathcal{B}$ aborts in step (g) of the **Signing Oracle Simulation Phase** 4g.
- Event_4: $\mathcal{B}$ aborts in step (c) of the **Challenge Phase** 5c.
- Event_5: $\mathcal{A}$ aborts in steps (d) or (e) of the **Challenge Phase** 5d5e.
- Event_6: $\mathcal{B}$ does not abort, $\mathcal{A}$ outputs a valid ring-signature forgery, but the extracted Σ-protocol transcript is invalid.

Since the simulation of the corruption oracle is perfect except when $\mathcal{B}$ aborts, and the simulation of the signing oracle is perfect except when $\mathcal{B}$ aborts, $\mathcal{B}$'s simulation for $\mathcal{A}$ is therefore indistinguishable whenever $\neg\mathsf{Event}_2 \wedge \neg\mathsf{Event}_3$. Denote $p_1 = \Pr[\neg\mathsf{Event}_2 \wedge \neg\mathsf{Event}_3]$. It is clear that

$$p_1 = \Pr[\neg\mathsf{Event}_2] \cdot \Pr[\neg\mathsf{Event}_3].$$

We analyze the probabilities of $\Pr[\neg\mathsf{Event}_2]$ and $\Pr[\neg\mathsf{Event}_3]$ as follows:

- $\Pr[\neg\mathsf{Event}_2]$. Since i^* is chosen randomly, when $\mathcal{A}$ makes q_c queries to O_{cor}, the upper bound of $\Pr[\neg\mathsf{Event}_2]$ is
$$(1-\frac{1}{N})(1-\frac{1}{N-1})\cdots(1-\frac{N-q_c}{N-q_c}+1) = \frac{N-q_c}{N}.$$
- $\Pr[\neg\mathsf{Event}_3]$. Due to the randomness of salt, when $\mathcal{A}$ makes q_s queries to O_{sig}, we have
$$\Pr[\neg\mathsf{Event}_3] \geq 1 - \frac{q_s \cdot (q_s + q_{H^{\mathsf{FS}}})}{2^{2\lambda}}.$$

Then,

$$p_1 \geq (1 - \frac{q_s \cdot (q_s + q_{H^{\mathsf{FS}}})}{2^{2\lambda}}) \cdot \frac{N-q_c}{N}.$$

When $\mathcal{B}$'s simulations are perfect, we analyze

$$p_2 := \Pr[\mathsf{Event}_4 | \mathsf{Event}_1].$$

Note that Event_4 implies $pk^* \notin \mathcal{PK}$, while Event_1 implies $\mathcal{PK} \notin Q$. Therefore,

$$p_2 = \prod_{i=0}^{n-1}(1 - \frac{1}{N-q_c-i}) = \frac{N-q_c-n}{N-q_c}.$$

According to results in probability theory, we have

$$\Pr[\mathsf{Event}] \geq p_1 \cdot (1-p_2)\Pr[\mathsf{Event}_1] - \Pr[\mathsf{Event}_5] - \Pr[\mathsf{Event}_6] - \epsilon_{\mathsf{ext}}.$$

Next, we analyze $\Pr[\mathsf{Event}_5]$ and $\Pr[\mathsf{Event}_6]$:

- $\Pr[\mathsf{Event}_5]$. The node v_e is not found in L_G^{MT}, which happens with probability at most $\frac{1}{2^{2\lambda}}$, and across all M values, the probability is $\frac{M}{2^{2\lambda}}$. The challenge h is not found in L_G^{FS}, which happens with probability at most $\frac{1}{2^{2\lambda}}$. Hence, $\Pr[\mathsf{Event}_5] \leq \frac{1+M}{2^{2\lambda}}$.
- $\Pr[\mathsf{Event}_6]$. It denotes the probability that $\mathcal{B}$ does not abort, $\mathcal{A}$ outputs a valid ring signature forgery, but the extracted Σ-protocol transcript is not a valid proof. By the randomness of $\mathcal{G}_{\mathsf{MT}}$ and $\mathcal{G}_{\mathsf{FS}}$, this happens with probability at most $\frac{(q_{H^{\mathsf{FS}}})^2+(q_{H^{\mathsf{MT}}})^2}{2^{2\lambda}}$.

Finally, we conclude that

$$\frac{n}{N}(1-\frac{q_s\cdot(q_s+q_{H^{\mathsf{FS}}})}{2^{2\lambda}})\cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathcal{A}}(1^\lambda)$$
$$\leq \mathsf{Adv}^{\mathcal{R}_{\beta\mathcal{L}^t}}_{\mathcal{B}}(1^\lambda)+\frac{(q_{H^{\mathsf{FS}}})^2+(q_{H^{\mathsf{MT}}})^2+M+1}{2^{2\lambda}}+\epsilon_{\mathsf{ext}}.$$

□

C.12 Proof of Theorem 32

Proof. The construction of $\mathcal{B}$ is described as follows.

