# Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures From VOLE-in-the-Head*

Carsten Baum[1,2], Lennart Braun[1], Cyprien Delpech de Saint Guilhem[3], Michael Klooß[4†], Emmanuela Orsini[5], Lawrence Roy[1], and Peter Scholl[1]

[1] Aarhus University
[2] Technical University of Denmark
[3] imec-COSIC, KU Leuven
[4] Aalto University
[5] Bocconi University

**Abstract.** We present a new method for transforming zero-knowledge protocols in the designated verifier setting into public-coin protocols, which can be made non-interactive and publicly verifiable. Our transformation applies to a large class of ZK protocols based on oblivious transfer. In particular, we show that it can be applied to recent, fast protocols based on *vector oblivious linear evaluation* (VOLE), with a technique we call *VOLE-in-the-head*, upgrading these protocols to support public verifiability. Our resulting ZK protocols have linear proof size, and are simpler, smaller and faster than related approaches based on MPC-in-the-head.

To build VOLE-in-the-head while supporting both binary circuits and large finite fields, we develop several new technical tools. One of these is a new proof of security for the SoftSpokenOT protocol (Crypto 2022), which generalizes it to produce certain types of VOLE correlations over large fields. Secondly, we present a new ZK protocol that is tailored to take advantage of this form of VOLE, which leads to a publicly verifiable VOLE-in-the-head protocol with only 2x more communication than the best, designated-verifier VOLE-based protocols.

We analyze the soundness of our approach when made non-interactive using the Fiat-Shamir transform, using round-by-round soundness. As an application of the resulting NIZK, we present FAEST, a post-quantum signature scheme based on AES. FAEST is the first AES-based signature scheme to be smaller than SPHINCS+, with signature sizes between 5.6 and 6.6kB at the 128-bit security level. Compared with the smallest version of SPHINCS+ (7.9kB), FAEST verification is slower, but the signing times are between 8x and 40x faster.

## Table of Contents



*This is the full version of [BBD+23], for citations please use the conference version.

†Research was conducted at Karlsruhe Institute of Technology.

## 1 Introduction

Zero-knowledge (ZK) proofs allow a prover to convince a verifier of the truth of a statement, without revealing any further information. Since their inception in 1985 [GMR85], ZK proofs have become an essential part of a cryptographer's toolbox, being used for a range of applications including CCA-secure encryption, digital signatures, anonymous credentials, anonymous cryptocurrencies and more.

MPC-in-the-Head (MPCitH) is a method of using secure multi-party computation (MPC) protocols to build efficient ZK proof systems [IKOS07]. The idea is for the prover to emulate, in its head, the execution of an MPC protocol related to the statement being proven. The verifier then checks parts of this execution in order to verify the truth of the statement. MPCitH proofs can be based on relatively simple MPC protocols, which often allows them to be very efficient in terms of computational complexity [GMO16, KKW18, BN20, dOT21]. MPCitH proofs are also public-coin[6], so can be easily made non-interactive (hence, publicly verifiable) using the Fiat-Shamir transform.

The main drawback of many MPCitH protocols is that their proof size scales linearly with the size of the (Boolean or arithmetic) circuit representation of the statement being proven. In practice, however, improvements that lowered the constants in linear-size proofs [GHS+21, KZ22] have allowed MPCitH to shine in settings where a small prover runtime is critical, and/or when proving statements of small-to-medium sized circuits, where the linear proof size may not have a big impact. Another advantage is that MPCitH protocols are usually based on standard, symmetric

[6] Meaning that the verifier's messages are always sampled uniformly at random.

cryptography, which can easily be made secure against quantum adversaries. This has led to them being used for post-quantum digital signature schemes such as Picnic [CDG+17] and follow-up constructions [BdK+21, KZ22, FJR22a]. These are non-interactive ZK proofs (NIZKs), where the verification key of the signature is the output $y$ of a one-way function $f$ while the secret the prover shows knowledge of is an input $x$ such that $y = f(x)$. These signatures can be particularly efficient if $f$ has a nice circuit representation that is efficient to evaluate in MPCitH [dDOS19, BdK+21, FJR22a].

One exception to the linear proof size is Ligero [AHIV17]. For a circuit $C$, Ligero achieves $O(\sqrt{|C|})$ proof size by building upon an actively secure, honest-majority MPC protocol. A major downside of Ligero is that the computational costs of the prover and verifier are higher, due to the need for many Reed-Solomon encoding operations and consistency checks, as well as the fact that addition gates are no longer "free" (unlike in most other MPCitH protocols). Ligero also has an inherent "startup cost", so that its proof size only drops below the size of linear proofs [GMO16, KKW18, BN20, dOT21] if $|C|$ is large enough.

*VOLE-based ZK.* Another approach to prover-efficient, linear-sized ZK is to use vector oblivious linear evaluation (VOLE), a tool which has recently seen a lot of progress [BCGI18, WYKW21]. VOLE-based proofs use preprocessed random VOLE correlations to implement highly efficient proofs with a commit-and-prove type structure, where VOLE is used to commit to the witness, and then relations are proven about the commitments using information-theoretic techniques [WYKW21, DIO21, YSWW21, BMRS21, WYY+22, BBMHS22]. These proofs communicate as little as 1 field element per multiplication gate and are computationally very efficient. They also permit optimizations such as efficient verification of low-degree polynomials [YSWW21] or disjunctions [BMRS21].

The main disadvantage of all these protocols is that they are inherently designated verifier proofs. This is because they require the verifier to keep a secret state, namely his parts of the VOLE correlation, to ensure soundness.

*Succinct ZK.* ZK proofs can be done with poly-logarithmic or even constant size, using techniques such as SNARKs [GGPR13] and STARKs [BBC+17]. Later works obtain succinct proofs with a linear prover runtime [BCG20, XZS22], however, the hidden constants are fairly large, meaning that in practice, the prover in these constructions is usually slower than MPCitH. In this work, we instead focus on proofs with a fast prover runtime that only achieve low communication for small-to-medium sized statements.

### 1.1 Our Contributions

We present *VOLE-in-the-head*, a new approach to building efficient ZK. Like MPCitH, VOLE-in-the-head proofs are based on standard symmetric cryptographic primitives and are publicly verifiable. At the same time, they inherit the simplicity and expressiveness of VOLE-based protocols, which allows them to be much smaller and faster than previous MPC-in-the-head methods.

**From OT/VOLE-Based ZK to Public Verifiability.** We start by presenting a simple compiler that takes a ZK protocol in the OT-hybrid model and converts it into a publicly verifiable protocol. We then extend this to compile ZK protocols in the VOLE-hybrid model, by giving a suitable VOLE based on OT. Unfortunately, our approach is not compatible with previous LPN-based

**Table 1.** Comparison of linear-size zero-knowledge proof systems

| Protocol | Field* | Model | Comm./gate† | Assumption |
|---|---|---|---|---|
| VOLE-ZK [YSWW21]‡ | $\mathbb{F}_2$ | deg-$d$ constraints | 1 | LPN |
| VOLE-ZK [DIO21, YSWW21]‡ | $\mathbb{F}_p$ | deg-$d$ constraints | 1 | LPN |
| Limbo [dOT21] | $\mathbb{F}_2$ | Circuits (free XOR) | 42 (11) | Hash |
| Limbo [dOT21] | $\mathbb{F}_p$ | Circuits (free add) | 40 (11) | Hash |
| VOLE-in-the-head (§6.2) | $\mathbb{F}_2$ | deg-$d$ constraints | 16 (5) | Hash |
| VOLE-in-the-head (§6.1) | $\mathbb{F}_p$ | deg-$d$ constraints | 3 (2) | Hash |

* $p \approx 2^{64}$
† Soundness error at most $2^{-128}$ ($2^{-40}$). Cost is average number of field elements sent per AND/mult. gate, for a circuit with $2^{20}$ such gates.
‡ Designated-verifier only

VOLE protocols [BCGI18, BCG+19, WYKW21], since we require the prover to play the role of the OT sender, while in LPN-based VOLE, the prover is the OT receiver. We instead adapt the VOLE protocol from SoftSpokenOT [Roy22], where the VOLE sender is also the OT sender. This protocol is restricted to VOLEs where the sender's message is over a small field. To allow for more general ZK, we give a generalized version of the protocol that (with some constraints) works over large fields; to prove this secure, we devise a new proof strategy for SoftSpokenOT.

**Instantiations and Concrete Efficiency.** We give two main instantiations of our compiler, using VOLE-based ZK. The first is a variant of the QuickSilver protocol [YSWW21] using (subfield) VOLE over $\mathbb{F}_2$, aimed at Boolean computations. We tweak QuickSilver to allow mixing constraints over $\mathbb{F}_2$ and any extension field $\mathbb{F}_{2^k}$, which is particularly useful for AES. As shown in Table 1, the communication cost of the protocol is as small as 5 bits per AND gate (or, when proving low-degree relations, per bit of the witness) with 40-bit statistical security. The runtime of the prover should be at least as fast as QuickSilver; the only difference is that instead of LPN-based VOLE, we use VOLE-in-the-head based on SoftSpokenOT; despite the higher communication, this is computationally cheaper than LPN-based methods [Roy22].

Our second instantiation is designed for proving statements in $\mathbb{F}_{p^k}$ when $p^k$ is large. This case is more challenging, due to some subtle issues and limitations of our VOLE protocol over large fields. For $\kappa$-bit security and a finite field of size $\geq 2^\kappa$, the resulting protocol has roughly $2\times$ the communication cost of the best designated-verifier VOLE-based protocol.

**Application: Post-Quantum Signatures From AES.** As an example application, we present a new post-quantum signature scheme based on proving knowledge of an AES pre-image, called FAEST. FAEST significantly outperforms prior AES-based signatures [BdK+21, KZ22], while obtaining signature sizes under 7kB. We also compare FAEST to other efficient post-quantum schemes, showing that it achieves the smallest signature size, as well as better prover running time, among all the schemes based on symmetric primitives and code-based assumptions, while SPHINCS+[HBD+22] still has better verification time.

### 1.2 Technical Overview

We now give a more detailed overview of our techniques.

**Compiling OT-Based Zero-Knowledge Proofs.** At the heart of our approach is a compiler that starts with a ZK protocol based on oblivious transfer (OT) — where the prover is the OT sender and verifier is the receiver — and converts it into a publicly verifiable one. If the ZK protocol satisfies a natural public-coin type property, this is easily done by replacing the oblivious transfer with the prover committing to its OT messages; at the end of the proof, the verifier simply sends its OT choices in the clear to the prover, who opens the corresponding commitments. When using standard 1-out-of-2 OT, this approach (which is similar to e.g. MPC-in-the-head [IKOS07] and homomorphic commitments [CDD+19]) does not seem enough to transform ZK based on VOLE, because it is not compatible with efficient, LPN-based VOLE used in these ZK constructions: in these, the prover plays the role of the OT receiver and not the sender.

We therefore present a generalized version of the compiler, which starts with a ZK protocol based on $(N-1)$-out-of-$N$ OT (all-but-one OT) on random strings, for some parameter $N$. By having the prover commit to a key for a puncturable PRF that defines $N$ pseudorandom strings, the prover can later reveal all-but-one of these with only $O(\log N)$ communication by opening a punctured key.

**Fiat–Shamir and Signatures.** We show how to apply the notion of round-by-round soundness [CCH+19, CMS19] to our compiled protocols when made non-interactive using the Fiat-Shamir transformation. To this end, we interpret our interactive protocol as an IOP (only for soundness, not for SHVZK), by treating the (malicious) prover's OT inputs as PCP-strings and the verifier's OT choices as queries to the PCP oracles. This allows also to build a Picnic-like post-quantum signature scheme based on AES.

**Using VOLE Instead of OT.** VOLE can be viewed as an arithmetic form of OT, where one party, who we call the prover $\mathcal{P}$, learns a pair $\mathbf{u} \in \mathbb{F}_p^\ell, \mathbf{v} \in \mathbb{F}_{p^k}^\ell$, while the verifier $\mathcal{V}$ learns a random $\varDelta \in \mathbb{F}_{p^k}$ and $\mathbf{q} = \mathbf{u}\varDelta + \mathbf{v} \in \mathbb{F}_{p^k}^\ell$. VOLE is used in ZK proofs as a kind of linear homomorphic commitment; $\mathcal{P}$ is committed to the vector $\mathbf{u}$ towards the verifier, and cannot open any component of $\mathbf{u}$ to a different value without guessing $\varDelta$. Note that $\mathbf{u}$ can be chosen in a small subfield $\mathbb{F}_p$, while $\varDelta$ is in an extension $\mathbb{F}_{p^k}$, giving soundness error $p^{-k}$.

Our goal is to find a suitable VOLE protocol based on $(N-1)$-out-of-$N$ OT, so we can use our compiler to transform VOLE-based ZK. We adapt the construction of [Roy22], which is based on the observation that if $N = p^k$, then a single $(N-1)$-out-of-$N$ OT can be converted into a VOLE correlation. Let $t_0, \dots, t_{N-1} \in \mathbb{F}_p$ be the messages held by $\mathcal{P}$, and let $\varDelta \in [1..N]$ be a random index chosen by $\mathcal{V}$, who then learns $t_x$ for all $x \neq \varDelta$. The idea is that $\varDelta$ can be viewed as the secret in a VOLE correlation, given by:

$$q = \sum_{x \in \mathbb{F}_{p^k} \setminus \{\varDelta\}} t_x(\varDelta - x) = \sum_{x \in \mathbb{F}_{p^k}} t_x(\varDelta - x) = \sum_{x \in \mathbb{F}_{p^k}} t_x \varDelta + \sum_{x \in \mathbb{F}_{p^k}} t_x(-x) = u\varDelta + v$$

$\mathcal{P}$ can compute $u$ and $v$, while $\mathcal{V}$ gets $q = \varDelta u + v$. This approach is only efficient when $p^k$ is small; however, as long as $p$ is small, it can be extended to handle arbitrarily large $p^k$ with some extra communication, via parallel repetition. We can directly combine this VOLE protocol with our OT-based compiler to get publicly verifiable ZK. We call this technique *VOLE-in-the-head*.

This approach for small $p$ is loosely connected to MPCitH approaches like KKW [KKW18] and Limbo [dOT21]. We can view the OT setup as splitting the secret $u$ into $N$ shares $r_i$, where the verifier learns all-but-one of the shares. The key difference is that, instead of evaluating the circuit on all $N$ sets of shares using MPC, we compress the shares into a VOLE (-in-the-head) correlation. Then, by adopting the simple multiplication checks of VOLE-based proofs, the circuit verification procedure becomes much simpler than MPCitH checks, in terms of both communication and computation.

**Handling Large Fields.** Unfortunately, the above method is limited to efficiently proving constraints over small fields, since the VOLE from [Roy22] requires $O(p)$ computation. To circumvent this, we first observe that the VOLE idea can easily support any large field $\mathbb{F}_p$, under the constraint that the sender's secret $\varDelta$ is sampled from a small subset of $S_\varDelta \subseteq \mathbb{F}_p$.

One remaining issue is that even though we can now use a large field, $\varDelta$ has low entropy, which means the ZK protocol will have a large soundness error. We fix this by using an *encoded* form of VOLE, called subspace VOLE, where $\mathcal{P}$'s input $\mathbf{u}$ is committed as

$$\mathbf{q} = \mathcal{C}(\mathbf{u}) * \varDelta + \mathbf{v}$$

for some linear code $\mathcal{C}$. Here, $\varDelta$ is a vector of length $n_\mathcal{C}$ field elements and $*$ is the component-wise product. If the code has minimum distance $d_\mathcal{C}$ and the $n_\mathcal{C}$ VOLE secrets are independent (one for each symbol of $\mathcal{C}(\mathbf{u})$), then to open to a different codeword $\mathcal{C}(\mathbf{u}')$, a malicious $\mathcal{P}$ must guess $d_\mathcal{C}$ entries of $\varDelta$ instead of just one.

On its own, this type of subspace VOLE is incompatible with the standard VOLE used in previous ZK protocols. We therefore present a new ZK protocol based on subspace VOLE, with around $2\times$ overhead on standard VOLE-based protocols over large fields. Our protocol is based on a simple code-switching technique, which translates a vector that is committed under subspace VOLE, into one committed under (standard) VOLE-in-the-head. Using our compiler, we can then replace subspace VOLE with VOLE-in-the-head, obtaining a publicly verifiable protocol.

**Consistency Checking.** While adjusting the VOLE to handle large fields $\mathbb{F}_p$, with $S_\varDelta \subseteq \mathbb{F}_p$, is straightforward, we still need a consistency check to argue that subspace VOLE is secure against a malicious prover. Proving security of this consistency check is *not* straightforward, requiring a new analysis of the SoftSpokenOT consistency check. SoftSpokenOT makes much use of $\varDelta$ being uniform in a linear space, making the distribution invariant under invertible linear transformations. This invariance no longer holds when $\varDelta$ is sampled from an arbitrary subset $S_\varDelta$.

Another difficulty is proving that the consistency check works together with both the Fiat–Shamir transformation and the commitment-based "OT"s. The verifier's choice of a hash function for the consistency check and its revelation of $\varDelta$ at the end must take place in two separate rounds, and a tight bound for Fiat–Shamir requires these rounds to be analyzed separately from each other. However, combining these bounds together is essential to the proof in SoftSpokenOT, as attackers can trade off success probability between these two rounds.

We address both these issues by giving a column-by-column analysis of the security, in the style of [OOS17] (but without the issue in its proof), rather than the linear subspaces style of [Roy22]. By defining it on columns, we do not need $S_\varDelta$ to form a linear subspace; we only need each entry

$\Delta_i$ to be independently random. To fix the Fiat–Shamir analysis, we first define a property defined on subsets of columns of the prover's secret $\mathbf{U}$, and show that if this property is preserved by the hash function then the protocol is secure. We then bound the probability that this property is not preserved by the hash function, independently from $\Delta$.

## 2 Preliminaries

Here we recall some preliminaries that will be useful in the rest of the paper.

**Basic Notation.** The security parameter is denoted by $\lambda$, and given as an implicit input to all algorithms; all other parameters in our schemes are viewed as functions of $\lambda$. A function $f$ which satisfies $\lim_{\lambda\to\infty} \mathsf{negl}(\lambda)\cdot\lambda^c = 0$ for any constant $c$ is called negligible, and we use $\mathsf{negl}(\lambda)$ to denote such a function. We write $\mathsf{AdvDist}_{\mathcal{D}}^{X,Y}$ for the distinguishing advantage of algorithm $\mathcal{D}$ for probability ensembles $(X_{\lambda,z})$, $(Y_{\lambda,z})$, i.e. $\mathsf{AdvDist}_{\mathcal{D}}^{X,Y} = \Pr[\mathcal{D}(1^\lambda, z, X_{\lambda,z}) = 1] - \Pr[\mathcal{D}(1^\lambda, z, Y_{\lambda,z}) = 1]$.

Given two machines, $A, B$, we let $B^A(x)$ denote the output of machine $B$ on input $x$ and given oracle access to $A$.

For a set $S$, we denote by $s \leftarrow S$ the process of sampling $s$ from $S$ uniformly at random. For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$; for $a, b \in \mathbb{N}$ with $a \le b$, we use $[a..b] = \{a, \ldots, b\}$ and $[a..b) = \{a, \ldots, b-1\}$.

We use bold lower-case letters for column vectors and bold upper-case letters for matrices. We denote the $i$th row (resp. column) of a matrix $\mathbf{A}$ by $\mathbf{A}_i$ (resp. $\mathbf{A}^i$), by $\mathbf{A}_{[a..b]}$ (resp. $\mathbf{A}^{[a..b]}$) the submatrix of $\mathbf{A}$ containing rows (resp. columns) $a$ through $b$; we denote by $x_i$ the $i$-th component of vector $\mathbf{x}$ and $\mathbf{x}_{[a..b]}$ the vector of components $x_a, \ldots, x_b$. Given a vector $\mathbf{x}$, we denote by $\mathsf{diag}(\mathbf{x})$ the diagonal matrix having $\mathbf{x}$ on the diagonal.

**Linear Codes.** An $[n_{\mathcal{C}}, k_{\mathcal{C}}, d_{\mathcal{C}}]_p$ linear code $\mathcal{C}$ over $\mathbb{F}_p$ is a $k_{\mathcal{C}}$-dimensional subspace of $\mathbb{F}_p^{n_{\mathcal{C}}}$, where $n_{\mathcal{C}}$ is the length and $d_{\mathcal{C}}$ the minimum distance of the code, i.e., the minimum Hamming distance between any two codewords. A matrix $\mathbf{G}_{\mathcal{C}} \in \mathbb{F}_p^{k_{\mathcal{C}}\times n_{\mathcal{C}}}$ is a generator matrix for $\mathcal{C}$ if its rows are a basis for $\mathcal{C}$ as a linear subspace, that is $\mathcal{C} = \{\mathbf{x}^T\mathbf{G}_{\mathcal{C}} : \mathbf{x} \in \mathbb{F}_p^{k_{\mathcal{C}}}\}$. We denote $\mathcal{C}(\mathbf{x}) = \mathbf{x}^T\mathbf{G}_{\mathcal{C}}$. We let $\mathbf{T}_{\mathcal{C}} \in \mathbb{F}_p^{n_{\mathcal{C}}\times n_{\mathcal{C}}}$ contain $\mathbf{G}_{\mathcal{C}}$ in its first $k_{\mathcal{C}}$ rows, with the remaining rows chosen linearly independently, so that $\mathbf{T}_{\mathcal{C}}$ is invertible and forms a basis of $\mathbb{F}_p^{n_{\mathcal{C}}\times n_{\mathcal{C}}}$. We recall that a code $\mathcal{C}$ is systematic if it has a generator matrix $\mathbf{G}_{\mathcal{C}}$ of the form $[\mathbf{A} \,|\, \mathbf{I}_{k_{\mathcal{C}}}]$, where $\mathbf{I}_{k_{\mathcal{C}}}$ is the $k_{\mathcal{C}} \times k_{\mathcal{C}}$ identity matrix.

Given a matrix $\mathbf{A} \in \mathbb{F}_p^{n\times k_{\mathcal{C}}}$ and an $[n_{\mathcal{C}}, k_{\mathcal{C}}, d_{\mathcal{C}}]_p$ linear code $\mathcal{C}$, by abuse of notation, we will write $\mathcal{C}(\mathbf{A})$ to denote the $n \times n_{\mathcal{C}}$ matrix whose rows are the encoding $\mathcal{C}(\mathbf{A}_i)$, $i \in [n]$, of $\mathbf{A}$'s rows, i.e., $\mathcal{C}(\mathbf{A}) = \mathbf{A}\cdot\mathbf{G}_{\mathcal{C}}$.

**Universal Hash Functions.** Several of our protocols take advantage of linear structure to perform a consistency check, where a number of equations are checked at once by taking a random linear combination. For such checks, it is often more efficient to use a linear universal hash function rather then taking a truly random linear combination.

**Definition 1.** *A linear $\varepsilon$-almost universal family of hashes is a family of matrices $\mathcal{H} \subseteq \mathbb{F}_q^{r\times\ell}$ such that for any nonzero $\mathbf{v} \in \mathbb{F}_q^\ell$,*

$$\Pr_{\mathbf{H}\leftarrow\mathcal{H}}[\mathbf{H}\mathbf{v} = 0] \le \varepsilon.$$

We also borrow the notion of a $\mathbb{F}_p^\ell$-hiding hash from [Roy22].

**Definition 2.** *Let $p$ and $q = p^k$ be prime powers. A matrix $\mathbf{H} \in \mathbb{F}_q^{r\times(\ell+h)}$ is $\mathbb{F}_p^\ell$-hiding if the distribution of $\mathbf{Hv}$ is independent from $\mathbf{v}_{[1..\ell]}$ when $\mathbf{v} \leftarrow \mathbb{F}_p^{\ell+h}$. Equivalently, if $\mathbf{H}' \in \mathbb{F}_p^{rk\times(\ell+h)}$ is $\mathbf{H}$ reinterpreted as a $\mathbb{F}_p$-linear map, then the column space of $\mathbf{H}'$ must equal the column space of $\mathbf{H}'_{[\ell+1..\ell+h]}$. A hash family $\mathcal{H} \subseteq \mathbb{F}_q^{r\times(\ell+h)}$ is $\mathbb{F}_p^\ell$-hiding if every $\mathbf{H} \in \mathcal{H}$ is $\mathbb{F}_p^\ell$-hiding.*

## 2.1 Zero-Knowledge Proofs of Knowledge

We define zero-knowledge proof systems in the combined common reference string (CRS) and random oracle (RO) model, short CRS+RO model. Our definition roughly corresponds to real-ideal multi-use zero-knowledge proofs of knowledge definitions, i.e. our definitions are sequentially composable.

An *interactive zero-knowledge proof system $\Pi$* for NP-relation $\mathcal{R}$ is a tuple $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ of PPT algorithms, the setup algorithm $\mathsf{Setup}$ which generates the CRS, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$.

- $\mathsf{Setup}^H(1^\lambda) \to crs$: Given the security parameter, output a CRS.
- $\mathcal{P}^H(crs, \mathbb{x}, \mathbb{w})$ and $\mathcal{V}^H(crs, \mathbb{x})$ interact on common input $\mathbb{x}$. The prover's private input is $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. The verifier outputs a bit $b$ indicating whether it accepts (1) or rejects (0). The prover has no output.

Note that all algorithms have access to $crs$ and the random oracle(s), usually denoted $H$. Thus, we consider these as implicit inputs in the rest of the paper. We write $\mathsf{tr} \leftarrow \langle \mathcal{P}^H(x), \mathcal{V}^H(y)\rangle$ for the *transcript* of an interaction where $\mathcal{P}$ (resp. $\mathcal{V}$) has input $x$ (resp. $y$) and *implicit input crs* and access to $H$. We also write $b = \langle \mathcal{P}^H(x), \mathcal{V}^H(y)\rangle$ for the verifier's output. A proof system is *public-coin* if the verifier's messages are parts of its random tape and it outputs $b = \mathsf{Verify}^H(crs, \mathbb{x}, \mathsf{tr})$ for a PPT algorithm $\mathsf{Verify}$.

For concreteness, we provide explicit definitions of zero-knowledge and knowledge soundness in the combined CRS and RO model. In particular, our definitions allow CRS and RO dependent statements and are sequentially composable. For these properties in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid or $\mathcal{F}_{\mathsf{VOLE}}$-hybrid model,[7] we instead use the standard real-ideal notion of zero-knowledge proofs of knowledge. Most of these formal definitions are deferred to Appendix A.1.

*Remark 1.* Since CRS and RO are only used to "realize" $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$ in our compiler, but not our VOLE (Section 5) or ZK proofs Section 6, we need not consider combinations of CRS+RO model with $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid models, etc.

Since the notions are less common, we define public-coin and special honest-verifier ZK in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{N,\ell}$-hybrid model here. See Figure 1 for the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{N,\ell}$-functionality.

**Definition 3 (Public-coin verifier).** *A public-coin verifier $\mathcal{V}$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model sends parts of its random tape as messages to the prover or as choices to $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$. The output is computed via a function* $\mathsf{Verify}(\mathsf{view}_{\mathcal{V}})$ *depending on verifier's view at the end of the protocol.*

[7] We use standard notions of hybrid models, see for example [Lin17]. That is, we consider access to an (unbounded) number of instances (or sessions), distinguished of the hybrid functionality, which are distinguished by an identifier *sid*. Both CRS and RO model can be viewed as hybrid models as well, although we do not do this (and limit the CRS and RO to a single one in our protocols).

