Traceable Verifiable Random Functions

Dan Boneh, Aditi Partap, and Lior Rotem

Stanford University
{dabo,aditi712,lrotem}@cs.stanford.edu

Abstract. A threshold verifiable random function (threshold VRF) is a VRF where the evaluation key is secret shared among n parties, and a quorum of t parties is needed to evaluate the VRF. Threshold VRFs are used widely in practice in applications such as randomness beacons and deterministic wallets. Despite their long history, the question of accountability for leaking key shares in a threshold VRF has not been studied. Specifically, consider a set of f parties who use their key shares to create an evaluation box E that lets anyone evaluate the VRF at any point in the domain of the VRF. When f is less than the threshold t, this box E must also take as input $t - f$ additional evaluation shares. Our goal is to design a threshold VRF where there is a tracing algorithm that can trace any such box E to the coalition of f parties that created it, using only blackbox access to E. The risk of tracing should deter the coalition from selling such a box. Questions in this vein were previously explored in the context of threshold decryption and secret sharing. Here we define and study traceability for a threshold VRF.
Our traceable threshold VRF is built from a VRF based on Paillier encryption. The starting point for our tracing algorithm is the tracing technique of Boneh-Partap-Rotem (Crypto 2024) designed for tracing leaks in the context of secret sharing. However, there are multiple technical challenges in making this approach work, and we develop the necessary tools to overcome all these challenges. The end result is a threshold VRF with a provably secure tracing algorithm.

1 Introduction

A verifiable random function (VRF) [39] is a pseudorandom function whose evaluations are publicly verifiable: given a public verification key vk and an evaluation proof, anyone can verify that a VRF output was correctly computed using a secret evaluation key. VRF outputs are unique, in the sense that it is infeasible to produce valid proofs for two different VRF outputs for the same input. Threshold VRFs [23] are a natural threshold version of this notion. In a threshold VRF, the secret evaluation key is distributed among n parties, each holding a key share, such that any t of the n parties can jointly evaluate the function. Threshold VRFs have recently found important applications in blockchains, including for randomness beacons [27,15,18,22] and deterministic wallets [21,41]. Several constructions of threshold VRFs have been suggested over the years (see, for example, [37,23,24,27,22] and the references therein).

The need for accountability. A key security property missing from current threshold VRFs is accountability. Specifically, in settings where a coalition of parties collude and leak a function of their key shares, there should be a mechanism in place to identify these parties (and subsequently penalize them). Without accountability, parties have a risk-free incentive to leak such information, as they face no consequences for doing so. Introducing accountability would hopefully deter this behavior by ensuring that colluding parties can be caught, thus strengthening the overall security of the threshold VRF.

Over the years, the importance of such accountability mechanisms has been recognized by the cryptographic community in the context of other cryptographic primitives. This realization has lead to the development of traitor tracing schemes for public key encryption, a notion that has garnered substantial attention from the academic community (see, for example [11,28,49,48,30,31,17,13]). More recently, the ability to trace misbehaving parties who leak (a function) of their key shares was extended to the setting of threshold decryption [9], and similar notions were developed for secret sharing schemes [10,32]. Still, despite the fact the threshold VRFs have been around for over 20 years, and their increasingly prominent applications, the notion of accountability in threshold VRFs has so far not been explored.

1.1 This Work: Traceable Threshold VRFs

In this paper, we initiate the study of accountability in threshold VRFs. Our contributions are twofold:

- First, we formalize the requirement of accountability in threshold VRFs via the notion of **traceable threshold VRFs**. Our notion is inspired by former notions of accountability in threshold cryptography [10,32,9] and we will elaborate on it in a second.
- Second, we present a construction of a traceable threshold VRF in the random oracle model, that is derived from Paillier encryption [42]. This is the first threshold VRF to offer provable accountability guarantees.

Traceable threshold VRFs. Our definition for traceable threshold VRFs is inspired by previous works on tracing leaks in threshold cryptosystems [32,10,9]. In our model, $f < t$ corrupted parties collude, pull their key shares together, and sell an "evaluation box" E. This box is an algorithm that takes in an input x to the VRF, and $t - f$ additional evaluation shares. The box E then combines its knowledge of the key shares of the corrupted parties and the evaluation shares it received as input, to compute the VRF output $f_{\mathsf{ek}}(x)$ on x, where ek is the evaluation key sampled at setup. At a high level, we say that the box E is "good" if, given well-formed inputs, it outputs the correct function value with high probability. By now, modeling leaks as algorithms that break the security of the underlying primitive is a well accepted way to capture any "useful" leakage of the secret keys (for example, "decryption boxes" in traitor tracing schemes [19] and "reconstruction boxes" in traceable secret sharing schemes [32,10]).

We say that a threshold VRF is *traceable* if any such good box E can be traced back to the corrupted parties who contributed to it. This is formalized by requiring the existence of a tracing algorithm, Trace. This is an efficient algorithm that gets as input a tracing key tk and interacts with E via oracle access. At the end of this interaction Trace outputs the subset of corrupted parties. We ask that all corrupted parties are caught, and that no honest party is implicated. The formal definition is presented and discussed in Section 2.

A simple approach and its pitfalls. In Appendix B we present a simple generic tracing algorithm. At a high level, it works as follows: say that there are $f = t - 1$ corruptions. The tracing algorithm is equipped with n partial evaluation $w_1 = f_{\mathsf{ek}_1}(x), \dots, w_n = f_{\mathsf{ek}_n}(x)$ for some input x, one partial evaluation for each of the parties, where $\mathsf{ek}_1, \dots, \mathsf{ek}_n$ are the partial keys distributed to the parties. We also give it the true value $y = f_{\mathsf{ek}}(x)$ of the VRF at x. Then, the simple tracing algorithm feeds these partial evaluations to E one at a time: If E returns y on input w_i then party i must be innocent; otherwise, the box does not have enough information to predict y. In this way, we can exonerate all honest parties one at a time, and then we deem the remaining parties as the corrupted parties.

Though simple, the forgoing approach suffers from two major limitations. First, it requires that we reveal to the tracer the VRF value at the point x. Second, and more importantly, this tracing algorithm only works if E is *perfect.* By that, we mean that E always outputs the correct value when sufficiently-many additional evaluation shares are given as input. To see why, think of a box E that only outputs the correct evaluation if $H(f_{\mathsf{ek}_t}(x))$ starts with ℓ 0s, where $f_{\mathsf{ek}_t}(x)$ is the additional evaluation share that E gets as input, H is a hash function, and ℓ is some small integer. In this example, the simple tracing algorithm will end up accusing many honest parties! This is, even though the box outputs the correct VRF evaluation often enough (namely, with probability $2^{-\ell}$). We would like to correctly trace such a box back to the corrupted parties who created it. We next present our Paillier-based construction, that avoids these two shortcomings.

Our construction. We present our construction in three steps. We start with a base scheme, that already captures many of the main ideas behind our construction, but suffers from two unfavorable security restrictions, discussed below. Then, we show how to extend the base construction to lift these restrictions.

In Section 3 we present our base scheme. At its core, is a PRF based on Paillier encryption, previously used in [8]. This PRF uses the fact that decrypting a random Paillier ciphertext, derived from the VRF input, gives a pseudorandom function. Moreover, the homomorphic properties of Paillier encryption lend themselves to efficient proofs of correct decryption, which can be used to achieve public verifiability as needed for a VRF. First, we extend this observation to the threshold setting, by relying on threshold variants of Paillier encryption [26,34], modifying them to fit our needs. This gives us a threshold VRF.

Next, we augment this construction with a tracing algorithm to make it traceable. We would like to leverage the recent work of Boneh, Partap, and Rotem [10] (BPR) who showed how to trace leaks in Shamir secret sharing. Their tracing algorithm relies on detecting (carefully-planted) errors in the Lagrange interpolation process underlying the reconstruction in Shamir secret sharing. We observe that in threshold Paillier decryption, these errors are revealed in the group $\mathbb{Z}_N$ for a bi-prime modulus N.

Already in this base scheme, we need to overcome two main technical challenges. The first is that the BPR tracing algorithm for Shamir relies on list-decoding for Reed-Solomon codes. The group $\mathbb{Z}_N$ is not an integral domain, and list decoding algorithms for Reed-Solomon codes, that are typically defined over finite fields, do not naturally extend to it. To deal with this issue, we observe that full-fledged list-decoding is unnecessary for tracing. We then have to look under the hood of prominent list decoding algorithms, and use them in a way that respects the structure of $\mathbb{Z}_N$.

The second technical issue is that, in our definition, when the box E takes as input a VRF input x and partial evaluations $w_{f+1}, \ldots, w_t$, we allow E to require evaluation proofs that all of $w_{f+1}, \ldots, w_t$ were computed correctly for the input x. This makes the definition of traceability stronger and more realistic. However, the BPR algorithm requires feeding E with *wrong* partial evaluations, for which a valid evaluation proof cannot be constructed. To get around this problem, we encode some additional information in the tracing key. This information essentially acts as a trapdoor to the VRF verification key, and allows the tracer to forge partial evaluations proofs for false evaluations, overcoming this issue.

Lifting the two restrictions. The base scheme we just described only serves as a stepping stone. It suffers from two main restrictions, which we show how to lift. First, the added information in the tracing key has undesirable consequences; since it allows the tracer to forge evaluation proofs, it also gives it the ability to break the uniqueness property of the VRF. In Section 4 we show how to lift this restriction. By relying on techniques from Waters [47], we greatly limit the power of the trapdoor the tracer is endowed with. Specifically, we show how to modify the verification and tracing keys such that the tracer can only break uniqueness with respect to a very small (polynomial-size) subset of the VRF input domain, out of exponentially many possible inputs. Moreover, this bad set of inputs is uniformly random, so they will almost surely never come up in the above-mentioned applications of threshold VRFs.

Second, a fact we ignored thus far is that the above base scheme is only one-time traceable. That is, tracing is only guaranteed to work if the corrupted parties have observed no honest partial evaluations before constructing the box E. This is a highly unrealistic assumption. Hence, in Section 5, we show how to bootstrap our one-time traceable scheme into a full-fledged traceable threshold VRF. We present two methods. The first method takes inspiration from the recent work on bootstrapping one-time lattice-based multisignatures into many-time multisignatures [25]. The idea is to divide time into epochs, such that in each epoch, we use a different one-time traceable scheme. The verification key can be compressed by relying on succinct vector commitments [36,14]. This method assumes a synchronized notion of epochs among the parties, but this is often the case in the applications of threshold VRFs [27].

The second method relies on the properties of exponential decay. We set up T instances $f_1, \ldots, f_T$ of a one-time traceable VRF for some T that is super-logarithmic in the security parameter. Then, for each input x to the VRF, we use f_i with probability 2^{-i}. We decide which function to use deterministically using a hash function, so each input x is always mapped to the same function f_i. This approach only provides a fine-grained tracing guarantee: if the corrupted parties have observed q partial evaluations before leaking E, but E works for $\delta \gg 1 - 1/q$ fraction of the inputs, then there should be an index $i^* \in [T]$ such that: (1) E works for f_i, but (2) the corrupted parties have not observed partial evaluations with respect to f_{i^*}. Hence, tracing follows from the one-time traceability of f_{i^*}.

1.2 Discussion and Future Directions

Our work opens up several avenues for future work.

(1) Additional constructions. Perhaps the most natural open problem is to come up with additional constructions of traceable threshold VRFs. Specifically, it would be advantageous to have constructions based on assumptions that are believed to be post-quantum secure. A natural starting point in this context may be Regev's lattice-based public key encryption scheme [43]. Regev encryption exhibits some similar properties to Paillier, and so one could hope to port our techniques to Regev encryption. This would require, however, to overcome a myriad of technical difficulties that is unique to the lattice setting.

(2) Public-key tracing. For our constructions, the tracing key needs to be kept secret from the adversary, because knowledge of this key can allow it to evade tracing. (Note that other security properties including uniqueness and pseudorandomness are preserved even against adversaries with the tracing key). This is inherent to the BPR tracing algorithm which also relies on the tracing key being secret. An interesting future direction would be to build traceable threshold VRF constructions which support public-key tracing.

(3) The $f \geq t$ case. In this work, we focus solely on the case where the number f of corrupted parties is less than the threshold t. In the traceable secret sharing case [10] this is unavoidable, since any coalition of size $f \geq t$ can just reconstruct and leak the secret itself, which contains no

information about the leaking coalition. In the VRF setting too, a coalition of size $f \geq t$ can leak the VRF evaluation on a polynomial size set of *prespecified inputs* without being detected. However, what if the coalition leaks an evaluation box E that takes in a single input x in the domain of the VRF, and outputs its evaluation $y = f_{\mathsf{ek}}(x)$. If we assume a super-polynomial size domain, then tracing such a box to the corrupted coalition that constructed it is a very interesting direction that we leave for future work.

(4) An alternative view: VRFs as signatures. Another way to view our contributions is as the introduction of a new type of accountability for threshold signatures. Typically, when discussing accountable threshold signatures, the term refers to the ability of a signature to uniquely identify the quorum responsible for generating it (see [38,3,40,5,7,6,9] and references therein). It has long been noted that, under certain conditions, VRFs are equivalent to the concept of unique signatures [29,37]—signatures with the additional property that it is computationally hard to produce two distinct signatures for the same message. Building on this observation, our work can be understood more broadly as introducing a novel form of accountability for threshold signatures, specifically aimed at addressing the issue of leaking secret key shares. An interesting question in this regard, is whether lifting the uniqueness requirement can result in more efficient constructions of (non-unique) traceable threshold signatures.

2 Definitions of Traceable VRFs

In this section, we define the notion of a traceable threshold VRF.

A traceable threshold VRF is a tuple $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{EvalVerify}, \mathsf{Combine}, \mathsf{Verify}, \mathsf{Trace})$ of PPT algorithms. The algorithms $\mathsf{Eval}, \mathsf{Combine}, \mathsf{EvalVerify}, \mathsf{Verify}$ are the standard algorithms of threshold VRFs, for generating partial VRF evaluations, combining partial evaluations and verifying the evaluation proofs. We do restrict the presentation, however, to threshold VRF schemes where the key generation procedure works in a certain manner. This restriction is in line with the recent work of Boneh, Partap, and Rotem [10] on traceability in secret sharing schemes. Specifically, we have a $\mathsf{FuncSamp}$ algorithm that samples a function indexed by an evaluation key. This key is then secret shared among the parties via the $\mathsf{KeyShareGen}$ algorithm which produces one secret share at a time. To produce the shares of all n parties, this algorithm is invoked n times. But, many secret sharing schemes require that the n shares are correlated beyond just the underlying secret key, to ensure correctness. For example, in Shamir secret sharing, all shares must lie on the same degree $t-1$ polynomial whose free coefficient is the evaluation key. To account for this added correlation, we consider key share generation algorithms that take in a **correlation string** ρ as an additional input. In Shamir secret sharing, for example, this ρ specifies the other $t-1$ coefficients of the said polynomial. Correctness then needs to hold for every choice of ρ, and security is defined over a random choice of ρ.

We now formally define the syntax of traceable threshold VRF schemes:

– $\mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta}) \to (\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho)$ is a randomized algorithm that samples a function $f_{\mathsf{ek}} : \mathcal{Z}_\lambda \to \mathcal{Y}_{\mathsf{ek}}$ from the function space $\mathcal{F} = \{\{f_{\mathsf{ek}}\}_{\mathsf{ek} \in \mathcal{EK}_\lambda}\}_\lambda$ indexed by a key ek, where $\mathcal{EK} = \{\mathcal{EK}_\lambda\}_{\lambda \in \mathbb{N}}$ is the space of evaluation keys. It takes in the security parameter λ, the number n of potential evaluators, a threshold t and error parameters $\epsilon = \epsilon(\lambda)$ and $\delta = \delta(\lambda)$. The algorithm outputs an evaluation key ek that defines the function f_{ek}, the corresponding function verification key vk, a combiner key ck, a tracing key component tk^* and a correlation string

$\rho \in \{0,1\}^\kappa$ where $\kappa = \kappa(\lambda, n, t) \in \mathbb{N}$. We assume that this algorithm samples the evaluation key ek and the correlation string ρ uniformly randomly from $\mathcal{EK}_\lambda$ and $\{0,1\}^\kappa$ respectively.

- $\mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}, n, t, \rho) \to (\mathsf{ek}_i, \mathsf{tk}_i)$ is a randomized algorithm that outputs a single share of the evaluation key ek_i and a tracing key component tk_i. The overall tracing key is $\mathsf{tk} \leftarrow (\mathsf{tk}^*, \{\mathsf{tk}_i\}_{i\in[n]})$.
- $\mathsf{Eval}(\mathsf{ek}_i, \mathsf{vk}, z) \to (w_i, \pi_i)$ is the deterministic function evaluation algorithm. It takes as input an evaluation key share ek_i, the verification key vk and an input z from the function's input space $\mathcal{Z}$ and outputs an evaluation share w_i and a proof share π_i.
- $\mathsf{EvalVerify}(\mathsf{vk}, z, w_i, \pi_i) \to 0/1$ is the deterministic share verification algorithm, that takes in the verification key vk, an input z, a partial evaluation w_i, and a proof π_i. It outputs either 1, implying acceptance, or 0, implying rejection.
- $\mathsf{Combine}(\mathsf{ck}, w_{i_1}, \dots, w_{i_t}, \pi_{i_1}, \dots, \pi_{i_t}) \to (w, \pi)$ is the deterministic combiner algorithm, that takes in t evaluation shares $w_{i_1}, \dots, w_{i_t}$ and their respective proof shares $\pi_{i_1}, \dots, \pi_{i_t}$, and outputs a single combined evaluation $w \in \mathcal{Y}_{\mathsf{ek}}$ and an associated proof π.
- $\mathsf{Verify}(\mathsf{vk}, z, w, \pi) \to 0/1$ is the deterministic verification algorithm, that takes in the verification key vk, an input z, a VRF evaluation w, and a proof π. It outputs either 1, implying acceptance, or 0, implying rejection.
- $\mathsf{Trace}^E(\mathsf{tk}) \to \mathcal{I}$ is the randomized tracing algorithm. It takes as input a tracing key tk and it gets oracle (black-box) access to an evaluation box E. The algorithm outputs a subset $\mathcal{I} \subseteq [n]$ of identities of leaking parties. The exact role of E in our definition will become apparent in a minute.

A traceable threshold VRF should satisfy the standard notions of correctness, uniqueness, and pseudorandomness. We also define three new security notions, called tracer pseudorandomness, tracer uniqueness, and traceability.

Correctness. The correctness requirement for a traceable threshold VRF is the standard correctness property of threshold VRFs. That is, any t honestly-computed evaluation shares should correctly combine to the function's output.

Definition 1 (Correctness). *Let* $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{Combine}, \mathsf{EvalVerify}, \mathsf{Verify}, \mathsf{Trace})$ *be a traceable threshold VRF and let* $\eta = \eta(\lambda)$ *be a function of the security parameter. We say that* TTVRF *is* η***-correct*** *if for every* $\lambda \in \mathbb{N}$*, every input* z *in the input space of the function, every* $n \in \mathbb{N}$*, every* $\epsilon, \delta \in (0,1]$*, every* $0 < t \le n$*, and every subset* $\mathcal{J} = \{i_1, \dots, i_t\} \subseteq [n]$ *of size* t*, it holds that*

$$\Pr\left[\begin{matrix}\mathsf{Verify}(\mathsf{vk}, z, w, \pi) = 1 \ \wedge \\ \forall j \in [t],\ \mathsf{EvalVerify}(\mathsf{vk}, z, w_{i_j}, \pi_{i_j}) = 1\end{matrix}\right] \ge 1 - 1/\eta(\lambda),$$

where $(\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho) \leftarrow\!\$\ \mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta})$*,* $(\mathsf{ek}_{i_j}, \mathsf{tk}_{i_j}) \leftarrow\!\$\ \mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}, n, t, \rho)$ *and* $(w_{i_j}, \pi_{i_j}) \leftarrow\!\$\ \mathsf{Eval}(\mathsf{ek}_{i_j}, \mathsf{vk}, z)$ *for* $j = 1, \dots, t$*, and* $(w, \pi) \leftarrow \mathsf{Combine}(\mathsf{ck}, w_{i_1}, \dots, w_{i_t}, \pi_{i_1}, \dots, \pi_{i_t})$.

Uniqueness. A traceable threshold VRF is said to satisfy uniqueness, if for every input z in the input space, the verification algorithm accepts exactly one output. We define a relaxed notion called computational uniqueness, according to which different outputs that are accepted by Verify may exist, but they should be computationally hard to find (alongside corresponding accepting proofs).

Definition 2 (Uniqueness). *Let* $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{Combine}, \mathsf{EvalVerify}, \mathsf{Verify}, \mathsf{Trace})$ *be a traceable threshold VRF. We say that* TTVRF *has* ***computational uniqueness***

if for every PPT *algorithm* $\mathcal{A}$*, every* $\lambda \in \mathbb{N}$*, every* $n \in \mathbb{N}$*, every* $0 < t \leq n$*,every* $\epsilon, \delta \in (0, 1]$*, the function* $\mathsf{Adv}^{\mathrm{uniq}}_{\mathcal{A},\mathsf{TTVRF}}(\lambda)$ *defined below is negligible in* λ*:*

$$\Pr\left[\begin{array}{c} w_0 \neq w_1 \ \wedge \\ \mathsf{Verify}(\mathsf{vk}, z, w_0, \pi_0) = 1 \ \wedge \\ \mathsf{Verify}(\mathsf{vk}, z, w_1, \pi_1) = 1 \end{array}\middle|\begin{array}{c} (\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho) \leftarrow\!\!\$\ \mathsf{FuncSamp}\begin{pmatrix} 1^\lambda, n, t, \\ 1^{1/\epsilon}, 1^{1/\delta} \end{pmatrix} \\ \begin{pmatrix} z, (w_0, \pi_0), \\ (w_1, \pi_1) \end{pmatrix} \leftarrow\!\!\$\ \mathcal{A}\begin{pmatrix} 1^\lambda, \mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \\ \rho, n, t, 1^{1/\epsilon}, 1^{1/\delta} \end{pmatrix} \end{array}\right].$$

Tracer Uniqueness. For traceable threshold VRFs, we define another notion of uniqueness called tracer uniqueness, which means that uniqueness is maintained for a large fraction of the input space, even against a corrupt tracer. In other words, the tracing key tk only allows the tracer to break uniqueness for a negligible fraction of inputs. We formalize this by requiring that for any input z^* chosen by the adversary, the tracing key allows the adversary to break uniqueness for this input only with negligible probability. We allow the adversary to choose z^* adaptively – it is given the evaluation key, combiner key, verification key and the n key shares before outputting z^*.

Definition 3. *Let* TTVRF = (FuncSamp, KeyShareGen, Eval, Combine, EvalVerify, Verify, Trace) *be a traceable threshold VRF. We say that* TTVRF *satisfies* ***tracer uniqueness*** *if for every* PPT *algorithm* $\mathcal{A}$*, every* $\lambda \in \mathbb{N}$*, every* $n \in \mathbb{N}$*, every* $\epsilon, \delta \in (0, 1]$*, every* $0 < t \leq n$*, the function* $\mathsf{Adv}^{\mathrm{t-uniq}}_{\mathcal{A},\mathsf{TTVRF}}(\lambda)$ *defined below is negligible in* λ*:*

$$\Pr\left[\begin{array}{c} w_0 \neq w_1 \ \wedge \\ \mathsf{Verify}(\mathsf{vk}, z^*, w_0, \pi_0) = 1 \ \wedge \\ \mathsf{Verify}(\mathsf{vk}, z^*, w_1, \pi_1) = 1 \end{array}\middle|\begin{array}{c} (\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho) \leftarrow\!\!\$\ \mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta}) \\ \forall i \in [n] : (\mathsf{ek}_i, \mathsf{tk}_i) \leftarrow\!\!\$\ \mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}, n, t, \rho) \\ \mathsf{tk} \leftarrow (\mathsf{tk}^*, \mathsf{tk}_1, \ldots, \mathsf{tk}_n) \\ (z^*, \mathsf{state}) \leftarrow\!\!\$\ \mathcal{A}\begin{pmatrix} 1^\lambda, \mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \\ \rho, n, t, 1^{1/\epsilon}, 1^{1/\delta} \end{pmatrix} \\ ((w_0, \pi_0), (w_1, \pi_1)) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{state}, \mathsf{tk}) \end{array}\right]$$

On the tracer uniqueness definition. An avid reader might wonder why the adversary is forced to select the input z^* before seeing the tracing key, in the tracer uniqueness game. This is because of an inherent barrier to defining a stronger notion. Specifically, the tracer needs to give partial evaluations as input to the evaluation box E. But these evaluations need to appear valid (they need to pass verification), as otherwise the box might not work. As a result, the tracing key must allow the tracer to break either pseudorandomness or uniqueness on certain inputs – when combined with the leaked keys in the box, the input partial evaluations would either (a) reconstruct the correct output, which hurts pseudorandomness, or (b) produce a different, incorrect output, which hurts uniqueness. Nonetheless, our definition ensures that the set of inputs for which the tracer can break uniqueness must be distributed uniformly randomly, and constitute only a negligible fraction of the input space.