- **RO Simulation Phase.** $\mathcal{B}$ simulates $\mathcal{G}_{\mathsf{FS}}$ and $\mathcal{G}_{\mathsf{MT}}$ using the CStO technique [60].
- **Signing Oracle Simulation Phase.** In the step (g), $\mathcal{B}$ reprograms the $\mathcal{G}_{\mathsf{FS}}$ such that $\mathcal{G}_{\mathsf{FS}}((\mathtt{salt},\sigma_1,\mathtt{msg},\mathtt{ctr},\mathcal{PK}))=h$ rather than aborts.
- **Challenge Phase.** Once $\mathcal{A}$ outputs a forgery, $\mathcal{B}$ measures its internal state to get dababase D_{FS} and D_{MT}. To simplicity, we denote the dababase D such that $D(1,\cdot)=D_{\mathsf{FS}}(\cdot)$ and $D(2,\cdot)=D_{\mathsf{MT}}(\cdot)$. Then in step (d) and (e), $\mathcal{B}$ checks the database rather than the query lists.

Let $\mathbf{x}=\{(\mathtt{salt},e,\bar{z}_e,b,\Delta K'_e)\}_{e\in[M]}$ and $\mathbf{y}=\{v_e\}_{e\in[M]}$. Then, we deonte the database SUC and CL as follows:

$$\mathsf{SUC}:=\left\{\begin{array}{c}\exists\,\mathbf{x}\ s.t(1\|\mathbf{x},\mathbf{y})\in D\ satisifies\\ D:\|\mathcal{A}\ succeds\|=1\ for\ D^{-1}(h)\\ \wedge\ \Pi.\mathsf{Verify}^{\mathcal{H}}(pk^*,((\mathsf{com},A_\circ),S,Z),\mathsf{pp})\neq 1\end{array}\right\},$$

and

$$\mathsf{CL}:=\{D:\exists\ x\neq x'\ s.t\ D(x)=D(x')\neq\perp\}.$$

Similar to the analysis in the Theorem 31 and Theorem 26, we can get[6]

$$\Pr[\mathsf{Event}]\geq p\cdot\Pr[\mathsf{Event}_1]-\max_{\mathcal{A}}\Pr[D_G\in\mathsf{SUC}\cup\mathsf{CL}]$$
$$-\Pr[\mathcal{B}\ aborts|\mathcal{A}\ succeds]-\epsilon_{\mathsf{ext}}.$$

- Similar to the analysis in the proof of Theorem 31 , $p\geq(1-\frac{q_s\cdot(q_s+q_{H^{\mathsf{FS}}})}{2^\lambda})\cdot\frac{n}{N}$.
- According to Lemma 43, we have

$$\Pr[\mathcal{B}\ aborts|\mathcal{A}\ succeds]\leq\frac{2(M+1)}{2^{2\lambda}}.$$

- By basic properties of quantum transition capacity in Lemma 46 and Theorem 47, the upper bound of

$$\sqrt{\max_{\mathcal{A}}\Pr[D_G\in\mathsf{SUC}\cup\mathsf{CL}]}$$

is

$$\sum_{i=0}^{q-1}[\![\mathsf{SZ}_{\leq i}\backslash\mathsf{SUC}\rightarrow\mathsf{SUC}]\!]+\sum_{i=0}^{q-1}[\![\mathsf{SZ}_{\leq i}\backslash\mathsf{CL}\rightarrow\mathsf{CL}]\!].$$

Here, $q=q_{H^{\mathsf{FS}}}+q_{H^{\mathsf{MT}}}$.

  - $[\![\mathsf{SZ}_{\leq i}\backslash\mathsf{CL}\rightarrow\mathsf{CL}]\!]$. By Lemma 48 (Equation (8) in [25]), the collision probability of the QRO can be bounded as $[\![\mathsf{SZ}_{\leq i}\backslash\mathsf{CL}\rightarrow\mathsf{CL}]\!]\leq 2e\sqrt{\frac{(i+1)}{2^{2\lambda}}}\leq 2e\sqrt{\frac{q}{2^{2\lambda}}}$.

[6]Here, $\mathcal{B}\ aborts$ means that $\mathcal{B}$ aborts in steps (d) or (e) of the Challenge Phase.