**Definition 4 (SHVZK).** *Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a public-coin proof system in the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model. A semi-honest public-coin verifier $\hat{\mathcal{V}} = (\mathcal{V}, \mathcal{V}'')$ for $\Pi$ acts as follows.*

- *$\hat{\mathcal{V}}$ treats its auxiliary input as its random tape and splits into two parts $r', r''$.*
- *To handle all "normal" protocol messages, $\hat{\mathcal{V}}$ executes the honest verifier $\mathcal{V}(\mathbb{x}; r')$ with $r'$ as random tape.*
- *To handle the requests of $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ for corrupt verifiers, $\hat{\mathcal{V}}$ executes algorithm $\mathcal{V}''(\mathsf{view}; r'')$ which decides how to program the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ outputs $F^i$ to the prover, given the view $\mathsf{view}$ of $\mathcal{V}$ above and random tape $r''$.*

*A protocol in the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model is special honest-verifier zero-knowledge (SHVZK), if for every PPT semi-honest verifier $\hat{\mathcal{V}}$ as specified above, a* straightline *zero-knowledge simulator $\mathcal{S}$ exists such that $\mathsf{AdvZK}^{\Pi,\mathcal{S}}_{\hat{\mathcal{V}}}(\lambda)$ is negligible, with advantage defined analogously to Definition 17.*

### 2.2 Polynomial Constraint Systems

Our ZK protocols can prove NP relations defined by a set of degree-$d$ constraints on a witness. This is a natural generalization of the standard models of arithmetic circuits and rank-1 constraint systems (R1CS). A statement $\mathbb{x}$ for an NP relation $R$ is defined by a set of $t$ degree-$d$ polynomials $f_i \in \mathbb{F}_{p^k}[X_1, \ldots, X_\ell]_{\leq d}$, for $i \in [t]$. The witness is a vector $\mathbb{w} \in \mathbb{F}_p^\ell$, where

$$(\mathbb{x}, \mathbb{w}) \in R \Leftrightarrow f_i(\mathbb{w}) = 0, \forall i \in [t]$$

Note that even though the constraints are defined over an extension field $\mathbb{F}_{p^k}$, the witness is a vector over $\mathbb{F}_p$, embedded into $\mathbb{F}_{p^k}$ in the natural way.

### 2.3 Random Vector Commitment Schemes

We define vector commitments w.r.t. common reference strings (CRS) and random oracles (RO), so as to cover the plain model, CRS model, RO model, and CRS plus RO model simultaneously.

Informally, a vector commitment scheme is a two-phase protocol between two PPT machines, a *sender* and a *receiver*. In the first phase, also called *commitment phase*, it enables the sender to commit to a vector of messages while keeping it secret; in the second phase, called *decommitment phase*, a subset of indices of the commitment is opened. The commitment scheme satisfies two main properties: the *binding property* ensures that the sender cannot open the commitment in two different ways; the *hiding property* guarantees that the commit phase does not reveal any information about the committed message before opening and that messages at *unopened indices* are hidden, even after opening a subset of indices.

**Definition 5 (VC).** *Let $H$ be a random oracle. A* (non-interactive) vector commitment scheme $\mathsf{VC}$ *(with message space $\mathcal{M}$) in the CRS+RO model is defined by the following PPT algorithms:*

- $\mathsf{Setup}^H(1^\lambda, N) \to \mathsf{crs}$*: Given security parameter $\lambda$ and vector length $N = poly(\lambda)$ as input, output a commitment key* $\mathsf{crs}$.
- $\mathsf{Commit}^H_{\mathsf{crs}}() \to (\mathsf{com}, \mathsf{decom}, (m_1, \ldots, m_N))$*: Given* $\mathsf{crs}$ *as input, output a commitment* $\mathsf{com}$ *with opening information* $\mathsf{decom}$ *for messages $(m_1, \ldots, m_N) \in \mathcal{M}^N$.*
- $\mathsf{Open}^H_{\mathsf{crs}}(\mathsf{decom}, I)$*: On input* $\mathsf{crs}$*, opening* $\mathsf{decom}$ *and a subset $I \subseteq [N]$ of indices, output an opening* $\mathsf{decom}_I$ *for $I$.*

– $\mathsf{Verify}^H_{\mathsf{crs}}(\mathsf{com}, \mathsf{decom}_I, I) \to \{(m_i)_{i\in I}\} \cup \{\perp\}$: *Given* $\mathsf{crs}$*, a commitment* $\mathsf{com}$*, an opening* $\mathsf{decom}$ *for a subset* $I$ *as well as the subset* $I$*, either output the messages* $(m_i)_{i\in I}$ *at indices* $I$ *(accept the opening) or* $\perp$ *(reject the opening).*

*As indicated, all algorithms have access to the random oracle* $H$*.*

If in the definition above, instead of general $I \subseteq [N]$ only $I \in \mathcal{I}$ is allowed (otherwise, algorithms output $\perp$), then $\mathsf{VC}$ is restricted to $\mathcal{I}$-openings, where $\mathcal{I}$ is a fixed subset of $[N]$. If $\mathcal{I} = \{[N]\setminus\{i\} \mid i \in [N]\}$, then $\mathsf{VC}$ has *all-but-one* openings. We will explicitly state definitions for general $\mathsf{VC}$, but they apply to $\mathsf{VC}$s with $\mathcal{I}$-opening verbatim.

*Remark 2.* Our definition of $\mathsf{VC}$ is a *random* vector commitment scheme. By making the message vector an input to $\mathsf{Commit}$ instead of an output, one obtains the more common notion of (vector) commitments. However, our restricted definition is sufficient for our construction and has simpler security notions.

**Definition 6 (Correctness).** *A vector commitment scheme* $\mathsf{VC}$ *is (perfectly)* correct *if for all* $N = \mathsf{poly}(\lambda)$*, all oracles* $H$ *and all* $\lambda \in \mathbb{N}$*:*

$$\forall\, \mathsf{crs} \leftarrow \mathsf{Setup}^H(1^\lambda, N),\ \forall\, (\mathsf{com}, \mathsf{decom}, (m_1, \ldots, m_N)) \leftarrow \mathsf{Commit}^H_{\mathsf{crs}}()$$
$$\forall\, I \subseteq [N],\ \forall\, \mathsf{decom}_I \leftarrow \mathsf{Open}^H_{\mathsf{crs}}(\mathsf{decom}, I) : \mathsf{Verify}^H_{\mathsf{crs}}(\mathsf{com}, \mathsf{decom}, I) = (m_i)_{i\in I}$$

**Definition 7 (Extractable-Binding).** *Let* $\mathsf{VC}$ *be a vector commitment in the CRS+RO-model with RO* $H$*. Let* $(\mathsf{TSetup}, \mathsf{Ext})$ *be PPT algorithms such that:*

– $\mathsf{TSetup}^H(1^\lambda, N) \to (\mathsf{crs}, \mathsf{td})$: *Given security parameter* $\lambda$ *and vector length* $N$*, output a commitment key* $\mathsf{crs}$ *and a trapdoor* $\mathsf{td}$*.*
– $\mathsf{Ext}(\mathsf{td}, Q, \mathsf{com}) \to (m_i)_{i\in[N]}$: *Given the trapdoor* $\mathsf{td}$*, a set of query-response pairs of random oracle queries, and a commitment* $\mathsf{com}$*, output the committed messages. (*$\mathsf{Ext}$ *may output* $m_i = \perp$*, e.g. if committed value at index* $i$ *is invalid.)*

*For any* $N = \mathsf{poly}(\lambda)$*, define the straightline extractable-binding game for* $\mathsf{VC}$ *and stateful adversary* $\mathcal{A}$ *as follows:*

1. $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TSetup}^H(1^\lambda, N)$
2. $\mathsf{com} \leftarrow \mathcal{A}^H(1^\lambda, \mathsf{crs})$
3. $(m_1^*, \ldots, m_N^*) = \mathsf{Ext}(\mathsf{td}, Q, \mathsf{com})$*, where* $Q$ *is the set* $\{(x_i, H(x_i))\}$ *of query-response pairs of queries* $\mathcal{A}$ *made to* $H$*.*
4. $((m_i)_{i\in I}, \mathsf{decom}_I, I) \leftarrow \mathcal{A}^H(\mathsf{open})$
5. *Output* 1 *(success) if* $\mathsf{Verify}^H_{\mathsf{crs}}(\mathsf{com}, \mathsf{decom}_I, I) = (m_i)_{i\in I}$ *but* $m_i \neq m_i^*$ *for any* $i \in I$*. Else output* 0 *(failure).*

*We say* $\mathsf{VC}$ *is* straightline extractable *w.r.t.* $(\mathsf{TSetup}, \mathsf{Ext})$ *if*

1. $\{\mathsf{crs} \mid \mathsf{crs} \leftarrow \mathsf{Setup}^H(1^\lambda, N)\}$ *and* $\{\mathsf{crs} \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TSetup}^H(1^\lambda, N)\}$ *are computationally indistinguishable for any* $N = \mathsf{poly}(\lambda)$*. We denote the advantage of a distinguisher* $\mathcal{A}$ *by* $\mathsf{AdvDist}^{\mathsf{Setup},\mathsf{TSetup}}_{\mathcal{A}}$*.*
2. *Any PPT adversary* $\mathcal{A}$ *has negligible probability to win the extractable binding game. We denote the advantage, i.e. probability to win, by* $\mathsf{AdvEB}^{\mathsf{VC}}_{\mathcal{A}}$*.*

The definition of the hiding property for $\mathsf{VC}$ forces all unopened components of the message vector to be independent uniform (pseudo-)random elements.

**Definition 8 (Hiding (real-or-random)).** *Let* $\mathsf{VC}$ *be a vector commitment scheme in the CRS+RO-model with random oracle* $H$*. The* adaptive hiding *experiment for* $\mathsf{VC}$ *with* $N = \mathsf{poly}(\lambda)$ *and stateful* $\mathcal{A}$ *is defined as follows.*

1. $\mathsf{crs} \leftarrow \mathsf{Setup}^H(1^\lambda, N)$*,* $b^* \leftarrow \{0,1\}$
2. $(\mathsf{com}, \mathsf{decom}, (m_1^*, \ldots, m_N^*)) \leftarrow \mathsf{Commit}^H_{\mathsf{crs}}()$
3. $I \leftarrow \mathcal{A}^H(1^\lambda, \mathsf{crs}, \mathsf{com})$
4. $\mathsf{decom}_I \leftarrow \mathsf{Open}_{\mathsf{crs}}(\mathsf{decom}, I)$
5. $m_i \leftarrow m_i^*$ *for* $i \in I$*.*
6. *For* $i \notin I$ *set* $m_i \leftarrow \begin{cases} m_i^* & \text{if } b^* = 0 \\ \text{random from } \mathcal{M} & \text{if } b^* = 1 \end{cases}$
7. $b \leftarrow \mathcal{A}((m_i)_{i \in [N]}, \mathsf{decom}_I)$*.*
8. *Output* $1$ *(success) if* $b = b^*$*, else* $0$ *(failure).*

*In the* selective hiding *experiment,* $\mathcal{A}$ *must choose* $I$ *prior to receiving* $\mathsf{com}$*, i.e. steps 2 and 3 are swapped but* $\mathcal{A}$ *still learns* $\mathsf{crs}$*.*

*The advantage* $\mathsf{AdvSelHide}^{\mathsf{VC}}_{\mathcal{A}}$ *(resp.* $\mathsf{AdvAdpHide}^{\mathsf{VC}}_{\mathcal{A}}$*) of an adversary* $\mathcal{A}$ *is defined by* $\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}$ *in the selective (resp. adaptive) hiding experiment. We say* $\mathsf{VC}$ *is selectively (resp. adaptively)* hiding *if every PPT adversary* $\mathcal{A}$ *has negligible advantage.*

### 2.4 Extractable Functions

In analogy to extractable commitments, we define extractable function families.

**Definition 9.** *A* function (family) *in the CRS+RO model is a tuple* $(\mathsf{Setup}, \mathsf{Eval})$ *of PPT algorithms where*

- $\mathsf{Setup}^H(1^\lambda) \to crs$*: Given the security parameter* $\lambda$ *generate a CRS* $crs$*.*
- $\mathsf{Eval}^H_{crs}(x) \to y$*: Given a CRS* $crs$ *and input* $x \in \mathcal{X}_{crs}$*, output* $y$*. Here,* $\mathcal{X}_{crs}$ *is the efficiently recognizable domain of* $\mathsf{Eval}_{crs}$*.*

In the rest of this work, we will usually write $\mathsf{F}$ instead of $(\mathsf{Setup}, \mathsf{Eval})$ and $\mathsf{F}(x)$ instead of $\mathsf{F}.\mathsf{Eval}_{crs}(x)$. The definition of a *(straightline) extractable* function (family) $(\mathsf{Setup}, \mathsf{Eval})$ is analogous to extractable $\mathsf{VC}$. Again, there is a pair $(\mathsf{TSetup}, \mathsf{Ext})$ of trapdoor setup and straightline extractor algorithms. The adversary's goal is to find a value $y$ such that the extracted preimage $x'$ differs from the preimage $x$ which the adversary provides. (That is essentially the same as extractable binding, but with preimages instead of decommitments.) We refer to Appendix A.3 for more formal definitions.

## 3 Compiling $\binom{N}{N-1}$-OT-based zero-knowledge protocols

We begin by constructing a compiler that replaces random OT instances with random vector commitments $\mathsf{VC}$, where all but one committed (random) value is opened to the verifier. This will later allow us to design our protocols in the (more natural) random OT-hybrid model. Moreover, we believe that presenting this compilation step separately is of independent interest.

Towards achieving this, we first construct an efficient $\mathsf{VC}$ construction using a tree PRG, similar to prior works such as [KKW18, BdK+21]. We then describe which properties a protocol must have such that our compiler can be applied and specify which flavor of OT it must use. We then describe the compiler, which directly replaces every call to an Oblivious Transfer instance to an appropriate $\mathsf{VC}$ instance instead, and show that the resulting protocol is still a ZKPoK.

### 3.1 Tree-PRG Vector Commitments

We now give a $\mathsf{VC}$ construction with all-but-one openings, which we later directly use in our compiler. $\mathsf{VC}$ will make commitments to $N$ random seeds, where $N-1$ are later opened. Towards optimizing openings, we generate the seeds via a GGM tree of length-doubling PRGs, similarly to previous works [KKW18, BdK+21].

Let $\mathsf{PRG}\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG, $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$ be a collision-resistant hash function (CRHF), $\mathsf{G}\colon \{0,1\}^\lambda \to \{0,1\}^\lambda \times \{0,1\}^{2\lambda}$ be a PRG and CRHF and $N = 2^d$. We define the scheme $\mathsf{VC}_{\mathsf{GGM}}$ below. If $\mathsf{H}$ and $\mathsf{G}$ are instantiated as (independent) random oracles, then $\mathsf{VC}_{\mathsf{GGM}}$ is (secure) in the RO model (without CRS). If $\mathsf{H}$ is replaced by the identity function and $\mathsf{G}$ is an injective trapdoor function, then $\mathsf{VC}_{\mathsf{GGM}}$ is (secure) in the CRS model.

- $\mathsf{Setup}^H(1^\lambda, N = 2^d)$:
  1. Compute $\mathsf{crs}_\mathsf{G} \leftarrow \mathsf{G.Setup}(1^\lambda)$ resp. $\mathsf{crs}_\mathsf{H} \leftarrow \mathsf{H.Setup}(1^\lambda)$.
  2. Define $\mathsf{crs} = (\lambda, d, \mathsf{crs}_\mathsf{G}, \mathsf{crs}_\mathsf{H})$, which is implicitly input to all other algorithms. Moreover, $\mathsf{crs}_\mathsf{G}$ (resp. $\mathsf{crs}_\mathsf{H}$) are implicit inputs to $\mathsf{G}$, (resp. $\mathsf{H}$).
- $\mathsf{Commit}()$:
  1. Sample $k \leftarrow \{0,1\}^\lambda$ and let $k_0^0 \leftarrow k$
  2. For each level $i \in [d]$, for $j \in [0..2^{i-1})$, compute $(k_{2j}^i, k_{2j+1}^i) \leftarrow \mathsf{PRG}(k_j^{i-1})$
  3. Let $(\mathsf{sd}_0, \dots, \mathsf{sd}_{N-1}) \leftarrow (k_0^d, \dots, k_{N-1}^d)$
  4. Compute $(m_i, \overline{\mathsf{com}}_i) \leftarrow \mathsf{G}(\mathsf{sd}_i)$, for $i \in [0..N)$
  5. Compute $h \leftarrow \mathsf{H}(\overline{\mathsf{com}}_0, \dots, \overline{\mathsf{com}}_{N-1})$
  6. Output the commitment $\mathsf{com} = h$, the opening $\mathsf{decom} = k$ and the messages $(m_0, \dots, m_{N-1})$
- $\mathsf{Open}(\mathsf{decom} = k, I = [0..N) \setminus \{j^*\})$: (where $j^* \in [0..N)$):
  1. Write $j^* = \sum_{i=0}^{d-1} 2^i b_i$, for $b_i \in \{0,1\}$
  2. Define the prefixes of $j^*$ as $j^*|_i \leftarrow \sum_{k=0}^{i-1} 2^k b_k$ for $i \in [1..d]$
  3. Recompute $k_j^i$, for $i \in [d]$ and $j \in [0..2^i)$ as in $\mathsf{Commit}$
  4. Output the opening information $\mathsf{decom}_I = (\overline{\mathsf{com}}_{j^*}, \{k^i_{2j^*|_i + \bar{b}_i}\}_{i \in [d]})$
- $\mathsf{Verify}(\mathsf{com} = (h), \mathsf{decom}_I = (\overline{\mathsf{com}}_{j^*}, \{k^i_{2j^*|_i + \bar{b}_i}\}_{i \in [d]}), I = [0..N) \setminus \{j^*\})$:
  1. Recompute $\mathsf{sd}_i$ from $\mathsf{decom}_I$, for $i \neq j^*$, and compute $(m'_i, \overline{\mathsf{com}}'_i) \leftarrow \mathsf{G}(\mathsf{sd}_i)$
  2. Let $\overline{\mathsf{com}}'_{j^*} = \overline{\mathsf{com}}_{j^*}$
  3. If $h \neq \mathsf{H}(\overline{\mathsf{com}}'_0, \dots, \overline{\mathsf{com}}'_{N-1})$ output $\perp$. Otherwise output $(m'_i)_{i \in I}$

We denote the above (all-but-one) vector commitment scheme by $\mathsf{VC}_{\mathsf{GGM}}$. Clearly, $\mathsf{VC}_{\mathsf{GGM}}$ is perfectly correct. We prove that it is extractable-binding and hiding in the two lemmas below. Note that we use 1-based indexing in the definition of $\mathsf{VC}_{\mathsf{GGM}}$, because it is more suitable here than the 1-based indexing used in general $\mathsf{VC}$ definitions.

*Remark 3 (Optimizations).* Instead of hashing the $\overline{\mathsf{com}}_i$'s, they could be sent in the clear as well. However, for the equality check in $\mathsf{Verify}$, a (extractable) collision-resistant hash function $\mathsf{H}$ is sufficient. If multiple $\mathsf{VC}_{\mathsf{GGM}}$ commitments are made in parallel, one can hash the all $\overline{\mathsf{com}}_i^{(j)}$ where $j$ ranges over the parallel instances, or hash the hashes $\mathsf{com}^i$, further reducing communication.

**Lemma 1.** *Decompose* $\mathsf{G}\colon \{0,1\}^\lambda \to \{0,1\}^\lambda \times \{0,1\}^{2\lambda}$ *along the outputs into* $(\mathsf{G}_1, \mathsf{G}_2)$*, such that* $\mathsf{G}(x) = (\mathsf{G}_1(x), \mathsf{G}_2(x))$*. Suppose* $\mathsf{G}_2$ *and* $\mathsf{H}$ *are straightline extractable. Then* $\mathsf{VC}_{\mathsf{GGM}}$ *is straightline extractable-binding. More concretely, for any adversaries* $\mathcal{D}$*,* $\mathcal{A}$ *there exist adversaries* $\mathcal{D}_\mathsf{G}$*,* $\mathcal{D}_\mathsf{H}$*, resp.* $\mathcal{A}_\mathsf{G}$*,* $\mathcal{A}_\mathsf{H}$ *with roughly the same running time as* $\mathcal{D}$ *resp.* $\mathcal{A}$*, such that*

$$\mathsf{AdvDist}_{\mathcal{D}}^{\mathsf{VC}_{\mathsf{GGM}}.\mathsf{Setup},\mathsf{VC}_{\mathsf{GGM}}.\mathsf{TSetup}} \leq \mathsf{AdvDist}_{\mathcal{D}_\mathsf{G}}^{\mathsf{G}.\mathsf{Setup},\mathsf{G}.\mathsf{TSetup}} + \mathsf{AdvDist}_{\mathcal{D}_\mathsf{H}}^{\mathsf{H}.\mathsf{Setup},\mathsf{H}.\mathsf{TSetup}}$$
$$\mathsf{AdvEB}_{\mathcal{A}}^{\mathsf{VC}_{\mathsf{GGM}}} \leq N \cdot \mathsf{AdvExt}_{\mathcal{A}_\mathsf{G}}^{\mathsf{G}} + \mathsf{AdvExt}_{\mathcal{A}_\mathsf{H}}^{\mathsf{H}}$$

*where* $N = 2^d$ *is the vector length of* $\mathsf{VC}_{\mathsf{GGM}}$ *and the advantages are defined in Definition 23 and Definition 7.*

*Proof (Sketch).* The extraction algorithms $(\mathsf{TSetup}, \mathsf{Ext})$ are defined in the obvious way: $\mathsf{TSetup}$ uses the trapdoor setups for $\mathsf{G}$ (i.e. $\mathsf{G}_2$) and $\mathsf{H}$. Given a commitment $\mathsf{com} = h$, $\mathsf{Ext}$ first extracts the preimage $(\overline{\mathsf{com}}_0, \dots, \overline{\mathsf{com}}_{N-1})$ of $h$ under $\mathsf{H}$, and then extracts preimages $\mathsf{sd}_i$ for each of the $\overline{\mathsf{com}}_i$ under $\mathsf{G}_2$. Now, we derive the claimed advantages.

For indistinguishability of the setup, first replace the honest setup of $\mathsf{G}$ with the trapdoor setup, and then do the same for $\mathsf{H}$. These are direct reductions.

For extractability, observe that the extractor $\mathsf{Ext}$ of $\mathsf{VC}_{\mathsf{GGM}}$ only fails if:

- Extracting $\mathsf{H}$ for $\mathsf{com} = h$ yielded $(\overline{\mathsf{com}}_0, \dots, \overline{\mathsf{com}}_{N-1})$ (or $\perp$), but the adversary presented a (different) preimage during an opening of $\mathsf{com}$.
- Extracting $\mathsf{G}_2$ for some $\overline{\mathsf{com}}_i$ yielded $\mathsf{sd}_i$ (or $\perp$), but the adversary presented a (different) preimage during an opening of $\mathsf{com}$.

In both cases, the reduction to the extractable-binding property of $\mathsf{H}$ (resp. $\mathsf{G}_2$) is a straightforward (hybrid/guessing) argument. ◻

**Lemma 2.** *Suppose* $\mathsf{PRG}$ *and* $\mathsf{G}$ *are PRGs. Then* $\mathsf{VC}_{\mathsf{GGM}}$ *is all-but-one selectively hiding. More precisely, for any adversary* $\mathcal{A}$ *there exist adversaries* $\mathcal{A}_{\mathsf{PRG}}$*,* $\mathcal{A}_\mathsf{G}$*, with roughly the same running time as* $\mathcal{A}$*, such that* $\mathsf{AdvSelHide}_{\mathcal{A}_{\mathsf{PRG}}}^{\mathsf{VC}_{\mathsf{GGM}}} \leq d \cdot \mathsf{AdvPRG}_{\mathcal{A}_{\mathsf{PRG}}}^{\mathsf{PRG}} + \mathsf{AdvPRG}_{\mathcal{A}_\mathsf{G}}^{\mathsf{G}}$*, where* $N = 2^d$*.*

*Proof (Sketch).* In the all-but-one selective hiding experiment, the adversary's index vector $I = [N] \setminus i$ is chosen prior to commitment generation. Using knowledge of $i$, we can rely on the GGM construction being a secure puncturable PRF [BW13]. The respective security reduction replaces $\mathsf{PRG}$ calls along the path of $i$ to the root of the GGM construction by true randomness. This is possible since only the seeds for the co-path nodes for $i$ are given out as part of the commitment. Hence, a hybrid with $d = \log(N)$ steps replaces $\mathsf{sd}_i$ by a truly random output. In a final step, using that $\mathsf{G}$ is a PRG, and $\mathsf{G}(\mathsf{sd}_i)$ is a PRG output where $\mathsf{sd}_i$ is truly random and unknown to the adversary, we can replace the committed value $m_i$ by a truly random committed value. ◻

### 3.2 The Compiler

We now present our compiler, which takes ZKPoKs in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model which satisfy certain properties and replaces the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$ calls with $\mathsf{VC}$ instances. The $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$ functionality is given in Figure 1. Intuitively, our compiler uses the same approach as [CDD+19], that is, it replaces OTs by mere commitments.

To receive an OT output, the compiled verifier reveals its choice bit to the prover, which then opens the commitments as appropriate. Clearly, this limits the protocols which can be securely

**Functionality** $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$

The functionality interacts with a sender $\mathcal{P}$, a receiver $\mathcal{V}$ and an adversary $\mathcal{A}$ which may corrupt either of the parties.
It is parametrized by integers $N$ (number of choices) and $\ell$ (number of parallel instances).

1. Upon receiving $(\mathsf{init})$ from $\mathcal{P}$: For $i \in [\ell]$
   - Sample $F^i \leftarrow (\{0,1\}^\lambda)^{[N]}$.
   - If $\mathcal{P}$ is corrupted, receive $F^i \in (\{0,1\}^\lambda)^{[N]}$ from $\mathcal{A}$.
   - If $\mathcal{V}$ is corrupted, receive $x^i \in [N]$, $F^{i,*} \in (\{0,1\}^\lambda)^{[N]\setminus\{x^i\}}$ from $\mathcal{A}$ and set $F^i(x) = F^{i,*}(x)$ for all $x \in [N] \setminus \{x^i\}$, $i \in [\ell]$.
2. Send $F^1, \ldots, F^\ell$ to $\mathcal{P}$ and $(\mathsf{done})$ to $\mathcal{V}$.
3. Upon receiving $(\mathsf{get}, (x^1, \ldots, x^\ell))$ from $\mathcal{V}$, send $((F^i(x))_{x \neq x^i})_{i=1,\ldots,\ell}$ to $\mathcal{V}$.

**Figure 1.** $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ functionality adapted from SoftSpokenOT [Roy22]

compiled. Namely, the verifier's actions should not depend on any intermediate OT outputs, so that all choice bit queries can be delayed to the very end of the protocol, as part of a final verification step. (Otherwise, a malicious prover would gain the power to make its responses dependent on choice bits, making typical protocols completely insecure[8].) To formalize this protocol structure, we use a functionality $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$ which outputs $\mathsf{done}$ to the OT receiver when inputs are provided by the OT sender, instead of the value. Only later, by sending $\mathsf{get}$ to $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$, the actual value can be obtained by the OT receiver.