Pseudorandomness. A traceable threshold VRF should be pseudorandom in the following sense: even an adversary that holds $t - 1$ evaluation keys should not be able to distinguish the function's output on a new input z^* from a uniformly random value. This condition should hold, even if the adversary can observe the partial evaluations of the function on inputs of its choice, as long as it does not observe t secret keys (and can hence trivially predict the value of the function at z^*). Figure 1 presents a formal definition.

Definition 4 (Pseudorandomness). *A traceable threshold VRF scheme* TTVRF = (FuncSamp, KeyShareGen, Eval, Combine, EvalVerify, Verify, Trace) *is said to be pseudorandom if, for every* PPT *adversary* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\mathrm{rand}}_{\mathcal{A},\mathsf{TTVRF}}(\lambda) = \Pr\left[\mathbf{G}^{\mathrm{rand}}_{\mathcal{A},\mathsf{TTVRF}}(\lambda) = 1\right]$$

Tracer Pseudorandomness. For traceable threshold VRFs, it makes sense to require tracer pseudorandomness, which means that pseudorandomness is maintained even against the tracer. We note that due to the inherent barrier listed above, the tracing key $\mathtt{tk}$ can allow the adversary to break uniqueness for a certain set of inputs. Hence, we define tracer pseudorandomness as follows: the definition is parameterized by some function $\mathcal{S}$ that maps tracing keys to subsets of the inputs. Then, we allow the adversary to break pseudoranomness on inputs in a set $\mathcal{S}(\mathsf{tk})$ that is deterministically induced by the tracing key tk, but it should not break pseudoranomness on inputs outside this set. Figure 1 presents a formal definition.

Definition 5 (Tracer Pseudorandomness). *A traceable threshold VRF scheme* $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{Combine}, \mathsf{EvalVerify}, \mathsf{Verify}, \mathsf{Trace})$ *is said to be tracer-pseudorandom if, for every* PPT *adversary* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\mathrm{t\text{-}rand}}_{\mathcal{A},\mathsf{TTVRF}}(\lambda) = \Pr\left[\mathbf{G}^{\mathrm{t\text{-}rand}}_{\mathcal{A},\mathsf{TTVRF}}(\lambda) = 1\right]$$

```
Games $\mathbf{G}^{\mathrm{rand}}_{\mathsf{TTVRF}}$ and $\mathbf{G}^{\mathrm{t\text{-}rand}}_{\mathsf{TTVRF}}$
1 :  $(\mathsf{st}, n, t, 1^{1/\epsilon}, 1^{1/\delta}) \leftarrow \mathcal{A}(1^\lambda)$
2 :  $(\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho) \leftarrow\$ \mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta})$
3 :  for $i = 1, \dots, n : (\mathsf{ek}_i, \mathsf{tk}_i) \leftarrow\$ \mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}, n, t, \rho)$
4 :  $\mathsf{tk} \leftarrow (\mathsf{tk}^*, \mathsf{tk}_1, \dots, \mathsf{tk}_n)$
5 :  $(\mathsf{st}, z^*) \leftarrow\$ \mathcal{A}^{\mathsf{ekO}(\cdot), \mathsf{EvalO}(\cdot,\cdot,\cdot)}(\mathsf{st}, \mathsf{vk}, \mathsf{ck}, \boxed{\mathsf{tk}}\ )$
6 :  $b \leftarrow\$ \{0,1\}, m^* \leftarrow\$ \mathcal{Y}_{\mathsf{ek}}$
7 :  if $b = 0$ then $m^* \leftarrow f_{\mathsf{ek}}(z^*)$
8 :  $b' \leftarrow\$ \mathcal{A}(\mathsf{st}, m^*)$
9 :  $\mathsf{flag}_{\mathrm{rand}} \leftarrow (b' = b \wedge |\mathcal{Q}^{\mathsf{ek}}| < t \wedge |\mathcal{Q}^{\mathsf{eval}}(z^*)| = 0)$
10 : return $\mathsf{flag}_{\mathrm{rand}}\boxed{\wedge\ z^* \notin \mathcal{S}(\mathsf{tk})}$

Oracle $\mathsf{ekO}(i)$
1 :  $\mathcal{Q}^{\mathsf{ek}} \leftarrow \mathcal{Q}^{\mathsf{ek}} \cup \{i\}$
2 :  return $\mathsf{ek}_i$

Oracle $\mathsf{EvalO}(z, i)$
1 :  $(w_i, \pi_i) \leftarrow \mathsf{Eval}(\mathsf{ek}_i, \mathsf{vk}, z)$
2 :  $\mathcal{Q}^{\mathsf{eval}}(z) \leftarrow \mathcal{Q}^{\mathsf{eval}}(z) \cup \{i\}$
3 :  return $(w_i, \pi_i)$
```

Fig. 1. The security games $\mathbf{G}^{\mathrm{rand}}_{\mathsf{TTVRF}}$ and $\mathbf{G}^{\mathrm{t\text{-}rand}}_{\mathsf{TTVRF}}$ for a traceable threshold VRF scheme $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Combine}, \mathsf{EvalVerify}, \mathsf{Verify}, \mathsf{Trace})$. The differences between the two games are highlighted in red – in the game $\mathbf{G}^{\mathrm{t\text{-}rand}}_{\mathsf{TTVRF}}$, (i) the adversary is given the tracing key as an additional input on line 5. Additionally, (ii) in line 10, the adversary can win only if it breaks pseudorandomness for an input outside of $\mathcal{S}(\mathsf{tk})$, the set of inputs for which the tracing key allows breaking uniqueness.

Semi-adaptive adversaries. The security games as defined in Figure 1 allow for fully-adaptive adversaries, in the sense that they do not pose any restrictions on the order in which the adversary decides on its oracle queries. Proving security against such adversaries is known to be a challenging task, already for non-traceable threshold VRFs and signature schemes [35]. Since the problem of fully-adaptive adversaries is not at the focus of this work, we consider semi-adaptive adversaries, which are restricted to issuing all secret-key queries before issuing their partial evaluation queries. This is captured by modifying the security games as follows. The game maintains a bit b_e denoting whether a partial evaluation query has been issued by the adversary. On input (i), the ekO oracle will check if $b_e = 1$; if so, it will ignore this query, returning $\perp$. Otherwise, it will continue as defined in Figure 1.
On uf-0 vs uf-1. The pseudorandomness games defined above consider uf-0 security [2] (originally defined as low-threshold in [45]), where the adversary is not allowed to query partial evaluations on the challenge input z^*. This definition can be naturally extended to define a stronger notion called uf-1 security (referred to as high-threshold schemes in [45]), where such queries are permitted. However, since achieving uf-1 security is not the primary focus of this work, we choose to consider the uf-0 definition. Nonetheless, in Section 3, we discuss how our constructions can be proven to be uf-1 secure, by adapting techniques from [4].

Traceability. In addition, a traceable threshold VRF should provide *traceability*. Suppose a coalition $\mathcal{I} \subseteq [n]$ of parties, of size $f < t$, gets together and constructs an evaluation box E using their shares. This E is an algorithm that takes in an input z along with $t - f$ evaluation shares and outputs the combined VRF evaluation at z. Intuitively, if this E is a "good" evaluation box, then it should be possible to trace it back to the parties who "contributed" their shares to it. "Good" here means that, given the necessary additional information, the box outputs the correct value of the function with high probability for many of the inputs. This is formally defined below. As we discussed in the introduction, tracing E back to the corrupted parties should be done given only black-box access to it.

More formally, say that the evaluation key shared among the parties is ek^*. Then, E is a good evaluation box if there is a large subset of inputs such that, for any z in that subset, and for random $t-f$ evaluation shares for z computed using secret shares of ek^*, denoted $(w'_1, \pi'_1), \ldots, (w'_{t-f}, \pi'_{t-f})$, it holds that $E(z, (w'_1, \pi'_1), \ldots, (w'_{t-f}, \pi'_{t-f}))$ outputs the correct VRF evaluation at z corresponding to ek^* with high probability. Definition 6 below formally defines good evaluation boxes.

Definition 6 (Good evaluation boxes). *Let* $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{Combine}, \mathsf{EvalVerify}, \mathsf{Verify}, \mathsf{Trace})$ *be a traceable threshold VRF scheme. Let* $\lambda \in \mathbb{N}$*, let* $n, t, f \in \mathbb{N}$ *such that* $0 < f < t \leq n$*, and let* $\kappa = \kappa(\lambda, n, t)$*. For* $\epsilon, \delta \in (0, 1]$*, an evaluation key* ek^**, a correlation string* $\rho \in \{0,1\}^\kappa$*, and a corresponding verification key* vk*, we say that an evaluation box* E *is* $(n, t, f, \mathsf{ek}^*, \rho, \mathsf{vk}, \epsilon, \delta)$*-good if there exists a subset* $\mathcal{Z}_E \subseteq \mathcal{Z}$ *of size at least* $\delta \cdot |\mathcal{Z}|$*, such that for all* $z \in \mathcal{Z}_E$*,*

$$\Pr\left[E(z, (w'_1, \pi'_1), \ldots, (w'_{t-f}, \pi'_{t-f})) = f_{\mathsf{ek}^*}(z)\right] \geq \epsilon,$$

where the probability is taken over $(\mathsf{ek}'_i, \mathsf{tk}'_i) \leftarrow\$ \mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}^*, n, t, \rho)$ *for* $i = 1, \ldots t - f$ *and the random coins of* E*, and* $(w'_i, \pi'_i) \leftarrow \mathsf{Eval}(\mathsf{ek}'_i, \mathsf{vk}, z)$ *for all* $i \in 1, \ldots, t - f$*.*

Note that an alternate definition might consider a single parameter (equal to $\epsilon\delta$) that characterizes the probability that the box outputs the correct evaluation. But we define ϵ and δ separately due to technical reasons which will become clearer in Section 5. Equipped with the above definition, the traceability experiment is presented in Figure 2. We define two notions of traceability called many-time and one-time, which differ in whether or not the adversary is allowed to see any partial evaluations of the VRF. Looking ahead, we will start by presenting a scheme that provides traceability only if the adversary does not observe partial evaluations of honest parties on any input before constructing the box E (and is thus one-time traceable), and then we will show how to bootstrap such a scheme into a scheme that does not have this restriction (and is thus many-time traceable).

```
Games $\mathbf{G}^{\text{trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda)$ and $\mathbf{G}^{\text{trace-1}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda)$
1 : $(n, t, \mathcal{I}, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$   // $\mathcal{A}$ chooses the parties $\mathcal{I} = \{i_1, \ldots, i_f\} \subseteq [n]$ to corrupt
2 : $(\mathsf{ek}^*, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho) \leftarrow\!\!\$\ \mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta})$
3 : for $i = 1, \ldots, n$ : $(\mathsf{ek}_i, \mathsf{tk}_i) \leftarrow\!\!\$\ \mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}^*, n, t, \rho)$
4 : $\mathsf{tk} := (\mathsf{tk}^*, \mathsf{tk}_1, \ldots, \mathsf{tk}_n)$
5 : $E \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{EvalO}(\cdot,\cdot)}(\mathsf{state}, \mathsf{ck}, \mathsf{vk}, \mathsf{ek}_{i_1}, \ldots, \mathsf{ek}_{i_f})$ where $\mathcal{I} = \{i_1, \ldots, i_f\}$
6 : if $E$ is not $(n, t, f, \mathsf{ek}^*, \rho, \mathsf{vk}, \epsilon, \delta)$-good then return 0
7 : $(\mathcal{I}') \leftarrow\!\!\$\ \mathsf{Trace}^{E(\cdot)}(\mathsf{tk})$
8 : if $\mathcal{I} = \mathcal{I}'$ then return 0, else return 1
```

Fig. 2. The one-time and many-time tracing experiments for a traceable threshold VRF scheme TTVRF and an adversary $\mathcal{A}$. The difference between the two experiments is highlighted in red – the adversary gets access to the partial evaluation oracle EvalO in the many-time tracing experiment. (The oracle is as defined in Figure 1)

Definition 7 (Traceability). *Let* $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{Combine}, \mathsf{EvalVerify}, \mathsf{Verify}, \mathsf{Trace})$ *be a traceable threshold VRF scheme. Let* $\epsilon = \epsilon(\lambda)$ *and* $\delta = \delta(\lambda)$ *be functions of the security parameter. We say that* TTVRF *is one-time (and many-time) traceable if for every probabilistic polynomial time adversary* $\mathcal{A}$*, the following function is negligible in* λ *for* $b = 0$ *(and* 1*, respectively):*

$$\mathsf{Adv}^{\text{trace-b}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = \Pr\left[\mathbf{G}^{\text{trace-b}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1\right]$$

We make the following observations about the definition:

– **Adaptive adversaries:** For simplicity, our definition requires that the adversary chooses the set of corrupted parties all at once, non-adaptively. However, observe that in symmetric traceable threshold VRF schemes, as we consider here, all shares come from the same distribution, and hence querying for shares adaptively would give no additional power to the adversary.
– **The output of E:** Our definition requires that a good evaluation box E outputs the full VRF evaluation $f_{\mathsf{ek}}(z)$, and not some function thereof. For example, if E outputs only a single bit of $f_{\mathsf{ek}}(z)$, our definition makes no guarantees as to the ability to trace E back to the corrupted subset. Looking ahead, our tracing algorithms will indeed rely on E outputting the full evaluation $f_{\mathsf{ek}}(z)$ and not, say, its first bit. This is justified in real-world applications of threshold VRFs, such as in blockchain consensus, where the entire VRF output is used to

determine the behavior of the next few blocks. In such cases, the adversary would need the entire output of the VRF to benefit from its early knowledge of this value. Nevertheless, tracing boxes that only output some bits (or a function) of the VRF evaluation is an important future direction.

– **Private tracing:** In the tracing experiment, the adversary is not given access to the tracing key tk. This is because, for our constructions, knowledge of this key can allow an adversary to evade tracing. We leave the problem of constructing traceable threshold VRF schemes with public tracing as an avenue for future work.

Extending to the random oracle (RO) model. All of the syntactic and security definitions in this section readily extend to the random oracle model by granting all algorithms, including the adversary $\mathcal{A}$, oracle access to a function H chosen uniformly at random from a family $\mathcal{H}$ of functions. In the correctness and security definitions (Definitions 1 to 7), all probabilities are then also taken over the choice of H.

2.1 A Useful Fact About Good Evaluation Boxes

In this section, we introduce a slightly different notion of traceability, which we call *universal traceability.* This notion is easier to work with when proving traceability of traceable threshold VRF schemes, and — as we will shortly see — any scheme that is universally traceable is also traceable in some formal sense. As we will argue, universal traceability also makes sense as a notion of traceability in its own right.

Looking ahead, universal traceability says that a *universally good* evaluation box can be traced back to the subset of parties who manufactured it. Informally, an evaluation box is universally good if it correctly evaluates a VRF function corresponding to a random evaluation key $\hat{\mathsf{ek}}$ with high probability, when given partial evaluations derived from a random sharing of $\hat{\mathsf{ek}}$ as input. Observe that these partial VRF evaluations might not be correct with respect to the original verification key vk for the VRF (which corresponds to the VRF key ek). To be able to successfully run the evaluation box with these evaluations in-spite of this issue, we define universally good boxes only with respect to VRFs that define a $\mathsf{Sim}(\mathsf{vk}, z, w)$ algorithm, that can construct valid proofs for an evaluation share w generated using a random share of a random key $\hat{\mathsf{ek}}$. We formalize this requirement below.

For a subset $\mathcal{EK}^* \subseteq \mathcal{EK}_\lambda$ of evaluation keys, the verification key vk and an input z, we use $(w_i', \pi_i') \leftarrow\!\!\$\ \mathcal{EV}_{\lambda,n,t}(\mathcal{EK}^*, \mathsf{vk}, z)$ to denote the process of sampling a random secret share for a random evaluation key in this subset, and then using it to generate a partial evaluation share, and a corresponding accepting proof using the Sim algorithm. More formally, the process: $\rho \leftarrow\!\!\$\ \{0,1\}^\kappa$, $\hat{\mathsf{ek}} \leftarrow\!\!\$\ \mathcal{EK}^*$, $(\mathsf{ek}_i, \cdot) \leftarrow\!\!\$\ \mathsf{KeyShareGen}(1^\lambda, \hat{\mathsf{ek}}, n, t, \rho)$, $(w_i, \cdot) \leftarrow \mathsf{Eval}(\mathsf{ek}_i, \mathsf{vk}, z)$ and $\pi_i \leftarrow\!\!\$\ \mathsf{Sim}(\mathsf{vk}, z, w_i)$. Let $\Gamma_{ek} = \{\mathcal{EK}_1, \mathcal{EK}_2, \ldots\}$ be a partition of the space $\mathcal{EK}_\lambda$. For a key $\mathsf{ek} \in \mathcal{EK}_\lambda$, we denote by $\Gamma_{ek}(\mathsf{ek})$ the unique subset that contains ek. To be able to define universally good boxes, we require that the distribution of the output of $(w_i', \pi_i') \leftarrow\!\!\$\ \mathcal{EV}_{\lambda,n,t}(\mathcal{EK}^*, \mathsf{vk}, z)$ is indistinguishable from that of an honestly generated evaluation share. We formalize this requirement in the definition below.

Definition 8 (Simulatable Threshold VRF). *We say that a threshold VRF scheme with a* Sim *algorithm (as defined above) satisfies simulatability with respect to a partition* Γ_{ek} *of the evaluation key space, if for every* $\lambda \in \mathbb{N}$*, every* $0 < t < n \in \mathbb{N}$*, every* $\mathsf{ek} \in \mathcal{EK}_\lambda$*, every* $\rho \in \{0,1\}^\kappa$ *where* $\kappa = \kappa(\lambda, n, t)$ *and every* vk *that corresponds to* ek *and* ρ*, and for every* $z \in \mathcal{Z}_\lambda$ *we have that*

$$(w, \pi) \approx^c (w', \pi')$$

where $\mathsf{ek}_i \leftarrow\!\!\$\ \mathsf{KeyShareGen}(\mathsf{ek}, n, t, \rho)$ *and* $(w, \pi) \leftarrow\!\!\$\ \mathsf{Eval}(\mathsf{ek}_i, z)$*, and* $(w', \pi') \leftarrow\!\!\$\ \mathcal{EV}_{\lambda,n,t}(\Gamma_{ek}(\mathsf{ek}), \mathsf{vk}, z)$*.*

We formally define the notion of universally-good boxes in Definition 9 below.

Definition 9 (Universally-good evaluation boxes). *Let* TTVRF *be a simulatable and traceable threshold VRF scheme, as defined above. Let* $\lambda \in \mathbb{N}$*, let* $n, t, f \in \mathbb{N}$ *such that* $0 < f < t \le n$*, and let* $\kappa = \kappa(\lambda, n, t)$*. For* $\epsilon, \delta \in (0, 1]$*,* f *key shares* $\mathbf{ek} = (\mathsf{ek}_1, \ldots, \mathsf{ek}_f)$*, a subset* $\mathcal{EK}^* \subseteq \mathcal{EK}_\lambda$*, we say that an evaluation box* E *is* $(n, t, \mathbf{ek}, \mathsf{ck}, \mathsf{vk}, \mathcal{EK}^*, \epsilon, \delta)$*-good if there exists a subset* $\mathcal{Z}_E \subseteq \mathcal{Z}$ *of size at least* $\delta \cdot |\mathcal{Z}|$ *such that for all* $z \in \mathcal{Z}_E$*, the following probability is at least* ϵ*:*

$$\Pr\left[\mathsf{Combine}\left(\begin{array}{c}\mathsf{ck}, sh_1, \ldots, sh_f,\\ (w'_1, \pi'_1), \ldots, (w'_{t-f}, \pi'_{t-f})\end{array}\right) = (w, \cdot) \middle| E\left(\begin{array}{c}z, (w'_1, \pi'_1), \ldots,\\ (w'_{t-f}, \pi'_{t-f})\end{array}\right) = w\right]$$

where $(\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}, \rho) \leftarrow\!\!\$\ \mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta})$*,* $\mathcal{EK}^* \leftarrow \Gamma_{ek}(\mathsf{ek})$ *,* $(\mathsf{ek}_i, \cdot) \leftarrow\!\!\$\ \mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}, n, t, \rho)$*,* $sh_i \leftarrow \mathsf{Eval}(\mathsf{ek}_i, \mathsf{vk}, z)$ *for all* $i \in [f]$*, and the probability is taken over* $(w'_i, \pi'_i) \leftarrow\!\!\$\ \mathcal{EV}_{\lambda,n,t}(\mathcal{EK}^*, \mathsf{vk}, z)$ *for* $i = 1, \ldots, t - f$ *and the random coins of* E*.*

We now use the notion of universally good evaluation boxes to define universal traceability. Figure 3 presents the formal universal traceability experiment. It is almost identical to the tracing security experiment from Fig. 2 other than the fact that the evaluation box E is required to be universally good with respect to $\Gamma_{ek}(\mathsf{ek}^*)$, where ek^* is the key used to generate the secret shares in the experiment (rather than just good with respect to the key ek^*).

```
Games $\mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda)$ and $\mathbf{G}^{\text{univ-trace-1}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda)$
1 : $(n, t, \mathcal{I}, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$   // $\mathcal{A}$ chooses the parties $\mathcal{I} = \{i_1, \ldots, i_f\} \subseteq [n]$ to corrupt
2 : $(\mathsf{ek}^*, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho) \leftarrow\!\!\$\ \mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta})$
3 : for $i = 1, \ldots, n$ : $(\mathsf{ek}_i, \mathsf{tk}_i) \leftarrow\!\!\$\ \mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}^*, n, t, \rho)$
4 : $\mathsf{tk} := (\mathsf{tk}^*, \mathsf{tk}_1, \ldots, \mathsf{tk}_n)$
5 : $E \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{EvalO}(\cdot,\cdot)}(\mathsf{state}, \mathsf{ck}, \mathsf{vk}, \mathsf{ek}_{i_1}, \ldots, \mathsf{ek}_{i_f})$ where $\mathcal{I} = \{i_1, \ldots, i_f\}$
6 : $\mathbf{ek} := (\mathsf{ek}_{i_1}, \ldots, \mathsf{ek}_{i_f})$
7 : if $E$ is not $(n, t, \mathbf{ek}, \mathsf{ck}, \mathsf{vk}, \Gamma_{ek}(\mathsf{ek}^*), \epsilon, \delta)$-universally-good then return 0
8 : $(\mathcal{I}') \leftarrow\!\!\$\ \mathsf{Trace}^{E(\cdot)}(\mathsf{tk})$
9 : if $\mathcal{I} = \mathcal{I}'$ then return 0, else return 1
```

Fig. 3. The one-time and many-time universal tracing experiments for a traceable threshold VRF scheme TTVRF and an adversary $\mathcal{A}$. The difference from the experiments in Figure 2 is marked in blue. (The EvalO oracle is as defined in Figure 1, and the adversary gets access to this oracle only in the many-time experiment)

Definition 10 (Universal Traceability). *Let* $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{EvalVerify}, \mathsf{Combine}, \mathsf{Verify}, \mathsf{Trace})$ *be a traceable threshold VRF scheme. Let* $\epsilon = \epsilon(\lambda)$ *and* $\delta = \delta(\lambda)$ *be functions of the security parameter. We say that* TTVRF *is one-time (and many-time) universally traceable if for every probabilistic polynomial time adversary* $\mathcal{A}$*, the following function is negligible in* λ *for* $b = 0$ *(and* 1*, respectively):*

$$\mathsf{Adv}^{\text{univ-trace-b}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = \Pr\left[\mathbf{G}^{\text{univ-trace-b}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1\right]$$

We argue that Definition 10 is very reasonable. Informally, the definition says that if E is universally good with respect to the equivalence class $\Gamma_{ek}(\mathsf{ek}^*)$ of the *real evaluation key* ek^* (i.e., the one used to sample the real secret shares sampled by the challenger), then it should be traced back to the corrupted subset. Intuitively, since $\mathcal{A}$ only sees at most $t-1$ key shares, and does not have any information about the real correlation string ρ or the real evaluation key ek^*, it is very reasonable to consider an evaluation box E that almost always fails on $\Gamma_{ek}(\mathsf{ek}^*)$ to be a bad reconstruction box. To make this intuition precise, we prove below that for a certain natural class of traceable secret sharing schemes (as the ones we will construct), if a traceable threshold VRF scheme satisfies universal traceability (Definition 10), then it also satisfies standard traceability (Definition 7) with related parameters.

Bidirectional Threshold VRF schemes. To relate the notion of universal traceability to that of standard traceability, we define a subclass of threshold VRF schemes, which we call *bidirectional*. This is a generalization of the notion of simulatability defined in Definition 8 above.