– $[\![\mathsf{SZ}_{\le i}\backslash\mathsf{SUC} \to \mathsf{SUC}]\!]$. By Theorem 47, we obtain the bound

$$[\![\mathsf{SZ}_{\le i}\backslash\mathsf{SUC} \to \mathsf{SUC}]\!] \le \sqrt{\frac{10qM}{2^{2\lambda}}}.$$

In conclusion, we derive the following bound:

$$\frac{n}{N}(1-\frac{q_s\cdot(q_s+q_{H^{\mathsf{FS}}})}{2^\lambda})\cdot \mathsf{Adv}^{\mathsf{EUF}}_{\mathcal{A}}(1^\lambda) \le \mathsf{Adv}^{\mathcal{R}_{\beta\mathcal{L}^t}}_{\mathcal{B}}(1^\lambda) + \frac{8e^2q^3+20Mq^3+2M+2}{2^{2\lambda}} + \epsilon_{\mathsf{ext}},$$

where $q = q_{H^{\mathsf{FS}}} + q_{H^{\mathsf{MT}}}$. □

C.13 Proof of Theorem 33

Proof. Let hash function $\mathcal{H}_{\mathsf{FS}}$ be modeled as a random oracle. $\mathcal{B}$ runs signature setup to generate R-pp and gives R-pp to $\mathcal{A}$. $\mathcal{B}$ simulates the oracles similarly as in unforgeability game. Except that in anonymity game, $\mathcal{A}$ is given all public-secret key pairs and can insert maliciously generated keys.

Challenge. $\mathcal{A}$ gives $\mathcal{B}$ a message $\mathtt{msg}^*$, a set of public keys $\mathcal{PK}^*$ with n public keys, and two signer indices i_0, i_1 w.r.t. the same ring $\mathcal{PK}^*$. $\mathcal{B}$ computes the signature as follows:

(1) $\mathcal{B}$ randomly selects $h_1,\dots,h_n \xleftarrow{\$} \{0,1\}^\lambda$ and expands each seed to $(\bar{\ell}_e^{(i)})_{e\in[M]} \leftarrow \mathsf{expand}(h_i)$, where $\bar{\ell}_e^{(i)} \in [0, L-1]$.

(2) $\mathcal{B}$ computes $(\bar{z}_e, \bar{c}_e) = \left(\left\lfloor \frac{\bar{\ell}_e^{(i)}}{c} \right\rfloor, \bar{\ell}_e^{(i)} \pmod c\right)$. Then it computes $\bar{z}_e = \sum_{i=1}^n \bar{z}_e^{(i)}$, $\bar{c}_e = \sum_{i=1}^n \bar{c}_e^{(i)}$, and $\bar{\ell}_e = \bar{z}_e c + \bar{c}_e$.

(3) $\mathcal{B}$ randomly selects $\Delta K_e^* \xleftarrow{\$} \mathbb{F}_p$ and computes

$$o_e* = I_{\bar{\ell}_e} \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell} + \sum_{\ell=0,\ell\neq\bar{\ell}_e}^{L-1} r_{e,\ell}, I_\ell + \Delta K_e^*.$$

(4) $\mathcal{B}$ generates $b_e \xleftarrow{\$} \{0,1\}$ and set $\Delta K_{e,\bar{z}_e}^{b_e} = \Delta K_e^*$ where $\Delta K_{e,\bar{z}_e}^{1-b_e} \xleftarrow{\$} \mathbb{F}_p$. For other $j \in [0, z-1]\backslash\{\bar{z}_e\}$,

$$(\Delta K_{e,j}^{b_e}, \Delta K_{e,j}^{1-b_e}) \xleftarrow{\$} \mathbb{F}_p^2.$$

(5) $\mathcal{B}$ computes $T_e^* = \mathcal{L}_0^t(o_e^*) - \sum_{i=1}^n pk_{\bar{\ell}_e^{(i)}}^{(i)}$.

(6) $\mathcal{B}$ computes all other first round components following the protocol.

(7) Finally, $\mathcal{B}$ programs the random oracle $\mathcal{H}$, which on input $(\mathtt{com}, \mathtt{com}_{\mathsf{MT}}, (T_e^*)_{e\in[M]})$, outputs h such that $h[:128] = \bigoplus_{i=1}^n h_i$ and the last 128 bits are randomly chosen by $\mathcal{B}$.

(8) $\mathcal{B}$ follows the remaining steps to generate σ honestly.

Then, $\mathcal{B}$ returns the simulated signature σ to $\mathcal{A}$ and randomly picks $\mathtt{bit} \xleftarrow{\$} \{0,1\}$.