**Definition 10.** *A ZKPoK $(\mathcal{P}, \mathcal{V})$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model is* OT-admissible*, if the following holds:*

1. *The prover $\mathcal{P}$ always plays the role of the sender in $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$.*
2. *The verifier $\mathcal{V}$ can be split into two phases $\mathcal{V}_1$, $\mathcal{V}_2$, where:*
   - $\mathcal{V}_1$*(inputs)* never *sends* $\mathsf{get}$ *to* $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$ *(and outputs a state* $\mathtt{state}$ *for* $\mathcal{V}_2$*).*
   - $\mathcal{V}_2(\mathtt{state})$ only *sends* $\mathsf{get}$ *to* $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$ *(and outputs the verdict).*

**Compiler.** We describe the compiler under the assumption that the prover (resp. verifier) never sends multiple messages/choice bits or $\mathsf{get}$ to the same $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ instance, as they would be ignored by $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ (and hence may be ignored by the compiler). By $\mathsf{O2C}[\Pi]$, we denote the result of protocol compilation, where $\Pi$ is an interactive proof system in the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model. We also write $(\mathcal{P}_{\mathsf{O2C}}, \mathcal{V}_{\mathsf{O2C}}) \coloneqq \mathsf{O2C}[(\mathcal{P}, \mathcal{V})]$ to denote the compiled prover and verifier. We make the assumption that each $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ instance has a unique identifier *sid*.

*Changes to any setup.*
1. A trusted entity securely runs $\mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, N)$ and sends it to $\mathcal{P}_{\mathsf{O2C}}, \mathcal{V}_{\mathsf{O2C}}$.
2. Additionally, run any setup that the original protocol may require.

*The compiled prover $\mathcal{P}_{\mathsf{O2C}}$.*
- Let $\mathcal{P}_{\mathsf{O2C}}$ run a copy of $\mathcal{P}$ and forward all messages of $\mathcal{P}$, except for messages to/from $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-instances.

[8] As an example, MPC-in-the-head approaches [IKOS08, IPS08] leaking their watch-lists during execution allow a cheating prover to specifically maul unopened parties.

– When $\mathcal{P}$ would send $(\mathsf{init})$ to $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ with identifier $sid$, $\mathcal{P}_{\mathsf{O2C}}$ instead first runs $(\mathsf{com}^i_{sid}, \mathsf{decom}^i_{sid}, (m^i_1, \ldots, m^i_N)) = \mathsf{Commit}()$ for $i \in [\ell]$ and returns the functions $(F^i)_{i\in[\ell]}$ to $\mathcal{P}$ where $F^i(x) = m^i_x$. Then $\mathcal{P}_{\mathsf{O2C}}$ sends $(sid, \mathsf{init}, (\mathsf{com}^i_{sid})_{i\in\ell})$ to $\mathcal{V}_{\mathsf{O2C}}$.
– Upon receiving $(sid, \mathsf{get}, (x^1, \ldots, x^\ell))$ from $\mathcal{V}_{\mathsf{O2C}}$, with $x^i \in [N]$, $i \in [\ell]$, compute $\mathsf{decom}^i_{sid,x} = \mathsf{Open}(\mathsf{decom}^i_{sid}, [N] \setminus \{x^i\})$ and send $(sid, \mathsf{get}, (\mathsf{decom}^i_{sid,x})_{i\in[\ell]})$ to $\mathcal{V}_{\mathsf{O2C}}$.

*Changes to the verifier $\mathcal{V}$.*

– Let $\mathcal{V}_{\mathsf{O2C}}$ run a copy of $\mathcal{V}$ and forward the messages of $\mathcal{V}$, except for messages to/from $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-instances.
– Upon receiving $(sid, \mathsf{init}, (\mathsf{com}^i)_{i\in[\ell]})$ from $\mathcal{P}_{\mathsf{O2C}}$, store $\mathsf{com}^i$ as $\mathsf{com}^i_{sid}$. Then pass $(\mathsf{done})$ on to the simulated $\mathcal{V}$ in place of $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ with identifier $sid$.
– Upon $\mathcal{V}$ sending $(\mathsf{get}, (x^i_{sid})_{i\in[\ell]})$ to $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ with identifier $sid$, send $(sid, \mathsf{get}, (x^i_{sid})_{i\in[\ell]})$ to $\mathcal{P}_{\mathsf{O2C}}$.
– Upon receiving $(sid, \mathsf{get}, (\mathsf{decom}^i_{sid,x})_{i\in[\ell]})$, for $i \in [\ell]$, compute the following messages $\mathtt{out}^i \leftarrow \mathsf{Verify}(\mathsf{com}^i_{sid}, \mathsf{decom}^i_{sid,x}, [N] \setminus \{x^i_{sid}\})$. If $\mathtt{out}^i = \perp$ for any $i \in [\ell]$ then reject. Else, output $\mathtt{out}^i = (m^i_x)_{x\in[N]\setminus\{x^i_{sid}\}}$ as the function values of $F$ to $\mathcal{V}$.

*Remark 4.* The compiler $\mathsf{O2C}$ preserves public-coin verifiers.

*Remark 5 (Public-coin $\implies$ OT-admissible).* Any public-coin verifier is automatically OT-admissible. To see this, observe that a public-coin verifier by definition chooses challenges which are independent of the OT outputs, hence any call to $\mathsf{get}$ can be delayed until the final response is received, and then all OT outputs are gathered for $\mathsf{Verify}(\mathsf{view}_{\mathcal{V}})$. Clearly, this "modification" does not affect $\mathcal{V}$'s visible behaviour in any way.

**Security of the Compiler for Interactive Protocols.** For interactive zero-knowledge protocols $\Pi$ in the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model, our compiler is able to translate them into protocols with almost identical security parameters for SHVZK[9] and knowledge soundness, essentially by replacing $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ instances with vector commitments (in the CRS and/or RO model).

**Lemma 3.** *Suppose the commitment scheme* $\mathsf{VC}$ *used in* $\mathsf{O2C}$ *is perfectly correct, straightline extractable-binding and real-or-random hiding in the CRS+RO model. Let $\Pi$ be an OT-admissible proof system. Then, in the CRS+RO model,* $\mathsf{O2C}[\Pi]$*, when compiled with* $\mathsf{VC}$ *as specified above, satisfies the following properties.*

– *Correctness, if $\Pi$ is correct.*
– *Public-coin, if $\Pi$ is public-coin.*
– *Knowledge soundness, with asymptotic error $\kappa(\lambda) + \mathsf{negl}(\lambda)$, if $\Pi$ has a black-box knowledge extractor $\mathcal{E}_\Pi$ with knowledge error $\kappa(\lambda)$.*
– *SHVZK, if $\Pi$ has a black-box SHVZK simulator $\mathcal{S}_\Pi$.*

*Moreover, if $\Pi$ has a straightline extractor or a special extractor, so has* $\mathsf{O2C}[\Pi]$*.*

[9] We note that a modification of the compiler, which additionally forces the verifier to commit to its OT choices (with an extractable-binding commitment scheme), yields full zero-knowledge. However, this modification does not preserve public-coin.

*Proof.* Let $\Pi = (\mathcal{P}, \mathcal{V})$ and $\mathsf{O2C}[\Pi] = (\mathcal{P}_{\mathsf{O2C}}, \mathcal{V}_{\mathsf{O2C}})$.

CORRECTNESS. Observe that replacing the outputs $F^i$ chosen by an instance of $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ by pseudo-random outputs generated as $\mathsf{VC}$ outputs is an indistinguishable change (in the honest protocol execution), by a direct reduction to the real-or-random hiding property. More precisely, the hiding property asserts that all *unopened* commitments are (pseudo-)random, and via a hybrid argument over the messages indices $1, \ldots, N$, one shows that given only the random messages $m_i$ from $\mathsf{VC}$ *without* any decommitments, the tuple $(m_1, \ldots, m_N)$ is pseudo-random. Therefore, the correctness error of $\mathsf{O2C}[\Pi]$ differs by a negligible amount from the correctness error of $\Pi$. If $\Pi$ is perfectly complete, so is $\mathsf{O2C}[\Pi]$.

SPECIAL HONEST-VERIFIER ZERO-KNOWLEDGE. By assumption, there exists a black-box simulator $\mathcal{S}_\Pi$ for special honest verifiers for the original protocol $\Pi$. We now describe how to translate a distinguishing attack of a corrupted $\widehat{\mathcal{V}}_{\mathsf{O2C}}$ on an honest $\mathcal{P}_{\mathsf{O2C}}$ in $\mathsf{O2C}[\Pi]$ into an attack of a corrupted $\hat{\mathcal{V}}$ on an honest $\mathcal{P}$ in $\Pi$. For this, we construct a simulator $\mathcal{S}_{\mathsf{O2C}}$ for the compiled protocol as follows:

1. Initially, $\mathcal{S}_{\mathsf{O2C}}$ will generate the commitment key $\mathsf{crs}$ by running $\mathsf{Setup}^H$ to generate $\mathsf{crs}$.
2. Then $\mathcal{S}_{\mathsf{O2C}}$ runs the simulator $\mathcal{S}_\Pi$ for the SHVZK adversary $\hat{\mathcal{V}} = (\mathcal{V}, \mathcal{V}'')$ against $\Pi$, where $\hat{\mathcal{V}}$ is defined as follows:
   – $\mathcal{V}$ is the honest verifier, as required for SHVZK adversaries (cf. Definition 4). Suppose the random/auxiliary tape of $\widehat{\mathcal{V}}_{\mathsf{O2C}}$ is interpreted such that the choice bits are $x^i$ for $i = 1, \ldots, \ell$.
   – $\mathcal{V}''$ programs the $F^i$-values of $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$. This means that $\mathcal{V}''$ executes, for $i = 1, \ldots, \ell$,
   $$(\mathsf{com}^i, \mathsf{decom}^i, (F^i(1), \ldots, F^i(N))) \leftarrow \mathsf{Commit}^H_{crs}(),$$
   in order to generate the prover's inputs $F^i$ to be used in $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ (except for $F^i(x^i)$, which the functionality chooses randomly).
3. After obtaining the view $\mathsf{view}_{\hat{\mathcal{V}}}$, $\mathcal{S}_{\mathsf{O2C}}$ transforms it into the view $\mathsf{view}_{\widehat{\mathcal{V}}_{\mathsf{O2C}}}$ for the protocol $\mathsf{O2C}[\Pi]$. This is done by replaying $\hat{\mathcal{V}}$ to recover
   – the $\mathsf{VC}$ commitments $\mathsf{com}$ to reconstruct $(\mathsf{init}, \mathsf{com}^1, \ldots, \mathsf{com}^\ell)$ (purportedly sent by $\mathcal{P}_{\mathsf{O2C}}$ in $\mathsf{view}_{\widehat{\mathcal{V}}_{\mathsf{O2C}}}$).
   – the $\mathsf{VC}$ decommitments $(\mathsf{get}, \mathsf{decom}^1, \ldots, \mathsf{decom}^\ell)$ (purportedly sent by $\mathcal{P}_{\mathsf{O2C}}$ in $\mathsf{view}_{\widehat{\mathcal{V}}_{\mathsf{O2C}}}$).

   By construction, the choice bits that $\hat{\mathcal{V}}$ uses and those of $\widehat{\mathcal{V}}_{\mathsf{O2C}}$ coincide. Moreover, by correctness of $\mathsf{VC}$, the prover's decommitments are accepted by $\widehat{\mathcal{V}}_{\mathsf{O2C}}$.[10] Thus, the (re)construction of a view of $\widehat{\mathcal{V}}_{\mathsf{O2C}}$ succeeds.

Clearly, $\mathcal{S}_{\mathsf{O2C}}$ runs in polynomial time if $\mathcal{S}_\Pi$ runs in polynomial time. We argue indistinguishability via hybrid games as follows.

The first game $\mathsf{G}_1$ is the real protocol $\mathsf{O2C}[\Pi]$ running with the special honest verifier $\widehat{\mathcal{V}}_{\mathsf{O2C}}$. (The output is the verifier's view).

In the second game $\mathsf{G}_2$, for the $\ell$-parallel invocations of $\mathsf{VC}$ for a compiled $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ call of $\mathcal{P}_{\mathsf{O2C}}$ we output a truly random $F^i(x^i) = m_{x^i}$ to $\mathcal{P}_{\mathsf{O2C}}$, that it uses throughout the protocol. Here $(x^i)_{i \in [\ell]}$ are the honest verifier's choice bits in this call. Since by construction of the compiled verifier $\mathcal{V}_{\mathsf{O2C}}$,

[10] If $\mathsf{VC}$ is not perfectly correct, this must be taken into account. E.g., by adapting $\hat{\mathcal{V}}$ to deal with it (as it can predict which index is opened).

all $x^i$ are fixed beforehand in the SHVZK setting, this reduces to selective real-or-random hiding of $\mathsf{VC}$ via a straightforward sequence of hybrids. We find

$$\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1] \leq \mathsf{poly}(\lambda) \cdot \ell(\lambda) \cdot \mathsf{AdvSelHide}^{\mathsf{VC}}_{\mathcal{A}'}(\lambda)$$

for suitable hybrid adversary $\mathcal{A}'$, where $\mathsf{poly}$ bounds the number of sessions/instances of $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ used in $\Pi$, and hence $\mathsf{poly} \cdot \ell$ bounds the total number of $\mathsf{VC}$ instances in the hybrid argument.

In the third game $\mathsf{G}_3$, we consider the SHVZK verifier $\hat{\mathcal{V}}$ for $\Pi$ described in the simulation above, instead of using $\widehat{\mathcal{V}}_{\mathsf{O2C}}$. More precisely, the special fixed randomness of $\hat{\mathcal{V}}$ is just that of $\widehat{\mathcal{V}}_{\mathsf{O2C}}$. The prover's commitments are generated by $\hat{\mathcal{V}}$. And the output of $\mathsf{G}_3$ is the translation of the view of $\hat{\mathcal{V}}$ to a view of $\widehat{\mathcal{V}}_{\mathsf{O2C}}$ in the same way as outlined above for the simulator. This change is merely conceptual, and thus, $\mathsf{G}_2$ and $\mathsf{G}_3$ have identically distributed output.[11] Also observe that $\mathsf{G}_3$ essentially corresponds to the real distribution of the SHVZK game for $\hat{\mathcal{V}}$ against protocol $\Pi$.

Finally $\mathsf{G}_4$ is using the simulator $\mathcal{S}_\Pi$ to generate messages instead of the implicit execution of $\Pi$ with $\hat{\mathcal{V}}$. Indistinguishability of $\mathsf{G}_3$ and $\mathsf{G}_4$ follows by security of the SHVZK simulator $\mathcal{S}_\Pi$. Hence $\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_4 = 1] \leq \mathsf{AdvZK}^{\Pi}_{(\mathcal{A},\hat{\mathcal{V}})}(\lambda)$.

Strictly speaking, we still need to show that this simulation composes sequentially. However, since the SVHZK simulator does not use any trapdoors, this is automatic. Overall, we find that if $\mathcal{A}$ makes at most $Q_{\mathcal{S}}$ queries to the oracle $\mathcal{O}$ in the zero-knowledge game, then by a hybrid argument

$$\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_4 = 1] \leq Q_{\mathcal{S}} \cdot \Big(\mathsf{poly}(\lambda) \cdot \ell(\lambda) \cdot \mathsf{AdvSelHide}^{\mathsf{VC}}_{\mathcal{A}'}(\lambda) + \mathsf{AdvZK}^{\Pi}_{(\mathcal{A},\hat{\mathcal{V}})}(\lambda)\Big)\,.$$

Knowledge Soundness. By assumption, there exists an knowledge-extractor $\mathcal{E}_\Pi$ for the original protocol $\Pi$ which is straightline and black-box. Similar to zero-knowledge, we translate an attack against $\mathsf{O2C}[\Pi]$ into an attack against $\Pi$ by instantiating a suitable adversary. We rely on the extractor $\mathsf{Ext}$ for $\mathsf{VC}$, which requires the CRS trapdoor and all random oracle queries. Since $\mathcal{E}_{\mathsf{O2C}}$ implements the CRS setup and learns all random oracle queries, this is trivial to provide. Intuitively, $\mathcal{E}_{\mathsf{O2C}}$ simply runs $\mathsf{VC.Ext}$ to extract the commitments and uses these values as input to $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$. With this, translation of $\widehat{\mathcal{P}}_{\mathsf{O2C}}$ to $\hat{\mathcal{P}}$ is straightforward.

1. Initially, $\mathcal{E}_{\mathsf{O2C}}$ for $\mathsf{O2C}[\Pi]$ runs $(crs, \mathsf{td}) \leftarrow \mathsf{TSetup}^H(1^\lambda, N)$ to create a trapdoored commitment key $crs$.
2. $\mathcal{E}_{\mathsf{O2C}}$ defines a malicious prover $\hat{\mathcal{P}}$ against $\Pi$ which runs the malicious prover $\widehat{\mathcal{P}}_{\mathsf{O2C}}$ against $\mathsf{O2C}[\Pi]$ in its head and acts as follows:
   – By abuse of formalism, we let $\hat{\mathcal{P}}$ access the random oracle $H$, even though it is defined in the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model (without other setups, in particular, without access to a RO). Formally, this is justified (and does not pose any problems) since we consider a *black-box* extractor $\mathcal{E}_\Pi$. Thus, we could "embed" the function table of $H$ (or the algorithm sampling $H$) within $\hat{\mathcal{P}}$, as this is opaque to $\mathcal{E}_\Pi$.
   – Whenever $\widehat{\mathcal{P}}_{\mathsf{O2C}}$ sends a message $m = (sid, \mathsf{init}, (\mathsf{com}^i)_{i\in[\ell]})$, then $\hat{\mathcal{P}}$ runs $\mathsf{VC.Ext}(\mathsf{td}, Q, \mathsf{com}^i)$ on all $i$ to obtain the committed functions $F^i$. Then $\hat{\mathcal{P}}$ sends $(\mathsf{init}, F^1, \dots, F^\ell)$ to $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$.
   – Whenever $\widehat{\mathcal{P}}_{\mathsf{O2C}}$ sends a message $m = (sid, \mathsf{get}, (\mathsf{decom}^i_{x_i})_{i\in[\ell]})$, then $\hat{\mathcal{P}}$ runs $\mathsf{VC.Verify}$ and aborts if verification succeeds, but the opened messages are different from the previously extracted messages. (Note that if $F^i(x) = \bot$, i.e., extraction failed, but $\mathsf{VC.Verify}$ passed, this also leads to an abort.) If verification succeeds, then $\hat{\mathcal{P}}$ allows delivery of the OT outputs.

[11] This is obvious if $\mathsf{VC}$ is perfectly correct, but if it isn't, then $\mathcal{S}_{\mathsf{O2C}}$ can correctly generate an aborting view given $\mathsf{view}_{\hat{\mathcal{V}}}$ during the translation process.

– All other messages of $\widehat{\mathcal{P}}_{\mathsf{O2C}}$ are forwarded by $\hat{\mathcal{P}}$.
3. Finally, $\mathcal{E}_{\mathsf{O2C}}$ outputs the witness (or $\perp$) which the extractor $\mathcal{E}_{\Pi}$ for $\Pi$ outputs when run on $\hat{\mathcal{P}}$.

We argue by game hops. Game $\mathsf{G}_1$ is the real game which outputs 1 if the verifier accepts.

In game $\mathsf{G}_2$, we replace the CRS. Note that since $\mathsf{VC}$ is extractable-binding, changing the CRS to a trapdoored key changes is indistinguishable, i.e. $\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1] \leq \mathsf{AdvDist}_{\mathcal{D}}^{\mathsf{Setup},\mathsf{TSetup}}$ for straightforward PPT $\mathcal{D}$.

In game $\mathsf{G}_3$, we extract all $\mathsf{VC}$ commitments made during a protocol run and upon $\mathsf{Verify}$ we check whether an unveiled commitment broke the extractable-binding property. If such a break happens (or extraction fails), the game aborts. The probability that $\mathsf{VC.Ext}$ fails or extractable-binding is broken (and hence game $\mathsf{G}_3$ aborts) can be bounded via a hybrid argument, reducing to the extractable-binding property. As the only changes between games $\mathsf{G}_2$ and $\mathsf{G}_3$ are aborts, we find $\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1] \leq \mathsf{poly} \cdot \mathsf{AdvEB}_{\mathcal{B}}^{\mathsf{VC}}$ for a straightforward hybrid PPT adversary $\mathcal{B}$, assuming $\mathsf{poly}$ upper-bounds the number of $\mathsf{VC}$ commitments made during an execution of $\widehat{\mathcal{P}}_{\mathsf{O2C}}$.

In game $\mathsf{G}_4$, we employ the extractor $\mathcal{E}_{\Pi}$ for $\Pi$, i.e. we now consider the actual the extraction procedure. This is the ideal game. Observe that $\hat{\mathcal{P}}$ convinces the verifier $\mathcal{V}$ if and only if $\widehat{\mathcal{P}}_{\mathsf{O2C}}$ convinces $\mathcal{V}_{\mathsf{O2C}}$. And the probability to convince is the probability that $\mathsf{G}_3$ outputs 1. Thus, by knowledge soundness of $\Pi$ the probability that $\mathcal{E}_{\Pi}$ outputs a witness is at most $\kappa$ less than the probability that $\hat{\mathcal{V}}$ outputs 1 in game $\mathsf{G}_3$, i.e. $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq \kappa$.

Overall, we find

$$\begin{aligned}\mathsf{AdvKE}_{\mathcal{A},\hat{\mathcal{P}}}^{\Pi,\mathcal{E}}(\lambda) &= \mathsf{Real}_{\mathcal{A}}(\lambda) - \mathsf{Ideal}_{\mathcal{A}}(\lambda)\\ &\leq \kappa + \mathsf{AdvDist}_{\mathcal{D}}^{\mathsf{Setup},\mathsf{TSetup}}(\lambda) + \mathsf{poly}(\lambda) \cdot \mathsf{AdvEB}_{\mathcal{B}}^{\mathsf{VC}}(\lambda).\end{aligned}$$

Since $\hat{\mathcal{P}}$ is PPT, the black-box extractor $\mathcal{E}_{\Pi}$, and hence $\mathcal{E}_{\mathsf{O2C}}$ is efficient.[12] Moreover, it is not hard to see that if $\mathcal{E}_{\Pi}$ is an straightline extractor, then $\mathcal{E}_{\mathsf{O2C}}$ can be made straightline by translating the messages directly, i.e. actively taking the role of $\hat{\mathcal{P}}$ in the protocol $\Pi$. Indeed, this even simplifies the setting. If $\mathcal{E}$ is a special extractor, so is $\mathcal{E}_{\Pi}$ (since the verifier's messages are never modified). □

## 4 Fiat–Shamir Transformation

In this section, we prove that a protocol $\Pi$ that has been compiled using the $\mathsf{O2C}[\Pi]$-transformation from Section 3 can securely be made non-interactive using the Fiat–Shamir transformation. For this, we reinterpret our protocols as a form of IOPs [BCS16, RRR16], which are multi-round extensions of PCPs. This allows us to apply the concept of round-by-round (RBR) soundness [CCH+19, CMS19] to analyze the security loss due to the Fiat-Shamir transform. We make this change of perspective to IOPs, because it is easier to define RBR knowledge in this setting, and the relation to RBR soundness from [CCH+19] will be clear.[13] The definition of RBR knowledge for IOPs from [CMS19] does not apply to our protocols (at least not immediately). Thus, we define and use on a different notion of RBR knowledge. To avoid a full-fledged definition of IOP parameters and security notions, we consider a simplified notion of IOP where the verifier is given *all PCP oracles in the plain* (instead of letting it query a limited number of positions). While this may sound insecure, we only use the

[12] Strictly speaking, we have to consider the implementation of the random oracle with which $\hat{\mathcal{P}}$ interacts to justify that. If the using the usual lazy sampling approach is used, efficiency is clear.

[13] Moreover, while possible, defining a notion of RBR knowledge in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{N,\ell}$-hybrid model makes things unwieldy.

IOP point-of-view to prove *soundness* of the Fiat–Shamir transformation. The zero-knowledge property is much simpler to see.

**Definition 11 (Interactive Oracle Proof (IOP)).** *A (public-coin)* interactive oracle proof *(IOP) [BCS16, RRR16] for NP-relation $\mathcal{R}$ is a pair of PPT ITMs $\Pi = (\mathcal{P}, \mathcal{V})$, which are defined as follows: The* prover *$\mathcal{P}$ throughout the protocol sends strings $\overline{m} = (m_i, f_i)$ in each round, where $m_i \in \{0,1\}^*$ is called* message *and $f_i\{0,1\}^*$ is called* oracle. *The* verifier *$\mathcal{V}$ learns $m_i$ in round $i$, but not $f_i$. It sends random challenges $\gamma_i \in \mathcal{C}_i$ in round $i$ in response. In a final step, the receiver learns all $f_i$ and outputs $b = \mathsf{Verify}(\mathbb{x}, ((m_0, \ldots, m_n), (f_0, \ldots, f_n), (\gamma_1, \ldots, \gamma_n)))$.*

*Remark 6 (From OT to IOP).* The mapping from a protocol in the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model with *corrupted prover* and *honest verifier* to an IOP is as follows: The prover, i.e. adversary, inputs $(F^1, \ldots, F^\ell) \in ((\{0,1\}^\lambda)^N)^\ell$ to $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ corresponding to an oracle $f_i$ of the IOP. The verifier's picks $(x^1, \ldots, x^\ell) \in [N]^\ell$ and looks at the corresponding values in $(f_0, \ldots, f_n)$ to compute the output bit. (Sequential (or parallel) sessions/instances of $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$ are handled in the obvious manner.)

**Definition 12 (Transcripts).** *We use following notation for partial transcripts of IOPs, namely, $\mathsf{tr} = (\overline{\boldsymbol{m}}, \boldsymbol{\gamma})$ where*

- *$\overline{\boldsymbol{m}} = (m_i, f_i)_{i=0}^{\ell}$ are the messages sent by the prover,*
- *$\boldsymbol{\gamma} = (\gamma)_{i=1}^{\ell}$ are the messages sent by the receiver.*

*We write $\mathsf{tr} \parallel (m, f)$ if the prover is about to move and sends $(m, f)$. Analogously, we write $\mathsf{tr} \parallel \gamma$ for the verifier. For a public-coin IOP, a transcript is called* full, *if the verifier outputs its verdict (and halts) given the transcript.*

**Definition 13.** *Let $(\mathcal{P}, \mathcal{V})$ be an IOP. A* bad challenge function $\mathsf{badch}$ *is a (deterministic not necessarily efficiently computable) function which takes as input a statement $\mathbb{x}$ and a partial transcript of an execution of $\Pi$, and outputs a bit such that the following holds:*

1. *$\mathsf{badch}(\mathbb{x}, (\emptyset, \emptyset)) = 0$, i.e. if no challenge is input to $\mathsf{badch}$, then it outputs $0$.*
2. *If $\mathsf{tr}$ is a prefix of $\mathsf{tr}'$, then $\mathsf{badch}(\mathbb{x}, \mathsf{tr}) = 1 \implies \mathsf{badch}(\mathbb{x}, \mathsf{tr}') = 1$, i.e. a bad challenge occurring is "a monotone event".*

*Moreover, we say that $\mathsf{badch}$ has* (round-by-round) error *$\kappa$ if for any transcript $\mathsf{tr}$ where the verifier moves next, we have*

3. *$\Pr[\mathsf{badch}(\mathbb{x}, \mathsf{tr} \parallel \gamma) = 1 \mid \mathsf{badch}(\mathbb{x}, \mathsf{tr}) = 0] \leq \kappa$, i.e. the probability that a good transcript turns bad is bounded by $\kappa$, where the probability is over $\gamma$.*

Our definition of round-by-round knowledge soundness asserts that, unless a bad challenge event occurred, an accepting verifier ensures that a witness can be extracted from the transcript.