Definition 11. *Let* TTVRF *be a traceable threshold VRF scheme. Let* $\gamma = \gamma(\lambda)$, $n = n(\lambda)$ *and* $t = t(\lambda)$ *be functions of the security parameter* $\lambda \in \mathbb{N}$. *Let* Γ_{ek} *be a partition of the space of evaluation keys* $\mathcal{EK}_\lambda$. *We say that* TTVRF *is* (Γ_{ek}, γ)*-bidirectional if, for every* $\lambda \in \mathbb{N}$, *every* $\epsilon, \delta \in (0,1]$, *every* $\mathcal{EK}^* \in \Gamma_{ek}$,

$$\mathsf{SD}\left(\left(\begin{array}{c} z, \mathsf{ek}_1, \ldots, \mathsf{ek}_f, \\ (w_{f+1}, \pi_{f+1}), \ldots, (w_t, \pi_t) \end{array}\right), \left(\begin{array}{c} z, \mathsf{ek}'_1, \ldots, \mathsf{ek}'_f, \\ (w'_{f+1}, \pi'_{f+1}), \ldots, (w'_t, \pi'_t) \end{array}\right)\right) \leq \gamma(\lambda)$$

where $z \leftarrow\$ \mathcal{Z}$, $\mathsf{ek} \leftarrow\$ \mathcal{EK}^*$, $\rho \leftarrow\$ \{0,1\}^\kappa$ *and* vk *is a corresponding verification key and,*

- $\mathsf{ek}_j \leftarrow\$ \mathsf{KeyShareGen}(\mathsf{ek}, n, t, \rho)$ *for* $i = 1, \ldots, t$ *and* $(w_i, \pi_i) \leftarrow \mathsf{Eval}(\mathsf{ek}_i, z)$ *for* $i = f+1, \ldots, t$.
- $(w'_i, \pi'_i) \leftarrow\$ \mathcal{EV}_{\lambda,n,t}(\mathcal{EK}^*, \mathsf{vk}, z)$ *for* $i = f+1, \ldots, t$ *and for all* i *in* $[f]$, $\rho_i \leftarrow\$ \{0,1\}^\kappa$, $\hat{\mathsf{ek}}^{(i)} \leftarrow\$ \mathcal{EK}^*$, $(\mathsf{ek}'_i, \cdot) \leftarrow\$ \mathsf{KeyShareGen}(\hat{\mathsf{ek}}^{(i)}, n, t, \rho_i)$.

Looking ahead, the proofs $\{\pi_i\}$ in our scheme will be based on the Fiat-Shamir transform in the random oracle model. For simplicity of presentation, we do not consider the random oracle model in the definition above, but it can easily be generalized to include the list of random oracle queries done by the adversary, as part of the distributions.

From universal traceability to traceability. As mentioned earlier, we will now relate standard traceability (Definition 7) to universal traceability (Definition 10) for bidirectional threshold VRF schemes.

Lemma 1. *Let* $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{EvalVerify}, \mathsf{Combine}, \mathsf{Verify}, \mathsf{Trace})$ *be a traceable threshold VRF scheme that satisfies* ν_1*-correctness and* (Γ_{ek}, ν_2)*-bidirectionality for negligible functions* ν_1, ν_2 *of the security parameter* $\lambda \in \mathbb{N}$ *and partition* Γ_{ek} *of* $\mathcal{EK}_\lambda$. *Let* $\epsilon = \epsilon(\lambda)$ *and* $\delta = \delta(\lambda)$ *be functions of the security parameter, and let* $\mathcal{A}$ *be an adversary. Assume that* $\mathsf{Adv}^{\mathrm{trac-b}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) \geq 2\epsilon$. *Then, there exists negligible functions* $\nu = \nu(\cdot), \nu' = \nu'(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$ *and every* $b \in \{0,1\}$, *it holds that*

$$\mathsf{Adv}^{\text{uni-trac}}_{\mathcal{A},\mathsf{TTVRF},\Gamma,\Gamma_{ek},\epsilon',\delta'}(\lambda) \geq \frac{1}{2} \cdot \epsilon \cdot \delta \cdot \mathsf{Adv}^{\mathrm{trac}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) - \nu(\lambda)$$

where $\delta' = \epsilon' \geq \frac{1}{2}\epsilon^2\delta - \nu'$.

The proof essentially follows the proof of [10][Lemma 1].

3 A One-Time Traceable VRF from Paillier Encryption

In this section, we present a one-time traceable threshold VRF based on Paillier encryption. For ease of presentation, we start by presenting a scheme with two caveats: (1) the tracing key provides too much information, to the point where the tracer can break the uniqueness of the VRF; and (2) the scheme only enjoys one-time traceability. We stress, however, that this is just for the sake of presentation, as this base scheme is quite involved already. Jumping ahead, in Section 4 we show how to enhance this base scheme with tracer uniqueness; and in Section 5, we discuss how to bootstrap it to achieve many-time traceability.

Our starting point: Traceable secret sharing. The starting point for our construction is the recent work by Boneh, Partap, and Rotem [10] (BPR), who showed how to trace leaks in Shamir threshold secret sharing [44]. At a very high level, their idea was as follows. The tracer in their setting gets black-box access to an algorithm R that has f secret shares "hardcoded" in it. R gets additional $t - f$ shares as input, and it outputs the reconstruction of the combined $f + (t - f) = t$ shares. The BPR tracing algorithm leverages the fact that in Shamir secret sharing, each secret share is an evaluation $(x, y = h(x))$ of a polynomial h, and the secret is the result of interpolating the polynomial from these evaluations at the point 0.

Let us focus first on the case where R is a perfect reconstruction box, and it always outputs the result of the interpolation at 0, based on the t shares. Let us further assume for simplicity that $f = t - 1$, and so R takes one more share (x, y) as input. The observation of BPR is that from two correlated queries to R, namely queries of the form (x, y) and $(x, y+\sigma)$, one can obtain the Lagrange coefficient λ_x of x in the interpolation of h at 0. They then observed that this coefficient depends on the x-values of the corrupted parties; in particular, $\lambda_x = \prod_{i=1}^{t-1} x_i/(x_i - x)$, where $x_1, \ldots, x_{t-1}$ are the x-values of the corrupted parties. Then, one can think of λ_x^{-1} as the evaluation at x of $\psi(X)$, a low-degree polynomial, whose roots are $x_1, \ldots, x_{t-1}$. Thus, repeating this trick t times for different values of x allows one to interpolate the polynomial ψ, factor it to find its roots, and then map these roots back to the corrupted parties. If R is imperfect, and may err occasionally, we may end up with some wrong evaluations of λ_x^{-1}. In that case, BPR showed that one can rely on list decoding for Reed-Solomon codes to find the polynomial ψ.

A first attempt. At first glance, it is very tempting to try and use the Shamir tracing algorithm of BPR in existing constructions of threshold VRFs, in which the secret keys of the parties are Shamir secret shares of a global secret in a finite field. Perhaps the first obvious such VRF is the DDH-based threshold VRF [37], defined via $z \mapsto H(z)^{\mathsf{ek}}$, where H is a hash function mapping inputs to a DDH-hard cyclic group of prime order p, and $\mathsf{ek} \in \mathbb{Z}_p$ is the global function key which is secret shared via Shamir secret sharing to derive the individual evaluation keys. Indeed, adapting the tracing algorithm of BPR to work with a pirate evaluation box E for this VRF, we can obtain evaluations of the form $H(z)^{\lambda_x}$ for many values of x. Alas, we immediately encounter a problem. Since λ_x is in the exponent, we cannot rely on list-decoding — which involves non-linear operations — to recover the ψ polynomial.

Paillier to the rescue? To circumvent the problem described above, we look at another VRF, namely a VRF based on Paillier encryption. Boneh, Haitner, and Lindell [8] recently observed that the function defined via $z \mapsto \mathsf{Dec}_{\mathsf{sk}}(H(z))$, where Dec is the decryption algorithm for Paillier encryption, can be proven to be a VRF. Recall that in Paillier encryption, the public key is an RSA bi-prime modulus $N = p \cdot q$ and the secret key is the order $\phi(N) = (p - 1)(q - 1)$ of the multiplicative group $\mathbb{Z}_N^*$. Plaintexts then live in $\mathbb{Z}_N$ and ciphertexts in $\mathbb{Z}_{N^2}$. To encrypt a message

$m \in \mathbb{Z}_N$, one samples $r \leftarrow\$ \mathbb{Z}_N$ and computes the ciphertext $\mathsf{ct} \leftarrow (N+1)^m \cdot r^N \mod N^2$. To decrypt, one computes $m \leftarrow L_N(\mathsf{ct}^{\phi(N)})/\phi(N) \mod N$, where $L_N(x) = ((x \mod N^2) - 1)/N$. Intuitively, decryption is done via projecting the ciphertext onto the subgroup of $Z^*_{N^2}$ that is generated by $(N+1)$ to obtain $(N+1)^m$. Since discrete log is easy in that subgroup, this efficiently yields m in $\mathbb{Z}_N$.

The reader might already notice a significant advantage of the Paillier-based VRF over the DDH-based one: evaluations live in the additive group $\mathbb{Z}_N$, and not in the exponent of a dlog-hard group! This gives hope that we can leverage this fact to force the BPR Shamir tracing algorithm [10] to go through. However, there are major barriers that make translating this hope into a traceable threshold VRF construction a significant challenge:

1. Unlike most DDH-hard groups, the group $\mathbb{Z}^*_{N^2}$ is a group of unknown order. This makes thresholdizing the Paillier-based VRF a much more cumbersome task. Still, previous works have looked at this problem in the context of threshold decryption [26,34], and we leverage some of their observations with the necessary adjustments.
2. The group $\mathbb{Z}_N$ is not an integral domain, and list decoding algorithms for Reed-Solomon codes, that are typically defined over finite fields, do not naturally extend to it.
3. We still need to make this function verifiable. Somewhat paradoxically, at the same time, we need the tracing algorithm to be able to fool the pirate evaluation box E and feed it wrong evaluations. This is needed in order to make our version of the BPR tracing algorithm go through.

We now present our construction and discuss how we overcome these problems.

3.1 Our Scheme

In this section, we present the workings of our Paillier-based threshold VRF construction. In subsequent sections, we will explain how we trace leaks in this VRF.

To construct a VRF from Paillier encryption, we sample an RSA modulus $N = pq$, where p and q are safe primes, and set the evaluation key to be the decryption key $\mathsf{ek} = \phi(N)$. To evaluate the VRF at $z \in \{0,1\}^\lambda$, we simply hash z to the ciphertext space, and use the evaluation key to decrypt it. More formally, let $\mathsf{H}_0 : \{0,1\}^\lambda \to \mathcal{QR}_{N^2}$ be a hash function [1]. We restrict the ciphertexts to the group of quadratic residues mod N^2 to allow efficient proofs for partial evaluation, as we will see later in this section. Then, we define the ciphertext corresponding to z as $ct_z = \mathsf{H}_0(z)$, and the VRF function is the decryption of ct_z.

To thresholdize the above construction, we use Shamir secret sharing over $\mathbb{Z}_{N\phi(N)/4}$ to distribute the secret $\phi(N)/4$: we sample a uniformly random polynomial h of degree $t-1$, with coefficients $a_0, a_1, \ldots, a_{t-1}$ such that $h(0) = \beta \cdot \phi(N)/4$. Here, β is a blinding factor sampled uniformly from $\mathbb{Z}_N$. Each party $i \in [n]$ is uniquely associated with an evaluation point $x_i \in \mathbb{Z}_N$ and is given the share $(x_i, h(x_i))$. This share can be used to partially evaluate the VRF, by raising the ciphertext ct_z to the power $h(x_i)$. Any t parties can combine their partial evaluations via Lagrange interpolation in the exponent. More formally, let us consider an example where parties $1, 2, \ldots, t$ want to evaluate the VRF. Then, the Lagrange coefficient for party i is:

[1] Note that the range of a hash function modeled as a random oracle cannot depend on the parameter of the scheme N. To get around this, we can use a hash function with a very large range: $\mathsf{H}^* : \{0,1\}^\lambda \to \{0, \ldots, B\}$ where $B = 2^\lambda \cdot 2^{2\gamma\lambda} \gg N$, and γ is a parameter such that $n \leq 2^{\gamma\lambda}$ for all λ and n sampled as in KeyGen. Then we define $\mathsf{H}_0(z)$ as $(\mathsf{H}^*(z) \bmod N^2)^2$.

$$\hat{\lambda}_i^{[t]} = \prod \frac{x_j}{x_j - x_i}$$

Unfortunately, these coefficients cannot be computed directly in $\mathbb{Z}_{N\phi(N)/4}$ since $\phi(N)$ is unknown. Instead, we first compute $\nu'_{[t]}$ as follows:

$$\nu'_{[t]} = \prod_{i,j \in [t], i<j} (x_i - x_j)$$

Observe that we can easily compute $\lambda_i^{[t]} = \nu'_{[t]} \cdot \hat{\lambda}_i^{[t]}$ for all $i \in [t]$, since $\nu'_{[t]}$ divides all the pairwise difference terms in the denominator of the Lagrange coefficients. Hence, to combine partial evaluation shares, the combiner raises the share of party i to the power $\lambda_i^{[t]}$. This results in $d = ct_z^{\beta \cdot \phi(N) \cdot \nu'_{[t]}/4}$. Note that this falls into the subgroup with easy discrete log! So, the VRF output can be computed as follows:

$$f_{\mathsf{ek}}(z) = \frac{L_N(d)}{\nu'_{[t]}}$$

Lastly, the distribution of any $t-1$ shares is the same for any two secret keys because of perfect secrecy of Shamir secret sharing over $\mathbb{Z}_{N\phi(N)/4}$.

To generate partial evaluation proofs, we use Chaum-Pedersen style proofs over the group of quadratic residues $\mathcal{QR}_{N^2}$. We first use integer commitments [20] to commit to all the coefficients of the polynomial $h(X)$. Let $v \in \mathbb{Z}_{N^2}$ be a uniform random element in the group of quadratic residues $\mathcal{QR}_{N^2}$. Then, the commitment to each coefficient a_j of $h(X)$ is v^{a_j} for all $j \in \{0, \ldots, t-1\}$. To prove that an evaluation share (x_i, w_i) is correct, we can compute $v' = v^{h(x_i)} = \prod_{j \in \{0,\ldots,t-1\}} (v^{a_j})^{x_i^j}$, and then use a non-interactive zero-knowledge proof for the following discrete-log equality relation:

$$\mathcal{R}_{eq} = \{((v, v', w, w'); (y)) : v' = v^y \wedge w' = w^y\} \tag{1}$$

where $w = ct_z$ and $w' = w_i$. We describe an efficient proof system $(\mathcal{P}_{eq}, \mathcal{V}_{eq})$ for the above relation in Appendix A.

We do need to make one modification to Shamir secret sharing over integers to support tracing, which we now discuss.

Random evaluation points. Typically in Shamir secret sharing, each party i is deterministically associated with its evaluation point x_i. A natural choice is to set $x_i = i$. As observed by [10], this approach cannot allow efficient tracing as per our definition. We refer to [10] for a detailed reasoning for the same.

To avoid this problem, we consider a variant of Shamir secret sharing, wherein each x_i is sampled uniformly at random from $\mathbb{Z}_N$. Since this set is large enough, this has a very small impact on the correctness of the scheme. The benefit is that now a share $(x_i, h(x_i))$ cannot be linked to party i without knowing the dealer's randomness. As we will now see, this allows us to overcome the impossibility argument sketched in [10].

3.2 Tracing perfect boxes via polynomial interpolation

To understand our tracing algorithm, let us first start with a perfect evaluation box, i.e. a box that always outputs the correct VRF evaluation. We also make a simplifying assumption that

$f = t-1$. In this case, the box E gets as input $z \in \mathcal{Z}$ and one partial evaluation, and outputs the full evaluation. This output is the result of reconstructing the value $h(0)$ in the exponent using the $t-1$ evaluations of h hardcoded in the box and the additional evaluation share given as input. Let us denote the shares in the box as $(x_i, y_i = h(x_i))$ for $i = 1, \ldots, t-1$. So, when given z and $(x_t, ct_z^{h(x_t)})$, a perfect box E will output:

$$w = \frac{L_N\left(\prod_{i\in[t]} ct_z^{\lambda_i \cdot h(x_i)}\right)}{\nu'} \bmod N$$

where ν' is the product of pairwise differences: $\nu' = \prod_{i,j\in[t],i<j}(x_i - x_j)$ and λ_i is the Lagrange coefficient : $\lambda_i = \nu' \cdot \prod_{j\in[t],j\neq i} \frac{x_j}{x_j - x_i}$ for all $i \in [t]$.

Recall that the ciphertext ct_z is a quadratic residue mod N^2. Hence we can denote $ct_z = (g^m \cdot r^N)^2$ for some $m \in \mathbb{Z}_N$ and $r \in \mathbb{Z}_{N^2}$, where $g = 1 + N$. Then, the above equals:

$$w = \frac{L_N(ct_z^{\nu'\cdot\beta\cdot\phi(N)/4})}{\nu'} \bmod N = m\beta\phi(N)/2 \bmod N \tag{2}$$

Following the Shamir tracing approach in [10], we now run the box with a slightly different input: z and $(x_t, ct_z^{h(x_t)} \cdot g)$. Note that this is an incorrect evaluation share, but let us assume that the tracing key can be used to generate valid proofs for such shares. Then, with these inputs, a perfect box must output:

$$w' = \frac{L_N\left(ct_z^{\lambda_t \cdot h(x_t)} \cdot g^{\lambda_t} \cdot \prod_{i\in[t-1]} ct_z^{\lambda_i \cdot h(x_i)}\right)}{\nu'} \bmod N$$

We can simplify the above to get:

$$w' = \frac{L_N\left(ct_z^{\nu'\cdot\beta\cdot\phi(N)/4} \cdot g^{\lambda_t}\right)}{\nu'} \bmod N = w + \frac{\lambda_t}{\nu'} \tag{3}$$

Subtracting Eq. (2) from Eq. (3), and rearranging, we get:

$$(w' - w)^{-1} = \prod_{i\in[t-1]} \frac{x_i - x_t}{x_i} \bmod N \tag{4}$$

We now consider the univariate polynomial $p(x) = \prod_{i\in[t-1]} \frac{x_i - X}{x_i} \in \mathbb{Z}_N[X]$ in the indeterminate X. Observe that the x_i values of all the corrupted parties are roots of the polynomial p. Moreover, we can interpret Eq. (4) as an evaluation of p at the point x_t we fed to E. Repeating the above with additional $t-1$ fresh x_t values, would give us t evaluations of p. Since p is of degree $t-1$, if the true x_i values of all the parties are given to the tracer as part of tk, the tracer can now interpolate p at each x_i and blame party i if $p(x_i) = 0$.

Generating valid proofs for invalid shares. The algorithm above requires the tracer to be able to feed the evaluation box E with incorrect partial evaluations. Recall that our traceability definition mandates that whenever E takes a partial evaluation, it also takes a proof asserting its validity (this only makes the definition stronger). So the question that remains is: how can the tracer feed E with incorrect evaluations without being detected?

To allow the tracer to do so, we augment the verification key vk and the tracing key tk as follows. We modify the verification key to include a random element b in a prime-order, discrete-log-hard, group $\mathbb{G}$ generated by a generator a. We also include the discrete log of b with respect to a as part of the tracing key. Then, instead of using the plain proofs of discrete log equality (as in $\mathcal{R}_{eq}$) describe above, we use proofs that assert that either the aforesaid discrete log equality holds, or the prover knows the discrete log of b with respect to a. This is captured by the following relation:

$$\mathcal{R}'_{eq} = \left\{ \big((v, v', w, w', a, b); (y, u)\big) : \begin{matrix} (v' = v^y \wedge w' = w^y) \vee \\ b = a^u \end{matrix} \right\} \tag{5}$$

This idea is that $u = dlog_a(b)$ allows the tracer to produce a valid proof for any partial evaluation. Since we use zero-knowledge proofs, this goes undetected, and the tracing algorithm described above can go through. On the other hand, a party that does not hold the (secret) tracing key, does not know the discrete log $u = dlog_a(b)$ and hence can only prove the validity of correct partial evaluations. We present an efficient proof system $(\mathsf{P}'_{eq}, \mathsf{V}'_{eq})$ for the relation $\mathcal{R}'_{eq}$ in Appendix A.2.

For now, the augmented tracing key allows the tracer to break uniqueness for all inputs. However, as mentioned, we get around this issue in Section 4.

3.3 Tracing imperfect boxes

We now describe the full tracing algorithm. Notice that the informal overview from the previous section inherently assumed that the evaluation box E is always correct, and in particular, that all evaluations of the polynomial $p(X)$ are correct. To work for imperfect evaluation boxes, that output the correct VRF evaluation only with a certain non-negligible probability, we need an additional idea. This is because standard polynomial interpolation with erroneous evaluations might fail or give us the wrong evaluation, which can lead to the tracer blaming honest parties or not finding all the traitors. As observed by [10], this problem of interpolating a polynomial of bounded degree from a set of evaluation points with errors is equivalent to the list decoding problem for Reed Solomon codes [46,33]. Unfortunately, as mentioned earlier in this section, since our polynomial $p(X)$ is defined over $\mathbb{Z}_N$, existing list decoding algorithms that work over finite fields cannot be used directly. We will now describe how we can modify Reed Solomon list decoding algorithms to work for our setting.[2]

Let $\{(x'_j, y_j)\}_{j\in[m]}$ denote the partially-erroneous list of evaluations of $p(X)$ that we get by querying the box m times, as described in the previous section. Following Reed-Solomon list decoding algorithms, we first construct a low-degree bivariate polynomial $Q(x, y)$ such that $Q(x'_j, y_j) = 0$ for all $j \in [m]$. Then, the list of factors of $Q(X, Y)$ of the form $Y - p^{(j)}(X)$ includes all polynomials $p^{(j)}(X)$ of degree $t - 1$ that agree with a pre-determined fraction of the evaluation points. Specifically, $Y - p(X)$ must be a factor of $Q(X, Y)$, where $p(X)$ is the polynomial defined in the previous section, and has all the corrupt x_i values as roots. We now make a crucial observation that, similarly to the perfect case, $Q(x_i, 0)$ must evaluate to zero for all the corrupt x_i values.

Additionally, we show that with high probability, $Q(x_j, 0) \neq 0$ if party j is honest, and so j will not be blamed by our tracing algorithm. To see this, first observe that $Q'(X) = Q(X, 0)$ is a low-degree univariate polynomial. Now there are two possibilities:

1. The fraction of roots of $Q'(X)$ in $\mathbb{Z}_N$ is non-negligible. In this case, it has been shown that Q' can be used to factor N (see, for example, Aggarwal and Maurer [1]). By the hardness of factoring N, this can only happen with a very small probability.

[2] To be clear, we show how the main ideas behind list decoding can be used for tracing over $\mathbb{Z}_N$. We do not, however, present a list decoding algorithm over $\mathbb{Z}_N$.

2. The fraction of roots of $Q'(X)$ in $\mathbb{Z}_N$ is negligible. In this case, we observe that the x-values $\{x_i\}$ of the key shares of honest parties are statistically independent from the view of the box. Hence, the probability that the x_i value for an honest party i is a root of this polynomial is negligible as well.

So, the tracer can simply iterate over the x_i values of all the parties and blame those with $Q(x_i, 0) = 0$.

One subtlety that may arise is that Q may be of the form $Q(X, Y) = Y^k \cdot \hat{Q}(X, Y)$ for some $k > 0$. Then, $Q(X, 0)$ is equal to zero for all values of X. Fortunately, this is easy to fix, by simply dividing out the largest power of Y from Q. Specifically, in the above example, we would work with the polynomial $\hat{Q}$, and now the analysis described before applies.

Figures 4, 5 and 6 formally describe our one-time traceable threshold VRF from Paillier. It relies on a group generation algorithm $\mathsf{GroupGen}(1^\lambda) \to (\mathbb{G}, a, q^*)$ which takes as input the security parameter and outputs a cyclic group $\mathbb{G}$ generated by a, with prime order $q^* > N^2$.

3.4 Correctness and Security

We now prove correctness, uniqueness, pseudorandomness, tracer pseudorandomness and one-time traceability for this scheme.

Correctness. Correctness follows directly from the completeness of the proof system $(\mathsf{P}'_{eq}, \mathsf{V}'_{eq})$ and correctness of the secret sharing scheme. Specifically, we have,

$$d = \prod_{j\in[t]} \hat{w}_{i_j}^{\lambda_j} = \prod_{j\in[t]} ct_z^{h(x_i)\cdot\lambda_j} = ct_z^{\nu'\cdot h(0)} = ct_z^{\nu'\cdot\beta\cdot\phi(N)/4}$$

Let $ct_z = \mathsf{H}_0(z) = (g^m r^N)^2$ for some $m, r \in \mathbb{Z}_N$. Then, we get that,

$$w = \frac{L_N(d)}{\nu'} = \frac{L_N(g^{2m\nu\beta\cdot\phi(N)/4})}{\nu'} = m\beta\cdot\phi(N)/2 \bmod N$$

Hence, w is indeed the correct VRF evaluation, and we get the same evaluation for all subsets $\mathcal{J} \subseteq [n]$ of size t.

Uniqueness. Theorem 1 below proves uniqueness of the scheme based on special soundness of the proof system and the hardness of discrete log in the group $\mathbb{G}$ (Definition 12).