Hybrid Argument. It is obvious that the simulated signature passes the verification, we omit the detail of the simulation correctness. We show that the distribution of simulated ring signature is indistinguishable from the real distribution in a hybrid argument, assume BAVC is hiding.

- H0(**Real**) Honest prover (knows K w.r.t. $pk_\pi \in \mathcal{PK}$) follows Algorithm **??** and **??** to output a real signature

$$\sigma = (\mathtt{salt}, \mathtt{ctr}, \mathtt{decom}_S, \mathtt{path}_I, (\Delta K_e, v_e, d_e)_{e\in[M]}), (h_i)_{i\in[n]}).$$

Here, the signer runs BAVC and Merkle tree commitment honestly such that

$$r_e = \sum_{\ell=0}^{L-1} r_{e,\ell},\ T_e = \mathcal{L}_0^t(r_e) - \sum_{i=1, i\neq\pi}^{n} pk_{\bar{\ell}_e^{(i)}}^{(i)}$$

and

$$\Delta K_e = (K - I_{\bar{\ell}_e} + I_{\bar{\ell}_e^{(\pi)}}) r_e + \sum_{\ell=0}^{L-1} I_\ell r_{e,\ell},$$
$$v_e = \mathcal{H}(\mathtt{salt}, e, \bar{z}_e, 0/1, \Delta K_e \pm bcr_e).$$

- H1(**Randomizing unopened leaves for BAVC**). For each $e \in [M]$, replace the unopened message $r_{e,\bar{\ell}_e}$ with an independent, uniformly sampled $r'_{e,\bar{\ell}_e} \xleftarrow{\$} \mathbb{F}_p$. Because BAVC is hiding, replacing unopened message with fresh random values yields a distribution for the new signature that is indistinguishable from the real signature, with distinguishing advantage at most $\mathsf{AdvHD}_{\mathcal{A}}^{\mathsf{BAVC}}$. Concretely, after the substitution, we have

$$r'_e = \sum_{\ell\neq\bar{\ell}_e} r_{e,\ell} + r_{e,\bar{\ell}_e'},\ T'_e = \mathcal{L}_0^t(r'_e) - \sum_{i=1, i\neq\pi}^{n} pk_{\bar{\ell}_e^{(i)}}^{(i)}$$

and

$$\Delta K'_e = (K - I_{\bar{\ell}_e} + I_{\bar{\ell}_e^{(\pi)}}) r'_e + \sum_{\ell\neq\bar{\ell}_e} I_\ell r_{e,\ell} + I_{\bar{\ell}_e} r_{e,\bar{\ell}_e},$$
$$v_e = \mathcal{H}_{\mathsf{MT}}(\mathtt{salt}, e, \bar{z}_e, 0/1, \Delta K'_e \pm bcr'_e)$$

  Since $r'_{e,\bar{\ell}_e}$ is uniform and independent, r'_e is uniform over $\mathbb{F}_p$. Hence, T'_e is the Legendre or t-th power residue PRF of a uniform input (minus a fixed constant value). For all $j \in [0, z-1]$, since we can write $\Delta K_{e,j}^{b'} = A_{e,j}^b r'_{e,\bar{\ell}_e} + B_{e,j}$ for $b \in \{0,1\}$. As long as $A_{e,j}^b \neq 0$, which happens with probability at most $2ML/p$, $\Delta K_{e,j}^{0'}$ and $\Delta K_{e,j}^{1'}$ are all uniformly random from the adversary's view.
- H2**(Simulator)**. Replacing ΔK_e with independent uniformly random ΔK_e^* and set $T_e^* = \mathcal{L}_0^t(o_e^*) - \sum_{i=1}^n pk_{\bar{\ell}_e^{(i)}}$ as in simulation does not change the distribution of signature from H2. Furthermore, since the value committed in v_e and other Merkle tree nodes are uniformly random in H2, the simulation is a perfect hop from H1.

Output $\mathcal{A}$ outputs a guess $\mathtt{bit}'$. Observe that $\mathtt{bit}$ is not used in the generation of σ, hence $\mathcal{A}$ wins with probability $1/2$.

Probability Analysis. We now analyze the probability that $\mathcal{B}$ aborts in the above simulation. For q_h queries made to $\mathcal{H}$ and q_s queries made to signing oracle, $\mathcal{B}$ aborts if the simulated σ_1 was queried. Due to the uniformly random salt of length 2λ bit, for each signing oracle query, the probability that $\mathcal{A}$ has queried σ_1 is at most $\frac{q_h+q_s}{2^{2\lambda}}$. Then, for all q_s queries, the aborting probability is at most $\frac{q_s(q_s+q_h)}{2^{2\lambda}}$. □