**Definition 14.** *Let $(\mathcal{P}, \mathcal{V})$ be an IOP for $\mathcal{L}$ and $\mathsf{badch}$ be a bad challenge function. Then $(\mathcal{P}, \mathcal{V})$ has* round-by-round knowledge error *$\kappa$ with extractor $\mathsf{Ext}$ if*

- *$\mathsf{badch}$ has round-by-round error $\kappa$*
- *For every full transcript $\mathsf{tr}$, if $\mathsf{badch}(\mathbb{x}, \mathsf{tr}) = 0$ and $\mathcal{V}.\mathsf{Verify}(\mathbb{x}, \mathsf{tr}) = 1$, then $(\mathbb{x}, \mathsf{Ext}(\mathbb{x}, \mathsf{tr})) \in \mathcal{R}$.*
- *$\mathsf{Ext}$ is a PPT algorithm.*

*Remark 7.* If the IOP $\Pi = (\mathcal{P}, \mathcal{V})$ comes from reinterpreting a protocol in the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model as an IOP (Remark 6), then Definition 14 requires that $\Pi$ has a *special* extractor $\mathsf{Ext}$ (Definition 20).

For concreteness, we define one variant of the Fiat–Shamir transformation for which our results apply.

**Definition 15.** *Let $\Pi$ be a public-coin proof system. Then the* Fiat–Shamir transformation *of $\Pi$ with random oracle $H$ is denoted by $\mathsf{FS}[\Pi]$ and yields a NIZK $\mathsf{NIZK}$ which is defined as follows:*

- $\mathsf{NIZK.Setup}$ *is identical to $\Pi.\mathsf{Setup}$.*
- $\mathsf{NIZK.Prove}^H(\mathbb{x}, \mathbb{w})$ *runs the interactive prover $\Pi.\mathcal{P}^H(\mathbb{x}, \mathbb{w})$ and computes verifier challenges as $\gamma_{i+1} = H(\mathbb{x}, (m_0, \dots, m_i))$, where $m_0, \dots, m_1$ the messages the prover would send. The proof is $\pi = (m_0, \dots, m_\mu)$, assuming a $\mu$-round protocol.*
- $\mathsf{NIZK.Verify}^H(\mathbb{x}, \pi)$ *reconstructs the challenges $\gamma_{i+1} = H(\mathbb{x}, (m_0, \dots, m_i))$, where $\pi = (m_0, \dots, m_\mu)$. Then it constructs the interactive transcript $\mathsf{tr}$ and outputs $\Pi.\mathsf{Verify}(\mathbb{x}, \mathsf{tr})$.*

*If $\Pi$ is in the CRS+RO model, then we consider the obvious generalization of $\mathsf{FS}$, and we also include crs in the hashes, i.e. $\gamma_{i+1} = H(crs, \mathbb{x}, (m_0, \dots, m_i))$. Moreover, we generally want independent random oracles for the Fiat–Shamir transformation and $\Pi$.*

Instead of doing a modular analysis by (1) compiling an interactive proof system $\Pi$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model via $\mathsf{O2C}^{H_{\mathsf{O2C}}}[\Pi]$ to the CRS+RO model (using $\mathsf{VC}$), (2) establishing its round-by-round knowledge error, and (3) applying this to $\mathsf{FS}^{H_{\mathsf{FS}}}[\mathsf{O2C}^{H_{\mathsf{O2C}}}[\Pi]]$, we now combine this analysis into one Lemma.[14] Since both the Fiat-Shamir transform and the $\mathsf{VC}$ within the $\mathsf{O2C}$ compiler use random oracles (but these are separate instances), we denote them as $H_{\mathsf{FS}}$ and $H_{\mathsf{O2C}}$.

**Lemma 4** ($\mathsf{FS} \circ \mathsf{O2C}$)**.** *Suppose $\Pi = (\mathcal{P}, \mathcal{V})$ is a $\mu$-round* public-coin *proof system for $\mathcal{R}$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model with* special *extractor $\mathsf{Ext}_\Pi$ and round-by-round knowledge error $\kappa$. Let $M$ denote an upper bound on the number of $\mathsf{VC}$ commitments sent during an (interactive) run of $\mathsf{O2C}[\Pi]$. Then $\mathsf{NIZK} = (\mathcal{P}', \mathcal{V}') = \mathsf{FS}^{H_{\mathsf{FS}}}[\mathsf{O2C}^{H_{\mathsf{O2C}}}[(\mathcal{P}, \mathcal{V})]]$ is a non-interactive proof system in the CRS+RO model such that:*

1. *There exists a special extractor $\mathsf{Ext}_{\mathsf{NIZK}}$ such that for any adversary $\mathcal{A}$ which makes at most $Q_{\mathsf{FS}}$ (resp. $Q_{\mathsf{O2C}}$, $Q_{\mathsf{Verify}}$) queries to $H_{\mathsf{FS}}$ (resp. $H_{\mathsf{O2C}}$, $\mathsf{Verify}$-oracle) the advantage $\mathsf{AdvKE}^{\mathsf{NIZK}}_{\mathcal{A}}$ of $\mathcal{A}$ is bounded by*
$$\mu \cdot (Q_{\mathsf{FS}} + Q_{\mathsf{Verify}}) \cdot \kappa + M \cdot Q_{\mathsf{Verify}} \cdot \mathsf{AdvEB}^{\mathsf{VC}}_{\mathcal{A}'}[Q_{H_{\mathsf{O2C}}}] + \mathsf{AdvDist}^{\mathsf{VC.Setup},\mathsf{VC.TSetup}}_{\mathcal{D}} \tag{1}$$
*where $\mathcal{A}'$ and $\mathcal{D}$ are adversaries with roughly the same running time as $\mathcal{A}$.*
2. *Suppose that $\Pi$ is SHVZK. Moreover, suppose that the first message which the prover of $\mathsf{O2C}[\Pi]$ sends is $\delta$-unpredictable, i.e. has min-entropy $-\log(\delta)$. Then $\mathsf{NIZK}$ is a zero-knowledge in the (programmable) ROM.*

*Proof.* We start by proving knowledge soundness.
Knowledge soundness. To simplify our analysis, we modify the adversary as follows: $\mathcal{A}$ will never make duplicate queries to $H_{\mathsf{FS}}$ or $H_{\mathsf{O2C}}$ (as it can simply cache the response). Moreover,

[14] Otherwise, we would need to define IOPs in the CRS+RO model (since this is the setting of $\mathsf{O2C}^{H_{\mathsf{O2C}}}[\Pi]$). We want to avoid extending the notion of IOPs in this work.

instead of considering the protocol $\Pi$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model, we interpret $\Pi$ as an IOP, cf. Remark 6. Hence, we write $\mathsf{Ext}_{\mathsf{IOP}}$ (instead of $\mathsf{Ext}_{\Pi}$) for the assumed special extractor of protocol $\Pi$.

The special extractor $\mathsf{Ext}_{\mathsf{NIZK}}$ for $\mathsf{NIZK} = \mathsf{FS}^{H_{\mathsf{FS}}}[\mathsf{O2C}^{H_{\mathsf{O2C}}}[(\mathcal{P}, \mathcal{V})]]$ is defined as follows:

- Given $(\mathbb{x}, \pi)$ with $\mathsf{NIZK.Verify}^{H_{\mathsf{FS}}, H_{\mathsf{O2C}}}(crs, \mathbb{x}, \pi) = 1$, extract all commitments of the prover's messages in $\pi = \overline{\boldsymbol{m}}$.
- Assemble an IOP transcript denoted $\mathsf{asIOP}^{H_{\mathsf{FS}}}(\pi)$ from the extracted values. For messages where extraction (partially) failed, dummy values are used.
- Output $\mathbb{w} \leftarrow \mathsf{Ext}_{\mathsf{IOP}}(\mathbb{x}, \mathsf{asIOP}^{H_{\mathsf{FS}}}(\pi))$.

Now, we argue about the knowledge error of $\mathsf{NIZK}$ via game hops. Let game $\mathsf{G}_1$ be defined like real knowledge soundness game (Definition 22) for $\mathsf{NIZK}$.

In game $\mathsf{G}_2$, the CRS $crs$ is generated using $\mathsf{VC.TSetup}$ instead of $\mathsf{VC.Setup}$. The obvious distinguisher $\mathcal{D}$, namely just running the "real" knowledge experiment but with the $crs$ provided by the challenger, yields $\Pr[\mathsf{G}_1 = 1] \leq \Pr[\mathsf{G}_2 = 1] + \mathsf{AdvDist}_{\mathcal{D}}^{\mathsf{VC.Setup}, \mathsf{VC.TSetup}}$.

In game $\mathsf{G}_3$, whenever the adversary calls $\mathsf{Verify}(\mathbb{x}, \pi)$ where $\pi = \overline{\boldsymbol{m}}$ is a complete proof, all (fresh) $\mathsf{VC}$ commitments in $\overline{\boldsymbol{m}}$ are extracted. If the oracle call $\mathsf{Verify}(\mathbb{x}, \pi)$ returns 1, that is, if $\mathsf{NIZK.Verify}^{H_{\mathsf{FS}}, H_{\mathsf{O2C}}}(crs, \mathbb{x}, \pi) = 1$, but any extraction of commitments in $\pi$ is inconsistent with the decommitments contained in $\pi$ (in particular, if extraction yields $\perp$ but $\mathcal{A}$ successfully decommits), the experiment immediately outputs 0 (making $\mathcal{A}$ lose). Note that there are (at most) $Q_{\mathsf{Verify}}$ oracle calls, with at most $M$ commitments to extract in each call, hence at most $M \cdot Q_{\mathsf{Verify}}$ calls to $\mathsf{VC.Ext}$ in $\mathsf{G}_2$. By a hybrid argument, this change reduces to an adversary against the straightline extractability of $\mathsf{VC}$ used in $\mathsf{O2C}$. Thus

$$\Pr[\mathsf{G}_2 = 1] \leq \Pr[\mathsf{G}_3 = 1] + M \cdot Q_{\mathsf{Verify}} \cdot \mathsf{AdvEB}_{\mathcal{A}}^{\mathsf{VC}}[Q_{H_{\mathsf{O2C}}}].$$

Game $\mathsf{G}_4$ is defined as the ideal NIZK knowledge experiment, with special extractor $\mathsf{Ext}_{\mathsf{NIZK}}$ as described in the beginning. Recall that we interpret our protocol in $\mathsf{G}_3$ as a $\mu$-round IOP. Now, consider some call $\mathsf{Verify}(\mathbb{x}, \pi)$ of $\mathcal{A}$ where $\mathsf{Verify}^{H_{\mathsf{FS}}, H_{\mathsf{O2C}}}(crs, \mathbb{x}, \pi) = 1$. The proof $\pi$ is then a sequence of prover messages $\overline{\boldsymbol{m}}$ which yields (via Fiat–Shamir with $H_{\mathsf{FS}}$) a full accepting transcript of $\mathsf{O2C}[\Pi]$. Since in game $\mathsf{G}_3$, we ensured that $\mathsf{Ext}$ extracts all $\mathsf{VC}$ commitments (that is, the adversary cannot open to a message which was not extracted), this allows us to recover the committed values, which correspond to the IOP oracles $f_i$ (which are the malicious OT inputs). By definition, $\mathcal{E}$ recomputes such an IOP transcript $\mathsf{asIOP}^{H_{\mathsf{FS}}}(\mathbb{x}, \pi)$ for $(\mathbb{x}, \pi)$ and runs the IOP extractor to obtain a witness for $\mathbb{x}$ from $\mathsf{asIOP}^{H_{\mathsf{FS}}}(\mathbb{x}, \pi)$. Moreover, we can recover partial IOP transcripts $(\overline{\boldsymbol{m}}, \boldsymbol{\gamma})$ from random oracle queries $H_{\mathsf{FS}}(\mathsf{tr})$ where $\mathsf{tr}$ is a partial transcript of $\mathsf{O2C}[\Pi]$.

By definition of round-by-round knowledge soundness, for any full IOP transcript $\mathsf{tr}$ which does not contain a bad challenge, i.e. with $\mathsf{badch}(\mathbb{x}, \mathsf{tr}) = 1$, the extraction of the IOP succeeds. Thus, the failure event $F = \{\mathsf{G}_3 \neq \mathsf{G}_4\}$ is bounded by the probability that $\mathsf{badch}(\mathsf{asIOP}^{H_{\mathsf{FS}}}(\mathbb{x}, \pi)) = 1$ occurs for some query $\mathsf{Verify}^{H}(crs, \mathbb{x}, \pi) = 1$. By monotonicity, there is a unique first bad prefix of $\mathsf{asIOP}^{H_{\mathsf{FS}}}(\mathbb{x}, \pi)$. If the adversary always queries all prefixes of a transcript to $H_{\mathsf{FS}}$ in order, and if it also checks that $\mathsf{Verify}$ accepts if called, then by a simple guessing argument one can embed round-by-round knowledge and obtain $\Pr[F] \leq Q_{\mathsf{FS}} \cdot \kappa$. However, an adversary might try to "guess" challenges, i.e. proceed without ever querying $H_{\mathsf{FS}}$ to know the challenge. This can be handled by replacing $\mathcal{A}$ with an adversary $\mathcal{A}'$ which emulates $\mathcal{A}$ but always queries all (unqueried) prefixes. This increases the number of queries to $\mu \cdot (Q_{\mathsf{FS}} + Q_{\mathsf{Verify}})$ in the worst case. Thus, we find

$$\Pr[\mathsf{G}_3 = 1] \leq \Pr[\mathsf{G}_4 = 1] + \mu \cdot (Q_{\mathsf{FS}} + Q_{\mathsf{Verify}}) \cdot \kappa$$

By a union bound over all games, the knowledge soundness bound follows.

Zero-Knowledge. We sketch the proof of zero-knowledge. As shown in Lemma 3, the compiled protocol $\mathsf{O2C}[\Pi]$ inherits SHVZK and public-coin from $\Pi$, with a (straightline black-box) SHVZK simulator. Thus, by choosing a random tape for $\mathcal{V}_{\mathsf{O2C}}$, and by programming random oracle responses according to corresponding challenges, (multi-instance) zero-knowledge of $\mathsf{FS}^{H_{\mathsf{FS}}}[\mathsf{O2C}^{H_{\mathsf{O2C}}}[\Pi]]$ reduces to SHVZK for $\mathsf{O2C}[\Pi]$ via a hybrid argument. More precisely, either programming a partial transcript $\mathsf{tr}$ fails, because $H^{\mathsf{FS}}(\mathsf{tr})$ is defined, or the SHVZK game can be embedded. By $\delta$-unpredictability (i.e. min-entropy $-\log(\delta)$) of the first message of the prover in $\mathsf{O2C}[\Pi]$, (and since $H_{\mathsf{FS}}$ and $H_{\mathsf{O2C}}$ are independent), the probability that programming fails is bounded by $\delta \cdot (Q_{\mathsf{FS}} + Q_{\mathcal{S}}) \cdot Q_{\mathcal{S}}$, which is negligible. Since $\mathsf{O2C}[\Pi]$ is SHVZK by Lemma 3, this shows that $\mathsf{NIZK}$ is zero-knowledge. More precisely, the advantage of $\mathcal{A}$ against zero-knowledge of $\mathsf{NIZK} = \mathsf{FS}^{H_{\mathsf{FS}}}[\mathsf{O2C}^{H_{\mathsf{O2C}}}[\Pi]]$ is bounded by

$$\mathsf{AdvZK}_{\mathcal{A}}^{\mathsf{NIZK}}(\lambda) \leq \delta \cdot (Q_{\mathsf{FS}} + Q_{\mathcal{S}}) \cdot Q_{\mathcal{S}} + \mathsf{AdvZK}_{\mathcal{A}'}^{\mathsf{O2C}[\Pi]}(\lambda)$$

where $\mathcal{A}'$ denotes the straightforward SHVZK adversary, and $Q_{\mathcal{S}}$ denotes a (polynomial) bound for the number (adaptive) simulations $\mathcal{A}$ can see.

*Remark 8.* The loss of $\mu \cdot Q_{\mathsf{Verify}} \cdot \kappa$ is inherent to our notion of round-by-round (knowledge) soundness. Suppose $\kappa = k/2^{\lambda} > 0$ and consider a $\mu$-round protocol for a language $\mathcal{L}$ in P, where the prover just sends $(\mathsf{ack})$, and the verifier sends random challenges $\gamma_i \leftarrow [0, 2^{\lambda 1})$. Finally, the verifier accepts if any $\gamma_i$ satisfies $\gamma_i < k$, or if $\varkappa \in \mathcal{L}$. Clearly, the round-by-round (knowledge) soundness error is $\kappa$, but the soundness error is $\mu\kappa$. Moreover, a prover can generate a random transcript and wins with probability at least $\mu\kappa$ even without querying the random oracle. From this example, it also follows that the term $\mu \cdot (Q_{\mathsf{FS}} + Q_{\mathsf{Verify}}) \cdot \kappa$ in Lemma 4 is essentially tight.

## 5 Generalized Subspace VOLE Protocol

The VOLE protocols presented in SoftSpokenOT [Roy22] achieve subspace VOLE over a polynomial-order field $\mathbb{F}_q$. Here, we generalize to exponentially large fields $\mathbb{F}_q$, with the limitation that the receiver's secret $\Delta$ must be sampled from a subset $S_\Delta \subseteq \mathbb{F}_q^{n_\mathcal{C}}$, such that the projected set $S_\Delta^i$, which contains the $i$-th coordinate of every element of $S_\Delta$, has polynomial size. We also reorder the operations so that the protocols fit the $(\mathsf{get})$, $(\mathsf{init})$ model required for ZKP compilation (Definition 10). The subspace VOLE functionality we realize is in Figure 2.

### 5.1 VOLE with Small-Domain $\Delta$

The first step is to construct VOLE (that is, subspace VOLE where $n_\mathcal{C} = k_\mathcal{C} = 1$) from oblivious transfer. In Figure 3, we've adapted the small field VOLE from SoftSpokenOT to work over an arbitrary field $\mathbb{F}_q$, as long as $\Delta$ is sampled from a polynomial-sized subset $S_\Delta$. The security proof requires very few changes, so we have deferred it to the appendix.

**Theorem 1.** *The protocol $\Pi_{\mathsf{small\text{-}VOLE}}^{p,q,S_\Delta,\ell}$, given in Figure 3, securely realizes $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathbb{F}_p,\ell,\{2^{S_\Delta}\}}$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{N,1}$-hybrid model, with malicious security.*[15]

[15] Note that setting $\mathcal{L} = \{2^{S_\Delta}\}$ is equivalent to no leakage, i.e., not allowing a corrupt $\mathcal{P}$ to perform a selective failure attack.

**Functionality** $\mathcal{F}^{p,q,S_\Delta,\mathcal{C},\ell,\mathcal{L}}_{\mathsf{sVOLE}}$

The functionality interacts with a sender $\mathcal{P}$, a receiver $\mathcal{V}$ and an adversary $\mathcal{A}$.
It is parametrized by integers $\ell$ and $p, q$, such that $q = p^k$, as well as an $[n_\mathcal{C}, k_\mathcal{C}, d_\mathcal{C}]_p$ linear code $\mathcal{C}$ over $\mathbb{F}_p$ and a generator matrix $\mathbf{G}_\mathcal{C} \in \mathbb{F}_p^{k_\mathcal{C} \times n_\mathcal{C}}$ for $\mathcal{C}$.

1. Upon receiving $(\mathsf{init})$ from $\mathcal{P}$ and $\mathcal{V}$, sample $\mathbf{U} \leftarrow \mathbb{F}_p^{\ell \times k_\mathcal{C}}$, $\mathbf{V} \leftarrow \mathbb{F}_q^{\ell \times n_\mathcal{C}}$ and $\Delta \leftarrow S_\Delta \subseteq \mathbb{F}_q^{n_\mathcal{C}}$ and set $\mathbf{Q} := \mathbf{V} + \mathbf{U}\mathbf{G}_\mathcal{C}\mathsf{diag}(\Delta)$.
   – If $\mathcal{P}$ is corrupt, receive $\mathbf{U}, \mathbf{V}$ from $\mathcal{A}$, and recompute $\mathbf{Q}$ as above.
   – If $\mathcal{V}$ is corrupt, receive $\Delta, \mathbf{Q}$ from $\mathcal{A}$ and compute $\mathbf{V} := \mathbf{Q} - \mathbf{U}\mathbf{G}_\mathcal{C}\mathsf{diag}(\Delta)$.
   – Send $(\mathbf{U}, \mathbf{V})$ to $\mathcal{P}$.
   – If $\mathcal{P}$ is corrupt, receive a leakage query $L \in \mathcal{L}$ from $\mathcal{A}$.
2. Upon receiving $(\mathsf{get})$ from $\mathcal{V}$, if $\Delta \notin L$, send $(\mathsf{check\text{-}failed})$ to $\mathcal{V}$ and abort. Otherwise, send $(\Delta, \mathbf{Q})$ to $\mathcal{V}$.

**Figure 2.** Subspace VOLE functionality adapted from SoftSpokenOT [Roy22]

**Protocol** $\Pi^{p,q,S_\Delta,\ell}_{\mathsf{small\text{-}VOLE}}$

Requires $S_\Delta = \{f_1, \ldots, f_N\} \subseteq \mathbb{F}_q$, where $N = |S_\Delta| = \mathsf{poly}(\lambda)$. Also requires $S_\Delta \setminus \{f_1\}$ to span $\mathbb{F}_q$, viewed as a vector space over $\mathbb{F}_p$. Let $\mathsf{PRG} : \{0,1\}^\lambda \to \mathbb{F}_p^\ell$ be a PRG.

On $(\mathsf{init})$, $\mathcal{P}$ does as follows:
1. Call $\mathcal{F}^{N,1}_{\mathsf{OT}\text{-}\bar{1}}$ with $(\mathsf{init})$, and receive the messages $s_1, \ldots, s_N \in \{0,1\}^\lambda$.
2. For $i \in [N]$, let $\mathbf{t}_{f_i} = \mathsf{PRG}(s_i)$.
3. Compute and output $\mathbf{u} := \sum_{x \in S_\Delta} \mathbf{t}_x$ and $\mathbf{v} := -\sum_{x \in S_\Delta} \mathbf{t}_x x$.

On $(\mathsf{init})$, $\mathcal{V}$ passes the $(\mathsf{init})$ message on to $\mathcal{F}^{N,1}_{\mathsf{OT}\text{-}\bar{1}}$.

On $(\mathsf{get})$, $\mathcal{V}$ does as follows:
1. Sample $j \leftarrow [N]$, and let $\Delta = f_j$.
2. Call $\mathcal{F}^{N,1}_{\mathsf{OT}\text{-}\bar{1}}$ with input $(\mathsf{get}, j)$, and receive $s_i$ for $i \in [N] \setminus \{j\}$.
3. For $i \in [N] \setminus \{j\}$, let $\mathbf{t}_{f_i} := \mathsf{PRG}(s_i)$.
4. Compute $\mathbf{q} := \sum_{x \in S_\Delta \setminus \{\Delta\}} \mathbf{t}_x(\Delta - x)$.
5. Output $(\Delta, \mathbf{q})$.

**Figure 3.** VOLE protocol in the $(\mathsf{init}), (\mathsf{get})$ model, with $\Delta$ from a small domain $S_\Delta$. Note that we notate the subspace VOLE here with vectors $\mathbf{u}$ instead of matrices $\mathbf{U}$, because they all only have 1 column.

*Proof.* The protocol and its proof are very similar to SoftSpokenOT [Roy22, Theorem 3.1], with sampling $\Delta$ from $S_\Delta$ being the only important difference. As with SoftSpokenOT, the following equation shows correctness, and will be useful for all cases of the security proof.

$$\mathbf{q} = \sum_{x \in S_\Delta \setminus \{\Delta\}} \mathbf{t}_x(\Delta - x) = \sum_{x \in S_\Delta} \mathbf{t}_x(\Delta - x) = \sum_{x \in S_\Delta} \mathbf{t}_x \Delta - \sum_{x \in S_\Delta} \mathbf{t}_x x = \mathbf{u}\Delta + \mathbf{v} \tag{2}$$

*Both honest.* The ideal functionality will proceed by sampling $\mathbf{u}$, $\mathbf{v}$, and $\Delta$, and then compute $\mathbf{q}$ according to the VOLE correlation. Equation (2) shows that the correct $\mathbf{q}$ will be computed, so we only need to show that the distribution of $(\mathbf{u}, \mathbf{v}, \Delta)$ is correct.

The PRG seeds $\{s_i\}_{i \in [N]} \in \{0,1\}^\lambda$ are independent and uniformly random, and are only given to the PRG, so the PRG outputs $\{\mathbf{t}_x\}_{x \in S_\Delta}$ are also independent and uniformly random. The vectors $(\mathbf{u}, \mathbf{v})$ are a $\mathbb{F}_p$-linear map applied to $\{\mathbf{t}_x\}_{x \in S_\Delta}$. Since $\{x\}_{x \in S_\Delta \setminus \{f_1\}}$ spans $\mathbb{F}_q$ as a $k$-dimensional vector space over $\mathbb{F}_p$, $\{(1, x)\}_{x \in S_\Delta}$ spans a $k+1$-dimensional vector space representing all possible values of $(\mathbf{u}, \mathbf{v})$. Therefore, $\mathbf{u} \in \mathbb{F}_p^\ell$ and $\mathbf{v} \in \mathbb{F}_q^\ell$ are independent and uniformly random. Finally, $\Delta$ is sampled by $\mathcal{V}$ as a uniformly random element of $S_\Delta$.

**Protocol** $\Pi_{\mathsf{VOLE}}^{p,q,S_\Delta,\mathcal{C},\ell}$

$\mathcal{H} \subseteq \mathbb{F}_q^{r\times(\ell+h)}$ is a family of $\ell$-hiding, $\varepsilon$-universal linear hash functions.
$\mathcal{L}$ must contain all single variable constraints: $\{\Delta \in S_\Delta \mid \Delta_i = y\} \in \mathcal{L}, \forall i, y$.

On (init), $\mathcal{P}$ and $\mathcal{V}$ run the following protocol:

1. $\mathcal{P}$ & $\mathcal{V}$: Send (init) to $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathbb{F}_p^{n_\mathcal{C}},\ell+h,\{2^{S_\Delta}\}}$.
2. $\mathcal{P}$: Receive $\mathbf{U}' \in \mathbb{F}_p^{(\ell+h)\times n_\mathcal{C}}$ and $\mathbf{V} \in \mathbb{F}_q^{(\ell+h)\times n_\mathcal{C}}$ from $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathbb{F}_p^{n_\mathcal{C}},\ell+h,\{2^{S_\Delta}\}}$.
3. $\mathcal{P}$: Compute $[\mathbf{U}\,\mathbf{C}] := \mathbf{U}'\mathbf{T}_\mathcal{C}^{-1}$ and send the correction $\mathbf{C} \in \mathbb{F}_q^{(\ell+h)\times(n_\mathcal{C}-k_\mathcal{C})}$.
4. $\mathcal{P}$: Output $(\mathbf{U}_{[1..\ell]}, \mathbf{V}_{[1..\ell]})$.
5. $\mathcal{V}$: Sample and send a uniformly random challenge $\mathbf{H} \leftarrow \mathcal{H}$.
6. $\mathcal{P}$: Send $\widetilde{\mathbf{U}} := \mathbf{HU}$ and $\widetilde{\mathbf{V}} := \mathbf{HV}$.