Definition 12. *Let* $\mathsf{GroupGen}$ *be a group generator. The discrete log assumption holds with respect to* $\mathsf{GroupGen}$ *if, for all* PPT *adversaries* $\mathcal{A}$*, the following function is negligible in* λ*:*

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathcal{A},\mathsf{GroupGen}}(\lambda) := \Pr[\mathcal{A}(\mathbb{G}, a, q^*, a^u) = u]$$

where the probability is taken over the random choice of the generator a*, the random choice of* $u \in \mathbb{Z}_{q^*}$*, and the randomness used by* $\mathcal{A}$*.*

Theorem 1. *Let* $\mathsf{GroupGen}$ *be a group generator. For every* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that,*

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathcal{B},\mathsf{GroupGen}}(\lambda) \geq (\mathsf{Adv}^{\mathrm{uniq}}_{\mathcal{A},\mathsf{OT}\text{-}\mathsf{P}_1}(\lambda))^2 - 4/N^2 - 1/q'$$

where $N = N(\lambda)$*,* $N = pq, p > q$*,* $q = 2q' + 1$ *and* $q' = q'(\lambda)$ *is a function of the security parameter.*

One-Time Traceable threshold VRF $\mathsf{OT\text{-}P}_1$ without tracer uniqueness.

$\underline{\mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta}):}$

1. Sample two safe primes $p, q \leftarrow\!\!\$\ \mathcal{P}_\lambda$ s.t. $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are primes. Here, $\mathcal{P}_\lambda$ is the set of all λ-bit safe primes.
   Compute $N = pq$ and $\phi(N) = (p-1)(q-1) = 4p'q'$.
   Let $\mathsf{H}_0 : \{0,1\}^\lambda \to \mathcal{QR}_{N^2}$ be a hash function, and let $g = 1 + N \in \mathbb{Z}_{N^2}$ and sample $v \leftarrow\!\!\$\ \mathcal{QR}_{N^2}$.
2. Sample $\beta \leftarrow\!\!\$\ \mathbb{Z}_N^*$ and sample coefficients of the polynomial to be used for Shamir secret sharing: $a_1, \ldots, a_{t-1} \leftarrow\!\!\$\ \mathbb{Z}_{Np'q'}$.
   Define a polynomial of degree $t-1$: $h(X) = p'q'\beta + \Sigma_{i \in [t-1]} a_i \cdot X^i \in \mathbb{Z}_{Np'q'}[X]$.
   Compute $v_i \leftarrow v^{a_i}$ for all $i \in [t-1]$, and $v_0 \leftarrow v^{p'q'\beta}$.
3. Sample $z_1, \ldots, z_m \leftarrow\!\!\$\ \{0,1\}^\lambda$ where $m = m(n, 1/\epsilon, 1/\delta, \lambda)$.
4. For each $j \in [m]$, sample $x_{j,1}, \ldots, x_{j,t-1}, \alpha_{j,1}, \ldots, \alpha_{j,t-1} \leftarrow\!\!\$\ \mathbb{Z}_N$. Then, compute $ct_j \leftarrow \mathsf{H}_0(z_j)$, and for each $i \in [t-1]$, $w_{j,i} \leftarrow (x_{j,i}, ct_j^{h(x_{j,i}) + p'q'(\alpha_{j,i} - \beta)})$.
5. Sample a cyclic group of prime order: $(\mathbb{G}, a, q^*) \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$ such that $q^* > N^2$ and sample $u \leftarrow\!\!\$\ \mathbb{Z}_{q^*}$.
   Compute $b \leftarrow a^u$.
6. Output $\mathsf{ek} \leftarrow p'q'\beta, \rho \leftarrow (\{a_1, \ldots, a_{t-1}\}), \mathsf{ck} \leftarrow (N)$, $\mathsf{vk} \leftarrow (N, v, \{v_i\}_{i \in \{0,\ldots,t-1\}}, a, b)$ and the tracing key:
$$\mathsf{tk} \leftarrow (N, a, b, u, \{(z_j, \{w_{j,i}\}_{i \in [t-1]})\}_{j \in [m]}, v, \{v_i\}_{i \in \{0,\ldots,t-1\}}).$$

$\underline{\mathsf{KeyShareGen}(1^\lambda, \mathsf{ek}, n, t, \rho):}$

1. Parse ρ as $(\{a_1, \ldots, a_{t-1}\})$. Sample $x \leftarrow\!\!\$\ \mathbb{Z}_N$.
2. Compute $y \leftarrow \mathsf{ek} + \Sigma_{j \in \{1,\ldots,t-1\}} a_j \cdot x^j$. Output $\mathsf{ek}_i \leftarrow (x, y)$ and $\mathsf{tk}_i \leftarrow x$.

$\underline{\mathsf{Eval}(\mathsf{ek}_i, \mathsf{vk}, z):}$

1. Parse vk as $(N, v, \{v_i\}_{i \in \{0,\ldots,t-1\}}, a, b)$ and ek_i as (x_i, y_i). Compute $ct_z \leftarrow \mathsf{H}_0(z)$. Compute $v' \leftarrow v^{y_i}$, $w'_i \leftarrow ct_z^{y_i}$.
2. Output $w_i \leftarrow (x_i, w'_i)$ and $\pi_i \leftarrow \mathsf{P}'_{eq}((v, v', ct_z, w'_i, a, b), (y_i, \perp))$.

$\underline{\mathsf{Combine}(\mathsf{ck}, w_{i_1}, \ldots, w_{i_t}, \pi_{i_1}, \ldots, \pi_{i_t}):}$

1. Parse w_{i_j} as $(x_{i_j}, \hat{w}_{i_j})$ for all $j \in [t]$. Parse ck as (N).
2. Compute $\nu' \leftarrow \prod_{j,k \in [t], i_j < i_k} (x_{i_j} - x_{i_k})$.
3. Compute the integer Lagrange coefficients for each $j \in [t]$:
$$\lambda_j = \left(\frac{\nu'}{\prod_{k \in [t], k \neq j} (x_{i_k} - x_{i_j})} \right) \cdot \prod_{k \in [t], k \neq j} x_{i_k}$$
4. Compute $d = \prod_{j \in [t]} \hat{w}_{i_j}^{\lambda_j}$.
5. Output $w \leftarrow \frac{L_N(d)}{\nu'} \bmod N$ and $\pi \leftarrow \{(w_{i_j}, \pi_{i_j})\}_{j \in [t]}$.

Fig. 4. The $\mathsf{FuncSamp}$, $\mathsf{KeyShareGen}$, Eval and $\mathsf{Combine}$ algorithms for our Paillier-based one-time traceable threshold VRF scheme $\mathsf{OT\text{-}P}_1$.

```
$\mathsf{EvalVerify}(\mathsf{vk}, z, w_i, \pi_i)$:

1. Parse vk as $(N, v, \{v_i\}_{i \in \{0,\dots,t-1\}}, a, b)$ and $w_i$ as $(x_i, w'_i)$.
2. Compute $v' \leftarrow \prod_{j \in \{0,\dots,t-1\}} v_j^{x_i^j}$. Let $w \leftarrow \mathsf{H}_0(z)$.
3. Output $\mathsf{V}'_{eq}((v, v', w, w'_i, a, b), \pi_i)$.

$\mathsf{Verify}(\mathsf{vk}, z, w, \pi)$:

1. Parse vk as $(N, v, \{v_i\}_{i \in \{0,\dots,t-1\}}, a, b)$, $\pi$ as $\{(w_i, \pi_i)\}_{i \in \mathcal{J}}$ and $w_i$ as $(x_i, \hat{w}_i)$ for all $i \in \mathcal{J}$.
2. Output 0 if $|\mathcal{J}| < t$, or if, for some $i \in \mathcal{J}$, $\mathsf{EvalVerify}(\mathsf{vk}, z, w_i, \pi_i) = 0$.
3. Output 0 if $w \neq L_N(d)/\nu'$, where $d = \prod_{i \in \mathcal{J}} \hat{w}_i^{\lambda_i}$, and the lagrange coefficients $\lambda_i$ and $\nu'$ are computed as
   in Steps 3 and 2 of the Combine algorithm respectively. Otherwise, output 1.
```

Fig. 5. The EvalVerify and Verify algorithms for our Paillier-based one-time traceable threshold VRF scheme $\mathsf{OT\text{-}P_1}$.

```
$\mathsf{Trace}^E(\mathsf{tk}, f)$:

1. Parse tk as $(N, a, b, u, \{z_j, \{w_{j,i}\}_{i \in [t-1]}\}_{j \in [m]}, v, \{v_i\}_{i \in \{0,\dots,t-1\}}, \{x_i^*\}_{i \in [n]})$, and $w_{j,i}$ as $(x_{j,i}, \hat{w}_{j,i})$ for all $j \in [m], i \in [t-1]$. Let $g = (1+N) \bmod \mathbb{Z}_{N^2}$.
2. For $\ell = 1, \dots, m$:
   (a) For $i \in \{1, \dots, t-f\}$: set $sh_i \leftarrow (x_{\ell,i}, \hat{w}_{\ell,i})$.
   (b) Sample $\eta_\ell \leftarrow\$ \mathbb{Z}_{N^2/4}$ and set $sh'_{t-f} \leftarrow (x_{\ell,t-f}, \hat{w}'_{\ell,t-f} = \hat{w}_{\ell,t-f} \cdot g^{2\eta_\ell})$. Let $x'_\ell \leftarrow x_{\ell,t-f}$.
   (c) For $i \in [t-f]$, compute $v_{\ell,i} \leftarrow \prod_{j \in \{0,\dots,t-1\}} v_j^{x_{\ell,i}^j}$ and $\pi_{\ell,i} \leftarrow \mathsf{P}'_{eq}((v, v_{\ell,i}, \mathsf{H}_0(z_\ell), \hat{w}_{\ell,i}, a, b), (\perp, u))$.
   (d) Compute $\pi'_{\ell,t-f} \leftarrow \mathsf{P}'_{eq}((v, v_{\ell,t-f}, \mathsf{H}_0(z_\ell), \hat{w}'_{\ell,t-f}, a, b), (\perp, u))$.
   (e) Query $E$ on $(z_\ell, (sh_1, \pi_1), \dots, (sh_{t-f}, \pi_{t-f}))$ and $(z_\ell, (sh_1, \pi_1), \dots, (sh_{t-f-1}, \pi_{t-f-1}), (sh'_{t-f}, \pi'_{t-f}))$.
       Let $w_\ell$ and $w'_\ell$ be $E$'s responses respectively.
   (f) If $w'_\ell = w_\ell$ or $\eta_\ell = 0$ or $x_{\ell,i} = x_{\ell,j}$ for some $i \neq j \in [t-f]$ or if $\ell > 1$ and $x_{\ell,t-f} = x_{i,t-f}$ for some $i < \ell$,
       then terminate and output $\perp$.
   (g) Otherwise, let $y_\ell \leftarrow \frac{2\eta_\ell}{w'_\ell - w_\ell} \cdot \prod_{i=1}^{t-f-1} \frac{x_{\ell,i}}{x_{\ell,i} - x'_\ell} \bmod N$.
3. Let $L = \{x'_\ell, y_\ell\}_{\ell \in [m]}$ and let $D = \lceil \sqrt{2fm} \rceil$. Compute a bivariate polynomial $Q(x,y) \in \mathbb{Z}_N[X,Y]$ with
   $(1, t-1)$-weighted degree bounded by $D$ such that $Q(x'_\ell, y_\ell) = 0$ for all $\ell \in [m]$.
4. Let $k \in \mathbb{Z}$ be the largest non-negative integer such that $Q(x,y)$ divides $y^k$, i.e., $Q(x,y) = y^k \cdot Q'(x,y)$ for
   some polynomial $Q' \in \mathbb{Z}_N[X,Y]$.
5. Let $\hat{Q}(x) = Q'(x, 0)$ and $\mathcal{I} = \phi$. For each $i \in [n]$, if $\hat{Q}(x_i^*) = 0$, then add $i$ to $\mathcal{I}$, i.e., $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$.
6. Output $\mathcal{I}$.
```

Fig. 6. The Trace algorithm for our one-time traceable threshold VRF $\mathsf{OT\text{-}P_1}$.

The proof of Theorem 1 is presented in Appendix D.1.

Pseudorandomness. We prove pseudorandomness in Theorem 2 based on the semantic security of the Paillier decryption system. We denote by $\mathsf{Adv}^{\mathrm{ss}}_{\mathcal{B},\mathsf{PE}}(\lambda)$ the advantage of an adversary in the semantic security game for Paillier. We refer to [12, Section 9.2.2] for a formal definition. For the proof, we take inspiration from [26], but we make some changes to be able to handle random evaluation points.

Theorem 2. *For every* PPT *adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that,*

$$\mathsf{Adv}^{\mathrm{rand}}_{\mathcal{A},\mathsf{OT\text{-}P_1}}(\lambda) \leq q_H \cdot \mathsf{Adv}^{\mathrm{ss}}_{\mathcal{B},\mathsf{PE}}(\lambda) + 2t/q$$

where $q_H = q_H(\lambda)$ is an upper bound on the number of random oracle queries by $\mathcal{A}$, and $N = pq$, $p > q$, where $p = p(\lambda)$ and $q = q(\lambda)$ are functions of the security parameter and $t = t(\lambda)$ denotes the threshold.

Theorem 2 is proved in Appendix D.2.
Achieving uf-1 security. As mentioned in Section 2, our construction can be proven to achieve uf-1 security by adapting ideas from [4]. More details can be found in Appendix C.

One-time Traceability. Theorem 3 below proves one-time universal tracing security of the above scheme. It proves that $\mathsf{OT\text{-}P_1}$ is one-time traceable with respect to the following partition of the space of evaluation keys: $\Gamma_{ek} = \{\{p'q'\alpha\}_{\alpha\in\mathbb{Z}_N}\}$. We rely on the factoring assumption for the proof, which states that it is hard for a PPT adversary to factor an RSA modulus $N = pq$. We denote by $\mathsf{Adv}^{\text{factor}}_{\mathcal{B}}(\lambda)$ the advantage of an adversary $\mathcal{B}$ in the factoring problem, where p and q are chosen to be λ-bit primes. We refer to [12] for a formal definition.

Theorem 3. *For every PPT adversary $\mathcal{A}$, for every $\lambda, N \in \mathbb{N}$, for every $m, D \in \mathbb{N}$ such that $\sqrt{2fm} < D < m < N$, for every $\epsilon, \delta \in (0,1]$ such that $\epsilon\delta \geq max\left(\sqrt{\frac{4}{q} + \frac{m+1+t(t-f)}{N}}, \sqrt{\frac{2D}{m}}\right)$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{OT\text{-}P_1},\epsilon,\delta}(\lambda) \leq \mathsf{Adv}^{\text{factor}}_{\mathcal{B}}(\lambda) + \frac{(n-f)\cdot poly(\lambda)}{N} + e^{-\frac{\epsilon^2\delta^2 m}{2}\cdot(1-\frac{1}{r})^2}$$

where $r = \frac{\epsilon^2\delta^2 m}{2D}$, $N = pq$, where $p = p(\lambda), q = q(\lambda)$ are large primes, and $n = n(\lambda)$ and $f = f(\lambda)$ are upper bounds on the number of parties and corruptions, respectively.

The parameters m, D can be set such that the advantage of the adversary is exponentially small. Specifically, we can set $D = \lceil\sqrt{2mf}\rceil$ and $m = \lceil\frac{32f\lambda}{\epsilon^4\delta^4}\rceil$. This gives us $r \geq 2$, meaning that the term $e^{-\frac{\epsilon^2\delta^2 m}{2}\cdot(1-\frac{1}{r})^2}$ is bounded by $e^{\frac{-\epsilon^2\delta^2 m}{8}}$. Moreover, since $m > \frac{8\lambda}{\epsilon^2\delta^2}$, this term is bounded by $e^{-\lambda}$. The formal proof for the above theorem can be found in Appendix D.3.

On traceability vs universal traceability. Recall that in Lemma 1, we prove that if a threshold VRF scheme satisfies universal traceability, then it also satisfies standard traceability. Hence, by combining Lemma 1 and Theorem 3, our scheme satisfies standard traceability.

Learning f. If the number of corruptions f is not known, the tracer can learn it by simply trying $f = t-1, t-2, \ldots, 1$ until it reaches a value that 'works'; that is, a value of f for which the above algorithm finds exactly f corrupted parties. We scale the number of z_i values in the tracing key by an order of $t-1$ to be able to use an independent set of inputs for each trial. More formally, suppose that the real number of corruptions is f^*. For each value $f > f^*$ that Trace tries, outputting a subset $\mathcal{I}$ of size f means outputting at least one honest party. By the analysis in the proof of Theorem 3, the probability that it outputs such a subset is at most $(n-f)\cdot \text{poly}(\lambda)/N \leq n\cdot\text{poly}(\lambda)/N$. Moreover, when Trace tries $f = f^*$, then Theorem 3 tells us that it will fail to output the correct subset with probability at most $e^{-\lambda} + \mathsf{Adv}^{\text{factor}}_{\mathcal{B}}(\lambda) + n\cdot\text{poly}(\lambda)/N$ (for the choices of parameters discussed above). Hence, by a union bound, the probability that Trace correctly traces E back the corrupted subset is at least $1 - (e^{-\lambda} + 2n^2\cdot\text{poly}(\lambda)/N + \mathsf{Adv}^{\text{factor}}_{\mathcal{B}}(\lambda))$.

Tracer Pseudorandomness. Theorem 4 proves tracer pseudorandomness for all inputs z not included in the tracing key. The proof follows that of Theorem 2, and is included in Appendix D.4.

Theorem 4. *For every* PPT *adversary $\mathcal{A}$, there exists another* PPT *adversary $\mathcal{B}$ such that,*

$$\mathsf{Adv}^{\text{t-rand}}_{\mathcal{A},\mathsf{OT\text{-}P_1}}(\lambda) \leq q_H \cdot \mathsf{Adv}^{\text{ss}}_{\mathcal{B},\mathsf{PE}}(\lambda) + 2t/q$$

where $q_H = q_H(\lambda)$ is an upper bound on the number of random oracle queries by $\mathcal{A}$, and $N = pq$, $p > q$, where $p = p(\lambda)$ and $q = q(\lambda)$ are functions of the security parameter and $t = t(\lambda)$ denotes the threshold.

4 Tracer Uniqueness

Recall that in the simplified scheme presented in the last section, the tracing key contained the discrete log of b with respect to a, which allowed the tracer to forge proofs for arbitrary VRF evaluations on every input z. In other words, the tracer could break uniqueness for all inputs z. As promised, in this section we present how to modify the scheme to achieve tracer uniqueness.

As a first step, consider sampling a large number of elements $b_1, \ldots, b_\mu$ in the group $\mathbb{G}$, for some parameter μ, and including all of them as part of the verification key. We then give the discrete log of only b_1 to the tracer. For any input z, the relation $\mathcal{R}'_{eq,z}$ is defined with respect to $b_{\mathsf{H}(z)}$, where $\mathsf{H} : \{0,1\}^\lambda \to [\mu]$ is a hash function. Specifically, the prover proves either knowledge of discrete log of $b_{\mathsf{H}(z)}$ or proves discrete log equality, as in the last section. Now, the tracer can only produce valid evaluation proofs for the subset of z values such that $\mathsf{H}(z) = 1$. This is a step in the right direction, but it still means that the tracer can break uniqueness for a $1/\mu$ fraction of the inputs z.

To amplify tracer uniqueness, we use a technique by Waters [47], building on the above idea. More formally, we rely on a hash function $\mathsf{H}_1 : \{0,1\}^\lambda \to \{0,1\}^\mu$. For an input z to the VRF, the corresponding relation $\mathcal{R}'_{eq,z}$ is defined with respect to $b_z := \prod_{j\in\lambda}(b_j)^{\mathsf{H}_1(z)_j}$. In other words, the prover needs to prove either knowledge of discrete log of b_z or prove discrete log equality for the partial evaluation share. Now, we can give the tracer the discrete log of b_z for only a small number m of input values z. Namely, m can be polynomial in $(n, 1/\epsilon, 1/\delta, \lambda)$, and in particular, it can be set to be much smaller than μ. As we discuss below, setting the parameters correctly ensures that the tracer can only forge proofs for a negligible fraction of the input space, while keeping the verification key small. Figure 7 describes the $\mathsf{FuncSamp}$, Eval and $\mathsf{EvalVerify}$ algorithms for our one-time traceable threshold VRF $\mathsf{OT\text{-}P_2}$ with tracer uniqueness.

Theorem 5 below proves m-tracer uniqueness for our threshold VRF $\mathsf{OT\text{-}P_2}$, based on the discrete log problem defined in Definition 12, wherein all the hash functions are modeled as a random oracle. To provide intuition, let us use $\{z_j\}_{j\in[m]}$ to refer to the m input values in the tracing key, and z^* to denote the input for which the adversary is able to break uniqueness, in addition to the z_i values. Then, we analyze two cases based on the hash of z^*, $\mathsf{H}_1(z^*)$:

– $\mathsf{H}_1(z^*)$ falls in the span of the m bit strings $\{\mathsf{H}_1(z_j)\}_{j\in[m]}$ corresponding to the inputs in the tracing key. In the random oracle model, $\mathsf{H}_1(z^*)$ is a uniform random string in $\{0,1\}^\mu$. Hence, this case only happens with probability proportional to $m(q^*)^m/2^\mu$, where q^* is the order of the cyclic group $\mathbb{G}$. As we discuss below, the parameter μ can be set so that this probability is negligible in λ.
– $\mathsf{H}_1(z^*)$ is independent of all the bit strings corresponding to $\{z_j\}_{j\in[m]}$. In this case, we can program the discrete log challenge b to be equal to b_{z^*}, i.e. the target with respect to which the relation $\mathcal{R}'_{eq,z^*}$ is defined, for evaluation proofs for z^*. Next, we observe that the adversary must have output a valid proof for an invalid partial evaluation share at input z^*, to win the

```
$\underline{\mathsf{FuncSamp}(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta})\text{:}}$

  1. Run Steps 1 to 4 as in the FuncSamp algorithm in Figure 4.
  2. Sample a cyclic group of prime order: $(\mathbb{G}, a, q^*) \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$ such that $q^* > N^2$.
  3. Let $\mu = \mu(n, 1^{1/\epsilon}, 1^{1/\delta}, \lambda)$. Sample $u_1, \ldots, u_\mu \leftarrow\!\!\$\ \mathbb{Z}_{q^*}$. For each $j \in [\mu]$, compute $b_j \leftarrow a^{u_j}$.
  4. For each $j \in [m]$, let $s_j \leftarrow \mathsf{H}_1(z_j)$. Compute $u_j \leftarrow \Sigma_{i\in[\mu]} s_{j,i} \cdot u_i$.
  5. Output $\mathsf{ek} \leftarrow p'q'\beta$, $\rho \leftarrow \{a_1, \ldots, a_{t-1}\}$, $\mathsf{vk} \leftarrow (N, v, v_0 \leftarrow v^{\mathsf{ek}}, \{v^{a_i}\}_{i\in[t-1]}, a, \{b_j\}_{j\in[\mu]})$, $\mathsf{ck} \leftarrow N$ and

     $\mathsf{tk} \leftarrow (\mathcal{Z}^* := \{z_j\}_{j\in[m]}, N, a, b, \{(u_j, \{w_{j,k}\}_{k\in[t-1]})\}_{j\in[m]}, v, \{v_i\}_{i\in\{0,\ldots,t-1\}})$.

$\underline{\mathsf{Eval}(\mathsf{ek}_i, \mathsf{vk}, z)\text{:}}$

  1. Parse $\mathsf{vk}$ as $(N, v, v_0, \{v_i\}_{i\in[t-1]}, a, \{b_j\}_{j\in[\mu]})$ and $\mathsf{ek}_i$ as $(x_i, y_i)$.
  2. Compute $ct_z \leftarrow \mathsf{H}_0(z)$. Set $v' \leftarrow v^{y_i}$ and $w'_i \leftarrow ct_z^{y_i}$.
  3. Compute $s_z \leftarrow \mathsf{H}_1(z)$. Set $b \leftarrow \prod_{j\in[\mu]} b_j^{s_{z,j}}$.
  4. Output $w_i \leftarrow (x_i, w'_i)$ and $\pi_i \leftarrow \mathsf{P}'_{eq}((v, v', ct_z, w'_i, a, b), (y_i, \perp))$.

$\underline{\mathsf{EvalVerify}(\mathsf{vk}, z, w_i, \pi_i)\text{:}}$

  1. Parse $\mathsf{vk}$ as $(N, v, v_0, \{v_i\}_{i\in[t-1]}, a, \{(b_j)\}_{j\in[\mu]})$ and $w_i$ as $(x_i, w'_i)$.
  2. Compute $v' \leftarrow \prod_{j\in\{0,\ldots,t-1\}} v_j^{x_i^j}$. Let $w \leftarrow \mathsf{H}_0(z)$.
  3. Let $s_z \leftarrow \mathsf{H}_1(z)$ and compute $b \leftarrow \prod_{j\in[\mu]} b_j^{s_{z,j}}$.
  4. Output $\mathsf{V}'_{eq}((v, v', w, w'_i, a, b), \pi_i)$.
```

Fig. 7. The FuncSamp, Eval and EvalVerify algorithms for our one-time traceable threshold VRF OT-P$_2$ with tracer uniqueness. The Trace algorithm is the same as in Figure 6, with the only difference being that the tracer uses u_j to produce valid proofs for partial evaluations for z_j.

uniqueness game. Using this proof and the honest evaluation proof, we can apply the knowledge extractor for the relation $\mathcal{R}'_{eq,z^*}$ to recover the discrete logarithm of the challenge b.

The full proof can be found in Appendix D.5.