On (get), $\mathcal{V}$ does as follows:

1. Send (get) to $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathbb{F}_p^{n_\mathcal{C}},\ell+h,\{2^{S_\Delta}\}}$, and receive $\Delta \in S_\Delta$ and $\mathbf{Q}' = \mathbf{U}'\Delta + \mathbf{V}$.
2. Compute $\mathbf{Q} := \mathbf{Q}' - [0\,\mathbf{C}]\mathbf{T}_\mathcal{C}\mathsf{diag}(\Delta)$.
3. Abort if $\widetilde{\mathbf{V}} \neq \mathbf{HQ} - \widetilde{\mathbf{U}}\mathbf{G}_\mathcal{C}\mathsf{diag}(\Delta)$.
4. Output $(\Delta, \mathbf{Q}_{[1..\ell]})$.

**Figure 4.** Subspace VOLE protocol in the (init), (get) model, with each $\Delta_i$ from a small domain $S_\Delta$.

*Malicious* $\mathcal{V}$. This case is similar to the previous, except that the adversary gets to see every seed $s_i$, except for $s_j$. Therefore, we can only rely on $\mathbf{t}_\Delta$ being random. Luckily, this is sufficient to make $\mathbf{u} = \sum_{x\in S_\Delta} \mathbf{t}_x$ uniformly random in $\mathbb{F}_p$. And $\mathbf{v}$ will be set correctly, by Equation (2).

More precisely, have the simulator program $\Delta$ and $\mathbf{q}$ into the ideal functionality, which will sample $\mathbf{u}$ randomly and set $\mathbf{v} = \mathbf{q} - \mathbf{u}\Delta$. In both the real and ideal worlds, $\mathbf{u}$ is indistinguishable from random, and $\mathbf{v}$ is computed correctly according to the VOLE correlation. Therefore, these worlds are indistinguishable.

*Malicious* $\mathcal{P}$. Let the simulator pretend to be $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{N,1}$, and receive $s_1, \ldots, s_N$ from $\mathcal{P}$. Next, compute $\mathbf{u}$ and $\mathbf{v}$, then send these to $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathbb{F}_p,\ell,\{2^{S_\Delta}\}}$. Both the real and ideal worlds will sample $\Delta$ uniformly from $S_\Delta$, then compute $\mathbf{q} = \mathbf{u}\Delta + \mathbf{v}$, by Equation (2). Therefore, the real and ideal worlds are indistinguishable.

### 5.2 Subspace VOLE with Small-Domain $\Delta$

**From $\mathbb{F}_p$ subspace VOLE to $\mathbb{F}_p^n$.** Suppose $S_\Delta = S_\Delta^1 \times \cdots \times S_\Delta^n$, where each $S_\Delta^i \subseteq \mathbb{F}_q$. By running $n$ parallel instances of $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta^i,\mathbb{F}_p,\ell,\{2^{S_\Delta^i}\}}$, we can obtain a single instance of subspace[16] VOLE for $S_\Delta$, where the vectors $\mathbf{u} \in \mathbb{F}_p^\ell, \mathbf{v} \in \mathbb{F}_q^\ell, \mathbf{q} \in \mathbb{F}_q^\ell$ have been stacked into matrices $\mathbf{U} \in \mathbb{F}_p^{\ell\times n}, \mathbf{V} \in \mathbb{F}_q^{\ell\times n}, \mathbf{Q} \in \mathbb{F}_q^{\ell\times n}$. This is exactly the functionality $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathbb{F}_p^n,\ell,\{2^{S_\Delta}\}}$. We skip the trivial security proof for this transformation.

**From $\mathbb{F}_p^n$ VOLE to $\mathcal{C}$ subspace VOLE.** However, we want to construct an *actual* subspace VOLE, where the rows of $\mathbf{U}$ can be constrained to lie in the subspace defined by an arbitrary linear code $\mathcal{C}$. When $\mathcal{P}$ is honest this is easily achieved with derandomization, but $\mathcal{P}$ could lie, leading to rows of $\mathbf{U}$ that are not in $\mathcal{C}$.

In Figure 4, we adapt the SoftSpokenOT consistency check to our problem, rearranging the protocol to fit the (init), (get) model, and restricting $\Delta$ to be sampled from a subset $S_\Delta \subseteq \mathbb{F}_q^{n_\mathcal{C}}$.

[16]In this case, the "subspace" is just the whole vector space $\mathbb{F}_p^n$.

We've made two changes that drastically change the proof of security for malicious $\mathcal{V}$: first, $\Delta$ is sampled from an arbitrary set $S_\Delta$, while SoftSpokenOT's analysis requires that $\Delta$ be uniform in a linear space, because it makes much use of $\Delta$'s distribution being invariant under invertible linear transformations.

Second, we want our bound to be compatible with Fiat–Shamir, which lets the adversary restart the proof between sampling $\mathbf{H}$ and the consistency check as many times as it wants. This means that our analysis of the hash $\mathbf{H}$ must be independent of $\Delta$, so we give a bad event for $\mathbf{H}$ and show that it's unlikely on its own, whether or not the adversary succeeds in guessing (part of) $\Delta$. These two changes unfortunately make our bound looser than SoftSpokenOT's, but it is good enough to be practically useful.

**Theorem 2.** *The protocol $\Pi^{p,q,S_\Delta,\mathcal{C},\ell}_{\textsf{small-VOLE}}$, given in Figure 4, securely realizes the functionality $\mathcal{F}^{p,q,S_\Delta,\mathcal{C},\ell,\mathcal{L}}_{\textsf{sVOLE}}$ using $\mathcal{F}^{p,q,S_\Delta,\mathbb{F}_p^{n_\mathcal{C}},\ell+h,\{2^{S_\Delta}\}}_{\textsf{sVOLE}}$. The distinguisher has advantage at most $\varepsilon\binom{n_\mathcal{C}}{k_\mathcal{C}+1}$. This advantage comes from a single bad event in the malicious $\mathcal{P}$ case that is decided once $\mathbf{H}$ is sampled.*

*Proof.* By comparing with the SoftSpokenOT subspace VOLE, notice that the adversary can only gain an additional advantage when $\mathcal{V}$ is honest. Indeed, when $\mathcal{V}$ is malicious, the underlying $\mathcal{F}_{\textsf{sVOLE}}$ functionality lets the adversary choose $\Delta$ however it wants, so it makes no difference what distribution an honest $\mathcal{V}$ would sample $\Delta$ from.

*Both honest.* This case follows easily from SoftSpokenOT's security proof, because the only change is to $\Delta$'s distribution. $\Delta$ is passed straight through from the underlying $\mathcal{F}_{\textsf{sVOLE}}$ functionality, so our protocol's distribution is identical to conditioning SoftSpokenOT on $\Delta$ being in $S_\Delta$. Because SoftSpokenOT is perfectly secure in the honest–honest case (in the $\mathcal{F}_{\textsf{sVOLE}}$-hybrid model), our protocol then realizes $\mathcal{F}^{p,q,\mathbb{F}_q^{n_\mathcal{C}},\mathcal{C},\ell,\mathcal{L}}_{\textsf{sVOLE}}$, but with the distribution conditioned on $\Delta \in S_\Delta$. This conditioned distribution is exactly the same as $\mathcal{F}^{p,q,S_\Delta,\mathcal{C},\ell,\mathcal{L}}_{\textsf{sVOLE}}$.

*Malicious $\mathcal{P}$.* While our consistency-checking protocol is most directly based on SoftSpokenOT, for this case our simulator and proof also take inspiration from OOS [OOS17]. In particular, our proof is based on erasure decoding, like with OOS. SoftSpokenOT pointed out an error in OOS's proof, so to use their proof technique we will need to patch this error. We will use a union bound to fix the problem, which is why our security bound is considerably looser than OOS's.

We present the simulator in Figure 5. The simulator first extracts $\mathbf{U}''$, the derandomization of $\mathbf{U}'$, and samples a challenge $\mathbf{H}$. Based on $\mathbf{U}''$, $\mathbf{H}$, and the errors $(\bar{\mathbf{U}}, \bar{\mathbf{V}})$ in the adversary's consistency check messages $(\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}})$, we can rewrite the consistency check as follows.

$$\begin{aligned}
\widetilde{\mathbf{V}} &= \mathbf{HQ} - \widetilde{\mathbf{U}}\mathbf{G}_\mathcal{C}\mathsf{diag}(\Delta) \\
\widetilde{\mathbf{V}} &= \mathbf{HV} + \mathbf{HU}''\mathsf{diag}(\Delta) - \widetilde{\mathbf{U}}\mathbf{G}_\mathcal{C}\mathsf{diag}(\Delta) \\
-\bar{\mathbf{V}} &= \bar{\mathbf{U}}\,\mathsf{diag}(\Delta)
\end{aligned}$$

From this equation, the simulator extracts guesses $\Delta_i^*$ for $\Delta_i$ for all $i$ in a subset $G$ of the columns. The consistency check is equivalent to the correctness of these guesses.

The guessed columns $G$ represent lies that the adversary has made while derandomizing $\mathbf{U}$. The simulator attempts to extract $\mathcal{P}$'s real output $\mathbf{U}^*$ by erasure decoding $\mathbf{U}''$ using only the columns that are not in $G$, hoping that these columns represent the truth about $\mathbf{U}$, as the erasure removes the lies present in the consistency check. That is, let $\mathcal{C}_{-G}$ be the punctured code created by removing all columns in $G$ from $\mathcal{C}$, and let $\mathbf{U}''_{-G}$ be the corresponding punctured matrix. If

On (init) sent to $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathbb{F}_p^{n_\mathcal{C}},\ell+h,\{2^{S_\Delta}\}}$:

1. Receive $\mathbf{U}', \mathbf{V}$ from $\mathcal{A}$ and send them to $\mathcal{P}$.
2. Receive $\mathbf{C}$ from $\mathcal{P}$.
3. Compute the derandomization $\mathbf{U}'' := \mathbf{U}' - [0\,\mathbf{C}]\mathbf{T}_\mathcal{C}$. We have:
$$\begin{aligned}\mathbf{Q} &= \mathbf{Q}' - [0\,\mathbf{C}]\mathbf{T}_\mathcal{C}\mathsf{diag}(\Delta)\\ &= \mathbf{V} + \mathbf{U}'\mathsf{diag}(\Delta) - [0\,\mathbf{C}]\mathbf{T}_\mathcal{C}\mathsf{diag}(\Delta)\\ &= \mathbf{V} + \mathbf{U}''\mathsf{diag}(\Delta).\end{aligned}$$
4. Sample and send a uniformly random challenge $\mathbf{H} \leftarrow \mathcal{H}$ to $\mathcal{P}$.
5. Receive the response $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$ from $\mathcal{P}$.
6. Compute the response errors: $\bar{\mathbf{U}} := \mathbf{H}\mathbf{U}'' - \widetilde{\mathbf{U}}\mathbf{G}_\mathcal{C}$ and $\bar{\mathbf{V}} := \mathbf{H}\mathbf{V} - \widetilde{\mathbf{V}}$.
7. The consistency checking equation is $-\bar{\mathbf{V}} = \bar{\mathbf{U}}\,\mathsf{diag}(\Delta)$. Abort if no solutions for $\Delta$ exist. Otherwise, there exists a set $G$ of guessed columns (the nonzero columns of $\bar{\mathbf{U}}$) and values $\Delta_i^*$ such that the solution set is $\{\Delta \in S_\Delta \mid \forall i \in G.\, \Delta_i = \Delta_i^*\}$.
8. Decode $\mathbf{U}''_{-G}$ with $\mathcal{C}_{-G}$ to get $\mathbf{U}^* \in \mathbb{F}_p^{\ell \times k_\mathcal{C}}$, aborting if any row of $\mathbf{U}''_{-G}$ isn't in $\mathcal{C}_{-G}$.
9. Recover $\mathbf{V}^* := \mathbf{V} + (\mathbf{U}'' - \mathbf{U}^*\mathbf{G}_\mathcal{C})\mathsf{diag}(\Delta^*)$ from the adversary's guesses. We have $\mathbf{V}^* = \mathbf{V} + \mathbf{U}''\mathsf{diag}(\Delta) - \mathbf{U}^*\mathbf{G}_\mathcal{C}\mathsf{diag}(\Delta) = \mathbf{Q} - \mathbf{U}^*\mathbf{G}_\mathcal{C}\mathsf{diag}(\Delta)$ if the consistency check passes, because $\mathbf{U}'' - \mathbf{U}^*\mathbf{G}_\mathcal{C}$ is zero except for the columns in $G$, and $\Delta_i = \Delta_i^*$ for $i \in G$.
10. Send $\{\Delta \in S_\Delta \mid \forall i \in G.\, \Delta_i = \Delta_i^*\} \in \mathcal{L}$ to $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathcal{C},\ell,\mathcal{L}}$.
11. Send $\mathbf{U}^*_{[1..\ell]}, \mathbf{V}^*_{[1..\ell]}$ to $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathcal{C},\ell,\mathcal{L}}$.

**Figure 5.** Simulator for $\Pi_{\mathsf{small\text{-}VOLE}}^{p,q,S_\Delta,\mathcal{C},\ell}$ with malicious $\mathcal{P}$.

$|G| \geq d_\mathcal{C}$, then $\mathcal{C}_{-G}$ may have 0 minimum distance, which means that decoding isn't unique; it is sufficient to pick an arbitrary decoding. The simulator can then extract the $\mathcal{P}$'s other real output $\mathbf{V}^*$ as $\mathbf{V} + (\mathbf{U}'' - \mathbf{U}^*\mathbf{G}_\mathcal{C})\mathsf{diag}(\Delta^*)$, using that $\mathbf{U}''$ must match $\mathbf{U}^*\mathbf{G}_\mathcal{C}$ on all columns that were not erased, and that $\Delta_i = \Delta_i^*$ for the erased columns $i \in G$. If this all works, the simulation is perfect – the consistency check is correctly represented by the leakage test, and $(\mathbf{U}^*, \mathbf{V}^*, \mathbf{Q}, \Delta)$ satisfy the subspace VOLE correlation.

The flaw is that some lies might not be present in the consistency check. That is, there may be some row of $\mathbf{U}''_{-G}$ that isn't in $\mathcal{C}_{-G}$, making the erasure decoding fail. The simulator aborts in this case. Next we present a bad event that must occur for the erasure decoding to fail, and then bound it's probability. This bad event is described in terms of the set of all circuits $\mathfrak{C}$ in the matroid represented by the columns of $\mathbf{G}_\mathcal{C}$. Recall that the definition of a circuit in a matroid is a set $c$ that is linearly dependent, and is minimal in the sense that all proper subsets are independent. That is, $\mathfrak{C}$ is the collection of all subsets $c$ of columns of $\mathbf{G}_\mathcal{C}$ such that $c$ is linearly dependent, but every $c' \subsetneq c$ is linearly independent.

**Bad event:** For all $c \in \mathfrak{C}$, all rows of $\mathbf{U}''_c$ must be in $\mathcal{C}_c$ (the punctured code containing only the columns in $c$) if and only if the rows of $\mathbf{H}\mathbf{U}''_c$ are all in $\mathcal{C}_c$. The bad event triggers if this does not hold.

The simulator fails if a row $\mathbf{u}_{-G}$ of $\mathbf{U}''_{-G}$ is not in $\mathcal{C}_{-G}$. For this to occur, there must be some vector $\mathbf{p}_{-G}$ in column space of the parity check matrix $\mathbf{P}_{\mathcal{C}_{-G}}$ of $\mathcal{C}_{-G}$ (equivalently, $\mathbf{p}_{-G}$ is in the null space of $\mathbf{G}_{\mathcal{C}_{-G}}$) such that $\mathbf{u}_{-G} \cdot \mathbf{p}_{-G} \neq 0$. Out of all such $\mathbf{p}_{-G}$, pick a maximally sparse $\mathbf{p}_{-G}$, minimizing the number of nonzero entries. The set $c$ of nonzero entries of $\mathbf{p}_{-G}$ forms a circuit in $\mathfrak{C}$. The parity check matrix $\mathbf{P}_{\mathcal{C}_c}$ of $\mathcal{C}_c$ is then $\mathbf{p}_c$. We have $\mathbf{u}_c \cdot \mathbf{p}_c = \mathbf{u}_{-G} \cdot \mathbf{p}_{-G} \neq 0$, yet $\mathbf{H}\mathbf{U}''_c\mathbf{p}_c = \mathbf{H}\mathbf{U}''_{-G}\mathbf{p}_{-G} = 0$, because the rows of $\mathbf{H}\mathbf{U}''_{-G}$ are in $\mathcal{C}_{-G}$. Therefore, the rows of $\mathbf{H}\mathbf{U}''_c$ are in $\mathcal{C}_c$, but the rows of $\mathbf{U}''_c$ are not all in $\mathcal{C}_c$, so the bad event must trigger.

Next, we bound the probability of the bad event occurring for any fixed $c \in \mathcal{C}$. Let $\mathbf{p}_c \in \mathbb{F}_p^{|c|}$ be the parity check matrix of $\mathcal{C}_c$, i.e., a vector such that $\mathbf{u}_c \in \mathcal{C}_c$ if and only if $\mathbf{u}_c \cdot \mathbf{p}_c = 0$. Then all rows of $\mathbf{U}''_c$ are in $\mathcal{C}_c$ if and only if $\mathbf{U}''_c \mathbf{p}_c = 0$, and similarly all rows of $\mathbf{H}\mathbf{U}''_c$ are all in $\mathcal{C}_c$ if and only if $\mathbf{H}\mathbf{U}''_c \mathbf{p}_c = 0$. The first clearly implies the second, so we only need to bound the probability that $\mathbf{U}''_c \mathbf{u}_p \neq 0$ but $\mathbf{H}\mathbf{U}''_c \mathbf{p}_c = 0$. Since $\mathbf{H}$ is sampled from an $\varepsilon$-almost universal family, if $\mathbf{U}''_c \mathbf{p}_c \neq 0$ we have that $\Pr[\mathbf{H}\mathbf{U}''_c \mathbf{p}_c = 0] \leq \varepsilon$.

Finally, $|\mathcal{C}| \leq \binom{n_{\mathcal{C}}}{k_{\mathcal{C}}+1}$ [DSL04, Theorem 2.1], so a union bound shows that the bad event occurs with probability at most $\varepsilon\binom{n_{\mathcal{C}}}{k_{\mathcal{C}}+1}$.

## 6 Zero-Knowledge from Generalized Subspace VOLE

We give two instantiations of the compiler from Section 3, by presenting two public coin, interactive ZK protocols in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model. The first one allows to prove statements over large fields using the generalized subspace VOLE given in Section 5. We start by describing a general ZK protocol for degree-2 relations, $\Pi^t_{\mathsf{2D-LC}}$, as specified in Figure 6 and, in Appendix B.1, we show how to generalize it to any degree-$d$ polynomials, for small $d$.

Our second protocol, $\Pi^t_{\mathsf{2D\text{-}Rep}}$, can be seen as a variant of the QuickSilver protocol [YSWW21] and is more tailored for proving statements over small fields. It permits to prove degree-2 constraints over any extension field $\mathbb{F}_{2^r}$. The protocol is described in Section 6.2 and its security is stated in Theorem 4.

### 6.1 ZK for Degree-2 Relations from Generalized sVOLE

Our 7-round ZK protocol for degree-2 relations allows for circuit satisfiability over any large field, while also cheaply proving useful operations like inner products, without unrolling them to a circuit. We highlight that the protocol uses subspace VOLE for a general code, rather than the trivial 1-dimensional code $\mathbb{F}_p$ (or the repetition code) used in previous VOLE-based ZK constructions [BMRS21, YSWW21]. The main challenge here is that, while VOLE with the repetition code can be viewed as a linearly homomorphic commitment scheme for messages in $\mathbb{F}_p$, with a general code, we only get a restricted form of homomorphic commitment to vectors in $\mathbb{F}_p^{k_{\mathcal{C}}}$, where linear operations must be applied across the vectors.

We let $\mathcal{C}$ be an $[n_{\mathcal{C}}, k_{\mathcal{C}}, d_{\mathcal{C}}]_p$ linear code with large enough distance, which for simplicity is given in systematic form. We assume the witness $\mathbf{w}$ can be divided into $\ell$ vectors $(\mathbf{w}_1, \dots, \mathbf{w}_\ell) \in (\mathbb{F}_p^{k_{\mathcal{C}}})^\ell$, where we also write $\mathbf{w} \in \mathbb{F}_p^{k_{\mathcal{C}}\ell}$ to mean the concatenation of these vectors. Let $\mathbb{F}_p[X]_{\leq 2} := \mathbb{F}_p[X_1, \dots, X_{k_{\mathcal{C}}\ell}]_{\leq 2}$ be the set of polynomials over $\mathbb{F}_p$ in $k_{\mathcal{C}}\ell$ variables with degree at most 2. Notice each $f_i \in \mathbb{F}_p[X]_{\leq 2}$ can be written as $f_i = f_{i,0} + f_{i,1} + f_{i,2}$ such that $\deg(f_{i,h}) = h$. The prover $\mathcal{P}$ wants to prove that $f_i(\mathbf{w}) = 0$, for $i \in [t]$. Here we consider the case where $p$ is large.

The intuition of the scheme is as follows. Let $S'_\Delta$ be a polynomially sized subset of $\mathbb{F}_p$ and $S_\Delta = (S'_\Delta)^{n_{\mathcal{C}}}$. First, both $\mathcal{P}$ and $\mathcal{V}$ call the subspace VOLE functionality $\mathcal{F}_{\mathsf{sVOLE}}^{p,\mathcal{C},S_\Delta,2\ell+2,\mathcal{L}}$, so that $\mathcal{P}$ receives the matrices $\mathbf{U} \in \mathbb{F}_p^{(2\ell+2)\times k_{\mathcal{C}}}, \mathbf{V} \in \mathbb{F}_p^{(2\ell+2)\times n_{\mathcal{C}}}$, while the verifier $\mathcal{V}$ gets the notification $\mathsf{done}$. Let $\mathbf{W}$ be the $\ell \times k_{\mathcal{C}}$ matrix whose $i$th are $\mathbf{w}_i$. The idea is to use the first $\ell+1$ rows of the output of the ideal functionality to commit to the witness, and the remaining $\ell+1$ rows as auxiliary random commitments. More precisely, we split the matrices as

$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \end{pmatrix} \quad \text{and} \quad \mathbf{U} = \begin{pmatrix} \mathbf{U}_1 \\ \mathbf{R} \end{pmatrix},$$

**Protocol $\Pi^t_{\mathsf{2D-LC}}$**

The protocol is parametrized by an $[n_\mathcal{C}, k_\mathcal{C}, d_\mathcal{C}]_p$ linear code $\mathcal{C}$, set $S_\Delta = (S'_\Delta)^{n_\mathcal{C}} \subset \mathbb{F}_p^{n_\mathcal{C}}$ and a leakage space $\mathcal{L}$ (used in $\mathcal{F}_{\mathsf{sVOLE}}$).

INPUTS: Both parties hold a set of polynomials $f_i \in \mathbb{F}_p[X_1, \ldots, X_{k_\mathcal{C}\ell}]_{\leq 2}$, $i \in [t]$. $\mathcal{P}$ also holds a witness $\mathbf{w} \in \mathbb{F}_p^{k_\mathcal{C}\ell}$ such that $f_i(\mathbf{w}) = 0$, for all $i \in [t]$.

**Round 1.** $\mathcal{P}$ does as follows:

1. $\mathcal{P}$ and $\mathcal{V}$ call $\mathcal{F}_{\mathsf{sVOLE}}^{p,p,S_\Delta,\mathcal{C},2(\ell+1),\mathcal{L}}$, $\mathcal{P}$ receives $\mathbf{U} \in \mathbb{F}_p^{(2\ell+2)\times k_\mathcal{C}}$, $\mathbf{V} \in \mathbb{F}_p^{(2\ell+2)\times n_\mathcal{C}}$, while $\mathcal{V}$ gets the message $\mathsf{done}$.
2. $\mathcal{P}$ sets $\mathbf{V}_1 = \mathbf{V}_{[1..\ell+1]}$, $\mathbf{V}_2 = \mathbf{V}_{[\ell+2..2\ell+2]}$ and $\mathbf{R} = \mathbf{U}_{[\ell+2..2\ell+2]}$
3. $\mathcal{P}$ commits to its witness by sending $\mathbf{D} = \mathbf{W} - \mathbf{U}_{[1..\ell]}$.

**Round 2.** $\mathcal{V}$ samples $\boldsymbol{\chi} \leftarrow \mathbb{F}_p^t$ and sends it to $\mathcal{P}$.

**Round 3.** $\mathcal{P}$ proceeds as follows.

1. For each $i \in [t]$, compute

$$\begin{aligned} g_i(Y) &:= \sum\nolimits_{h\in[0,2]} f_{i,h}(\mathbf{r}_1 + \mathbf{w}_1 \cdot Y, \ldots, \mathbf{r}_\ell + \mathbf{w}_\ell \cdot Y) \cdot Y^{2-h} \\ &= \sum\nolimits_{h\in[0,1]} A_{i,h} \cdot Y^h \end{aligned}$$

2. Compute $\widetilde{\mathbf{b}} = \sum_{i\in[t]} \chi_i \cdot A_{i,0} + \mathbf{r}_{\ell+1}$ and $\widetilde{\mathbf{a}} = \sum_{i\in[t]} \chi_i \cdot A_{i,1} + \mathbf{u}_{1,\ell+1}$, where $\mathbf{u}_{1,i}$ is the $i$th row of $\mathbf{U}$.
3. Send $(\widetilde{\mathbf{b}}, \widetilde{\mathbf{a}})$ to $\mathcal{V}$.

**Round 4.** $\mathcal{V}$ samples $\Delta' \leftarrow \mathbb{F}_p$ and sends it to the prover.

**Round 5.** $\mathcal{P}$ sends $\mathbf{S} = \mathbf{R} + \mathbf{U}_{[1..\ell+1]} \cdot \Delta' \in \mathbb{F}_p^{(\ell+1)\times n_\mathcal{C}}$ to $\mathcal{V}$

**Round 6.** $\mathcal{V}$ samples $\boldsymbol{\eta} \leftarrow \mathbb{F}_p^{\ell+1}$ and sends it to $\mathcal{P}$

**Round 7.** $\mathcal{P}$ computes $\widetilde{\mathbf{v}} = \boldsymbol{\eta}^\top(\mathbf{V}_2 + \mathbf{V}_1 \cdot \Delta')$ and sends it to $\mathcal{V}$.

**Verification.** $\mathcal{V}$ runs the following checks.

1. *Check the constraints:*
   - Compute $\mathbf{S}' = \mathbf{S} + \begin{bmatrix} \mathbf{D} \\ 0 \end{bmatrix} \cdot \Delta' = \mathbf{R} + \begin{bmatrix} \mathbf{W} \\ \mathbf{u}_{\ell+1} \end{bmatrix} \cdot \Delta'$.
   - For each $i \in [t]$, compute
   $$\mathbf{c}_i(Y) = \sum_{h\in[0,2]} f_{i,h}(\mathbf{s}'_1, \ldots, \mathbf{s}'_\ell) \cdot Y^{2-h}.$$
   - Let $\widetilde{\mathbf{s}} = \sum_{i\in[t]} \chi_i \cdot \mathbf{c}_i(\Delta') + \mathbf{s}'_{\ell+1}$.
   - Check that $\widetilde{\mathbf{s}} = \widetilde{\mathbf{b}} + \widetilde{\mathbf{a}} \cdot \Delta'$.
2. *Check the opening of* $\mathbf{S}$*:*
   - Call $\mathcal{F}_{\mathsf{sVOLE}}^{p,p,S_\Delta,\mathcal{C},2\ell+1,\mathcal{L}}$ on input $(\mathsf{get})$ and obtain $\Delta \in S_\Delta$ and $\mathbf{Q} \in \mathbb{F}_p^{(2\ell+2)\times n_\mathcal{C}}$ such that $\mathbf{Q} = \mathbf{V} + \mathcal{C}(\mathbf{U}) \cdot \mathsf{diag}(\Delta)$
   - Set $\mathbf{Q}_1 = \mathbf{Q}_{[1..\ell+1]}$ and $\mathbf{Q}_2 = \mathbf{Q}_{[\ell+2..2\ell+2]}$.
   - Check that
   $$\boldsymbol{\eta}^\top(\mathbf{Q}_2 + \mathbf{Q}_1 \cdot \Delta') = \widetilde{\mathbf{v}} + \boldsymbol{\eta}^\top \cdot \mathcal{C}(\mathbf{S}) \cdot \mathsf{diag}(\Delta)$$

**Figure 6.** ZK for Arbitrary Degree-2 Relation

where each sub-matrix consists of $\ell + 1$ rows. Hence, $\mathcal{P}$ commits to the witness by sending $\mathbf{D} = \mathbf{W} - \mathbf{U}_{1,[1..\ell]}$.

The idea is that $\mathcal{P}$ will run a VOLE-based ZK proof "in-the-head", as if $\mathbf{U}_1$ and $\mathbf{R}$ were a set of VOLE outputs where $\mathcal{V}$ held $\mathbf{S} = \mathbf{R}_1 + \mathbf{U}_1 \cdot \Delta'$ for some random $\Delta' \in \mathbb{F}_p$. Even though $\mathcal{V}$ does not (yet) have $\mathbf{S}$, it can send a random challenge for the proof and get the prover's response. We then have $\mathcal{V}$ send a random $\Delta'$, and have $\mathcal{P}$ open $\mathbf{S}$ so that $\mathcal{V}$ can check the proof. $\mathcal{V}$ can verify that $\mathbf{S}$ was opened reliably using the original subspace VOLE instance — if $\mathcal{P}$ tries to cheat, it must guess at least $d_\mathcal{C}$ entries of the secret $\Delta \in S_\Delta$.

The underlying VOLE-based proof that is run in-the-head is essentially the same as the protocol for proving degree-2 constraints from QuickSilver [YSWW21], and can be seen in round 3 (for $\mathcal{P}$) and the first part of round 7 (for $\mathcal{V}$). Once $\mathcal{P}$ receives the random challenge $\boldsymbol{\chi} \leftarrow \mathbb{F}_p^t$ it computes:

$$\begin{aligned} g_i(Y) &= \sum_{h\in[0,2]} f_{i,h}(\mathbf{r}_1 + \mathbf{w}_1 \cdot Y, \dots, \mathbf{r}_\ell + \mathbf{w}_\ell \cdot Y) \cdot Y^{2-h} \\ &= \sum_{h\in[0,2]} f_{i,h}(\mathbf{w}_1, \dots, \mathbf{w}_\ell) \cdot Y^2 + \sum_{h\in[0,1]} A_{i,h} \cdot Y^h \\ &= f_i(\mathbf{w}_1, \dots, \mathbf{w}_\ell) \cdot Y^2 + \sum_{h\in[0,1]} A_{i,h} \cdot Y^h, \end{aligned}$$

where $A_{i,h} \in \mathbb{F}_p^{k_\mathcal{C}}$ is the aggregated coefficient of $Y^h$. The key observation is that, if the prover $\mathcal{P}$ is honest, then $f_i(\mathbf{w}_1, \dots, \mathbf{w}_\ell) = 0$ and $g_i(Y) = \sum_{h\in[0,1]} A_{i,h} \cdot Y^h$. Using the challenge $\boldsymbol{\chi}$, $\mathcal{P}$ computes and sends to $\mathcal{V}$

$$\widetilde{\mathbf{a}} = \sum_{i\in[t]} \chi_i \cdot A_{i,1} + \mathbf{u}_{1,\ell+1} \qquad \widetilde{\mathbf{b}} = \sum_{i\in[t]} \chi_i \cdot A_{i,0} + \mathbf{r}_{\ell+1},$$

where $\mathbf{u}_{\ell+1}$ and $\mathbf{r}_{\ell+1}$ are extra rows of the original VOLE output used to mask the check values.

Next, $\mathcal{V}$ sends a challenge $\Delta' \in \mathbb{F}_p$ to $\mathcal{P}$, who opens the matrix $\mathbf{S} = \mathbf{R} + \mathbf{U}_{[1..\ell+1]} \cdot \Delta'$. This will be used as $\mathcal{V}$'s "VOLE-in-the-head" output, to check the QuickSilver proof values just sent by $\mathcal{P}$. First, though, it needs $\mathcal{P}$ to prove that $\mathbf{S}$ was sent correctly. To do this, $\mathcal{V}$ sends the last challenge $\boldsymbol{\eta} \leftarrow \mathbb{F}_p^{\ell+1}$, and gets $\widetilde{\mathbf{v}} = \boldsymbol{\eta}^\top(\mathbf{V}_2 + \mathbf{V}_1 \cdot \Delta')$ in response.[17] This is later verified in the second part of round 7, once $\mathcal{V}$ learns the subspace VOLE output $\mathbf{Q} = \mathbf{V} + \mathcal{C}(\mathbf{U}) \cdot \mathsf{diag}(\Delta)$. $\mathcal{V}$ can then use this to check the subspace VOLE relation between $\widetilde{\mathbf{v}}$ and $\boldsymbol{\eta}^\top\mathcal{C}(\mathbf{S})$, ensuring that $\mathbf{S}$ was correctly sent.

In the first part of the verification, $\mathcal{V}$ first computes

$$\mathbf{S}' = \mathbf{S} + \begin{bmatrix} \mathbf{D} \\ 0 \end{bmatrix} \cdot \Delta',$$

to adjust its subspace VOLE output to be a valid commitment to the prover's input $\mathbf{W}$. It then uses the rows of $\mathbf{S}'$ to compute polynomials $c_i(Y)$, similarly to the prover's polynomials $g_i(Y)$. These are used to check the constraints, by taking a linear combination and verifying that they form a valid VOLE correlation.

We can formally prove the result below.

**Theorem 3.** *The protocol $\Pi^t_{\mathsf{2D-LC}}$ (Figure 6) is a SHVZKPoK for arbitrary degree-2 relations with soundness error $3/p + 2\,|S'_\Delta|^{-d_\mathcal{C}}$ in the $\mathcal{F}_{\mathsf{sVOLE}}^{p,S_\Delta,\mathcal{C},2(\ell+1),\mathcal{L}}$-hybrid model.*

*Proof.* Correctness follows by inspection of the protocol. We now consider the case of a *malicious prover* and describe a simulator $\mathcal{S}$ as follows.

– $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{sVOLE}}$, it receives $\mathbf{U}$ and $\mathbf{V}$ from $\mathcal{A}$, samples $\Delta \leftarrow S_\Delta$ at random and computes $\mathbf{Q} = \mathbf{V} + \mathcal{C}(\mathbf{U}) \cdot \mathsf{diag}(\Delta)$.

[17] The challenge $\boldsymbol{\eta}$ is only used to save communication. $\mathcal{P}$ could instead directly send $\mathbf{V}_2 + \mathbf{V}_1 \cdot \Delta$ for $\mathcal{V}$ to check.

– It receives $\mathbf{D}$ from $\mathcal{A}$ and extracts the witness $\widetilde{\mathbf{w}}$. Then, it executes the rest of the protocol as an honest verifier would do. If the honest verifier aborts, then $\mathcal{S}$ returns $\perp$, otherwise it accepts $\widetilde{\mathbf{w}}$ as valid witness.

The ideal and real executions are identically distributed. Hence, we only need to bound the probability that the real-world verifier accepts a proof when the witness obtained by $\mathcal{S}$ is not valid. Let $\mathcal{P}^*$ be a malicious prover that commits to a witness $\widetilde{\mathbf{w}}$ such that exists at least one $f_i$ such that $f_i(\widetilde{\mathbf{w}}) = \mathbf{e}_i \neq 0$. We want to compute the probability that an honest verifier accepts the proof. In this case, once $\mathcal{P}^*$ receives the challenges $\boldsymbol{\chi} \leftarrow \mathbb{F}_p^t$, and later $\boldsymbol{\eta} \leftarrow \mathbb{F}_p^\ell$, from $\mathcal{V}$, it can send incorrect values $\widetilde{\mathbf{a}}^*, \widetilde{\mathbf{b}}^*, \widetilde{\mathbf{v}}_1^*$ and $\widetilde{\mathbf{v}}_2^*$, i.e.

$$\begin{aligned}
\widetilde{\mathbf{b}}^* =& \sum_{i \in [t]} \chi_i \cdot A_{i,0} + \mathbf{r}_{\ell+1} + E_{\widetilde{\mathbf{a}}} \\
\widetilde{\mathbf{a}}^* =& \sum_{i \in [t]} \chi_i \cdot A_{i,1} + \mathbf{u}_{1,\ell+1} + E_{\widetilde{\mathbf{b}}} \\
\widetilde{\mathbf{v}}^* =& \boldsymbol{\eta}^\top \cdot (\mathbf{V}_1 + \mathbf{V}_2 \cdot \Delta') + E_{\widetilde{\mathbf{v}}}
\end{aligned}$$

where $E_{\widetilde{\mathbf{a}}}, E_{\widetilde{\mathbf{b}}} \in \mathbb{F}_p^{kc}$ and $E_{\widetilde{\mathbf{v}}} \in \mathbb{F}_p^{nc}$ are adversarially chosen values. In addition, in Round 5, $\mathcal{P}^*$ can either send a matrix $\mathbf{S}^* = (\mathbf{R} + \mathbf{E_R}) + (\mathbf{U} + \mathbf{E_U}) \cdot \Delta'$ not consistent with the values previously committed or send a correct $\mathbf{S}$.

We consider the last two cases above separately, and first analyse the case when $\mathcal{P}$ sends the correct matrix $\mathbf{S}$ corresponding to the values previously committed. In particular, this means that $\mathcal{P}$ will always pass the final consistency check. Given $\mathbf{S}$ and $\mathbf{D}$, $\mathcal{V}$ honestly computes

$$\mathbf{c}_i(\Delta') = f_i(\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_\ell) \cdot \Delta'^2 + A_{i,1} \cdot \Delta' + A_{i,0}$$

and then

$$\widetilde{\mathbf{s}} = \sum_i \chi_i \cdot \mathbf{e}_i \cdot \Delta'^2 + \big(\sum_i \chi_i \cdot A_{i,1} + \Delta' + \mathbf{u}_{\ell+1}\big) \cdot \Delta' + \sum_i \chi_i \cdot A_{i,0} + \mathbf{r}_{\ell+1}$$

Finally, it must hold that $\widetilde{\mathbf{s}} = \widetilde{\mathbf{a}}^* \cdot \Delta' + \widetilde{\mathbf{b}}^*$. More precisely, we should have:

$$\begin{aligned}
& \sum_i \chi_i \cdot \mathbf{e}_i \cdot \Delta'^2 + \big(\sum_i \chi_i \cdot A_{i,1} + \mathbf{U}_{\ell+1}\big) \cdot \Delta' + \sum_i \chi_i \cdot A_{i,0} + \mathbf{R}_{\ell+1} = \\
& \big(\sum_{i \in [t]} \chi_i \cdot A_{i,1} + \mathbf{U}_{\ell+1} + E_{\widetilde{\mathbf{a}}}\big) \cdot \Delta' + \sum_{i \in [t]} \chi_i \cdot A_{i,0} + \mathbf{U}_{1,\ell+1} + E_{\widetilde{\mathbf{b}}} \\
\iff & \sum_i \chi_i \cdot \mathbf{e}_i \cdot \Delta'^2 = E_{\widetilde{\mathbf{a}}} \cdot \Delta' + E_{\widetilde{\mathbf{b}}}.
\end{aligned}$$

Again, we can have two different cases.

- $\sum_i \chi_i \mathbf{e}_i = 0$: since we are assuming that exists at least one $\mathbf{e}_i$ such that at least one of its coordinate $e_{i,j}$ is non-zero, then $\Pr[\sum_i \chi_i \mathbf{e}_i = 0] \leq 1/p$.
- $\sum_i \chi_i \mathbf{e}_i \neq 0$: if this is the case, the probability that the relation above holds is at most $2/p$, since $\Delta' \leftarrow \mathbb{F}_p$ and unknown to the prover until Round 4.

Now, we can consider the case where a malicious prover sends an incorrect matrix $\mathbf{S}^*$ and bound the probability $\mathcal{P}$ will pass the MAC check. Recall that $S'_\Delta$ is a polynomially sized subset of $\mathbb{F}_p$ and $S_\Delta = (S'_\Delta)^{n_\mathcal{C}}$. Let $\mathbf{S}^* = \mathbf{R} + \mathbf{U} \cdot \Delta' + \mathbf{E_S}$ be the value sent by the prover. The following must hold:

$$\begin{aligned}
&\boldsymbol{\eta}^\top(\mathbf{Q}_2 + \mathbf{Q}_1 \cdot \Delta') = \widetilde{\mathbf{v}} + \mathbf{E}_{\widetilde{\mathbf{v}}} + \boldsymbol{\eta}^\top \cdot \mathcal{C}(\mathbf{S} + \mathbf{E_S}) \cdot \mathsf{diag}(\boldsymbol{\Delta}) \\
\iff\ & -\mathbf{E}_{\widetilde{\mathbf{v}}} = \boldsymbol{\eta}^\top \mathcal{C}(\mathbf{E_S}) \cdot \mathsf{diag}(\boldsymbol{\Delta})
\end{aligned}$$

If $\mathbf{E}_{\widetilde{\mathbf{v}}} \neq 0$, the relation above is satisfied with probability at most $|S'_\Delta|^{-d_\mathcal{C}}$. This is because $\mathsf{diag}(\boldsymbol{\Delta})$ is unknown to $\mathcal{P}$ and each entry of $\boldsymbol{\Delta}$ is sampled uniformly from $S'_\Delta$ in $\mathcal{F}_{\mathsf{sVOLE}}$. So, the probability that at least $d_\mathcal{C}$ coordinates of $\mathcal{C}(\mathbf{E_S}) \cdot \mathsf{diag}(\boldsymbol{\Delta}) \in \mathbb{F}_p^{n_\mathcal{C}}$ are equal to the corresponding coordinates of $\mathbf{E}_{\widetilde{\mathbf{v}}}$ is at most $|S'_\Delta|^{-d_\mathcal{C}}$.

Otherwise, if $\mathbf{E}_{\widetilde{\mathbf{v}}} = 0$, then we must have $\boldsymbol{\eta}^\top \mathcal{C}(\mathbf{E_S}) \cdot \mathsf{diag}(\boldsymbol{\Delta}) = 0$. This happens either if $\mathcal{C}(\mathbf{E_S}) = 0$, in which case $\mathbf{S} = \mathbf{S}^*$, or if exist at least two indices $i, j$ such that $\eta_j \cdot \mathcal{C}(\mathbf{E}_{\mathbf{S},j}) = -\eta_i \cdot \mathcal{C}(\mathbf{E}_{\mathbf{S},i})$ which happens again with probability at most $|S_\Delta|^{-d_\mathcal{C}}$, since $\eta_i$ and $\eta_j$ are unknown to the prover when it commits to $\mathbf{S}$ and $\mathcal{C}(\mathbf{E}_{\mathbf{S},j})$ and $\mathcal{C}(\mathbf{E}_{\mathbf{S},i})$ are non-zero codewords. Summing up, the probability for a malicious prover to pass the verification step is at most $3/p + 2\,|S_\Delta|^{-d_\mathcal{C}}$. □

*Boosting the soundness.* We can improve the soundness of our protocol by using a challenge $\boldsymbol{\chi}$ in an extension field $\mathbb{F}_q$, where $q = p^r, r \leq k_\mathcal{C}$. However, this requires the masking values in $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ to be in $\mathbb{F}_q$ too. This can be achieved by calling the VOLE functionality with parameters $2(\ell + r)$ instead of $2(\ell + 2)$ and lifting rows $[\ell + 1..\ell + r]$ and $[2\ell + 1..2\ell + r]$ to $\mathbb{F}_q$. To boost the soundness related to challenge $\Delta'$, the most efficient way is that of sending two (or more) challenges $\Delta', \Delta''$ and repeating the check twice (or more, if needed).

We describe a more efficient protocol for small values of $p$ in the next section.

*Communication cost.* In $\Pi^t_{\mathsf{2D-LC}}$, given in Figure 6, other than the cost of the sVOLE step, the prover has to send the initial commitment, consisting of a matrix in $\mathbb{F}_p^{\ell \times k_\mathcal{C}}$, then two vectors in $\mathbb{F}_p^{k_\mathcal{C}}$, one vectors in $\mathbb{F}_p^{n_\mathcal{C}}$ and finally a matrix in $\mathbb{F}_q^{\ell \times k_\mathcal{C}}$. Summing up the cost is

$$\mathsf{CommCost}_{\Pi^t_{\mathsf{2D-LC}}} = \mathsf{CommCost}_{\mathsf{sVOLE}} + (2\ell + 2) \cdot \Big(k_\mathcal{C} \cdot \log_2 p\Big) + n_\mathcal{C} \cdot \log_2 p.$$

Note that is roughly 2 times the cost of QuickSilver and other VOLE-based protocols in the designated-verifier setting. Using our protocol from Section 5, $\mathsf{CommCost}_{\mathsf{sVOLE}}$ is dominated by $\ell \cdot (n_\mathcal{C} - k_\mathcal{C})$ field elements, so this part is sublinear in the witness length $(\ell \cdot k_\mathcal{C})$ if $\mathcal{C}$ has a good enough rate.

### 6.2 ZK for Degree-2 from Small-Sized sVOLE

For small fields, the previous protocol would not perform so well, since we'd need many repetitions to achieve a good soundness error. Instead, a better approach is to adopt the QuickSilver protocol [YSWW21] with subspace VOLE based on the $[\tau, 1, \tau]$ repetition code. This avoids the need for the code-switching step of the previous protocol, with the additional $\Delta'$ challenge, since the ZK proof can be done directly on repetition coded VOLE. To help with our AES use-case, we generalize QuickSilver slightly to allow for proving constraints over an extension field $\mathbb{F}_{p^k}$, even when the witness is committed over $\mathbb{F}_p$.

**Protocol $\Pi^t_{\mathsf{2D\text{-}Rep}}$**

PARAMETERS: Code $\mathcal{C}_{\mathsf{Rep}} = [\tau, 1, \tau]_p$ with $\mathbf{G}_{\mathcal{C}} = (1 \ldots 1) \in \mathbb{F}_p^{1\times\tau}$. VOLE size $q = p^r$.
INPUTS: Polynomials $f_i \in \mathbb{F}_{p^k}[X_1, \ldots, X_\ell]_{\leq 2}$, $i \in [t]$. The prover $\mathcal{P}$ also holds a witness $\mathbf{w} \in \mathbb{F}_p^\ell$ such that $f_i(\mathbf{w}) = 0$ for all $i \in [t]$.

**Round 1.** $\mathcal{P}$ does the following:

1. Call the functionality $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathcal{C}_{\mathsf{Rep}},\ell+r\tau,\mathcal{L}}$ and receive $\mathbf{u} \in \mathbb{F}_p^{\ell+r\tau}, \mathbf{V} \in \mathbb{F}_q^{(\ell+r\tau)\times\tau}$.
   $\mathcal{V}$ receives $\mathsf{done}$.
2. Compute $\mathbf{d} = \mathbf{w} - \mathbf{u}_{[1..\ell]} \in \mathbb{F}_p^\ell$ and send $\mathbf{d}$ to $\mathcal{V}$.
3. For $i \in [\ell+1..\ell+r\tau]$, embed $u_i \hookrightarrow \mathbb{F}_{q^\tau}$.
   For $i \in [\ell+r\tau]$, lift $\mathbf{v}_i \in \mathbb{F}_q^\tau$ into $v_i \in \mathbb{F}_{q^\tau}$.
   For $i \in [\ell]$, also embed $w_i \hookrightarrow \mathbb{F}_{q^\tau}$.

**Round 2.** $\mathcal{V}$ sends challenges $\chi_i \in \mathbb{F}_{q^\tau}$, $i \in [t]$.

**Round 3.** $\mathcal{P}$ does the following:

1. For each $i \in [t]$, compute $A_{i,0}, A_{i,1} \in \mathbb{F}_{q^\tau}$ such that
$$c_i(Y) = \bar{f}_i(w_1, \ldots, w_n) \cdot Y^2 + A_{i,1} \cdot Y + A_{i,0}.$$
2. Compute
$$u^* = \sum_{i\in[r\tau]} u_i X^{i-1} \qquad v^* = \sum_{i\in[r\tau]} v_i X^{i-1},$$
   where $\mathbb{F}_{q^\tau} \simeq \mathbb{F}_p[X]/F(X)$.
3. Compute $\tilde{b} = \sum_{i\in[t]} \chi_i \cdot A_{i,0} + v^* \in \mathbb{F}_{q^\tau}$ and $\tilde{a} = \sum_{i\in[t]} \chi_i \cdot A_{i,1} + u^* \in \mathbb{F}_{q^\tau}$ and send $(\tilde{a}, \tilde{b})$ to $\mathcal{V}$.

**Verification.** $\mathcal{V}$ runs the following check:

1. Call $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathcal{C}_{\mathsf{Rep}},\ell+r\tau,\mathcal{L}}$ on input $(\mathsf{get})$ and obtain $\boldsymbol{\Delta} \in \mathbb{F}_q^\tau$, $\mathbf{Q} \in \mathbb{F}_q^{(\ell+r\tau)\times\tau}$ such that $\mathbf{Q} = \mathbf{V} + \mathbf{u}^T\mathbf{G}_{\mathcal{C}}\mathsf{diag}(\boldsymbol{\Delta})$.
2. Compute $\mathbf{Q}' = \mathbf{Q}_{[1..\ell]} + \mathbf{d}^T\mathbf{G}_{\mathcal{C}}\mathsf{diag}(\boldsymbol{\Delta}) = \mathbf{V}_{[1..\ell]} + \mathbf{w}^T\mathbf{G}_{\mathcal{C}}\mathsf{diag}(\boldsymbol{\Delta})$.
3. Lift $\boldsymbol{\Delta}, \mathbf{q}'_1, \ldots, \mathbf{q}'_\ell, \mathbf{q}_{\ell+1}, \ldots, \mathbf{q}_{\ell+r\tau} \in \mathbb{F}_q^\tau$ into $\Delta, q'_1, \ldots, q'_\ell, q_{\ell+1}, \ldots, q_{\ell+r\tau} \in \mathbb{F}_{q^\tau}$.
4. For each $i \in [t]$, compute
$$c_i(\Delta) = \sum_{h\in[0,2]} \bar{f}_{i,h}(q'_1, \ldots, q'_\ell) \cdot \Delta^{2-h}$$
5. Compute $q^* = \sum_{i\in[r\tau]} q_{\ell+i} \cdot X^{i-1}$ such that $q^* = v^* + u^*\Delta$.
6. Compute $\tilde{c} = \sum_{i\in[t]} \chi_i \cdot c_i(\Delta) + q^*$.
7. Check that $\tilde{c} \stackrel{?}{=} \tilde{a} \cdot \Delta + \tilde{b}$.

**Figure 7.** ZK protocol for $t$ degree-2 relations from small $\mathsf{sVOLE}$ with repetition code.

In Figure 7, we present our public-coin ZK protocol in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model over small- to medium-sized fields which uses a repetition code $\mathcal{C}_{\mathsf{Rep}}$ with parameters $[\tau, 1, \tau]$ for soundness amplification.

As for the $\Pi^t_{\mathsf{2D-LC}}$ protocol of Figure 6, protocol $\Pi^t_{\mathsf{2D\text{-}Rep}}$ of Figure 7 proves knowledge of a witness vector $\mathbf{w} \in \mathbb{F}_p^\ell$ which satisfies a set ot $t$ degree-2 constraints $f_i \in \mathbb{F}_{p^k}[X_1, \ldots, X_\ell]_{\leq 2}$ expressed over a degree-$k$ extension of $\mathbb{F}_p$. As before, we note that each $f_i$ can be expressed as $f_i = f_{i,0} + f_{i,1} + f_{i,2}$ such that $\deg f_{i,h} = h$.

The sVOLE functionality we use here is parametrised with prime $p$ and tag size $q = p^r$ which is a small- to medium-size extension (in practice we will take $q \approx 2^{10}$). Due to the non-cryptographic size of $q$, we design the protocol so that each output $u$ of $\mathcal{F}_{\mathsf{sVOLE}}$ receives $\tau$ independent tags $v_i \in \mathbb{F}_q$, each under a distinct VOLE key $\Delta_i \in \mathbb{F}_q$. In effect, this protocol uses the subfield VOLE functionality $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathcal{C}_{\mathsf{Rep}},\ell+r\tau,\mathcal{L}}$ of Figure 2 where $S_\Delta$ is the whole of $\mathbb{F}_q$. By taking $r\tau \approx \lambda$, we achieve soundness amplification.

The intuition of the scheme is that the prover will first use $\ell$ sVOLE outputs to mask its $\ell$ input values over the base field $\mathbb{F}_p$. After sending these masked inputs, the prover then embeds each input value $w_i \hookrightarrow \mathbb{F}_{q^\tau}$ together with lifting each corresponding tag vector $\mathbf{v}_i \in \mathbb{F}_q^\tau$ into $v_i \in \mathbb{F}_{q^\tau}$. This allows the prover to compute the following polynomial, for $i \in [t]$:

$$\begin{aligned} c_i(Y) &= \sum_{h \in [0,2]} \bar{f}_{i,h}(v_1 + w_1 \cdot Y, \ldots, v_\ell + w_\ell \cdot Y) \cdot Y^{2-h} \\ &= \bar{f}_i(w_1, \ldots, w_n) \cdot Y^2 + A_{i,1} \cdot Y + A_{i,0}, \end{aligned}$$

where $\bar{f}_i \in \mathbb{F}_{q^\tau}[X_1, \ldots, X_\ell]$ is the embedding of $f_i \in \mathbb{F}_{p^k}[X_1, \ldots, X_\ell]$ and $A_{i,h} \in \mathbb{F}_{q^\tau}$ is the aggregated coefficient of $Y^h$.

*Note.* The embedding of $f_i$ into $\bar{f}_i$ requires that $k \mid r\tau$, which we assume to be the case in our presentation of $\Pi^t_{\mathsf{2D\text{-}Rep}}$ in Figure 7. If $k \nmid r\tau$, then it is possible to truncate the embedded and lifted elements of $\mathbb{F}_{q^\tau}$ down to elements of $\mathbb{F}_{p^{\tilde{r}k}}$ for a suitable multiple $\tilde{r}k$.

Having received $t$ independent challenges $\chi_i \in \mathbb{F}_{q^\tau}$ from the verifier (after sending its masked inputs), the prover can compress its $t$ pairs of aggregated coefficients $(A_{i,0}, A_{i,1})_i$ by computing

$$b = \sum_{i \in [t]} \chi_i \cdot A_{i,0} \qquad a = \sum_{i \in [t]} \chi_i \cdot A_{i,1}.$$

Since $b$ and $a$ contain information about the input values $w_i$, the prover must also mask them with $v^* \in \mathbb{F}_{q^\tau}$ and $u^* \in \mathbb{F}_{q^\tau}$ respectively.