Theorem 5. *For every* PPT *adversary $\mathcal{A}$, for every $\mu > (m+1)\log q^*$, there exists a* PPT *adversary $\mathcal{B}$ such that,*

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathcal{B},\mathsf{GroupGen}}(\lambda) \geq \left(\frac{1}{q_H} \cdot \left(1 - \frac{m(q^*)^m}{2^\mu}\right) \cdot \mathsf{Adv}^{\text{t-uniq}}_{\mathcal{A},\mathsf{OT\text{-}P_2}}(\lambda)\right)^2 - 4/N^2 - 1/q'$$

where $q_H = q_H(\lambda)$ is an upper bound on the number of random oracle queries by $\mathcal{A}$, $m = m(n, 1/\epsilon, 1/\delta, \lambda)$ is the number of z values in the tracing key, $q^* = q^*(\lambda)$ is the size of the cyclic group $\mathbb{G}$, and $N = pq$ is an RSA modulus with $p = 2p+1$, $q = 2q'+1$ where p', q', p, q are all prime and $p > q$.

The parameter μ can be set such that the advantage of the adversary is exponentially small. Specifically, we can set $\mu = \lambda + (m+1)\log q^*$. Combining with the analysis in Theorem 3, we get $\mu = \lambda + \frac{33ft\lambda \log q^*}{\epsilon^4\delta^4}$.

5 Removing the One-Time Restriction

In this section we present two transformations that bootstrap a one-time traceable threshold VRF to a many-time traceable threshold VRF. Recall that in a many-time traceable threshold VRF, the

VRF remains traceable, even if the parties constructing the evaluation box E can observe many partial evaluations of the function before deciding on E. Note that the constructions from Sections 3 and 4 are **not** secure when the corrupted parties observe partial evaluations. This is because our tracing procedure inherently relied on the fact that the corrupted parties do not know the x_i-values used as evaluation points to derive honest parties' shares of ek. However, the partial evaluation of party i (with respect to any function input z) includes in particular the values x_i! Hence, after observing a single partial evaluation, the corrupted parties learn x_i, and can frame the ith party as if it took part in the construction of E.

Hence, the basic idea in both of the bootstrapping methods that we will present here, is to instantiate several copies of a one-time traceable VRF. In particular, each party will be associated with several x values: $x_{i,1}, x_{i,2}, \ldots$. Intuitively, each of our methods will use a different mechanism to enforce that: (1) the corrupted parties will have to "embed" all of their x values in the box; while (2) they will not get to observe all of the x values of the honest parties. This discrepancy will allow us to still separate out the corrupted parties from the honest parties during tracing.

5.1 A Many-Time Traceable VRF in the Synchronized Model

Our first transformation is simple and efficient. It works in settings in which time can be divided into epochs, and in each epoch, the VRF is only computed once, on a single input. These settings include, in particular, the use of threshold VRFs for multiparty randomness generation [27,15,18,22]. This modeling, and the subsequent transformation are inspired by the recent work of Fleischhacker, Simkin, and Zhang [25], bootstrapping one-time lattice-based multisignatures to many-time multisignatures.

Synchronized Threshold VRFs. To present the transformation and argue its security, we first need to define threshold VRFs in the synchronized model. The syntax of a synchronized threshold VRF is the same as threshold VRFs as defined in Section 2, but for the following modifications: the function sampling algorithm $\mathsf{FuncSamp}$ takes the number $T \in \mathbb{N}$ of epochs as an additional input; and the algorithms Eval, $\mathsf{EvalVerify}$, $\mathsf{Combine}$, Verify, all take an epoch $j \in [T]$ as an additional input. Our security notions readily extend to this settings:

- Pseudorandomness remains the same as Definition 4, but it should hold for all $T = \mathsf{poly}(\lambda)$ and $j \in [T]$.
- The definition of uniqueness remains as in Definition 2, but should hold separately for every epoch. That is, the adversary wins if it outputs an input z and two distinct evaluations $w \neq w'$ that verify with respect to the same epoch $j \in [T]$. We call this notion *synchronized uniqueness.*
- Traceability: here, we make an important change in definitions. In the synchronized model, an evaluation box E also take in an epoch $j \in [T]$ as input. We say that E is good with respect to epoch $j \in [T]$, if it is good (per Definition 6) when Eval takes j as input. Now, in order to win the traceability game (Fig. 2), an adversary $\mathcal{A}$ must output a box E that is good with respect to an epoch $j \in [T]$ *on which $\mathcal{A}$ had not previously queried its partial evaluation oracle* EvalO. This can be enforced in Line 6 of the security game. We call this traceability guarantee *synchronized traceability.*

The real-world implications of this revised traceability definition can be interpreted as follows. If at some epoch $j^* \in [T]$, a coalition of corrupted parties outputs an evaluation box E then, either:

- E is good with respect to at least one future epoch $j > j^*$, for which partial evaluations have not been published yet. In this case E can be traced back to the corrupted parties.

– E is not good with respect to any future epoch. In this case, E is benign, since it reveals nothing about the values of the VRF that can be of interest in the future. For example, if the VRF is used to sample shared randomness r_j for every epoch j, then the box E reveals nothing about future r_js.

The transformation. Let $\mathsf{TTVRF} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{EvalVerify}, \mathsf{Combine}, \mathsf{Verify}, \mathsf{Trace})$ be a one-time traceable threshold VRF scheme. We will now discuss how to transform it into a many-time traceable scheme in the synchronized model. Let $T = \mathsf{poly}(\lambda)$ be the number of epochs. The idea is to generate key material for T epochs by invoking $\mathsf{FuncSamp}$ and $\mathsf{KeyShareGen}$ independently for each of the T epochs.[3] The result is T verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_T$, and each party $i \in [n]$ holding T partial evaluation keys $\mathsf{ek}_{i,1}, \ldots, \mathsf{ek}_{i,T}$. In epoch $j \in [T]$, the parties use $\{\mathsf{ek}_{i,j}\}_{i\in[n]}$ to compute the evaluation of the function, and verification is done with respect to vk_j. The tracer also holds T tracing keys $\mathsf{tk}_1, \ldots, \mathsf{tk}_T$, one for each epoch.

One caveat of the construction as presented so far, is that the verification key vk is linear in the number T of epochs. However, this is easy to counter using vector commitments. Suppose we have a vector commitment scheme VC (for detailed syntax and security notions, see [36,14]). The new verification key vk' will be a vector commitment com to $(\mathsf{vk}_1, \ldots, \mathsf{vk}_T)$. We include $(\mathsf{vk}_1, \ldots, \mathsf{vk}_T)$ and com as part of the combiner key ck. Then, the evaluation proof for the VRF at epoch j also contains vk_j and an opening proof for the jth entry underlying com, proving that vk_j is indeed the jth verification key.

We denote the resulting construction by $\mathsf{SyncTTVRF}$. The pseudorandomness and traceability of $\mathsf{SyncTTVRF}$ in the synchronized model follow directly from the pseudorandomness and traceability of TTVRF (without synchronization). For uniqueness, we additionally have to rely on the security of the vector commitment scheme in use. Namely, we rely on the assumption that VC satisfies *position binding*: it is infeasible to come up with a commitment com, an index i, and opening proofs of com at index i to two distinct values. The synchronized uniqueness of $\mathsf{SyncVRF}$ is captured by the following theorem.

Theorem 6. *Suppose* TTVRF *satisfies uniqueness and* VC *satisfies position binding. Then* $\mathsf{SyncTTVRF}$ *satisfies synchronized uniqueness.*

Proof. The uniqueness of the VRF at each epoch is guaranteed by uniqueness of the underlying one-time traceable threshold VRF TTVRF and the position binding of the vector commitment. To see why, suppose that an adversary manages to break uniqueness with respect to epoch $j \in [T]$, this means that it outputs an input z, two *distinct* evaluations w, w', and corresponding *accepting* evaluation proofs $\pi = (\mathsf{vk}_j, \pi_{\mathsf{VC},j}, \pi_{\mathsf{TTVRF}})$ and $\pi' = (\mathsf{vk}'_j, \pi'_{\mathsf{VC},j}, \pi'_{\mathsf{TTVRF}})$. In these proofs, vk_j and vk'_j are the supposed TTVRF verification keys for epoch j; $\pi_{\mathsf{VC},j}$ and $\pi'_{\mathsf{VC},j}$ are the vector commitment opening proofs, proving that respectively that vk_j and vk'_j are the opening to the jth entry of com; and π_{TTVRF} and π'_{TTVRF} are the evaluation proofs of TTVRF for w and w', respectively. Let $\mathbf{ek}^*_j := (\mathsf{ek}^*_{1,j}, \ldots, \mathsf{ek}^*_{n,j})$ be the vector of real evaluation keys for epoch j, and let vk^*_j sampled at key generation time.

Consider two cases:

– If $\mathsf{vk}^*_j \neq \mathsf{vk}_j$ or $\mathsf{vk}^*_j \neq \mathsf{vk}'_j$, then assume without loss of generality that $\mathsf{vk}^*_j \neq \mathsf{vk}_j$. In this case, the adversary can be used to break the position binding of the vector commitment scheme.

[3] Once T epochs are exhausted, new keys need to be sampled.

This is because π is an accepting evaluation proofs, which, in particular, implies that $\pi_{\mathsf{VC},j}$ is an accepting opening proof for vk_j with respect to com. But, vk_j^* was the true jth coordinate of the vector to which com is a commitment to. Hence, the correctness of the vector commitment scheme stipulates that it is possible to compute a proof $\pi^*_{\mathsf{VC},j}$ proving that this is indeed the case. It follows that the tuple $(\mathsf{com}, j, \mathsf{vk}_j, \pi_{\mathsf{VC},j}, \mathsf{vk}_j^*, \pi^*_{\mathsf{VC},j})$ breaks the position binding of the vector commitment.

– If $\mathsf{vk}_j^* = \mathsf{vk}_j = \mathsf{vk}'_j$, then the adversary can be used to break the uniqueness of TTVRF. Let w^* be the real value of the VRF on input z at epoch j, as induced by $\mathbf{ek}_j$, and let π^* be the corresponding TTVRF evaluation proof. Since $w \neq w'$, it must hold that $w^* \neq w$ or $w^* \neq w'$. Assume without loss of generality that $w^* \neq w$. Then, since $w^* \neq w$ and π^* and π_{TTVRF} and both accepting TTVRF proofs with respect to $\mathsf{vk}_j^* = \mathsf{vk}_j$, the tuple $(z, (w^*, \pi^*), (w, \pi_{\mathsf{TTVRF}}))$ breaks the uniqueness of TTVRF.

5.2 The Exponential Method

Our second transformation does not assume the existence of epochs, and does not rely on a synchronized model. However, as a result, in the many-time traceable threshold VRF that we end up with, the tracing algorithm can only trace evaluation boxes E that are good with respect to almost all inputs z. That is, E is $(n, t, \mathbf{ek}, \mathsf{ck}, \mathsf{vk}, \mathcal{C}, \mathcal{EK}^*, \epsilon, \delta)$-good with $\delta > 1 - \mathsf{negl}(\lambda)$.

The idea is to initialize $\zeta \in \mathbb{N}$ base VRFs $\mathsf{ek}_1, \dots, \mathsf{ek}_\zeta$ sets of TTVRF keys, by invoking $\mathsf{FuncSamp}$ and $\mathsf{KeyShareGen}$ ζ times independently. For reasons that will become apparent in a second, we require that $\zeta = \omega(\log \lambda)$. The result is ζ verification keys $\mathsf{vk}_1, \dots, \mathsf{vk}_\zeta$, and each party $i \in [n]$ holding T partial evaluation keys $\mathsf{ek}_{i,1}, \dots, \mathsf{ek}_{i,\zeta}$. The tracing key is the concatenation of the ζ tracing keys $\mathsf{tk}_1, \dots, \mathsf{tk}_\zeta$. To decide which of the ζ base functions to use, we rely on hash function H_{map} mapping function inputs to indices in $\{1, \dots, \zeta\}$. To compute the output of the function on input z, we compute $j \leftarrow H_{\mathsf{map}}(z)$ and compute $f_{\mathsf{ek}_j}(z)$. We will think of H_{map} as a skewed random oracle with the following distribution: for any $z \in \mathcal{Z}$, and for every $j \in [\zeta - 1]$, we require that $\Pr[H_{\mathsf{map}}(z) = j] = 2^{-j}$. So that the probabilities sum to 1, we require that $\Pr[H_{\mathsf{map}}(z) = \zeta] = 2^{-\zeta+1}$.[4]

The crucial observation is if an adversary $\mathcal{A}$ issues at most a polynomial number $Q = \mathsf{poly}(\lambda)$ of partial evaluation queries with respect to inputs $z_1, \dots, z_Q$, then with overwhelming probability there is an index $j = O(\log \lambda)$ such that:

– $H_{\mathsf{map}}(z_i) \neq j$ for all $i \in [Q]$. In particular, this means that $\mathcal{A}$ does not observe any partial evaluations of the jth one-time traceable function f_{ek_j}.

– At the same time, since, by assumption, E is good for $1 - \mathsf{negl}(\lambda)$ fraction of the inputs $z \in \mathcal{Z}$, it should, in particular, be good for $1 - \mathsf{negl}(\lambda)$ fraction of the inputs in $\mathcal{Z}_j := \{H_{\mathsf{map}}(z) = j\}_{z \in \mathcal{Z}}$. This is because $j = O(\log \lambda)$ and thus (with overwhelming probability), a non-negligible fraction of the inputs in $\mathcal{Z}$ is mapped to j by H_{map}. Since E is good for $\mathcal{Z}_j$ and $\mathcal{Z}_j$ is sufficiently dense within $\mathcal{Z}$, it follows that E is good for a non-negligible fraction of the inputs in $\mathcal{Z}$.

Taking these two points together, we get that there exists an index $j \in [\zeta]$ such that E is a good box with respect to the jth function f_{ek_j}, and $\mathcal{A}$ has not seen partial evaluations of f_{ek_j} before

[4] Such a function is easy to construct given a function H whose output is uniform over $\{0,1\}^\zeta$. Given such a function, define the function H_{map} as follows. On input z, $H_{\mathsf{map}}(z)$ first computes $\sigma \leftarrow H(z) \in \{0,1\}^\zeta$. The output of $H_{\mathsf{map}}(z)$ is then the index of the first 1 in σ. If $H(z) = 0^\zeta$, then $H_{\mathsf{map}}(z) = \zeta$.

outputting E. Hence, we can rely on the one-time traceability of TTVRF to trace E back to the corrupted parties.

Backdooring H_{map}. The diligent reader might have noticed a loophole in the forgoing argument. The issue is this. Our construction of a one-time traceable threshold VRF from Sections 3 and 4 gives the tracer, as part of the tracing, values that are tied to specific function inputs (z values), sampled at key generation. Alas, if we sample the z values included in each $\mathsf{tk}_1, \dots, \mathsf{tk}_\zeta$ and the hash function H_{map} independently, then it is very likely that the following bad event occurs: for all the z values $z_{j,1}, \dots, z_{j,\kappa}$ included as part of tk_j it holds that $H_{\mathsf{map}}(z_{j,1}) \neq j, \dots, H_{\mathsf{map}}(z_{j,\kappa}) \neq j$. In contrast, our tracing algorithm requires that for every such $z_{j,m}$, we can get E's partial evaluation for f_{ek_j} on $z_{j,m}$. However, since $H_{\mathsf{map}}(z_{j,m}) \neq j$, querying the box E on $z_{j,m}$ will result in E's evaluation with respect to a different function $f_{\mathsf{ek}_{j'}}$.

To remedy this situation, we do not choose H_{map} independently from the z values in the tracing keys. Rather, we choose it in the following manner. Let H'_{map} be a hash function, modeled as a random oracle. We require that H'_{map} is skewed and exhibits an exponential decay as discussed above. We sample the tracing keys $\mathsf{tk}_1, \dots, \mathsf{tk}_\zeta$ independently, according to the key generation algorithms of TTVRF. For every $j \in [\zeta]$, let $z_{j,1}, \dots, z_{j,\kappa}$ be the function inputs that are included in tk_j. Suppose that the set of all $\zeta \cdot \kappa$ function inputs in tk is pairwise distinct (this is indeed the case with overwhelming probability in our construction). Then, we sample a function g from a family of $\zeta \cdot \kappa$-wise independent hash functions, subject to the fact $g \circ H'_{\mathsf{map}}(z_{j,m}) = j$ for any $j \in [\zeta]$ and $m \in [\kappa]$. Finally, we set $H_{\mathsf{map}} := g \circ H'_{\mathsf{map}}$.

References

1. D. Aggarwal and U. Maurer. Breaking RSA generically is equivalent to factoring. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 36–53, Cologne, Germany, Apr. 26–30, 2009. Springer, Berlin, Heidelberg, Germany.
2. M. Bellare, E. C. Crites, C. Komlo, M. Maller, S. Tessaro, and C. Zhu. Better than advertised security for non-interactive threshold signatures. In Y. Dodis and T. Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550, Santa Barbara, CA, USA, Aug. 15–18, 2022. Springer, Cham, Switzerland.
3. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *CCS'06*, pages 390–399. ACM, 2006.
4. M. Bellare, S. Tessaro, and C. Zhu. Stronger security for non-interactive threshold signatures: BLS and FROST. Cryptology ePrint Archive, Report 2022/833, 2022.
5. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography – PKC 2003*, page 31–46, 2003.
6. D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *ASIACRYPT'18*, pages 435–463, 2018.
7. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
8. D. Boneh, I. Haitner, and Y. Lindell. Exponent-VRFs and their applications. Cryptology ePrint Archive, Paper 2024/397, 2024.
9. D. Boneh, A. Partap, and L. Rotem. Accountability for misbehavior in threshold decryption via threshold traitor tracing. In L. Reyzin and D. Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 317–351, Santa Barbara, CA, USA, Aug. 18–22, 2024. Springer, Cham, Switzerland.
10. D. Boneh, A. Partap, and L. Rotem. Traceable secret sharing: Strong security and efficient constructions. In L. Reyzin and D. Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 221–256, Santa Barbara, CA, USA, Aug. 18–22, 2024. Springer, Cham, Switzerland.
11. D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Heidelberg, Germany.

12. D. Boneh and V. Shoup. A graduate course in applied cryptography, 2023. Cryptobook.
13. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499, Santa Barbara, CA, USA, Aug. 17–21, 2014. Springer, Berlin, Heidelberg, Germany.
14. D. Catalano and D. Fiore. Vector commitments and their applications. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72, Nara, Japan, Feb. 26 – Mar. 1, 2013. Springer, Berlin, Heidelberg, Germany.
15. Chainlink vrf: On-chain verifiable randomness. link.
16. M. Chase, M. Orrù, T. Perrin, and G. Zaverucha. Proofs of discrete logarithm equality across groups. Cryptology ePrint Archive, Report 2022/1593, 2022.
17. Y. Chen, V. Vaikuntanathan, B. Waters, H. Wee, and D. Wichs. Traitor-tracing from LWE made simple and attribute-based. In A. Beimel and S. Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 341–369, Panaji, India, Nov. 11–14, 2018. Springer, Cham, Switzerland.
18. K. Choi, A. Manoj, and J. Bonneau. SoK: Distributed randomness beacons. In *2023 IEEE Symposium on Security and Privacy*, pages 75–92, San Francisco, CA, USA, May 21–25, 2023. IEEE Computer Society Press.
19. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270, Santa Barbara, CA, USA, Aug. 21–25, 1994. Springer, Berlin, Heidelberg, Germany.
20. I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142, Queenstown, New Zealand, Dec. 1–5, 2002. Springer, Berlin, Heidelberg, Germany.
21. P. Das, S. Faust, and J. Loss. A formal treatment of deterministic wallets. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 651–668, London, UK, Nov. 11–15, 2019. ACM Press.
22. S. Das, B. Pinkas, A. Tomescu, and Z. Xiang. Distributed randomness using weighted VRFs. Cryptology ePrint Archive, Report 2024/198, 2024.
23. Y. Dodis. Efficient construction of (distributed) verifiable random functions. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 1–17, Miami, FL, USA, Jan. 6–8, 2003. Springer, Berlin, Heidelberg, Germany.
24. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431, Les Diablerets, Switzerland, Jan. 23–26, 2005. Springer, Berlin, Heidelberg, Germany.
25. N. Fleischhacker, M. Simkin, and Z. Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, *ACM CCS 2022*, pages 1109–1123, Los Angeles, CA, USA, Nov. 7–11, 2022. ACM Press.
26. P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In Y. Frankel, editor, *FC 2000*, volume 1962 of *LNCS*, pages 90–104, Anguilla, British West Indies, Feb. 20–24, 2001. Springer, Berlin, Heidelberg, Germany.
27. D. Galindo, J. Liu, M. Ordean, and J.-M. Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 88–102, 2021.
28. S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 2010*, pages 121–130, Chicago, Illinois, USA, Oct. 4–8, 2010. ACM Press.
29. S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 228–245, Santa Barbara, CA, USA, Aug. 16–20, 1993. Springer, Berlin, Heidelberg, Germany.
30. J. Gong, J. Luo, and H. Wee. Traitor tracing with $N^{1/3}$-size ciphertexts and $O(1)$-size keys from k-Lin. In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 637–668, Lyon, France, Apr. 23–27, 2023. Springer, Cham, Switzerland.
31. R. Goyal, V. Koppula, and B. Waters. Collusion resistant traitor tracing from learning with errors. In I. Diakonikolas, D. Kempe, and M. Henzinger, editors, *50th ACM STOC*, pages 660–670, Los Angeles, CA, USA, June 25–29, 2018. ACM Press.
32. V. Goyal, Y. Song, and A. Srinivasan. Traceable secret sharing and applications. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 718–747, Virtual Event, Aug. 16–20, 2021. Springer, Cham, Switzerland.
33. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *39th FOCS*, pages 28–39, Palo Alto, CA, USA, Nov. 8–11, 1998. IEEE Computer Society Press.

34. C. Hazay, G. L. Mikkelsen, T. Rabin, and T. Toft. Efficient rsa key generation and threshold paillier in the two-party setting. In O. Dunkelman, editor, *Topics in Cryptology – CT-RSA 2012*, pages 313–331, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
35. B. Libert, M. Joye, and M. Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In M. M. Halldórsson and S. Dolev, editors, *33rd ACM PODC*, pages 303–312, Paris, France, July 15–18, 2014. ACM.
36. B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517, Zurich, Switzerland, Feb. 9–11, 2010. Springer, Berlin, Heidelberg, Germany.
37. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612, Santa Barbara, CA, USA, Aug. 18–22, 2002. Springer, Berlin, Heidelberg, Germany.
38. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *CCS'01*, pages 245–254. ACM, 2001.
39. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130, New York, NY, USA, Oct. 17–19, 1999. IEEE Computer Society Press.
40. J. Nick, T. Ruffing, and Y. Seurin. Musig2: Simple two-round schnorr multi-signatures. In *Advances in Cryptology – CRYPTO' 21*, pages 189–221, 2021.
41. J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020*, pages 1717–1731, Virtual Event, USA, Nov. 9–13, 2020. ACM Press.
42. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Berlin, Heidelberg, Germany.
43. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.
44. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.
45. V. Shoup. Practical threshold signatures. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220, Bruges, Belgium, May 14–18, 2000. Springer, Berlin, Heidelberg, Germany.
46. M. Sudan. Maximum likelihood decoding of reed solomon codes. In *37th FOCS*, pages 164–172, Burlington, Vermont, Oct. 14–16, 1996. IEEE Computer Society Press.
47. B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Heidelberg, Germany.
48. H. Wee. Functional encryption for quadratic functions from k-lin, revisited. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 210–228, Durham, NC, USA, Nov. 16–19, 2020. Springer, Cham, Switzerland.
49. M. Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 652–682, Santa Barbara, CA, USA, Aug. 17–21, 2020. Springer, Cham, Switzerland.

A Evaluation Share Proofs

A.1 A Chaum-Pedersen style proof for $\mathcal{R}_{eq}$

Figure 8 presents the proof system for the relation $\mathcal{R}_{eq}$. It uses rejection sampling to reduce the size of the transcript, but it may cause the prover to abort. Nevertheless, the Fiat-Shamir transform can be applied, wherein the prover tries to run the protocol multiple times with different randomness until it succeeds.

Theorem 7 below proves completeness, honest-verifier zero-knowledge and soundness of the proof system. We consider a notion called no-abort honest verifier zero-knowledge, where the simulator returns a valid transcript or $\perp$, and indistinguishability needs to hold only for non aborted transcripts. While this is weaker than HVZK, it is still sufficient to achieve HVZK with the Fiat-Shamir

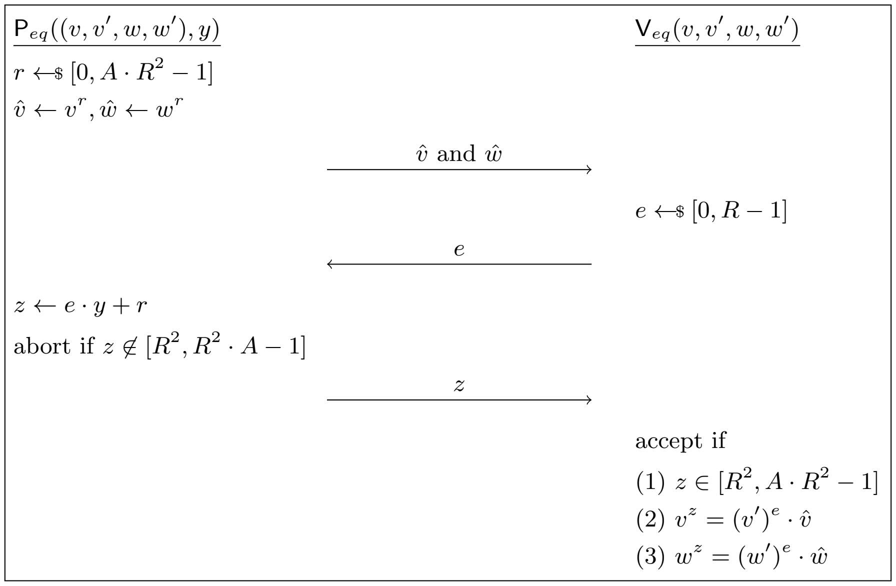

$\mathsf{P}_{eq}((v,v',w,w'),y)$
$\mathsf{V}_{eq}(v,v',w,w')$
$r \leftarrow\$ [0, A\cdot R^2-1]$
$\hat{v} \leftarrow v^r, \hat{w} \leftarrow w^r$
$\hat{v}$ and $\hat{w}$
$e \leftarrow\$ [0, R-1]$
e
$z \leftarrow e\cdot y + r$
abort if $z \notin [R^2, R^2\cdot A-1]$
z
accept if
(1) $z \in [R^2, A\cdot R^2-1]$
(2) $v^z = (v')^e \cdot \hat{v}$
(3) $w^z = (w')^e \cdot \hat{w}$

Fig. 8. A proof system for the relation $\mathcal{R}_{eq}$. The elements in the statement v, v', w, w' are all in the group of quadratic residues $\mathcal{QR}_{N^2}$, of order $Np'q'$, where $N = pq$, $p = 2p'+1, q = 2q'+1$ and p, q, p', q' are all prime. R is an upper bound on the value of y. For our VRF constructions, we set $R = N^2/4$. A is a parameter that we set to 256 to keep the abort probability low.

transform. Additionally, for proving soundness, we make the assumption that v is a generator of the set $\mathcal{QR}_{N^2}$, but note that a random element in $\mathcal{QR}_{N^2}$ is indeed a generator with overwhelming probability.

Theorem 7. *For every $A > 0$, the proof system $(\mathsf{P}_{eq}, \mathsf{V}_{eq})$ in Fig. 8 is a na-HVZK for the relation $\mathcal{R}_{eq}$.*

Proof. **Completeness.** In case the prover does not abort, correctness is straightforward:

$$v^z = v^{e\cdot y + r} = (v^y)^e \cdot v^r = (v')^e \cdot \hat{v}$$

and similarly for w, w'. Next, by [16][Lemma 2], the abort probability is exactly $1/A$. We set A to be 256 so that aborting is infrequent.

no-abort Honest-verifier Zero-knowledge (na-HVZK).

We construct a simulator $\mathsf{Sim}_{eq}(v, v', w, w') \to ((\hat{v}, \hat{w}), e, z)$ that matches the distribution of an accepting transcript between the honest prover and honest verifier, when the prover does not abort. The analysis in [16][Lemma 2] implies that when the honest prover does not abort, the quantity z is uniform in the set $[R^2, R^2 \cdot A - 1]$. Then, $\mathsf{Sim}_{eq}(v, v', w, w')$ works as follows:

```
1 :  $e \leftarrow\$ [0, R-1], z \leftarrow\$ [R^2, R^2\cdot A-1]$
2 :  $\hat{v} \leftarrow \frac{v^z}{(v')^e}$
3 :  $\hat{w} \leftarrow \frac{w^z}{(w')^e}$
```

Since both e and z are uniformly random in an honest transcript, the simulator above generates exactly the required distribution.

Soundness. Let (v, v', w, w') be an instance where the relation $\mathcal{R}_{eq}$ does not hold. Let $(\hat{v}, \hat{w}, e, z)$ be an accepting transcript.

Note that we assumed that v is a generator of $\mathcal{QR}_{N^2}$. This implies that:

$$v' = v^\gamma, w = v^\alpha, w' = v^\beta, \hat{v} = v^\delta, \hat{w} = v^\eta$$

for some $\gamma, \alpha, \beta, \delta, \eta \in [0, pqp'q' - 1]$ where $N = pq$, $p = 2p' + 1$, $q = 2q' + 1$, i.e. the order of the group $\mathcal{QR}_{N^2}$ is $pqp'q'$.

Since this is an accepting transcript, we have that:

$$z = e \cdot \gamma + \delta \bmod pqp'q'$$

$$z \cdot \alpha = e \cdot \beta + \eta \bmod pqp'q'$$

Multiplying the first equation by α and then subtracting from the second, we get,

$$e \cdot (\beta - \gamma \cdot \alpha) = \delta\alpha - \eta \bmod pqp'q'$$

If $\beta = \gamma \cdot \alpha \bmod pqp'q'$, then the instance is actually in $\mathcal{R}_{eq}$, which is a contradiction. Otherwise, $\beta - \gamma \cdot \alpha$ must be non zero modulo one of the primes in the set $\{p, q, p', q'\}$. Then, the above equation uniquely determines e modulo one of these primes. But since e is chosen uniformly randomly by the verifier, independent of the instance and the first message, the probability that the above equation holds for a random e is atmost $\max(1/p', 1/q')$. Without loss of generality, let us assume that $p > q$ and hence $p' > q'$. So the soundness error is bounded by $1/q'$. □

A.2 A Chaum-Pedersen style proof for $\mathcal{R}'_{eq}$

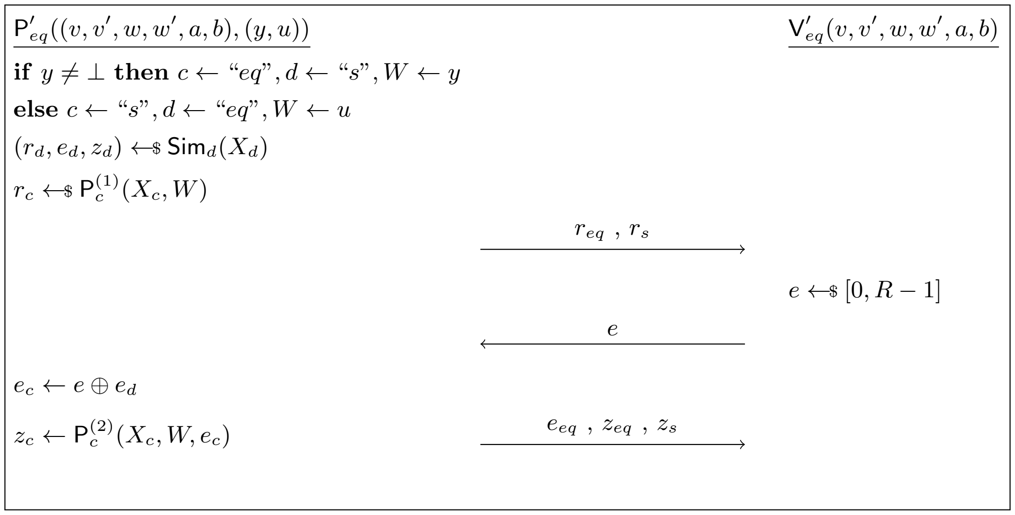
We construct a proof system for the relation $\mathcal{R}'_{eq}$ by running an OR proof [12][§ 19.7.2] over the proof system for $\mathcal{R}_{eq}$ and Schnorr's sigma protocol [12][Fig.19.1] for proof of knowledge of discrete log in the group $\mathbb{G}$. Let $(\mathsf{P}_s(u), \mathsf{V}_s(a, b))$ denote the Schnorr proof system. Let $\mathsf{Sim}_s(\mathbb{G}, a, b)$ be the simulator that produces transcripts from the same distribution as an honest prover, and let $\mathcal{E}_s$ be the extractor which can extract the witness from two transcripts with different challenges. Let $X_{eq} \leftarrow (v, v', w, w')$ and $X_s \leftarrow (G, H)$ denote the statements for the two proof systems. We use the same challenge space $[0, R-1]$ for both proof systems (this is fine due to our assumption that the order of $\mathbb{G}$ q is greater than N^2 and $R = N^2/4$). Additionally, we use $\mathsf{P}^{(1)}$ and $\mathsf{P}^{(2)}$ to denote the first and the second steps of the prover for any proof system.

Similar to Section A, we prove no-abort honest-verifier zero-knowledge.

Theorem 8. *For every $A > 0$, the proof system $(\mathsf{P}'_{eq}, \mathsf{V}'_{eq})$ in Fig. 9 is a na-HVZK for the relation $\mathcal{R}'_{eq}$.*

Proof. **Completeness.** This is implied by the completeness of $(\mathsf{P}_{eq}, \mathsf{V}_{eq})$ and $(\mathsf{P}_s, \mathsf{V}_s)$ and the correctness of the simulators.

na-HVZK. We construct a Simulator $\mathsf{Sim}'_{eq}(v, v', w, w', a, b)$ that matches the distribution of an accepting transcript, when the prover does not abort. It works as follows:



$\underline{\mathsf{P}'_{eq}((v, v', w, w', a, b), (y, u))}$ $\underline{\mathsf{V}'_{eq}(v, v', w, w', a, b)}$

if $y \neq \perp$ **then** $c \leftarrow \text{“}eq\text{”}, d \leftarrow \text{“}s\text{”}, W \leftarrow y$

else $c \leftarrow \text{“}s\text{”}, d \leftarrow \text{“}eq\text{”}, W \leftarrow u$

$(r_d, e_d, z_d) \leftarrow\!\!\$\ \mathsf{Sim}_d(X_d)$

$r_c \leftarrow\!\!\$\ \mathsf{P}_c^{(1)}(X_c, W)$

$\xrightarrow{r_{eq}\ ,\ r_s}$

$e \leftarrow\!\!\$\ [0, R-1]$

$\xleftarrow{e}$

$e_c \leftarrow e \oplus e_d$

$z_c \leftarrow \mathsf{P}_c^{(2)}(X_c, W, e_c)$ $\xrightarrow{e_{eq}\ ,\ z_{eq}\ ,\ z_s}$

Fig. 9. A proof system for the relation $\mathcal{R}'_{eq}$.

```
1 :  $(t_{eq}, e_{eq}, z_{eq}) \leftarrow\!\!\$\ \mathsf{Sim}_{eq}(v, v', w, w')$
2 :  $(t_s, e_s, z_s) \leftarrow\!\!\$\ \mathsf{Sim}_s(a, b)$
3 :  $e \leftarrow e_{eq} \oplus e_s$
4 :  Output $(t_{eq}, t_s, e, (e_{eq}, z_{eq}, z_s))$
```

Since both Sim_{eq} and Sim_s output the same distribution as the transcript of an honest prover when it does not abort, Sim'_{eq} also outputs the required distribution.

Special Soundness. The following lemma proves soundness.

Lemma 2. *There exists an extractor $\mathcal{E}$ such that, for every* PPT *adversary $\mathcal{A}$ that breaks soundness of $(\mathsf{P}'_{eq}, \mathsf{V}'_{eq})$,*

$$\Pr[a^u = b : u \leftarrow\!\!\$\ \mathcal{E}_{\mathcal{A}}(v, v', w, w', a, b)] \geq \epsilon^2 - 1/R - 1/q'$$

where ϵ is the probability with which $\mathcal{A}$ outputs an accepting transcript for the instance $(v, v', w, w', a, b) \notin \mathcal{R}'_{eq}$, $R = N^2/4$ is an upper bound on the value of y, and $N = pq$ where $p > q$, $q = 2q' + 1$ and $p = 2p' + 1$.

Proof. We construct $\mathcal{E}_{\mathcal{A}}$ as follows:

1. Run $((r_{eq}, r_s), st) \leftarrow\!\!\$\ \mathcal{A}$ to get the first message.
2. Sample $e \leftarrow\!\!\$\ [0, R-1]$. Run $\mathcal{A}$ with inputs e and st to get (e_{eq}, z_{eq}, z_s).
3. Sample $e' \leftarrow\!\!\$\ [0, R-1]$. Rewind $\mathcal{A}$ to the point right after it outputs the first message, and run it again with inputs e' and st to get (e'_{eq}, z'_{eq}, z'_s).
4. Abort if $\mathsf{V}'_{eq}((r_{eq}, r_s), e, (e_{eq}, z_{eq}, z_s)) = 0$ or if $\mathsf{V}'_{eq}((r_{eq}, r_s), e', (e'_{eq}, z'_{eq}, z'_s)) = 0$ or if $e = e'$.
5. Otherwise, if $(e \oplus e_{eq}) = (e' \oplus e'_{eq})$ then abort, else, run $\mathcal{E}_s$ with inputs $(r_s, e \oplus e_{eq}, z_s)$ and $(r_s, e' \oplus e'_{eq}, z'_s)$.

Let ϵ denote the probability with which $\mathcal{A}$ outputs an accepting transcript for the invalid instance (v, v', w, w'). Let E_f denote the event that the extractor does not abort on Step 4 above. In other words, we get two accepting transcripts with different challenges $e \neq e'$. Then, by the forking lemma, we have that

$$\Pr[\mathsf{E}_f] \geq \epsilon^2 - 1/R.$$

By total probability, we have that,

$$\Pr[\mathsf{E}_f] = \Pr[\mathsf{E}_f \wedge ((e \oplus e_{eq}) = (e' \oplus e'_{eq}))] + \Pr[\mathsf{E}_f \wedge ((e \oplus e_{eq}) \neq (e' \oplus e'_{eq}))]$$

In the case $(e \oplus e_{eq}) = (e' \oplus e'_{eq})$, observe that e'_{eq} must be equal to $e \oplus e' \oplus e_{eq}$. Since e' is chosen uniformly randomly, e'_{eq} is also uniformly random in $[0, R-1]$. By an analysis similar to that in the proof of Thm 7, since the instance is not in $\mathcal{R}'_{eq}$, we have that e'_{eq} is fixed modulo one of the primes in $\{p, q, p', q'\}$. The probability that a uniform random e'_{eq} satisfies that is at most $\max(1/p', 1/q')$. In other words,

$$\Pr[\mathsf{E}_f \wedge ((e \oplus e_{eq}) = (e' \oplus e'_{eq}))] \leq 1/q'$$

where we assume that $p' > q'$ without loss of generality.

Lastly, in the event $\mathsf{E}_f \wedge ((e \oplus e_{eq}) \neq (e' \oplus e'_{eq}))$, the extractor is able to successfully extract a valid witness for the instance (a, b) by running $\mathcal{E}_s$. This proves the lemma.

□

B A Simple and Generic Tracing Algorithm

In this section we show how to support tracing for any threshold VRF scheme. More precisely, for a given threshold VRF scheme we describe a simple tracing algorithm that can trace a *perfect* evaluation box. This algorithm is adapted from [10, App. A]. The main drawback of this tracing scheme, compared to the tracing constructions in the main body, is that this scheme can only trace a perfect box and requires storing several VRF evaluations in the tracing key. Hence, the tracer has pre-knowledge of the VRF evaluation at a number of points. This is undesirable if the VRF needs to be unpredictable at all points that have not yet been evaluated by the quorum. In particular, this traceable threshold VRF scheme does not have tracer pseudorandomness.

The tracing scheme. The basic idea is as follows. Suppose for now that the evaluation box E has $f = t - 1$ evaluation keys $\{\mathsf{ek}_{i_j}\}_{j \in [t-1]}$ hardcoded in it, and it takes as input the evaluation from one additional key ek'. The box is perfect meaning that it always produces a correct output when given a valid input. More precisely, the following condition

$$E\big(z, (w', \pi')\big) = w \quad \text{where} \quad w = f_{\mathsf{ek}}(z)$$

always holds whenever $\mathsf{Eval}(\mathsf{ek}', z) = (w', \pi')$ and ek' is a valid key share that is not one of the keys $\{\mathsf{ek}_{i_j}\}_{j \in [t-1]}$ in the box. The box E outputs $\perp$ when it is given fewer than t distinct shares as input, namely when (w', π') is derived from a key share ek' in the set of keys $\{\mathsf{ek}_{i_j}\}_{j \in [t-1]}$ that the box already has.

The tracing key, output by the setup algorithm, will contain some random z^* in $\mathcal{Z}_\lambda$ along with all the evaluation shares of the VRF at this z^*. In particular, the tracing key will contain z^* and $(w_i, \pi_i) \leftarrow \mathsf{Eval}(\mathsf{ek}_i, z^*)$ for all $i \in [n]$. To trace a box E, the tracer will run E on input $(z^*, (w_i, \pi_i))$ and get back some output w'_i or $\perp$ for every $i \in [n]$. Observe that if party i is honest, then a perfect box must output a valid output, namely $w'_i = f_{\mathsf{ek}}(z^*)$. On the other hand, if i is corrupt, namely $i \in \{i_j\}_{j \in [t-1]}$, then the box cannot output a valid VRF value at the point z^* because it does not have enough evaluation shares. Therefore, the tracer exonerates all the parties i for which the output of the box is valid, and blames all the remaining parties.

We generalize this strategy to any number of corrupted parties $1 \le f < t-1$ as follows. The setup algorithm generate $t-f-1$ dummy key shares $\{\mathsf{ek}'_j\}_{j\in[t-f-1]}$ by running the $\mathsf{KeyShareGen}$ algorithm $t-f-1$ times. Now for each $i \in [n]$ the tracer runs the evaluation box on input

$$E\big(z^*,\ \{\mathsf{Eval}(\mathsf{ek}'_j, z^*)\}_{j\in[t-f-1]},\ \mathsf{Eval}(\mathsf{ek}_i, z^*)\big)$$

As before, the tracer exonerates party i if the box outputs a valid evaluation.

Figure 10 presents the tracing scheme, called $\mathsf{TTVRF}_\mathsf{F}$, which is derived from a threshold VRF F. Since this tracing procedure is generic, it can be used to add traceability to any η-correct threshold VRF scheme F, as in Definition 1, as long as η is super-poly. Correctness and pseudorandomness of $\mathsf{TTVRF}_\mathsf{F}$ follow directly from that of the underlying VRF scheme F. Theorem 9 proves traceability of $\mathsf{TTVRF}_\mathsf{F}$.

```
$\underline{\mathsf{FuncSamp}'(1^\lambda, n, t, 1^{1/\epsilon}, 1^{1/\delta}) \to (\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho)}$:
  1. Run $(\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \rho) \leftarrow\$ \mathsf{F.FuncSamp}(1^\lambda, n, t))$.   // Setup of the underlying VRF
  2. Sample $z^* \leftarrow\$ \mathcal{Z}_\lambda$.
  3. for $j \in [t-1]$:
     (a) $\mathsf{ek}'_j \leftarrow \mathsf{F.KeyShareGen}(1^\lambda, \mathsf{ek}, n, t, \rho)$   // dummy key shares
     (b) $(w'_j, \pi'_j) \leftarrow \mathsf{F.Eval}(\mathsf{ek}'_j, z^*)$
  4. Set $\mathsf{tk}^* \leftarrow \big(z^*,\ \{(w'_j, \pi'_j)\}_{j\in[t-1]}\big)$   // $z^*$ and all $t-1$ dummy evals
  5. Output $(\mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \mathsf{tk}^*, \rho)$

$\underline{\mathsf{KeyShareGen}'(1^\lambda, \mathsf{ek}, \mathsf{tk}^*, n, t, \rho) \to (\mathsf{ek}_i, \mathsf{tk}_i)}$:
  1. $\mathsf{ek}_i \leftarrow \mathsf{F.KeyShareGen}(1^\lambda, \mathsf{ek}, n, t, \rho)$   // ShareGen of the underlying VRF
  2. $\mathsf{tk}_i \leftarrow \mathsf{F.Eval}(\mathsf{ek}_i, z^*)$
  3. Output $(\mathsf{ek}_i, \mathsf{tk}_i)$.   // the overall tracing key $\mathsf{tk}$ is $\mathsf{tk} \leftarrow (\mathsf{tk}^*, \mathsf{tk}_1, \dots, \mathsf{tk}_n)$.

$\underline{\mathsf{Trace}^E(\mathsf{tk}, f) \to \mathcal{I} \subseteq [n]}$:   // $E$ takes as input $(z^*, (w_1, \pi_1), \dots, (w_{t-f}, \pi_{t-f}))$
  1. Parse $\mathsf{tk}$ as $\Big(z^*,\ \{(w'_j, \pi'_j)\}_{j\in[t-1]},\ \{(w_i, \pi_i)\}_{i\in[n]}\Big)$. Let $\mathcal{I} \leftarrow [n]$.
  2. Let $(w^*, \pi^*) = \mathsf{F.Combine}(\{(w_i, \pi_i)\}_{i\in[t]})$.
  3. For $\ell = 1, \dots, n$:
     (a) Query the box $E$ as $w \leftarrow\$ E\big(z^*,\ \{(w'_j, \pi'_j)\}_{j\in[t-f-1]},\ (w_\ell, \pi_\ell)\big)$.
     (b) If $w = w^*$, then set $\mathcal{I} \leftarrow \mathcal{I} \setminus \{\ell\}$.
  4. Output $\mathcal{I}$
```

Fig. 10. The derived traceable threshold VRF scheme $\mathsf{TTVRF}_\mathsf{F}$ from a threshold VRF $\mathsf{F} = (\mathsf{FuncSamp}, \mathsf{KeyShareGen}, \mathsf{Eval}, \mathsf{EvalVerify}, \mathsf{Combine}, \mathsf{Verify})$.

Theorem 9. *Let F be a (standard, i.e., no tracing) threshold VRF scheme that is η-correct (as in Definition 1) for some super-polynomial function $\eta(\lambda)$. Further assume that the domain $\mathcal{Z}_\lambda$ has super-polynomial size in λ. Then for every adversary $\mathcal{A}$ the derived threshold VRF scheme $\mathsf{TTVRF}_\mathsf{F}$ in Figure 10 satisfies*

$$\mathsf{Adv}^{\text{trace-1}}_{\mathcal{A},\mathsf{TTVRF}_\mathsf{F},1,1}(\lambda) < \mathsf{negl}(\lambda).$$

Proof sketch. The proof follows from the fact that the box E must be perfect. First, we argue that the adversary can compute the evaluation of the VRF at the tracing point z^* with at most negligible probability. This follows from the super-polynomial size of $\mathcal{Z}_\lambda$. Since the adversary can guess z^* with at most negligible probability, it cannot query the VRF at z^*. Therefore, if it could predict the VRF output at z^* with non-negligible probability, then it could also be used to break the pseudorandomness of F.

It follows that if E is given fewer than $t-f$ evaluation shares from honest parties, then it cannot output the VRF value at z^*. Conversely, because E is perfect, whenever it is given $t-f$ valid evaluation shares from honest parties, it must output the VRF value at z^*.

Now, by the η-correctness property of F it follows that the evaluation shares $\{(w'_j, \pi'_j)\}_{j\in[t-1]}$ in the tracing key are all valid evaluation shares from honest parties. From this it follows that the tracing algorithm in Figure 10 will identify the correct set of corrupt parties. □

C Proving uf-1 security for $\mathsf{OT\text{-}P_1}$

In this section, we discuss how our $\mathsf{OT\text{-}P_1}$ construction can be proven to achieve uf-1 security. Following [4], we define a new (stronger) assumption in Definition 13. This is the DCR analog of the Vector-CDH assumption defined in [4].

To prove uf-1 pseudorandomness, the proof of Theorem 2 can be modified to rely on this new assumption. Specifically, we first consider t different events, based on the number ℓ of partial evaluation queries that the pseudorandomness adversary makes on the challenge input. By a simple averaging argument, we know that there must exist a value $\ell^* \in [t-1]\cup\{0\}$, such that the adversary makes exactly ℓ^* partial evaluation queries, and wins the pseudorandomness game with only a factor of $1/t$ loss in the advantage. We can then construct an adversary $\mathcal{B}$ that breaks the ℓ^*-Vector-DCR assumption using this adversary $\mathcal{A}$. $\mathcal{B}$ gets as input the modulus N, along with $\hat{v}, \hat{v}_0$, $\{\hat{v}_i\}_{i\in[\ell^*]}$ and a challenge "ciphertext" c and m^*. It samples $x_1, \ldots, x_{t-1}$ as in the original proof, but only samples $t-\ell^*-1$ y_i values, since these are enough to respond to the secret key queries of $\mathcal{A}$. In essence, $\hat{v}_i$ can be thought to replace $\hat{v}^{y_i}$ in the original proof. Then, $\mathcal{B}$ computes ν as in the proof, and sets v_0, v to be $\hat{v}_0^\nu$ and $\hat{v}^\nu$ respectively. It computes all the v_i terms in a similar manner as in the original proof, but uses $\hat{v}_i$ in place of $\hat{v}^{y_i}$ for $i \in [\ell^*]$. Next, when answering H_0 queries, it sets the hash of the challenge input $\hat{z}$ to be the challenge ciphertext c. Additionally, for any other $z \neq \hat{z}$, it sets the hash to be $\hat{v}^{r\cdot\nu}$ for some r which is uniformly randomly sampled from $N^2/4$. Note that these hash responses are statistically indistinguishable from a uniform random distribution. To answer partial evaluation queries on $\hat{z}$ for some $i \in [n]$, it simply queries the Eval of its challenger with α vector containing lagrange coefficients, to compute interpolation in the exponent. Lastly, $\mathcal{B}$ sends m^* to $\mathcal{A}$. This allows $\mathcal{B}$ to simulate the game to $\mathcal{A}$ with negligible statistical distance from the real uf-1 game. It can simply send $\mathcal{A}$'s output to its challenger. It can be seen that $\mathcal{B}$ will be able to break the ℓ^*-Vector-DCR assumption with non-negligible advantage if $\mathcal{A}$ breaks uf-1 pseudorandomness.