Due to the nature of the QuickSilver-style check that protocol $\Pi^t_{\mathsf{2D\text{-}Rep}}$ performs, these masks must also satisfy the VOLE correlation that $v^* = q^* - u^* \Delta$ for some $q^* \in \mathbb{F}_{q^\tau}$ and with $\Delta \in \mathbb{F}_{q^\tau}$ corresponding to the lift of the $\tau$ values $\Delta_i$. Since $u^*$ must be a uniform mask in $\mathbb{F}_{q^\tau}$, and not just $\mathbb{F}_p$ in which live the sVOLE output values $u$, the correlated values $u^*$ and $v^*$ must be constructed from $r\tau$ independent sVOLE correlations, similarly to the $\mathsf{VOPE}$ instruction of the $\Pi^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ protocol of Yang et al. [YSWW21]. This is why $\Pi_{\mathsf{2D\text{-}Rep}}$ requires a total of $\ell + r\tau$ sVOLE outputs.

After receiving the masked values $\tilde{a} = a + v^*$ and $\tilde{b} = b + u^*$, the verifier can then proceed to the final round of the protocol. First, it receives the sVOLE global keys $\boldsymbol{\Delta} \in \mathbb{F}_q^\tau$ and the value keys $\mathbf{Q} \in \mathbb{F}_q^{(\ell + r\tau) \times \tau}$ and adjusts the first $\ell$ rows of $\mathbf{Q}$ using the masked input values $\mathbf{d}$ it received from the prover in Round 1.

Next, after lifting $\boldsymbol{\Delta}$ and the $\ell$ adjusted rows $\mathbf{q}'_i$ into $\mathbb{F}_{q^\tau}$, the verifier can compute its half of the QuickSilver-style check:

$$c_i(\Delta) = \sum_{h \in [0,2]} \bar{f}_i(q'_1, \ldots, q'_\ell) \Delta^{2-h}.$$

Before checking this against $\tilde{a}$ and $\tilde{b}$ received from the prover, the verifier must also compress its $t$ values using the $\chi_i$ challenges and apply the corresponding $q^*$ mask, computed from the last $r\tau$ rows of the sVOLE output.

Finally, the verifier can verify the equality $\tilde{c} \stackrel{?}{=} \tilde{a} \cdot \Delta + \tilde{b}$, which will hold if the prover acted honestly.

**Theorem 4.** *The protocol $\Pi^t_{\mathsf{2D\text{-}Rep}}$ (Figure 7) is a SHVZKPoK for arbitrary degree-2 relations with soundness error $1/p^{r\tau} + 2|S_\Delta|^{-1}$ in the $\mathcal{F}^{p,q,\tau,\ell+\tau r}_{\mathsf{sVOLE}}$-hybrid model.*

*Proof.* We prove security of $\Pi^t_{\mathsf{2D\text{-}Rep}}$ in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model against (1) a malicious prover $\mathcal{P}^*$ and (2) a malicious verifier $\mathcal{V}^*$. In both cases we construct a simulator $\mathcal{S}$ which interacts with the malicious party $\mathcal{A}$.

*Malicious prover.* The simulator emulates $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathcal{C}_{\mathsf{Rep}},\ell+r\tau}$ for $\mathcal{A}$ by sampling $\tau$ uniform values $\Delta_i \in \mathbb{F}_q$, receiving $\mathbf{u} \in \mathbb{F}_p^{\ell+r\tau}$ and $\mathbf{V} \in \mathbb{F}_q^{(\ell+r\tau)\times\tau}$ from $\mathcal{A}$, and computing $\mathbf{Q} \in \mathbb{F}_q^{(\ell+r\tau)\times\tau}$ as shown in Figure 2. When $\mathcal{A}$ sends $\mathbf{d} \in \mathbb{F}_p^{\ell}$ in step 2 of Round 1, $\mathcal{S}$ computes $\mathbf{w} = \mathbf{d} + \mathbf{u}_{[1..\ell]}$. $\mathcal{S}$ then executes the rest of the protocol as an honest verifier, sampling the $t$ challenges $\chi_i \in \mathbb{F}_{q^\tau}$ at random and using $\boldsymbol{\Delta}$ and $\mathbf{Q}$ as defined during the emulation of $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,\mathcal{C}_{\mathsf{Rep}},\ell+r\tau}$ to perform the verification step. If the check at step 7 of the verification fails, $\mathcal{S}$ returns $\mathbf{w} = \perp$; if it passes, $\mathcal{S}$ outputs $\mathbf{w}$ as a valid witness.

Due to the honest sampling of the $\Delta_i$ and $\chi_i$ values, the view of $\mathcal{A}$ in the simulation is identically distributed to the real-world execution; also, if a real-world verifier would reject $\mathcal{A}$'s proof, so would the ideal-world verifier since $\mathcal{S}$ returns $\perp$. We therefore bound the error probability $\epsilon$ of a real-world verifier accepting the proof when in fact the witness $\mathbf{w}$ extracted by $\mathcal{S}$ does not satisfy the constraint system $f_i$.

Let $f_i(\mathbf{w}) = y_i$ for some $y_i \in \mathbb{F}_{p^k}$, for each $i \in [t]$, where $\mathbf{w}$ is extracted from $\mathcal{A}$ by $\mathcal{S}$ as above. Following from the definition of $c_i(\Delta)$, for $i \in [t]$, we have

$$\begin{aligned} c_i(\Delta) &= \bar{f}_i(\mathbf{w}) \cdot \Delta^2 + A_{i,1} \cdot \Delta + A_{i,0} \\ &= y_i \cdot \Delta^2 + A_{i,1} \cdot \Delta + A_{i,0}, \end{aligned}$$

where the embedded constraints $\bar{f}_i$ produce an embedded $y_i \hookrightarrow \mathbb{F}_{q^\tau}$.

At step 3 of Round 3, the simulator receives $\tilde{b}' = \tilde{b} + e_b$ and $\tilde{a}' = \tilde{a} + e_a$ from $\mathcal{A}$, where $\tilde{b}$ and $\tilde{a}$ are the values computed according to $\mathbf{w}$ and the additional $r\tau$ sVOLE correlations used for $u^*$ and $v^*$, and $e_b, e_a \in \mathbb{F}_{q^\tau}$ are error terms chosen by $\mathcal{A}$. By expanding the computation of $\tilde{c}$, we have:

$$\begin{aligned} \tilde{c} &= \sum_{i\in[t]} \chi_i \cdot c_i(\Delta) + q^* = \sum_{i\in[t]} \chi_i \cdot \left(y_i \cdot \Delta^2 + A_{i,1} \cdot \Delta + A_{i,0}\right) + v^* + u^*\Delta \\ &= \Delta^2 \cdot \sum_{i\in[t]} y_i\chi_i + \Delta \cdot \left(\sum_{i\in[t]} A_{i,1}\chi_i + u^*\right) + \sum_{i\in[t]} A_{i,0} + v^* \\ &= \Delta^2 \cdot \sum_{i\in[t]} y_i\chi_i + \Delta \cdot (\tilde{a}' - e_a) + (\tilde{b}' - e_b) \end{aligned}$$

For the real-world verifier to accept the proof, then it must hold that $\tilde{c} = \tilde{a}' \cdot \Delta + \tilde{b}'$, which then implies

$$\Delta^2 \cdot \sum_{i\in[t]} y_i\chi_i - \Delta e_a - e_b = 0. \tag{3}$$

If the random choice of $\chi_i$ is such that $\sum_{i\in[t]} y_i\chi_i = 0$, then the best prover strategy is to set $e_a = e_b = 0$, i.e. compute $\tilde{b}$ and $\tilde{a}$ honestly, which means that Equation (3) will hold with probability 1 for any value of $\Delta$. By assumption that the extracted witness $\mathbf{w}$ does not satisfy the constraint system, there must be at least one $y_i \neq 0$, which implies that $\sum_{i\in[t]} y_i\chi_i = 0$ with probability at most $1/p^{r\tau}$ since the $\chi_i \in \mathbb{F}_{q^\tau}$ are sampled after the $y_i$ are committed to.

If $\sum_{i\in[t]} y_i\chi_i \neq 0$ then Equation (3) can be solved for at most two values of $\Delta$ by the simulator based on the values it received from $\mathcal{A}$. If there are no solutions, $\mathcal{S}$ aborts; otherwise, it submits the solution(s) $\boldsymbol{\Delta}_i \in \mathbb{F}_q^\tau$ to its internally simulated $\mathcal{F}_{\mathsf{sVOLE}}$ which aborts if they differ from the sampled values.

Since $\Delta \leftarrow S_\Delta$ uniformly at random and it is hidden from $\mathcal{A}$, the solutions to Equation (3) will equal the sampled $\Delta$ with probability at most $2/|S_\Delta|$. Overall, the soundness is therefore bounded by $1/p^{r\tau} + 2|S_\Delta|^{-1}$.

*Malicious verifier.* The simulator emulates $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathcal{C}_{\mathsf{Rep}},\ell+r\tau}$ for $\mathcal{A}$ by receiving $\tau$ uniform values $\Delta_i \in \mathbb{F}_q$ and $\ell + r\tau$ outputs $\mathbf{Q} \in \mathbb{F}_q^{(\ell+r\tau)\times\tau}$. It then simulates Round 1 of the prover by sending $\ell$ uniform values $\mathbf{d} \in \mathbb{F}_p^\ell$ to $\mathcal{A}$. To simulate $\tilde{a}$ and $\tilde{b}$, $\mathcal{S}$ first computes $\tilde{c}$ according to the $\boldsymbol{\Delta}$ and $\mathbf{Q}$ it received before sampling $\tilde{a}$ at random and setting $\tilde{b} = \tilde{c} - \tilde{a} \cdot \Delta$.

Since $\mathbf{u}$ and $\mathbf{V}$ are kept secret from $\mathcal{A}$, the communication simulated by $\mathcal{S}$ is indistinguishable from a real-world execution. $\square$

*Communication cost.* In $\Pi^t_{\mathsf{2D\text{-}Rep}}$ of Figure 7, in addition to the cost of the sVOLE step, the prover sends the initial commitment $\mathbf{d} \in \mathbb{F}_p^\ell$, as well as two values $\tilde{a}, \tilde{b} \in \mathbb{F}_{q^\tau}$. Summing up the cost is

$$\mathsf{CommCost}_{\Pi^t_{\mathsf{2D\text{-}Rep}}} = \mathsf{CommCost}_{\mathsf{sVOLE}} + \ell \cdot \log_2 p + 2 \cdot r\tau \log_2 p.$$

In addition, the verifier sends $t$ values in $\mathbb{F}_{q^\tau}$ but this can be optimized in the interactive setting and does not contribute to the final proof size in the non-interactive setting.

We can extend both $\Pi^t_{\mathsf{2D-LC}}$ and $\Pi^t_{\mathsf{2D\text{-}Rep}}$ to handle degree-$d$ relations, for small $d$, with the technique described in Appendix B.1.

## 7 FAEST: AES-based Signature

We can use our non-interactive zero-knowledge protocol, obtained by applying the methodology described in Section 4 with the ZK scheme from small-sized subspace VOLE of Section 6.2 to build a Picnic-like post-quantum signature scheme based on AES.

More precisely, given a block cipher $E$, AES in our case, we define a family of one-way functions (OWF) $\{f_x\}$ such that $f_x(k) = E_k(x)$, where $E_k(x)$ denotes the encryption of $x$ under the key $k$. In this way, the private key $k$ and public values $(x, y)$, with $x$ sampled uniformly at random and $y = E_k(x)$, define the OWF relation $((x, y), k) \in R \iff E_k(x) = y$. Hence, a signature on a message $\mu$ is generated by binding $\mu$ with a non-interactive zero-knowledge proof of knowledge of $k$.

Recent works, starting with [GMO16, CDG+17], have used this approach to build efficient post-quantum secure signatures from MPCitH-based non-interactive zero-knowledge schemes [IKOS07]. One such scheme is Picnic [CDG+17, KKW18] which is a third-round alternate candidate for the NIST post-quantum standardization process. While Picnic relies on the LowMC block cipher [ARS+15] as the underlying OWF, a non-standard assumption, more recent works replaced LowMC with AES [dDOS19] or other well-studied problems such as the syndrome decoding (SD) [FJR22b].

Another scheme solely based on symmetric-key primitives is SPHINCS+ [HBD+22] which is one of three recently standardized by NIST, while the other two, Falcon [PFH+22] and Dilithium [LDK+22] are based on public-key lattice problems.

In the rest of this section, we first describe our signature scheme, FAEST, in more detail and then compare it to other post-quantum secure schemes.

### 7.1 The FAEST Signature Scheme

The main tool to build our signature scheme is a NIZK scheme, $\Pi_{\mathsf{FAEST}}$, obtained by applying Lemma 4 to the QuickSilver-style protocol $\Pi_{\mathsf{2D\text{-}Rep\text{-}OT}}$ set in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model, given by composing $\Pi^t_{\mathsf{2D\text{-}Rep}}$ of Figure 7 (in Section 6.2) with $\Pi^{p,q,S_\Delta,\mathcal{C},\ell}_{\mathsf{small\text{-}VOLE}}$ of Figure 4, where $S_\Delta = \mathbb{F}_q$, $q = p^r$, $\mathcal{C} = \mathcal{C}_{\mathsf{Rep}} = [\tau, 1, \tau]$ and $\mathcal{H}$ is an $\varepsilon$-universal hash family such that $\varepsilon\binom{\tau}{2} \leq 2/p^{r\tau}$. In particular, we prove the following result.

**Theorem 5.** *The $\Pi_{\mathsf{FAEST}}$ protocol, defined as*

$$\Pi_{\mathsf{FAEST}} = \mathsf{FS}^{H_{\mathsf{FS}}}[\mathsf{O2C}^{H_{\mathsf{O2C}}}[\Pi_{\mathsf{2D\text{-}Rep\text{-}OT}}]],$$

*is a zero-knowledge non-interactive proof system in the CRS+RO model with knowledge error*

$$2 \cdot (Q_{\mathsf{FS}} + Q_{\mathsf{Verify}}) \cdot \frac{2}{p^{r\tau}} + M \cdot (Q_{\mathsf{FS}} + Q_{\mathsf{Verify}}) \cdot \mathsf{AdvEB}^{\mathsf{VC}}_{\mathcal{A}'}[Q_{H_{\mathsf{O2C}}}] + \mathsf{AdvDist}^{\mathsf{VC.Setup},\mathsf{VC.TSetup}}_{\mathcal{D}},$$

*where $M$ is an upper bound on the number of* VC *commitments sent during a run of* $\mathsf{O2C}[\Pi_{\mathsf{2D\text{-}Rep\text{-}OT}}]$.

*Proof.* We apply Lemma 4 to the composed protocol $\Pi_{\mathsf{2D\text{-}Rep\text{-}OT}}$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model. First, by looking at the init phase of $\Pi_{\mathsf{small\text{-}VOLE}}$, we see that the sending of $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$ by the prover can be combined with the sending of $\mathbf{d}$ from $\Pi^t_{\mathsf{2D\text{-}Rep}}$. The combined protocol $\Pi_{\mathsf{2D\text{-}Rep\text{-}OT}}$ therefore contains a total of $\mu = 2$ verifier messages and runs in a total of $2\mu + 1 = 5$ rounds. Next, we analyse the round-by-round knowledge error $\kappa$ of $\Pi_{\mathsf{2D\text{-}Rep\text{-}OT}}$.

1. When the prover sends the correction $\mathbf{C}$ in $\Pi_{\mathsf{small\text{-}VOLE}}$, the probability that applying the universal hash $\mathbf{H} \leftarrow \mathcal{H}$ hides any cheating in $\mathbf{U}$ and $\mathbf{V}$ is $\varepsilon\binom{\tau}{2}$, where $\mathcal{H}$ is $\varepsilon$-universal.
2. Similarly, when the prover sends the correction $\mathbf{d}$ in $\Pi^t_{\mathsf{2D\text{-}Rep}}$, the probability that the random challenges $\chi_i$ hide a non-zero result of the constraint system is $1/p^{r\tau}$.
3. Given the responses $\tilde{a}, \tilde{b}$ for $\Pi^t_{\mathsf{2D\text{-}Rep}}$ and $\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}}$ for $\Pi_{\mathsf{small\text{-}VOLE}}$, the probability that the implicit challenge $\Delta$ hides errors in both of these at the same time is upper-bounded by the probability it hides errors in $\tilde{a}, \tilde{b}$ and is therefore at most $2/p^{r\tau}$.

Since $\mathcal{H}$ is such that $\varepsilon\binom{\tau}{2} \leq 2/p^{r\tau}$, this gives $\kappa = 2/p^{r\tau}$ as round-by-round knowledge error.

Finally, since both $\Pi_{\mathsf{small\text{-}VOLE}}$ and $\Pi^t_{\mathsf{2D\text{-}Rep}}$ are SHVZK, then the FS-trans-formed compiled protocol is indeed a NIZK in the programmable ROM. ◻

To implement the signature scheme FAEST, we expressed the AES-128 algorithm, including its key schedule, as a set of 200 degree-2 constraints over an extension $\mathbb{F}_{q^\tau}$ of $\mathbb{F}_{2^8}$. Since the AES S-box is a field inversion over $\mathbb{F}_{2^8}$ (for non-zero inputs), each constraint checks that the output of each inversion is the valid inverse of a linear combination of the outputs of the previous layer, where this linear combination represents the AES linear layer.

### 7.2 Implementation

We implemented FAEST in Rust[18] for AES-128 using $\mathbb{F}_{2^{128}}$ as well as the fields $\mathbb{F}_q$ with $q = 2^7, \ldots, 2^{11}$. In the implementation we used AES in counter mode and ChaCha20 as PRGs and the Blake3 hash and extendable output function. We ran our experiments on an Intel Core i9-9900

**Table 2.** Runtimes for signing and verification as well as signature sizes for FAEST with AES-128 and $\lambda = 128$.

| $q$ | $t_{\mathcal{P}}$ in ms | $t_{\mathcal{V}}$ in ms | $\lvert\mathsf{sign}\rvert$ in B |
|---|---|---|---|
| $2^7$ | 2.631 | 2.431 | 7 506 |
| $2^8$ | 2.279 | 2.109 | 6 583 |
| $2^9$ | 4.303 | 3.952 | 6 435 |
| $2^{10}$ | 6.447 | 5.941 | 5 803 |
| $2^{11}$ | 11.053 | 10.184 | 5 559 |

CPU. For comparison, we also benchmarked Limbo [dOT21] and SPHINCS+[HBD+22] on the same hardware. The runtimes and signatures sizes for FAEST are given in Table 2.

Varying the field size parameter $q$ gives a trade-off between computation time and signature size: With larger $q$ the signature size shrinks, but both the signer and the verifier need to perform computation linear in $q$. The fastest instantiation with about 2.2 ms for signing and verification is obtained by setting $q = 2^8$, which also exploits that an $\mathbb{F}_q$ element fits exactly into a byte.

### 7.3 Comparison with other PQ Signatures.

We focus on MPCitH protocols based on AES or code-based assumptions which we recall below.

*AES-based MPCitH signature schemes* While AES is the first natural choice as block cipher in MPCitH schemes, it leads to large signatures since the AES circuit over $\mathbb{F}_2$ is far more complex than LowMC in term of non-linear AND gates. In BBQ [dDOS19], it was proposed to evaluate AES directly on $\mathbb{F}_{2^8}$ instead, such that the only non-linear operation remaining are the S-box inversions; this reduced the proof size by about 40%. Further improvements were introduced in Banquet [BdK+21], Limbo [dOT21] and the Helium proof system [KZ22]. We report in Table 3 runtimes of Limbo [dOT21], as well as the numbers for the AES-based scheme Helium+AES from [KZ22], which outperforms Limbo. With FAEST, we managed to obtain signatures that are around 2× smaller then the fast version of Helium+AES, while having comparable runtimes for signing and verification. Compared with the short variant of Helium+AES, our two FAEST variants both perform faster and have around 35–45% smaller signatures.

*Non-Standard Variants of AES [DKR+22].* To reduce the size of AES-based signatures, Dobraunig et al. proposed new methods which also improved the overall performance of signature and verification [DKR+22]. Their approach differs from previous ones mainly with their use of different OWF that are more ZK-friendly. First, they show how to safely remove the key-schedule from the MPC protocol using the single-key Even-Mansour (EM) scheme [DKS12], effectively reducing the number of S-boxes from 200 to 160 for AES-128. Secondly, they propose a different variant of AES with larger S-boxes (LSAES), which is more amenable to zero-knowledge schemes over large fields. Finally, they describe a new OWF, Rain, specifically tailored for MPCitH schemes, which combines both the EM and LSAES tricks mentioned above and additionally modifies the AES linear layers. These techniques were also incorporated into subsequent improvements on the zero-knowledge side [KZ22] (building on [BN20]), and led to signatures as small as 5kB, with a conservative 4-round version of Rain. In Table 3, we show timings from [KZ22] for BN++Rain, a signature based on the 4-round version of Rain.

[18] Source code is available at https://github.com/faest-sign/faest-rs/tree/crypto-2023.

**Table 3.** Comparison of timings and signature sizes at the 128-bit security level for some standardized schemes from the NIST PQC standardization project, new alternatives and the designs explored in this work. The results for FAEST, Limbo [dOT21], and SPHINCS+ [HBD+22] were obtained on the system described in Section 7.2. The other numbers are taken from [FJR22b] and [KZ22].

| Scheme | $t_{\mathcal{P}}$ (ms) | $t_{\mathcal{V}}$ (ms) | \|sign\| (B) | Assumption |
|---|---|---|---|---|
| SDitH [FJR22b] (fast) | 13.40 | 12.70 | 17 866 | SD $\mathbb{F}_2$ |
| SDitH [FJR22b] (short) | 64.20 | 60.70 | 12 102 | SD $\mathbb{F}_2$ |
| SDitH [FJR22b] (fast) | 6.40 | 5.90 | 12 115 | SD $\mathbb{F}_{256}$ |
| SDitH [FJR22b] (short) | 29.50 | 27.10 | 8 481 | SD $\mathbb{F}_{256}$ |
| BN++Rain$_4$ [KZ22] (fast) | 2.52 | 2.36 | 5 536 | Rain$_4$ |
| BN++Rain$_4$ [KZ22] (short) | 4.79 | 4.53 | 4 992 | Rain$_4$ |
| Helium+AES [KZ22] (fast) | 9.87 | 9.60 | 11 420 | Hash/AES |
| Helium+AES [KZ22] (short) | 16.53 | 16.47 | 9 888 | Hash/AES |
| Limbo [dOT21] (fast) | 2.61 | 2.25 | 23 264 | Hash/AES |
| Limbo [dOT21] (short) | 24.51 | 21.82 | 13 316 | Hash/AES |
| SPHINCS+-SHA2 [HBD+22] (fast) | 4.40 | 0.40 | 17 088 | Hash |
| SPHINCS+-SHA2 [HBD+22] (short) | 88.21 | 0.15 | 7 856 | Hash |
| Falcon-512 [PFH+22] | 0.12 | 0.03 | 666 | Lattice |
| Dilithium2 [LDK+22] | 0.06 | 0.03 | 2 420 | Lattice |
| FAEST (**this work**, fast, $q = 2^8$) | 2.28 | 2.11 | 6 583 | Hash/AES |
| FAEST (**this work**, short, $q = 2^{11}$) | 11.05 | 10.18 | 5 559 | Hash/AES |

In parameter settings with a similar signature size, FAEST seems to perform several times slower than using Rain, while using the standard AES. However, the runtimes were obtained on different hardware, which prohibits an exact comparison. We could also use these alternative OWFs in FAEST — we estimate that Even-Mansour-based AES could reduce sizes by 10–15%, while using Rain could give a 30–40% reduction, giving smaller signatures than BN++Rain$_4$.

*Code-based MPCitH schemes.* In a recent work, Feneuil et al. proposed an MPCitH-based signature scheme where the 5-round ZK protocol is a PoK of a vector $x$ such that $y = Hx$, where $x$ is a vector with Hamming weight $wt(x)$ less than a fixed $t$ [FJR22b]. The resulting scheme, in addition to being competitive with SPHINCS+, outperforms all the known code-based signatures. Another very recent work, [AMGH+22], presents a new approach to amplify the soundness of MPCitH protocols. When applied to build code-based signature schemes, it shows concrete improvement over [FJR22b] in running time. In Table 3, we also include the scheme of Feneuil et al., reporting directly the estimations given in their paper [FJR22b]. Compared to this scheme, we achieve both smaller signature sizes and faster running time.

**Acknowledgments**

The work of Michael Klooß was supported by KASTEL Security Research Labs and by Helsinki Institute for Information Technology HIIT. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Emmmanuela Orsini, Lawrence Roy and Peter Scholl have been supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of the funders. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. Lennart Braun has been further supported by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC). Cyprien Delpech de Saint Guilhem is a Junior FWO Postdoctoral Fellow under project 1266123N and was also supported by CyberSecurity Research Flanders with reference number VR20192203. Peter Scholl was also supported by the Aarhus University Research Foundation, and the Independent Research Fund Denmark under project number 0165-00107B (C3PO).

## References


AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

AMGH+22. Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the SDitH. Cryptology ePrint Archive, Paper 2022/1645, 2022. https://eprint.iacr.org/2022/1645.

ARS+15. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, April 2015.

BBC+17. Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear PCPs. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 551–579. Springer, Heidelberg, April / May 2017.

BBD+23. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-Head. In *CRYPTO*. Springer, 2023.

BBMHS22. Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. Moz$\mathbb{Z}_{2^k}$arella: Efficient vector-OLE and zero-knowledge proofs over $\mathbb{Z}_{2^k}$. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 329–358. Springer, Heidelberg, August 2022.

BCG+19. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

BCG20. Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-time arguments with sublinear verification from tensor codes. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 19–46. Springer, Heidelberg, November 2020.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

BdK+21. Carsten Baum, Cyprien de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 266–297. Springer, Heidelberg, May 2021.

BMRS21. Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 92–122, Virtual Event, August 2021. Springer, Heidelberg.

BN20. Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 495–526. Springer, Heidelberg, May 2020.

BW13. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

CCH+19. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

CDD+19. Ignacio Cascudo, Ivan Damgård, Bernardo David, Nico Döttling, Rafael Dowsley, and Irene Giacomelli. Efficient UC commitment extension with homomorphism for free (and applications). In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 606–635. Springer, Heidelberg, December 2019.

CDG+17. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017.

CMS19. Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29. Springer, Heidelberg, December 2019.

dDOS19. Cyprien de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. BBQ: Using AES in picnic signatures. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 669–692. Springer, Heidelberg, August 2019.

DIO21. Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. In *2nd Conference on Information-Theoretic Cryptography (ITC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

DKR+22. Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 843–857. ACM Press, November 2022.

DKS12. Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 336–354. Springer, Heidelberg, April 2012.

dOT21. Cyprien de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge MPCitH-based arguments. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 3022–3036. ACM Press, November 2021.

DSL04. György Dósa, István Szalkai, and Claude Laflamme. The maximum and minimum number of circuits and bases of matroids. *Pure Mathematics and Applications*, 15(4):383–392, 2004. https://math.uni-pannon.hu/~szalkai/Szalkai-2006-DosaGy-PUMA.pdf.

FJR22a. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. Cryptology ePrint Archive, Report 2022/188, 2022. https://eprint.iacr.org/2022/188.