Definition 13 (The ℓ-Vector-DCR assumption). *We say that the ℓ-vector-DCR assumption is hard if, for all* PPT *adversaries $\mathcal{A}$, the following function is negligible in λ:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{v-dcr},\ell}(\lambda) = \left|\Pr[\mathbf{G}_{\mathcal{A}}^{v\text{-}dcr,\ell}(\lambda) = 1] - \frac{1}{2}\right|$$

where the game $\mathbf{G}_{\mathcal{A}}^{v\text{-}dcr,\ell}$ is as defined in Figure 11.

```
Game $\mathbf{G}^{v\text{-}dcr,\ell}$

1 :  $p, q \leftarrow\$ \mathcal{P}_\lambda$
2 :  $p' := (p-1)/2, q' := (q-1)/2, N := pq, g := 1 + N \in \mathbb{Z}_{N^2}$
3 :  $m, \beta, r \leftarrow\$ \mathbb{Z}_N, \hat{v} \leftarrow\$ \mathcal{QR}_{N^2}, b \leftarrow\$ \{0,1\}$
4 :  $\forall\ i \in [\ell], y_i \leftarrow\$ \mathbb{Z}_{Np'q'}, \hat{v}_i := \hat{v}^{y_i}$
5 :  $c := (g^m r^N)^2, \hat{v}_0 := \hat{v}^{p'q'\beta}, \theta := (p'q'\beta) \bmod N$
6 :  if $b = 0$ then $m^* \leftarrow\$ \mathbb{Z}_N$
7 :  else $m^* \leftarrow m\theta \bmod N$
8 :  $\mathcal{V} := \phi$
9 :  $b' \leftarrow\$ \mathcal{A}^{\mathsf{Eval}(\cdot)}(N, \hat{v}, \hat{v}_0, \{\hat{v}_i\}_{i\in[\ell]}, c, m^*)$
10 : return $b = b' \wedge (1, 0, \ldots, 0) \notin \mathrm{span}(\mathcal{V})$ w.r.t $\mathbb{Z}_{Np'q'}$

Oracle $\mathsf{Eval}(\boldsymbol{\alpha})$:

1 :  $\mathcal{V} \leftarrow \mathcal{V} \cup \{\boldsymbol{\alpha}\}$
2 :  $c' \leftarrow c^{\boldsymbol{\alpha}_0 \cdot p'q'\beta + \Sigma_{i\in[t]} \boldsymbol{\alpha}_i y_i}$
3 :  return $c'$
```

Fig. 11. The security game $\mathbf{G}^{v\text{-}dcr,\ell}$

D Postponed Proofs

D.1 Proof of Uniqueness for $\mathsf{OT\text{-}P_1}$, Theorem 1

Proof. We construct a discrete log algorithm $\mathcal{B}$. It gets as input a group description $(\mathbb{G}, a, q^*)$ along with a random element in the group b from its challenger.

We will first construct an algorithm $\mathcal{A}'$ which invokes $\mathcal{A}$ and plays the role of challenger to $\mathcal{A}$ in the uniqueness game, as follows:

- Run all the steps of the $\mathsf{FuncSamp}$ algorithm, except for Step 5 to get $\mathsf{ek}, \rho, \mathsf{ck}, \mathsf{tk}$ as in Step 6. Set $\mathsf{vk} \leftarrow (N, v, \{v_i\}_{i\in\{0,\ldots,t-1\}}, a, b)$.
- Run $\mathcal{A}$ with inputs $(1^\lambda, \mathsf{ek}, \mathsf{ck}, \mathsf{vk}, \rho, n, t, 1^{1/\epsilon}, 1^{1/\delta})$.
- Let E be the event that $\mathcal{A}$ outputs $(z, (w_0, \pi_0), (w_1, \pi_1))$ and wins the uniqueness game. Let $ct_z \leftarrow \mathsf{H}_0(z)$.
- Parse π_0 as $\{((x_{i,0}, w_{i,0}), \pi_{i,0})\}_{i\in\mathcal{J}}$ and π_1 as $\{((x_{i,1}, w_{i,1}), \pi_{i,1})\}_{i\in\mathcal{J}'}$
- For any $i \in \mathcal{J}$, if $w_{i,0} \neq ct_z^{h(x_{i,0})}$, then output

$$((v, v^{h(x_{i,0})}, ct_z, w_{i,0}, a, b), \pi_{i,0}).$$

  Otherwise, for any $i \in \mathcal{J}'$, if $w_{i,1} \neq ct_z^{h(x_{i,1})}$, then output

$$((v, v^{h(x_{i,1})}, ct_z, w_{i,1}, a, b), \pi_{i,1}).$$

- Otherwise, abort.

Let E_a denote the event that $\mathcal{A}'$ aborts.

Claim.

$$\Pr[\mathsf{E}] = \Pr[\mathsf{E} \wedge (\overline{\mathsf{E}}_a)]$$

Proof. By total probability, we have that,

$$\Pr[\mathsf{E}] = \Pr[\mathsf{E} \wedge (\overline{\mathsf{E}}_a)] + \Pr[\mathsf{E} \wedge (\mathsf{E}_a)]$$

If $\mathcal{A}'$ aborts, this implies that all the partial evaluation shares are correct for both w_0 and w_1. This means that,

$$\begin{aligned} w_0 =& \frac{L_N\left(\prod_{i\in\mathcal{J}} w_{i,0}^{\lambda_i}\right)}{\nu_0'} && (6)\\ =& \frac{L_N\left(\prod_{i\in\mathcal{J}} ct_z^{\lambda_j \cdot h(x_{i,0})}\right)}{\nu_0'} && (7)\\ =& \frac{L_N\left(ct_z^{\nu_0' \cdot h(0)}\right)}{\nu_0'} && (8)\\ =& 2m \cdot h(0) && (9)\end{aligned}$$

where $\nu_0' = \prod_{j,k\in[t],j<k}(x_{i_j} - x_{i_k})$ and $ct_z = (g^m r^N)^2$ for some $m, r \in \mathbb{Z}_N$. Similarly, we get that $w_1 = 2m \cdot h(0)$. This contradicts the fact that in the event E, $\mathcal{A}$ wins the uniqueness game, since $w_0 = w_1$. This proves the claim.

We now have a PPT algorithm $\mathcal{A}'$ that breaks the soundness of the proof system. Hence, $\mathcal{B}$ can simply run the extractor $\mathcal{E}$ constructed in the proof of Lemma 2 to extract the discrete log of b with respect to a, and return that to the challenger. This proves the theorem. □

D.2 Proof of Pseudorandomness for $\mathsf{OT\text{-}P_1}$, Theorem 2

Proof. We construct $\mathcal{B}$ that takes part in the semantic security game for Paillier decryption. It takes as input $pk = N$ from its challenger and simulates the game $\mathbf{G}^{\text{rand}}_{\mathsf{OT\text{-}P_1}}$ to $\mathcal{A}$ as follows:

- Receive $(n, t, 1^{1/\epsilon}, 1^{1/\delta})$ from $\mathcal{A}$.
- Sample $x_1, \ldots, x_{t-1} \leftarrow\$ \mathbb{Z}_N$ and $y_1, \ldots, y_{t-1} \leftarrow\$ \mathbb{Z}_{N^2/4}$. Compute $\nu = \prod_{j\in[t-1]} x_j \cdot \prod_{j,k\in[t-1],j<k}(x_j - x_k)$.
- Define h to be the polynomial in $\mathbb{Z}_{N^2/4}[X]$ of degree $t-1$ such that $h(x_i) = y_i$ for all $i \in [t-1]$, and $h(0) = \beta\phi(N)/4$ for some β. Note that $h(0)$ is unknown to $\mathcal{B}$.
- Sample $\theta, \alpha, r \leftarrow\$ \mathbb{Z}_N^*$. Let $g = 1 + N \in \mathbb{Z}_{N^2}$. Compute $\hat{v} \leftarrow (g^\alpha \cdot r^N)^2$ and set $v \leftarrow \hat{v}^\nu$. Here, $\theta = (\beta p' q') \bmod N$, where β, p', q' are all unknown to $\mathcal{B}$.
- Sample a cyclic group of prime order: $(\mathbb{G}, a, q^*) \leftarrow\$ \mathsf{GroupGen}(1^\lambda)$ such that $q^* > N^2$. Sample $u \leftarrow\$ \mathbb{Z}_{q^*}$ and set $b \leftarrow a^u$.
- Compute $v_0 \leftarrow (1 + 2\alpha\nu\theta N) \bmod N^2$.

– Consider the Vandermonde matrix $\boldsymbol{V} \in \mathbb{Z}^{(t-1)\times(t-1)}$ over $\{x_1, \ldots, x_{t-1}\}$:

$$\begin{bmatrix} x_1 & x_1^2 & \ldots & x_1^{t-1} \\ x_2 & x_2^2 & \ldots & x_2^{t-1} \\ . & . & \ldots & . \\ x_{t-1} & x_{t-1}^2 & \ldots & x_{t-1}^{t-1} \end{bmatrix}$$

Observe that $\det(\boldsymbol{V}) = \nu$. Let $\boldsymbol{W}$ be a matrix such that $\boldsymbol{V}^{-1} = \frac{1}{\nu}\boldsymbol{W}$. Then, $\mathcal{B}$ computes the verification key elements v_i using $\boldsymbol{W}$ as follows:

$$v_i \leftarrow \frac{\hat{v}^{\Sigma_{j\in[t-1]}(\boldsymbol{W}_{i,j}\cdot y_j)}}{(1+2\alpha\theta N)^{\Sigma_{j\in[t-1]}\boldsymbol{W}_{i,j}}}$$

for all $i \in [t-1]$. Next, Send $(\mathsf{vk} \leftarrow (N, v, \{v_i\}_{i\in\{0,\ldots,t-1\}}, a, b), \mathsf{ck} \leftarrow N)$ to $\mathcal{A}$.

– Sample messages $m_0, m_1 \leftarrow\!\$\ \mathbb{Z}_N$ and send them to the challenger. Receive a ciphertext c.

– Let $\mathcal{C} \leftarrow \phi$ denote the set of corrupt parties. Let q_H denote an upper bound on the number of random oracle queries issued by $\mathcal{A}$. $\mathcal{B}$ guesses the query corresponding to $\mathcal{A}$'s final output, by sampling $i^* \leftarrow\!\$\ [q_H]$. Let $e \leftarrow 0$ denote the number of hash queries so far, and let $\hat{z} \leftarrow \perp$. We define a map $\mathcal{M} : \{0,1\}^\lambda \to \mathbb{Z}_N \times \mathbb{Z}_N$ to store auxiliary information about random oracle responses. Let $\mathcal{I} : [n] \to \mathbb{Z}_N$ be a map that stores the x value for each party in $[n]$, and let $\mathcal{J} : [n] \to \mathbb{Z}_{N^2/4}$ be a map to store the corresponding y values for parties in $[n]$. Let $\mathcal{H} \leftarrow \perp$ be a set tracking all the honest parties for which $\mathcal{A}$ has queried partial evaluation queries.

– Reply to the oracle queries as follows:

- $\mathsf{H}_0(z)$. Set $e \leftarrow e+1$. If $e = i^*$, then set $\hat{z} \leftarrow z$ and output c^2. Otherwise, if $z \in \mathcal{M}$, then let $(m, r) := \mathcal{M}[z]$. Output $(g^m r^N)^{2\nu}$. Otherwise, sample $m \leftarrow\!\$\ \mathbb{Z}_N$ and $r \leftarrow\!\$\ \mathbb{Z}_N$, set $\mathcal{M}[z] \leftarrow (m, r)$ and return $(g^m r^N)^{2\nu}$.
- $\mathsf{ekO}(i)$. Add i to the set of corrupt parties: $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$. Let $j_i \leftarrow |\mathcal{C}|$. Abort if $j_i > t-1$. Otherwise, set $\mathcal{I}[i] \leftarrow x_{j_i}$, $\mathcal{J}[i] \leftarrow y_{j_i}$ and output (x_{j_i}, y_{j_i}).
- $\mathsf{EvalO}(z, i)$. We assume without loss of generality that $\mathcal{A}$ always queries $\mathsf{H}_0(z)$ before issuing a partial evaluation query for z. Additionally, recall that we are only considering a semi-adaptive adversary, meaning that all the ekO queries must be done before $\mathcal{A}$ calls the EvalO oracle. $\mathcal{B}$ proceeds as follows.
  - ∗ If $\hat{z} \neq \perp$ and $z = \hat{z}$ then abort.
  - ∗ Otherwise, if $i \in \mathcal{C}$ or $i \in \mathcal{H}$, then simply return $\mathsf{H}_0(z)^{\mathcal{J}[i]}$.
  - ∗ Else, if $|\mathcal{C}\cup\mathcal{H}| < (t-1)$, then, set $\mathcal{H} \leftarrow \mathcal{H} \cup \{i\}$, $\mathcal{I}[i] \leftarrow x_{|\mathcal{C}\cup\mathcal{H}|}$ and $\mathcal{J}[i] \leftarrow y_{|\mathcal{C}\cup\mathcal{H}|}$. Return $\mathsf{H}_0(z)^{\mathcal{J}[i]}$.
  - ∗ Otherwise, let $(m, r) = \mathcal{M}[z]$. Let $y = g^m \cdot r^N$. Next, if $i \in \mathcal{I}$, then set $x \leftarrow \mathcal{I}[i]$, otherwise sample $x \leftarrow\!\$\ \mathbb{Z}_N$ and set $\mathcal{I}[i] \leftarrow x$. Compute the evaluation share $d_{z,i} = \mathsf{H}_0(z)^{h(x)}$ using Lagrange interpolation in the exponent. More formally, first compute Lagrange coefficients in $\mathbb{Z}$:

$$\lambda_0 = \nu \cdot \prod_{i\in[t-1]} \frac{(x_i - x)}{x_i}, \lambda_i = \frac{\nu x}{x_i} \cdot \prod_{j\in[t-1], j\neq i} \frac{(x_j - x)}{(x_j - x_i)}\ \forall\ i \in [t-1]$$

    Then, compute the partial evaluation as follows:

$$d_{z,i} \leftarrow y^{2\Sigma_{i\in[t-1]}\lambda_i y_i} \cdot (1 + 2Nm\theta\lambda_0).$$

Let $v' \leftarrow \prod_{j\in\{0,1,\ldots,t-1\}} v_j^{x^j}$ and compute the evaluation proof $\pi_{z,i} \leftarrow\!\$\ \mathsf{P}'_{eq}((v, v', \mathsf{H}_0(z), d_{z,i}, a, b)$ $, (\perp, u))$. Output $((x, d_{z,i}), \pi_{z,i})$.

Eventually, $\mathcal{A}$ outputs z^*. $\mathcal{B}$ aborts if $z^* \neq \hat{z}$. Otherwise, $\mathcal{B}$ sends $2m_0\theta$ to $\mathcal{A}$. Let b' denote $\mathcal{A}$'s final response. $\mathcal{B}$ outputs b'.

First, we argue that if $\mathcal{B}$ does not abort, then the game simulated by $\mathcal{B}$ is statistically indistinguishable from that of the real pseudorandomness game. Specifically, the $\{x_i\}$ values and $h(0)$ are identically distributed, and each y_i value is sampled from $\mathbb{Z}_{N^2/4}$, which has statistical distance $\leq 1/p + 1/q \leq 2/q$ from uniform in $\mathbb{Z}_{Np'q'}$. Additionally, since ν is co-prime with $Np'q'$ with overwhelming probability (otherwise we can factor N), the verification key and the oracle responses are also identically distributed as in the real game. Lastly, if the bit b sampled by $\mathcal{B}$'s challenger is 0, then $\mathcal{B}$ has sent the correct VRF output with respect to z^*, otherwise, it is a random value, since m_0 was sampled randomly. Hence, $\mathcal{B}$'s simulation of the pseudorandomness game is statistically indistinguishable from the real game.

Next, observe that, if $\mathcal{A}$ wins the pseudorandomness game, and if $\mathcal{B}$ does not abort, then, $\mathcal{B}$ wins its semantic security game. This is because the challenge ciphertext c was planted as the hash of z^*.

Let E_a denote the event that $\mathcal{B}$ aborts. Since $\mathcal{B}$ only aborts if it incorrectly guesses i^*, we have that $\Pr[\mathsf{E}_a] = 1/q_H$. This proves the theorem.

D.3 Proof of One-time Universal traceability for $\mathsf{OT\text{-}P_1}$

Proof (of Theorem 3). Let $\mathcal{A}$ be an adversary taking part in the universal tracing experiment $\mathbf{G}^{\text{univ-trac-0}}_{\mathsf{OT\text{-}P}_1,\epsilon,\delta}$. Let G denote the event in which $\mathcal{A}$ outputs an evaluation box E that is $(n, t, \mathbf{ek}, \mathsf{ck}, \mathsf{vk}, \Gamma_{ek}(\mathsf{ek}), \epsilon, \delta)$-good, where n and t are chosen by $\mathcal{A}$ at the beginning of the experiment, and $\mathsf{ek}, \mathsf{ck}, \mathsf{vk}$ are as generated by the challenger, and $\mathbf{ek} = (\mathsf{ek}_{i_1}, \ldots, \mathsf{ek}_{i_f})$ are the key shares given to $\mathcal{A}$ by the challenger. Conditioned on $\neg\mathsf{G}$, the output of the experiment is 0 with probability 1, and so we condition the rest of the analysis on G. Let us denote $\mathsf{ek}_{i_j} = (x_j^*, y_j^*)$ for all $j \in [f]$, and let $\mathcal{I}$ be the set of parties corrupted by $\mathcal{A}$, i.e. $\{x_j^*\}_{j\in[|\mathcal{I}|]} = \{x_i\}_{i\in\mathcal{I}}$.

For $\ell = 1, \ldots, m$, let $\mathsf{E}_{\ell,\eta}$ be the event that $\eta_\ell = 0$. For $\ell \in [m]$ and $j \in \{2, \ldots, t-f\}$, let $\mathsf{E}_{\ell,O,j}$ be the event that $x_{\ell,j} = x_{\ell,i}$ for some $i \in [j-1]$. For $\ell \in [m]$, let E^*_ℓ denote the event that $x_{\ell,j} = x_i^*$ for some $i \in [f]$ and some $j \in [t-f]$. Lastly, let E'_ℓ be the event that $x'_\ell = x'_i$ for some $0 < i < \ell$.

Let $\mathsf{E}_{\ell,1}$ be the event that

$$(w_\ell, \cdot) = \mathsf{Combine}\begin{pmatrix} \mathsf{ck}, (sh^*_{\ell,1}, \pi^*_{\ell,1}), \ldots, (sh^*_{\ell,f}, \pi^*_{\ell,f}), \\ (sh_{\ell,1}, \pi_{\ell,1}), \ldots, (sh_{\ell,t-f}, \pi_{\ell,t-f}) \end{pmatrix}$$

where $(sh^*_{\ell,j}, \pi^*_{\ell,j}) \leftarrow \mathsf{Eval}(\mathsf{ek}_{i_j}, \mathsf{vk}, z_\ell)$ and $sh^*_{\ell,j} = (x_j^*, w_j^*)$ for all $j \in [f]$. Similarly, let $\mathsf{E}_{\ell,2}$ be the event that

$$(w'_\ell, \cdot) = \mathsf{Combine}\begin{pmatrix} \mathsf{ck}, (sh^*_{\ell,1}, \pi^*_{\ell,1}), \ldots, (sh^*_{\ell,f}, \pi^*_{\ell,f}), \\ (sh_{\ell,1}, \pi_{\ell,1}), \ldots, (sh_{\ell,t-f-1}, \pi_{\ell,t-f-1}), (sh'_{\ell,t-f}, \pi'_{\ell,t-f}). \end{pmatrix}$$

Then, in the event $\mathsf{E}_{\ell,1} \wedge \mathsf{E}_{\ell,2} \wedge \neg\mathsf{E}_{\ell,\eta} \wedge \neg\mathsf{E}^*_\ell \wedge \neg\mathsf{E}'_\ell \bigwedge_{j\in\{2,\ldots,t-f\}} (\neg\mathsf{E}_{\ell,O,j})$, we have that:

$$\begin{aligned} w_\ell &= \frac{1}{\nu'} L_N \left(\prod_{j\in[f]} (w_j^*)^{\lambda^*_{\ell,j}} \cdot \prod_{i\in[t-f]} w_{\ell,i}^{\lambda_{\ell,i}} \right) \\ &= \frac{1}{\nu'} L_N \left(\prod_{j\in[f]} (w_j^*)^{\lambda^*_{\ell,j}} \cdot \prod_{i\in[t-f]} (c_z^{h(x_{\ell,i})+p'q'\alpha_{\ell,i}})^{\lambda_{\ell,i}} \right) \\ &= \frac{1}{\nu'} L_N \left(c_z^{\nu' h(0)} \cdot c_z^{\Sigma_{i\in[t-f]} \lambda_{\ell,i} p'q'\alpha_{\ell,i}} \right) \end{aligned} \tag{10}$$

where $c_z = \mathsf{H}_0(z)$, $\nu' = \prod_{i,j\in[f],i<j}(x_i^* - x_j^*) \cdot \prod_{i,j\in[t-f],i<j}(x_{\ell,i} - x_{\ell,j}) \cdot \prod_{i\in[f],j\in[t-f]}(x_i^* - x_{\ell,j})$, $\lambda^*_{\ell,j} = \nu' \cdot \prod_{i\in[f]\setminus\{j\}} \frac{x_i^*}{x_i^*-x_j^*} \cdot \prod_{i\in[t-f]} \frac{x_{\ell,i}}{x_{\ell,i}-x_j^*}$ and $\lambda_{\ell,i} = \nu' \cdot \prod_{i\in[f]} \frac{x_i^*}{x_i^*-x_{\ell,i}} \cdot \prod_{j\in[t-f]\setminus\{i\}} \frac{x_{\ell,j}}{x_{\ell,j}-x_{\ell,i}}$. Additionally,

$$\begin{aligned} w'_\ell &= \frac{1}{\nu'} L_N \left(\prod_{j\in[f]} (w_j^*)^{\lambda^*_{\ell,j}} \cdot \prod_{i\in[t-f-1]} w_{\ell,i}^{\lambda_{\ell,i}} \cdot (w'_{\ell,t-f})^{\lambda_{\ell,t-f}} \right) \\ &= \frac{1}{\nu'} L_N \left(\prod_{j\in[f]} (w_j^*)^{\lambda^*_{\ell,j}} \cdot \prod_{i\in[t-f]} (c_z^{h(x_{\ell,i})+p'q'\alpha_{\ell,i}})^{\lambda_{\ell,i}} \cdot (g^{2\eta_\ell})^{\lambda_{\ell,t-f}} \right) \\ &= \frac{1}{\nu'} L_N \left(c_z^{\nu' h(0)} \cdot c_z^{\Sigma_{i\in[t-f]} \lambda_{\ell,i} p'q'\alpha_{\ell,i}} \cdot g^{2\eta_\ell \lambda_{\ell,t-f}} \right) \end{aligned} \tag{11}$$

Let us say that $c_z = (g^m r^N)^2$ for some $m, r \in \mathbb{Z}_N$. This means that the element inside L_N is in the subgroup with easy discrete log, for both Eqns 10 and 11. Hence, we can subtract the two and get the following:

$$\begin{aligned} w'_\ell - w_\ell &= \frac{1}{\nu'} L_N(g^{2\eta_\ell \lambda_{\ell,t-f}}) \\ &= 2\eta_\ell \prod_{j\in[f]} \frac{x_j^*}{x_j^* - x'_\ell} \cdot \prod_{i\in[t-f-1]} \frac{x_{\ell,i}}{x_{\ell,i} - x'_\ell} \end{aligned}$$

Let us define a polynomial $p^*(X) = \prod_{j\in[f]} \frac{x_j^*-X}{x_j^*}$. Then, the above equation implies that y_ℓ as computed in the ℓth iteration in the Trace algorithm, is the correct evaluation of $p^*(X)$ at $X = x'_\ell$. Hence, for each $\ell \in [m]$, we get a correct evaluation of $p^*(X)$ at a unique point x'_ℓ, in the event $\mathsf{E}_{\ell,1} \wedge \mathsf{E}_{\ell,2} \wedge \neg\mathsf{E}_{\ell,\eta} \wedge \neg\mathsf{E}^*_\ell \wedge \neg\mathsf{E}'_\ell \bigwedge_{j\in\{2,\ldots,t-f\}}(\neg\mathsf{E}_{\ell,O,j})$. We will now compute the probability of this event.