FJR22b. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Heidelberg, August 2022.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

GHS+21. Yaron Gvili, Julie Ha, Sarah Scheffler, Mayank Varia, Ziling Yang, and Xinyuan Zhang. TurboIKOS: Improved non-interactive zero knowledge and post-quantum signatures. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21, Part II*, volume 12727 of *LNCS*, pages 365–395. Springer, Heidelberg, June 2021.

GI08. Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Heidelberg, April 2008.

GMO16. Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

HBD+22. Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

IKOS08. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 433–442. ACM Press, May 2008.

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

KKW18. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

KZ22. Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Report 2022/588, 2022. https://eprint.iacr.org/2022/588.

LDK+22. Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Lin03. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, June 2003.

Lin17. Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer International Publishing, 2017.

OOS17. Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 381–396. Springer, Heidelberg, February 2017.

PFH+22. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Roy22. Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 657–687. Springer, Heidelberg, August 2022.

RRR16. Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 49–62. ACM Press, June 2016.

WYKW21. Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy*, pages 1074–1091. IEEE Computer Society Press, May 2021.

WYY+22. Chenkai Weng, Kang Yang, Zhaomin Yang, Xiang Xie, and Xiao Wang. AntMan: Interactive zero-knowledge proofs with sublinear communication. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2901–2914. ACM Press, November 2022.

XZS22. Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 299–328. Springer, Heidelberg, August 2022.

YSWW21. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.

## A Additional Preliminaries

### A.1 Zero-Knowledge Proofs of Knowledge

We recall that a witness relation for a language $L \in NP$ is a binary relation $R_L$ that is polynomially bounded, polynomial time recognizable and characterizes $L$ by $L = \{\mathbb{x} : \exists \mathbb{w} \text{ s.t. } (\mathbb{x}, \mathbb{w}) \in R_L\}$. We say that $\mathbb{w}$ is a witness for the statement $\mathbb{x} \in L$ if $(\mathbb{x}, \mathbb{w}) \in R_L$.

**Definition 16 (Correctness).** *A proof system* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *for* $\mathcal{R}$ *in the CRS+RO model has* correctness error $\gamma_{\mathsf{err}}$ *if for every (not necessarily efficient) adversary* $\mathcal{A}$

$$\Pr\left[\begin{matrix} crs \leftarrow \mathsf{Setup}^H(1^\lambda);\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}^H(1^\lambda, crs): \\ \langle \mathcal{P}^H(crs, \mathbb{x}, \mathbb{w}), \mathcal{V}^H(crs, \mathbb{x})\rangle = 1 \end{matrix}\right] \geq 1 - \gamma_{\mathsf{err}}(\lambda)$$

*We call* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ correct *if* $\gamma_{\mathsf{err}} = \mathsf{negl}(\lambda)$. *It is* perfectly correct *if* $\gamma_{\mathsf{err}} = 0$.

*Remark 9 (Sequential composition and proof systems in hybrid models).* In the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid models or $\mathcal{F}_{\mathsf{VOLE}}$-hybrid models, we rely on the *stand-alone real-ideal* notion of zero-knowledge proofs of knowledge (with static corruption). However, in the CRS+RO model (i.e. after our compilation), we define a game-based notion instead. The reason is that we want to *share* the setup (i.e. use a single CRS and RO for any number of protocol runs) and allow any party to participate. This requires a suitable definition of (non-interactive) zero-knowledge proof of knowledge. Since our compiled protocols are not UC-secure (because non-malleability is not ensured), it is simpler to use a game-based definition instead of an ideal functionality.

Our game-based definitions of zero-knowledge and proof of knowledge in the CRS+RO model below are made in such a way, that they compose sequentially. For zero-knowledge simulation, this is explicit in the definition, and for knowledge soundness, the adversary has the same (trapdoor) information as the extractor, hence composition is simple.

**Definition 17 (Zero-Knowledge).** *A simulator* $\mathcal{S}$ *for a proof system* $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *for* $\mathcal{R}$ *is a PPT algorithm with input a statement* $\mathbb{x}$ *for which* $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ *and implicit inputs* $1^\lambda$, $crs$, *and output a transcript* $\mathsf{tr}$. *Let* $\mathcal{A}$ *be a stateful algorithm and* $\hat{\mathcal{V}}$ *be an interactive deterministic algorithm. Let*

$$\mathsf{Real}_{\mathcal{A}}(\lambda) = \Pr\left[crs \leftarrow \mathsf{Setup}^H(1^\lambda);\ b \leftarrow \mathcal{A}^{H, \mathcal{O}_{\mathcal{P}}}(1^\lambda, crs):\ b = 1\right]$$
$$\mathsf{Ideal}_{\mathcal{A}}(\lambda) = \Pr\left[crs \leftarrow \mathsf{Setup}^H(1^\lambda);\ b \leftarrow \mathcal{A}^{H, \mathcal{O}_{\mathcal{S}}}(1^\lambda, crs):\ b = 1\right]$$

*The adversary has black-box access to the random oracle* $H$ *and to an oracle* $\mathcal{O}_{\mathcal{P}}$ *or* $\mathcal{O}_{\mathcal{S}}$, *which both take input* $(\mathbb{x}, \mathbb{w}, aux)$, *check if* $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ *(and output* $\perp$ *otherwise) and then*

- $\mathcal{O}_{\mathcal{P}}(\mathbb{x}, \mathbb{w}, aux)$ *outputs* $\mathsf{tr} \leftarrow \langle \mathcal{P}^H(crs, \mathbb{x}, \mathbb{w}), \hat{\mathcal{V}}^H(crs, \mathbb{x}, aux)\rangle$
- $\mathcal{O}_{\mathcal{S}}(\mathbb{x}, \mathbb{w}, aux)$ *outputs* $\mathsf{tr} \leftarrow \mathcal{S}^{H, \hat{\mathcal{V}}(crs, \mathbb{x}, aux)}(crs, \mathbb{x}, aux)$.

*The simulator* $\mathcal{S}$ *has black-box access to the next-message function for* $\hat{\mathcal{V}}(crs, \mathbb{x}, aux)$ *(and in particular, implements the random oracle for* $\hat{\mathcal{V}}$*). In the* programmable ROM, *the simulator* $\mathcal{S}$ *is allowed to* program *the random oracle* $H$ *on any fresh input to* $H$, *i.e. if* $H(m)$ *has not been queried by any party before, then* $\mathcal{A}$ *resp.* $\mathcal{S}$ *is free to choose* $H(m)$, *otherwise, programming fails.*

*We define the advantage of* $(\mathcal{A}, \hat{\mathcal{V}})$ *by* $\mathsf{AdvZK}^{\Pi,\mathcal{S}}_{\mathcal{A},\hat{\mathcal{V}}}(\lambda) = \mathsf{Real}_{\mathcal{A},\hat{\mathcal{V}}}(\lambda) - \mathsf{Ideal}_{\mathcal{A},\hat{\mathcal{V}}}(\lambda)$. *We call* $\mathcal{S}$ *(non-uniform)* zero-knowledge black-box *simulator for* $\Pi$ *if for any (non-uniform) PPT* $\mathcal{A}$ *and any DPT* $\hat{\mathcal{V}}$ *the advantage* $\mathsf{AdvZK}^{\Pi}_{\mathcal{A},\hat{\mathcal{V}}}$ *is negligible.*

*If* $\hat{\mathcal{V}}$ *is the honest verifier (and ignores aux), then* $\mathcal{S}$ *is called an* honest verifier zero-knowledge (HVZK) *simulator, if* $\mathcal{S}$ *is* straightline *and for every (non-uniform) PPT* $\mathcal{A}$*,* $\mathsf{AdvZK}^{\Pi}_{\mathcal{A},\mathcal{V}}$ *is negligible.*

*If similarly,* $\hat{\mathcal{V}}$ *interprets aux as the random tape for* $\mathcal{V}$ *and* $\Pi$ *is public-coin, then we call* $\mathcal{S}$ *a* special HVZK (SHVZK) *simulator if* $\mathcal{S}$ *is* straightline *and for every (non-uniform) PPT* $\mathcal{A}$*,* $\mathsf{AdvZK}^{\Pi}_{\mathcal{A},\hat{\mathcal{V}}}$ *is negligible..*

*Remark 10 (Abuse of notation).* Instead of pairs $(\mathcal{A}, \hat{\mathcal{V}})$ (resp. $(\mathcal{A}, \hat{\mathcal{P}})$), we often only speak of $\mathcal{A}$ (or $\hat{\mathcal{V}}$ resp. $\hat{\mathcal{P}}$) as the adversary. With universal machines for $\hat{\mathcal{V}}$ (resp. $\hat{\mathcal{P}}$), and their code as part of *aux*, this abuse of notation is justifiable. Similarly, we leave the trapdoor setups $\mathsf{TSetup}$ associated with extractors implicit.

We only define a specific form of knowledge soundness for *black-box* extraction in the *extractable* ROM.

**Definition 18 (Knowledge Soundness).** *Let* $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *be a public-coin proof system for* $\mathcal{R}$*. Let* $\mathsf{TSetup}$ *be a trapdoor PPT setup associated with extractor* $\mathcal{E}$*, which is a PPT algorithm with implicit inputs* $1^\lambda, crs, \mathsf{td}$*. Let* $\mathcal{A}$ *be a probabilistic algorithm and* $\hat{\mathcal{P}}$ *be a deterministic algorithm.*

$$\mathsf{Real}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c}(crs, \mathsf{td}) \leftarrow \mathsf{TSetup}^H(1^\lambda);\ (\mathbb{x}, aux) \leftarrow \mathcal{A}^H(1^\lambda, crs, \mathsf{td});\\ (out_{\hat{\mathcal{P}}}, b_{\mathcal{V}}) \leftarrow \langle \hat{\mathcal{P}}^H(\mathbb{x}, aux), \mathcal{V}^H(\mathbb{x})\rangle:\ b_{\mathcal{V}} = 1\end{array}\right]$$

$$\mathsf{Ideal}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c}(crs, \mathsf{td}) \leftarrow \mathsf{TSetup}^H(1^\lambda);\ (\mathbb{x}, aux) \leftarrow \mathcal{A}^H(1^\lambda, crs, \mathsf{td});\\ (out_{\hat{\mathcal{P}}}, b_{\mathcal{V}}, \mathbb{w}) \leftarrow \mathcal{E}^{H,\hat{\mathcal{P}}(\mathbb{x},aux)}:\ b_{\mathcal{V}} = 1 \wedge \mathcal{R}(\mathbb{x}, \mathbb{w}) = 1\end{array}\right]$$

*The extractor* $\mathcal{E}$ *has black-box access to the next-message function for* $\hat{\mathcal{P}}(crs, \mathbb{x}, aux)$*. Moreover, it is allowed to learn* all adversarial queries *made to* $H$*, i.e. those made by* $\mathcal{A}$ *and* $\hat{\mathcal{P}}$*.*

*W.l.o.g.* $\mathcal{E}$ *sets* $\mathbb{w} = \perp$ *if* $b_{\mathcal{V}} = 0$*. The advantage of* $(\mathcal{A}, \hat{\mathcal{P}})$ *is* $\mathsf{AdvKE}^{\Pi,\mathcal{E}}_{\mathcal{A},\hat{\mathcal{P}}}(\lambda) = \mathsf{Real}_{\mathcal{A}}(\lambda) - \mathsf{Ideal}_{\mathcal{A}}(\lambda)$*. We say* $\mathcal{E}$ *has* (asymptotic) knowledge error $\kappa_{\mathsf{err}} = \kappa_{\mathsf{err}}(\lambda)$ *if for any PPT pair* $(\mathcal{A}, \hat{\mathcal{P}})$*, there exists a negligible function* $\mathit{negl}(\lambda)$ *such that* $\mathsf{AdvKE}^{\Pi,\mathcal{E}}_{\mathcal{A},\hat{\mathcal{P}}} \le \kappa_{\mathsf{err}} + \mathit{negl}(\lambda)$*. We call* $\Pi$ knowledge sound *if an extractor with negligible knowledge error exists, and the associated trapdoored setup* $\mathsf{TSetup}$ *and real* $\mathsf{Setup}$ *are indistinguishable*[19]*.*

In practice, one wants not only knowledge soundness, but also the ability to continue the simulation, which is called witness-extended emulation [Lin03, GI08]. This can be achieved by introducing the requirements that the malicious prover's output $out_{\hat{\mathcal{P}}}$ in the real and ideal experiments are indistinguishable. Since all of our extractors are black-box and obtain $out_{\hat{\mathcal{P}}}$ by emulating an honest verifier, they trivially have perfect emulation, i.e. the distribution of $out_{\hat{\mathcal{P}}}$ is unchanged.

**Definition 19 (Straightline black-box simulation and extraction).** *A simulator (resp. extractor) for a proof system* $\Pi$ *is called* straightline*, if the next-message function is only used as in an actual interaction, i.e. if the malicious verifier (resp. prover) is never rewound.*

[19] Of course, omitting the $\mathsf{td}$ output of $\mathsf{TSetup}$, i.e., $\{crs \mid crs \leftarrow \mathsf{Setup}^H(1^\lambda)\} \overset{c}{\approx} \{crs \mid (crs, \mathsf{td}) \leftarrow \mathsf{TSetup}^H(1^\lambda)\}$.

In fact, our extractors satisfy the following even stronger notion, namely, they run the honest verifier completely unchanged. In other words, the extractor only needs the transcript and its trapdoor information; it does not require any (black-box) access to the malicious prover at all. We formalize this below.

**Definition 20 (Special extractor).** *Let $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a public-coin proof system for $\mathcal{R}$. Let* $\mathsf{TSetup}$ *be a trapdoor PPT setup associated with* special extractor $\mathsf{Ext}$*, which is a PPT algorithm with input a CRS trapdoor* $\mathsf{td}$*, a set $\mathcal{Q}$ of random-oracle query-response pairs, a statement $\mathbb{x}$, and a transcript* $\mathsf{tr}$*, and it outputs a witness $\mathbb{w}$ for $\mathbb{x}$ or $\perp$. The advantage of adversary $\mathcal{A}$ against knowledge soundness of special extractor* $\mathsf{Ext}$ *is*[20]

$$\Pr\left[\begin{array}{c} crs \leftarrow \mathsf{TSetup}^H(1^\lambda);\ (\mathbb{x}, aux) \leftarrow \mathcal{A}^H(1^\lambda, crs, \mathsf{td}); \\ (out_{\hat{\mathcal{P}}}, b_{\mathcal{V}}, \mathsf{tr}) \leftarrow \langle \hat{\mathcal{P}}^H(\mathbb{x}, aux), \mathcal{V}^H(\mathbb{x})\rangle; \\ \mathbb{w} \leftarrow \mathsf{Ext}(\mathsf{td}, \mathcal{Q}, \mathbb{x}, \mathsf{tr}):\ b_{\mathcal{V}} = 1 \wedge \mathcal{R}(\mathbb{x}, \mathbb{w}) = 0 \end{array}\right]$$

*where $\mathcal{Q}$ contains $\mathcal{A}$'s queries. In the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-hybrid model, the transcript contains the (malicious) prover's messages to the $\mathcal{F}^{N,\ell}_{\mathsf{OT}\text{-}\bar{1}}$-functionality.*

**Definition 21 (Non-interactive zero-knowledge (NIZK)).** *A proof system $\Pi$ where the prover sends only a single message, called* proof *and denoted by $\pi$, is called* non-interactive *(NI). We use the notation $\pi \leftarrow \Pi.\mathsf{Prove}^H_{crs}(\mathbb{x}, \mathbb{w})$ and $\Pi.\mathsf{Verify}^H_{crs}(\mathbb{x}, \pi)$ for computing and verifying the proof. If $\Pi$ is zero-knowledge, we call it a NIZK (proof system).*

For NI proof systems, we use a simplified notion of straightline extraction.

**Definition 22.** *Let $\Pi$ be a non-interactive proof system and $\mathcal{E}$ be a PPT algorithm with associated trapdoor setup* $\mathsf{TSetup}$*. Define the knowledge soundness experiment as follows:*

- $(crs, \mathsf{td}) \leftarrow \mathsf{TSetup}(1^\lambda)$
- $\mathcal{A}^{H, \mathsf{Verify}^H(crs,\cdot,\cdot)}(1^\lambda, crs, \mathsf{td})$
- *Return* 1 *if $\mathcal{A}$ made a call to* $\mathsf{Verify}$ *such that* $\mathsf{Verify}^H(crs, \mathbb{x}, \pi) = 1$ *call but $(\mathbb{x}, \mathcal{E}(\mathbb{x}, \pi, Q, \mathsf{td})) \notin \mathcal{R}$, where $Q$ contains all random oracle query-response pairs $(m, H(m))$ of $\mathcal{A}$. Else return* 0*.*

*The advantage $\mathsf{AdvKE}^{\Pi}_{\mathcal{A}}$ of $\mathcal{A}$ is defined as the probability that the experiment returns* 1*. If for every PPT $\mathcal{A}$ the advantage $\mathsf{AdvKE}^{\Pi,\mathcal{E}}_{\mathcal{A}}(\lambda)$ is negligible, and* $\mathsf{Setup}$ *and* $\mathsf{TSetup}$ *are indistinguishable, then we call $\Pi$ knowledge sound.*

### A.2 Commitments

With standard arguments, we can prove the following lemma.

**Lemma 5 (Selective all-but-one hiding implies adaptive all-but-one hiding).** *Consider real-or-random hiding security of a vector commitment* $\mathsf{VC}$ *for all-but-one sets, i.e. sets of the form $I = [N] \setminus \{i^*\}$. For any adversary $\mathcal{A}$ against all-but-one adaptive hiding, there is an adversary $\mathcal{B}$ against all-but-one selective hiding with roughly the same running time such that $\mathsf{AdvAdpHide}^{\mathsf{VC}}_{\mathcal{A}} \leq N \cdot \mathsf{AdvSelHide}^{\mathsf{VC}}_{\mathcal{B}}$.*

*Proof (Sketch).* The reduction $\mathcal{B}$ simply guesses $i^*$ and sends $I = [N] \setminus \{i^*\}$ to its (static) hiding challenger and receives $c$. Then $\mathcal{A}'$ runs $I' \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}, c)$. If the guess $i^*$ was wrong (i.e. $I \neq I'$), $\mathcal{A}'$ outputs a random bit. Otherwise it outputs whatever $\mathcal{A}$ eventually outputs.

[20] This coincides with the advantage in Definition 18 of the obvious (straightline) extractor $\mathsf{Ext}$ which can be constructed from $\mathsf{Ext}$.

### A.3 Extractable functions

We define when a function family is *extractable*. In a sense, this is a weakening of trapdoor functions where only in the security proof an invertibility trapdoor is needed. Hence, random oracles can be extractable functions.

**Definition 23.** *Let* $\mathsf{F} = (\mathsf{Setup}, \mathsf{Eval})$ *be a function family in the CRS+RO model. Let* $(\mathsf{TSetup}, \mathsf{Ext})$ *be pair of PPT algorithms such that*

– $\mathsf{TSetup}^H(1^\lambda) \to (crs, \mathsf{td})$*: Given security parameter* $\lambda$*, output a CRS* $crs$ *and trapdoor* $\mathsf{td}$*.*
– $\mathsf{Ext}(\mathsf{td}, Q, y) \to (m_i)_{i\in[N]}$*: Given the trapdoor* $\mathsf{td}$*, a set of query-response pairs of random oracle queries, and a purported image* $y$*, output a preimage* $x$*.*

*The extractability experiment (w.r.t.* $(\mathsf{TSetup}, \mathsf{Ext})$*) for stateful adversary* $\mathcal{A}$ *is defined as follows:*

1. $(crs, \mathsf{td}) \leftarrow \mathsf{TSetup}^H(1^\lambda)$
2. $y \leftarrow \mathcal{A}^H(1^\lambda, crs)$
3. $x' \leftarrow \mathsf{Ext}(\mathsf{td}, Q, y)$ *where* $Q$ *is the set* $\{(x_i, H(x_i))\}$ *of query-response pairs of queries* $\mathcal{A}$ *made to* $H$*.*
4. $x \leftarrow \mathcal{A}^H()$
5. *If* $x = x'$*, or* $\mathsf{Eval}_{crs}(x) \neq y$*, or* $x \notin \mathcal{X}_{crs}$*, output* 0*. Else output* 1 *(success).*

*The distinguishing advantage* $\mathsf{AdvDist}_{\mathcal{D}}^{(\mathsf{Setup},\mathsf{TSetup})}$ *for* $\mathsf{Setup}$ *and* $\mathsf{TSetup}$ *of an adversary* $\mathcal{D}$ *is defined as usual. The advantage* $\mathsf{AdvExt}_{\mathcal{A}}^{(\mathsf{TSetup},\mathsf{Ext})}$ *of an adversary* $\mathcal{A}$ *is defined by as probability to win the experiment. By abuse of notation, we write* $\mathsf{AdvExt}_{\mathcal{A}}^{\mathsf{F}}$ *if the algorithms are clear from the context. A function family is* (straightline) extractable *w.r.t.* $(\mathsf{TSetup}, \mathsf{Ext})$ *if for any PPT adversaries* $\mathcal{A}$ *and* $\mathcal{D}$*, their advantages* $\mathsf{AdvDist}_{\mathcal{D}}^{(\mathsf{Setup},\mathsf{TSetup})}$ *and* $\mathsf{AdvExt}_{\mathcal{A}}^{(\mathsf{TSetup},\mathsf{Ext})}$ *are negligible.*

Our notion of extractability is tailored to our setting and by definition straightline. We allow extraction to fail if $\mathcal{A}$ cannot produce a preimage, while requiring the exact same preimage if $\mathcal{A}$ has a preimage (implying a form collision resistance).

*Example 1 (Straightline extractable functions).*

1. Random oracles are extractable functions for any superpolynomial codomain $\mathcal{Y}$. Indeed, any $q$-query adversary has advantage at most $(q+1)^2/|\mathcal{Y}|$.
2. Any (injective) trapdoor one-way function $\mathsf{TDF}$ is straightline extractable. For this, let $\mathsf{TSetup}(1^\lambda)$ run the key generation algorithm for $\mathsf{TDF}$, and define $crs$ as the function key of $\mathsf{TDF}$ and $\mathsf{td}$ as the invertibility trapdoor.

## B Details on the Instantiations

### B.1 ZK from Generalized sVOLE for Arbitrary Degree-d Relations

We can easily generalize the method described in Section 6 for degree-2 relations to prove arbitrary degree-$d$ polynomials. Let $(\mathbf{w}_1, \ldots, \mathbf{w}_\ell) \in \mathbb{F}_p^{k_C \cdot \ell}$ be a witness and $f_i \in \mathbb{F}_p[X_1, \ldots, X_\ell]_{\leq d}$, $i \in [t]$, be the set of polynomials over $\mathbb{F}_p$ with degree at most $d$, we want to prove that $f_i(\mathbf{w}_1, \ldots \mathbf{w}_\ell) = 0$, for each $i \in [t]$.

We can proceed similarly to [YSWW21]. First, $\mathcal{P}$ and $\mathcal{V}$ call the sVOLE functionality with parameters $p, \mathcal{C}, 2(\ell+d)$, so that $\mathcal{P}$ obtains matrices $\mathbf{U} \in \mathbb{F}_p^{2(\ell+d)\times k_{\mathcal{C}}}$, $\mathbf{V} \in \mathbb{F}_p^{2(\ell+d)\times n_{\mathcal{C}}}$. As before, we can split these matrices as $\mathbf{U} = \begin{pmatrix} \mathbf{U}_1 \\ \mathbf{R} \end{pmatrix}$ and $\mathbf{V} = \begin{pmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \end{pmatrix}$, where each sub-matrix consists of $\ell + d - 1$ rows. The prover uses the first $\ell$ rows of $\mathbf{U}$ to commit to its witness, as described in Figure 6. Then, $\mathcal{P}$ opens $\mathbf{S} = \mathbf{R} + \mathbf{W} \cdot \Delta'$, where $\Delta' \leftarrow \mathbb{F}_p$ is the challenge sent by $\mathcal{V}$. The following relation holds:

$$\begin{aligned} g_i(Y) &= \sum_{h\in[0,d]} f_{i,h}(\mathbf{r}_1 + \mathbf{w}_1 \cdot Y, \ldots, \mathbf{r}_\ell + \mathbf{w}_\ell \cdot Y) \cdot Y^{d-h} \\ &= \sum_{h\in[0,d]} f_{i,h}(\mathbf{w}_1, \ldots, \mathbf{w}_\ell) \cdot Y^d + \sum_{h\in[0,d-1]} A_{i,h} \cdot Y^h \\ &= f_i(\mathbf{w}_1, \ldots, \mathbf{w}_\ell) \cdot Y^d + \sum_{h\in[0,d-1]} A_{i,h} \cdot Y^h, \end{aligned}$$

where $A_{i,h} \in \mathbb{F}_p^{k_{\mathcal{C}}}$ is the aggregated coefficient of $Y^h$.

The key observation is that, if the prover $\mathcal{P}$ is honest, then $f_i(\mathbf{w}_1, \ldots, \mathbf{w}_n) = 0$ and $g_i(Y) = \sum_{h\in[0,d-1]} A_{i,h} \cdot Y^h$. In order to send the aggregated values $\sum_{i\in[t]} A_{i,h}$, $h \in [0, d-1]$, as in $\Pi^t_{\mathsf{2D-LC}}$, $\mathcal{P}$ needs $d$ extra independent masks $\mathbf{a}_h$. These are computed using the matrices $\mathbf{U}_{1,[\ell+1..\ell+d-1]}$ and $\mathbf{R}_{[\ell+1..\ell+d-1]}$ as follows:

- $\mathcal{P}$ sets $p_1(Y) = \mathbf{r}_{\ell+1} + \mathbf{u}_{\ell+1} \cdot Y$ and iteratively computes $p_{i+1}(Y) = p_i(Y) \cdot (\mathbf{r}_{\ell+i+1} + \mathbf{u}_{\ell+i+1} \cdot Y)$, for each $i \in [2, d-2]$ and the coefficients $\mathbf{a}_h$, $h \in [d-1]$, such that $p_{d-1}(Y) = \sum_{[0,d-1]} \mathbf{a}_h \cdot Y^h$
- $\mathcal{V}$ locally computes $\mathbf{b}_1 = \mathbf{q}_{\ell+1}$ and $\mathbf{b}_{i+1} = \mathbf{b}_i \cdot \mathbf{q}_{i+1}$, $i \in [2, d-2]$. So that $\sum_{h\in[0,d-1]} \mathbf{a}_h \cdot (\Delta')^h = \mathbf{b}_{d-1}$.

After the first commitment to the witness, the proof proceeds with the prover sending values $\widetilde{\mathbf{a}}_h = \sum_{i\in[t]} \chi_i \cdot A_{i,h} + \mathbf{a}_h$, $h \in [0, d-1]$, where $\chi_i \in \mathbb{F}_p$ are the challenges sent by the verifier. After the opening of $\mathbf{S}$, the proof proceeds similarly to $\Pi^t_{\mathsf{2D-LC}}$. In particular, $\mathcal{V}$ first computes $\mathbf{c}_i(\Delta') = \sum_{[0,d]} f_{i,h}(\mathbf{s}'_1, \ldots, \mathbf{s}'_\ell) \cdot (\Delta')^{d-h}$, where $\mathbf{s}'_i$ are defined as in $\Pi^t_{\mathsf{2D-LC}}$, then it checks that

$$\sum_{h\in[0,d-1]} \chi_i \cdot \mathbf{c}_i + \mathbf{b}_{d-1} = \sum_{h\in[0,d-1]} \widetilde{\mathbf{a}}_h \cdot (\Delta')^h$$

and finally performs the consistency check.