We have that, for all $\ell \in [m]$, $\Pr[\mathsf{E}_{\ell,\eta}] = 1/N$. Next, since each of the $x_{\ell,i}$ values are sampled uniformly randomly, we get that $\Pr[\mathsf{E}_{\ell,O,j}] \le (j-1)/N$ for all $j \in [t-f]$, $\Pr[\mathsf{E}^*_\ell] \le f(t-f)/N$ and $\Pr[\mathsf{E}'_\ell] \le (\ell-1)/N \le m/N$.

Recall that E is $(n, t, \mathbf{ek}, \mathsf{ck}, \mathsf{vk}, \Gamma_{ek}(\mathsf{ek}), \epsilon, \delta)$-universally-good. Next, observe that each evaluation $sh_{\ell,i}$ is based on a random key share of a secret key $\alpha_{\ell,i}p'q' \in \mathcal{EK}^* = \Gamma_{ek}(p'q'\beta)$ where $\alpha \in \mathbb{Z}_N$, using the polynomial $h(X) + p'q'(\alpha_{\ell,i} - \beta)$. Additionally, the proofs $\pi_{\ell,i}$ are also distributed identically to honest proofs. Hence, $\Pr[\mathsf{E}_{\ell,1}] \ge \epsilon \cdot \delta$. And for $\mathsf{E}_{\ell,2}$, $sh'_{\ell,t-f}$ is uniformly random in $\mathcal{QR}_{N^2}$, and hence is identically distributed to a valid evaluation share. This implies that $\Pr[\mathsf{E}_{\ell,2}]$ is also $\geq \epsilon \cdot \delta$. We now compute the joint probability. Let us denote the tuple of random variables $(z, \{(x_{\ell,j}, \hat{w}_{\ell,j}, \pi_{\ell,j})\}_{j\in[t-f-1]}, x'_\ell)$ as $W \in \mathcal{W} = \{0,1\}^\lambda \times (\mathbb{Z}_N \times \mathcal{QR}_{N^2} \times \mathit{\Pi})^{t-f-1} \times \mathbb{Z}_N$, where $\mathit{\Pi}$ is the space of partial evaluation proofs. Then,

$$\begin{aligned}
\Pr[\mathsf{E}_{\ell,1} \wedge \mathsf{E}_{\ell,2}] &\geq \Pr_{\substack{w, \hat{w}_{\ell,t-f}, \pi_{\ell,t-f}, \\ \eta_\ell \leftarrow\$ \ \mathbb{Z}_{Np'q'}, \pi'_{\ell,t-f}}} [\mathsf{E}_{\ell,1} \wedge \mathsf{E}_{\ell,2}] - 2/q \qquad (12)\\
&\geq \Sigma_{w\in\mathcal{W}} \Pr[W = w] \cdot \Pr_{\substack{\hat{w}_{\ell,t-f}, \pi_{\ell,t-f}, \\ \eta_\ell, \pi'_{\ell,t-f}}} [\mathsf{E}_{\ell,1} \wedge \mathsf{E}_{\ell,2} | W = w] - 2/q\\
&\geq \Sigma_{w\in\mathcal{W}} \Pr[W = w] \cdot \Pr_{\substack{\hat{w}_{\ell,t-f}, \\ \pi_{\ell,t-f}}} [\mathsf{E}_{\ell,1} | W = w] \cdot \Pr_{\substack{\hat{w}_{\ell,t-f}, \eta_\ell, \\ \pi_{\ell,t-f}}} [\mathsf{E}_{\ell,2} | W = w] - 2/q \qquad (13)\\
&\geq \Sigma_{w\in\mathcal{W}} \Pr[W = w] \cdot \left(\Pr_{\hat{w}_{\ell,t-f}, \pi_{\ell,t-f}} [\mathsf{E}_{\ell,1} | W = w] \right)^2 - \frac{2}{q}\\
&\geq \mathbb{E}_w \left[\left(\Pr_{\hat{w}_{\ell,t-f}, \pi_{\ell,t-f}} [\mathsf{E}_{\ell,1} | W = w] \right)^2 \right] - \frac{2}{q}\\
&\geq \left(\mathbb{E}_w \left[\Pr_{\hat{w}_{\ell,t-f}, \pi_{\ell,t-f}} [\mathsf{E}_{\ell,1} | W = w] \right] \right)^2 - \frac{2}{q} \qquad (14)\\
&\geq \epsilon^2 \delta^2 - \frac{2}{q}
\end{aligned}$$

Eqn 12 follows from the fact that η_ℓ sampled randomly from $\mathbb{Z}_{N^2/4}$ has statistical distance $\leq 2/q$ to a uniform random distribution in $\mathbb{Z}_{Np'q'}$. Eqn 13 follows from the fact that the events $\mathsf{E}_{\ell,1}$ and $\mathsf{E}_{\ell,2}$ are independent conditioned on w if η_ℓ is sampled uniformly randomly in $\mathbb{Z}_{Np'q'}$. Eq. 14 follows from Jensen's inequality.

By combining the above and applying union bound, we get that,

$$\begin{aligned}
&\Pr[\mathsf{E}_{\ell,1} \wedge \mathsf{E}_{\ell,2} \wedge \neg\mathsf{E}_{\ell,\eta} \wedge \neg\mathsf{E}^*_\ell \wedge \neg\mathsf{E}'_\ell \bigwedge_{j\in\{2,\ldots,t-f\}} (\neg\mathsf{E}_{\ell,O,j})]\\
&\quad \geq \Pr[\mathsf{E}_{\ell,1} \wedge \mathsf{E}_{\ell,2}] - \Pr\left[\mathsf{E}_{\ell,\eta} \vee \mathsf{E}^*_\ell \vee \mathsf{E}'_\ell \bigvee_{j\in\{2,\ldots,t-f\}} \mathsf{E}_{\ell,O,j}\right]\\
&\quad \geq \epsilon^2\delta^2 - \frac{2}{q} - 1/N - f(t-f)/N - m/N - (t-f)^2/N\\
&\quad \geq \epsilon^2\delta^2/2
\end{aligned}$$

The last equation follows from our assumption that $\epsilon\delta \geq \sqrt{\frac{4}{q} + \frac{m+1+t(t-f)}{N}}$.

Let us define an indicator random variable $Z_\ell = \mathbb{1}[\mathsf{E}_{\ell,1} \wedge \mathsf{E}_{\ell,2} \wedge \neg\mathsf{E}_{\ell,\eta} \wedge \neg\mathsf{E}^*_\ell \wedge \neg\mathsf{E}'_\ell \bigwedge_{j\in\{2,\ldots,t-f\}} (\neg\mathsf{E}_{\ell,O,j})]$, which is 1 if and only if we get a correct evaluation of $p^*(X)$ at a unique point x'_ℓ. Since all the shares are sampled independently for each $\ell \in [m]$, we get that $\Pr[Z_\ell = 1] = E[Z_\ell] \geq \epsilon^2\delta^2/2$.

Let $Z = \Sigma_{\ell\in[m]} Z_\ell$. By the Chernoff bound, we have that for every $\zeta > 0$,

$$\Pr[Z \leq (1-\zeta) \cdot \epsilon^2\delta^2 m/2] \leq e^{-\frac{\epsilon^2\delta^2 m\zeta^2}{4}}$$

Let E_z be the event that $Z > D$. Then, since $m = 2 \cdot D \cdot r/(\epsilon^2\delta^2)$, we can set $\zeta = 1 - 1/r$ to get that $\Pr[\mathsf{E}_z]$ is at least $1 - e^{-\frac{\epsilon^2\delta^2 m}{2}\cdot(1-\frac{1}{r})^2}$. Hence, with this probability, the bivariate polynomial $Q(X,Y)$ will have as factors $Y - q(X)$, for all degree f polynomials $q(X)$ that agree with at least D evaluations out of the list L, including the polynomial $p^*(X)$. Additionally, we claim that with high probability, $\hat{Q}(x_i) \neq 0$ for all honest parties $i \in [n]$. More formally, let $\rho_{\hat{Q}}$ denote the number of unique roots of the polynomial $\hat{Q}[X] \in \mathbb{Z}_N[X]$. By total probability,

$$\begin{aligned}\Pr[\mathsf{E}_z \wedge \mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1] = {} & \Pr[\mathsf{E}_z \wedge \mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1 \wedge \rho_{\hat{Q}} = \text{superpoly}(\lambda)] \\ & + \Pr[\mathsf{E}_z \wedge \mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1 \wedge \rho_{\hat{Q}} = \text{poly}(\lambda)]\end{aligned}$$

Aggarwal and Maurer [1] prove that given a polynomial with poly(λ) degree, with non-negligible root density in $\mathbb{Z}_N$, it can be used to factor N. This means that we can construct an adversary $\mathcal{B}$ using $\mathcal{A}$ (with techniques similar to those in the proof of Theorem 2) such that,

$$\mathsf{Adv}^{\text{factor}}_{\mathcal{B}}(\lambda) \geq \Pr[\mathsf{E}_z \wedge \mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1 \wedge \rho_{\hat{Q}} = \text{superpoly}(\lambda)]$$

On the other hand, in the event $\mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1 \wedge \rho_{\hat{Q}} = \text{poly}(\lambda)$, $\hat{Q}$ has a polynomial number of roots. Then, observe that the x-values of key shares of honest parties, i.e. $\{x_i\}_{i \notin \mathcal{I}}$ are statistically independent from both the view of $\mathcal{A}$ and the shares sampled by the Trace algorithm. Hence, the probability that an honest party's x_i value is a root of $\hat{Q}(X)$ is bounded by poly$(\lambda)/N$. This means that,

$$\Pr[\mathsf{E}_z \wedge \mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1 \wedge \rho_{\hat{Q}} = \text{poly}(\lambda)] \leq \frac{(n-f)\cdot \text{poly}(\lambda)}{N}$$

Combining the above, we get that,

$$\begin{aligned}\mathsf{Adv}^{\text{univ}-\text{trace}-0}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) &= \Pr[\mathsf{E}_z \wedge \mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1] + \Pr[\neg\mathsf{E}_z \wedge \mathbf{G}^{\text{univ-trace-0}}_{\mathcal{A},\mathsf{TTVRF},\epsilon,\delta}(\lambda) = 1] \\ &\leq \mathsf{Adv}^{\text{factor}}_{\mathcal{B}}(\lambda) + \frac{(n-f)\cdot\text{poly}(\lambda)}{N} + e^{-\frac{\epsilon^2\delta^2 m}{2}\cdot(1-\frac{1}{r})^2}\end{aligned}$$

D.4 Proof of Tracer pseudorandomness for $\mathsf{OT}\text{-}\mathsf{P}_1$

Proof. The proof essentially follows the proof of Theorem 2. We construct $\mathcal{B}$ in the same way, but simulate the tracing key tk as follows:

- Sample $z_1, \ldots, z_m \leftarrow\$ \{0,1\}^\lambda$ where $m = m(n, 1/\epsilon, 1/\delta, \lambda)$.
- For each $j \in [m]$, sample $x_{j,1}, \ldots, x_{j,t-1}, \theta_{j,1}, \ldots, \theta_{j,t-1} \leftarrow\$ \mathbb{Z}_N$.
- For each $j \in [m]$, sample $m_j, r_j \leftarrow\$ \mathbb{Z}_N$. Compute $y^*_j \leftarrow g^{m_j} {r_j}^N$ and set $\mathcal{M}[z_j] \leftarrow (m_j, r_j)$.
- For every $j \in [m]$ and every $i \in [t-1]$, compute Lagrange coefficients as follows:

$$\lambda^{j,i}_0 = \nu \cdot \prod_{k\in[t-1]} \frac{x_k - x_{j,i}}{x_k}, \lambda^{j,i}_k = \frac{\nu x_{j,i}}{x_k} \cdot \prod_{\ell\in[t-1],\ell\neq k} \frac{x_\ell - x_{j,i}}{x_\ell - x_k}$$

Then, compute $w_{j,i} \leftarrow (y^*_j)^{2\Sigma_{k\in[t-1]}\lambda^{j,i}_k y_k} \cdot (1 + 2Nm_j\lambda_0\theta_{j,i})$.

– Send $\mathsf{tk} = (N, a, b, u, \{(z_j, \{w_{j,i}\}_{i\in[t-1]})\}_{j\in[m]}, v, \{v_i\}_{i\in\{0,\ldots,t-1\}})$ to $\mathcal{A}$, along with vk and ck.

We also modify how $\mathcal{B}$ responds to the random oracle queries by $\mathcal{A}$. Let $e \leftarrow 0$ denote the number of H_0 queries so far, and let $\hat{z} \leftarrow \perp$.

– $\mathsf{H}_0(z)$: Set $e \leftarrow e+1$. If $e = i^*$, and if $z = z_i$ for some $i \in [m]$, then abort; otherwise, set $\hat{z} \leftarrow z$ and output c^2 (where c is the challenge ciphertext received from). If $e \neq i^*$, then, if $z \in \mathcal{M}$, then let $(m, r) \leftarrow \mathcal{M}[z]$, output $(g^m r^N)^{2\nu}$; otherwise, sample $m, r \leftarrow\$ \mathbb{Z}_N$, set $\mathcal{M}[z] \leftarrow (m, r)$ and output $(g^m r^N)^{2\nu}$.

$\mathcal{B}$ responds to ek and EvalO queries in the same way as in the proof of Theorem 2.

Eventually, $\mathcal{A}$ outputs z^*. $\mathcal{B}$ aborts if $z^* \neq \hat{z}$. Otherwise, $\mathcal{B}$ sends $2\theta m_0$ to $\mathcal{A}$. Let b' denote $\mathcal{A}$'s response. $\mathcal{B}$ outputs b'.

As argued in the Theorem 2, if $\mathcal{B}$ does not abort, then the game simulated by $\mathcal{B}$ is statistically indistinguishable from that of the real tracer pseudorandomness game. Additionally, the tracing key is identically distributed to that in the real game.

Next, observe that if $\mathcal{A}$ wins the tracer pseudorandomness game, and if $\mathcal{B}$ does not abort, then $\mathcal{B}$ also wins its security game. Let E_a denote the event that $\mathcal{B}$ aborts. Since it only aborts if it incorrectly guesses i^*, we have that $\Pr[\mathsf{E}_a] = 1/q_H$. This proves the theorem. □

D.5 Proof of Tracer uniqueness for $\mathsf{OT\text{-}P_2}$

Proof (of Theorem 5). Consider an adversary $\mathcal{B}$ playing in the discrete log game. It gets as input a group description $(\mathbb{G}, a, q^*)$ along with a random element in the group b from its challenger.

We will first construct an algorithm $\mathcal{A}'$ which invokes $\mathcal{A}$ and plays the role of challenger to $\mathcal{A}$ in the tracer uniqueness game, as follows:

– Run Steps 1 to 4 as in the $\mathsf{FuncSamp}$ algorithm in Figure 4. Let $\mu = \mu(n, 1^{1/\epsilon}, 1^{1/\delta}, \lambda)$.
– Sample μ-bit vectors $s_1, \ldots, s_m, s^* \leftarrow\$ \{0,1\}^\mu$. Abort if the set of vectors $s_1, \ldots, s_m, s^*$ is not linearly independent.
– Otherwise, for all $i \in [\mu]$, let e_i denote a bit vector such that $e_{i,i} = 1$ and it is zero at all other positions. The span of the vectors $s_1, \ldots, s_m, s^*$ can only contain upto $m+1$ vectors out of $\{e_i\}_{i\in[\mu]}$. Without loss of generality, let us assume that the vectors $e_1, \ldots, e_{\mu-m-1}$ are not in the span.
– Sample $u_1, \ldots, u_m \leftarrow\$ \mathbb{Z}_{q^*}$ and sample $b_1, \ldots, b_{\mu-(m+1)} \leftarrow\$ \mathbb{G}$. Consider the following matrix $S \in \{0,1\}^{\mu\times\mu}$:

$$\begin{bmatrix} e_1 \\ \cdot \\ \cdot \\ e_{\mu-m-1} \\ s_1 \\ \cdot \\ \cdot \\ s_m \\ s^* \end{bmatrix}$$

By the arguments above, the matrix S is of rank μ. Let $R \in \mathbb{Z}_{q^*}^{\mu\times\mu}$ be the inverse of S in $\mathbb{Z}_{q^*}$. Then, compute b_i for all $i \in \{\mu-m, \ldots, \mu\}$ as follows:

$$b_i \leftarrow \left(\prod_{j\in[\mu-m-1]} b_j^{R_{i,j}} \right) \cdot a^{\Sigma_{j\in[m]} u_j \cdot R_{i,\mu-m-1+j}} \cdot b^{R_{i,\mu}}$$

– Send $\mathsf{ek}, \mathsf{ck}, \rho$, $\mathsf{vk} \leftarrow (N, v, v_0 \leftarrow v^{\mathsf{ek}}, \{v^{a_i}\}_{i\in[t-1]}, a, \{b_j\}_{j\in[\mu]})$ to $\mathcal{A}$.
– Let q_H denote an upper bound on the number of H_1 random oracle queries by $\mathcal{A}$. Sample $i^* \leftarrow\$ [q_H]$: this is $\mathcal{A}$'s guess for the random oracle query corresponding to the input z^* for which $\mathcal{A}$ will forge a proof.
– Define maps $H_1 : \{0,1\}^\lambda \to \{0,1\}^\mu$ and $H_0 : \{0,1\}^\lambda \to \mathcal{QR}_{N^2}$ to keep track of responses to H_1 and H_0 random oracle queries respectively. Let $c \leftarrow 0$ be a counter for the number of H_1 oracle queries so far. Let $\hat{z} \leftarrow \perp$. $\mathcal{A}$ responds to the oracle queries as follows:
  - $\mathsf{H}_1(z)$. Set $c \leftarrow c+1$. If $c = i^*$ and if $z \neq z_i$ for any $i \in [m]$, then set $\hat{z} \leftarrow z$ and output s^*. If $c = i^*$ but $z = z_i$ for some $i \in [m]$ then abort. Otherwise, if $z = z_i$ for some $i \in [m]$, output s_i. Lastly, if $z \in H_1$, then output $H_1(z)$; if not, sample a random bit string $s \leftarrow\$ \{0,1\}^\lambda$, set $H_1(z) \leftarrow s$ and output s.
  - $\mathsf{H}_0(z)$. If $z \in H_0$, then output $H_0(z)$. Otherwise, sample $c \leftarrow\$ \mathcal{QR}_{N^2}$, set $H_0(z) \leftarrow c$ and output c.
– At some point, $\mathcal{A}$ sends the input z^*. $\mathcal{A}'$ responds with the tracing key $\mathsf{tk} \leftarrow (\mathcal{Z}^* := \{z_j\}_{j\in[m]}, N, a, b, \{(u_j, \{w_{j,k}\}_{k\in[t-1]})\}_{j\in[m]}, v, \{v_i\}_{i\in\{0,\ldots,t-1\}})$.
– Eventually, $\mathcal{A}$ outputs $(w_0, \pi_0), (w_1, \pi_1)$. We assume without loss of generality that $\mathcal{A}$ queries the random oracle H_1 on z^* before sending the output. $\mathcal{A}'$ aborts if $z^* \neq \hat{z}$. Then, similar to Claim D.1, we know that there must be some partial evaluation proof π^* in either π_0 or π_1 which is valid, but for a false statement $(v, v', \mathsf{H}_0(\hat{z}), w, a, b^*)$, i.e. for an incorrect partial evaluation share for $\hat{z}$. $\mathcal{A}'$ simply outputs the following tuple corresponding to the false statement: $((v, v', \mathsf{H}_0(\hat{z}), w, a, b^*), \pi^*)$.

Observe that if $\mathcal{A}'$ does not abort, then, the element $b^* \in \mathbb{G}$ output by $\mathcal{A}'$ is equal to b. This is because we programmed $\mathsf{H}_1(\hat{z}) = s^*$, and by construction, $\prod_{i\in[\mu]} b_i^{s_i^*}$ is equal to b. Hence, we have constructed a PPT algorithm $\mathcal{A}'$ that breaks the soundness of the proof system for a statement containing (a, b). Now, $\mathcal{B}$ simply runs the extractor $\mathcal{E}$ constructed in the proof of Lemma 2 to extract the discrete log of b with respect to a, and returns that to its challenger.

Let $\hat{p}$ denote the probability that $\mathcal{A}'$ successfully outputs a valid proof for an invalid statement containing (a, b). By Lemma 2, we have that

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathcal{B},\mathsf{GroupGen}}(\lambda) \geq \hat{p}^2 - 4/N^2 - 1/q'$$

Let us now compute $\hat{p}$. Let E_a denote the event that $\mathcal{A}'$ aborts. Note that if $\mathcal{A}'$ does not abort, and if $\mathcal{A}$ wins the uniqueness game, then $\mathcal{A}'$ successfully outputs a valid proof. This means that,

$$\begin{aligned}\hat{p} &\geq \Pr[\mathbf{G}^{\text{t-uniq}}_{\mathcal{A},\mathsf{OT\text{-}P}_2,\epsilon,\delta}(\lambda) = 1 \wedge \neg \mathsf{E}_a] \\ &\geq \Pr[\mathbf{G}^{\text{t-uniq}}_{\mathcal{A},\mathsf{OT\text{-}P}_2,\epsilon,\delta}(\lambda) = 1] \cdot \Pr[\neg \mathsf{E}_a | \mathbf{G}^{\text{t-uniq}}_{\mathcal{A},\mathsf{OT\text{-}P}_2,\epsilon,\delta}(\lambda) = 1]\end{aligned}$$

Lastly, observe that $\mathcal{A}'$ aborts if either (a) it incorrectly guesses i^*, which only happens with probability $1 - 1/q_H$ in the event that $\mathcal{A}$ wins its uniqueness game, or (b) the vectors $s_1, \ldots, s_m, s^*$

are not linearly independent. This happens with probability $\leq \frac{m(q^*)^m}{2^\mu}$. Combining the above equations, we get,

$$\hat{p} \geq \mathsf{Adv}^{\mathrm{t-uniq}}_{\mathcal{A},\mathsf{OT\text{-}P}_2,\epsilon,\delta}(\lambda) \cdot (1/q_H) \cdot \left(1 - \frac{m(q^*)^m}{2^\mu}\right)$$

This proves the theorem. □

D.6 Proof of Theorem 6

The uniqueness of the VRF at each epoch is guaranteed by uniqueness of the underlying one-time traceable threshold VRF TTVRF and the position binding of the vector commitment. To see why, suppose that an adversary manages to break uniqueness with respect to epoch $j \in [T]$, this means that it outputs an input z, two *distinct* evaluations w, w', and corresponding *accepting* evaluation proofs $\pi = (\mathsf{vk}_j, \pi_{\mathsf{VC},j}, \pi_{\mathsf{TTVRF}})$ and $\pi' = (\mathsf{vk}'_j, \pi'_{\mathsf{VC},j}, \pi'_{\mathsf{TTVRF}})$. In these proofs, vk_j and vk'_j are the supposed TTVRF verification keys for epoch j; $\pi_{\mathsf{VC},j}$ and $\pi'_{\mathsf{VC},j}$ are the vector commitment opening proofs, proving that respectively that vk_j and vk'_j are the opening to the jth entry of com; and π_{TTVRF} and π'_{TTVRF} are the evaluation proofs of TTVRF for w and w', respectively. Let $\boldsymbol{\mathsf{ek}}^*_{\boldsymbol{j}} := (\mathsf{ek}^*_{1,j}, \ldots, \mathsf{ek}^*_{n,j})$ be the vector of real evaluation keys for epoch j, and let vk^*_j sampled at key generation time.

Consider two cases:

- If $\mathsf{vk}^*_j \neq \mathsf{vk}_j$ or $\mathsf{vk}^*_j \neq \mathsf{vk}'_j$, then assume without loss of generality that $\mathsf{vk}^*_j \neq \mathsf{vk}_j$. In this case, the adversary can be used to break the position binding of the vector commitment scheme. This is because π is an accepting evaluation proof, which, in particular, implies that $\pi_{\mathsf{VC},j}$ is an accepting opening proof for vk_j with respect to com. But, vk^*_j was the true jth coordinate of the vector to which com is a commitment to. Hence, the correctness of the vector commitment scheme stipulates that it is possible to compute a proof $\pi^*_{\mathsf{VC},j}$ proving that this is indeed the case. It follows that the tuple $(\mathsf{com}, j, \mathsf{vk}_j, \pi_{\mathsf{VC},j}, \mathsf{vk}^*_j, \pi^*_{\mathsf{VC},j})$ breaks the position binding of the vector commitment.
- If $\mathsf{vk}^*_j = \mathsf{vk}_j = \mathsf{vk}'_j$, then the adversary can be used to break the uniqueness of TTVRF. Let w^* be the real value of the VRF on input z at epoch j, as induced by $\boldsymbol{\mathsf{ek}}_{\boldsymbol{j}}$, and let π^* be the corresponding TTVRF evaluation proof. Since $w \neq w'$, it must hold that $w^* \neq w$ or $w^* \neq w'$. Assume without loss of generality that $w^* \neq w$. Then, since $w^* \neq w$ and π^* and π_{TTVRF} and both accepting TTVRF proofs with respect to $\mathsf{vk}^*_j = \mathsf{vk}_j$, the tuple $(z, (w^*, \pi^*), (w, \pi_{\mathsf{TTVRF}}))$ breaks the uniqueness of TTVRF.