# SoftSpokenOT: Communication–Computation Tradeoffs in OT Extension

Lawrence Roy*

February 17, 2022

**Abstract**

Given a small number of base oblivious transfers (OTs), how does one generate a large number of extended OTs as efficiently as possible? The answer has long been the seminal work of IKNP (Ishai et al., Crypto 2003) and the family of protocols it inspired, which only use Minicrypt assumptions. Recently, Boyle et al. (Crypto 2019) proposed the Silent-OT technique that improves on IKNP, but at the cost of a much stronger, non-Minicrypt assumption: the learning parity with noise (LPN) assumption. We present SoftSpokenOT, the first OT extension to improve on IKNP's communication cost in the Minicrypt model. While IKNP requires security parameter $\lambda$ bits of communication for each OT, SoftSpokenOT only needs $\lambda/k$ bits, for any $k$, at the expense of requiring $2^{k-1}/k$ times the computation. For small values of $k$, this tradeoff is favorable since IKNP-style protocols are network-bound. We implemented SoftSpokenOT and found that our protocol gives almost a 5× speedup over IKNP in the LAN setting.

Our technique is based on a novel silent protocol for vector oblivious linear evaluation (VOLE) over polynomial-sized fields. We created a framework to build maliciously secure $\binom{N}{1}$-OT extension from this VOLE, revisiting the existing work for each step. Along the way, we found several flaws in the existing work, including a practical attack against the consistency check of Patra et al. (NDSS 2017), while also making some improvements.

## 1 Introduction

Oblivious transfer (OT) is a basic building block of multi-party computation (MPC), and for many realistic problems, MPC protocols may require millions of OTs. [Bea96] introduced the concept of OT extension, where a small number of OTs called *base OTs* are processed to efficiently generate a much larger number of *extended OTs*. [IKNP03] (hereafter, IKNP) was the first OT extension protocol to make black-box use of its primitives, a significant improvement in efficiency. Because of its speed, it is still widely used for semi-honest OT extension.

However, IKNP has a bottleneck: communication. It transfers $\lambda$ bits for every extended random OT. Recent works under the heading of Silent OT [BCGI18, BCG+19b, BCG+19a, YWL+20, CRR21] have communication complexity that grows only *logarithmically* in the number of oblivious transfers. Consequently, they are favored when communication is slow. On the other hand, IKNP has the advantage for computational cost: of the Silent OT protocols, only Silver [CRR21] uses a comparable amount of computation to IKNP. Additionally, while IKNP uses only Minicrypt [Imp95] assumptions (i.e. the assumptions are all provable in the random oracle model), Silent OT is based on the learning parity with noise (LPN) problem, which is not Minicrypt. Efficient instantiations depend on highly

[*] Oregon State University. Address: `ldr709@gmail.com`. Supported by a DoE CSGF Fellowship.

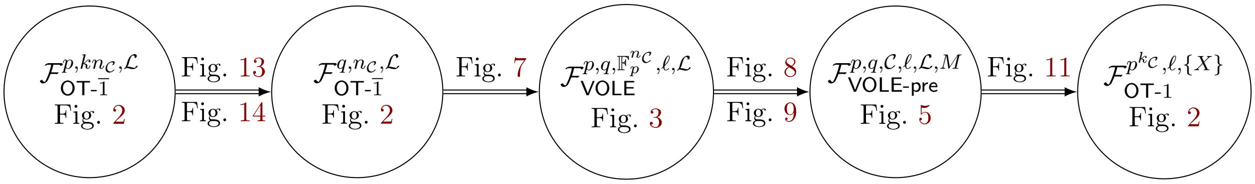

$\mathcal{F}^{p,kn_\mathcal{C},\mathcal{L}}_{\text{OT-}\bar{1}}$
Fig. 2
Fig. 13
Fig. 14
$\mathcal{F}^{q,n_\mathcal{C},\mathcal{L}}_{\text{OT-}\bar{1}}$
Fig. 2
Fig. 7
$\mathcal{F}^{p,q,\mathbb{F}_p^{n_\mathcal{C}},\ell,\mathcal{L}}_{\text{VOLE}}$
Fig. 3
Fig. 8
Fig. 9
$\mathcal{F}^{p,q,\mathcal{C},\ell,\mathcal{L},M}_{\text{VOLE-pre}}$
Fig. 5
Fig. 11
$\mathcal{F}^{p^{k_\mathcal{C}},\ell,\{X\}}_{\text{OT-1}}$
Fig. 2

Figure 1: Sequence of ideal functionalities and protocols used for OT extension. Here $q = p^k$ is the size of the small field VOLE, and $\mathcal{L} = \text{Affine}(\mathbb{F}_p^{kn_\mathcal{C}})$ is the set of allowed selective abort attacks against the base OT receiver. Protocols below the arrows are consistency checks needed for maliciously security.

structured versions of this problem, with the most efficient protocol, Silver, owing its efficiency to a novel variant of LPN that was introduced solely for that work. Compared with a tried-and-true block cipher like AES, these assumptions are too recent to have received as much cryptanalysis.

Improvements to IKNP also benefit a number of derived protocols. For maliciously secure OT extension, the main approach [KOS15] (hereafter, KOS) is to combine IKNP with a consistency check, although Silent OT can also achieve malicious security. [KK13] achieved $\binom{N}{1}$-OT extension by noticing that part of IKNP can be viewed as encoding the OT choice bits with a repetition code. They replaced it with a more sophisticated error correcting code. [OOS17] (hereafter, OOS) and [PSS17] (hereafter, PSS) then devised more general consistency checking protocols to achieve maliciously secure $\binom{N}{1}$-OT extension.

### 1.1 Our Results

Our technique, SoftSpokenOT, makes an asymptotic improvement over IKNP's communication cost. It is the first OT extension to do so in the Minicrypt model. For any parameter $k \geq 1$, SoftSpokenOT can implement $\binom{2}{1}$-OT maliciously secure extension using only $\lambda/k$ bits, compared to IKNP's $\lambda$ bits. This is a communication–computation tradeoff, as the sender in our protocol must *generate* $\lambda \cdot 2^k/k$ pseudorandom bits, while IKNP only needs to generate $2\lambda$ bits. In practice, fast hardware implementations of AES make IKNP network bound, so when $k$ is small (e.g. $k = 5$) this extra computation will have no effect on the overall protocol latency. And for $k = 2$, no extra computation is required, making it a pure improvement over IKNP. Asymptotically, setting $k = \Theta(\log(\ell))$ generates $\ell$ OTs with sublinear communication $\Theta\big(\frac{\lambda \cdot \ell}{\log(\ell)}\big)$, in polynomial time.

We present a sequence of protocols (Fig. 1), starting with base OTs, continuing through vector oblivious linear evaluation (VOLE), and ending at OT extension. First, we present a novel silent protocol for VOLE over polynomial-sized fields, which may be of independent interest. A VOLE generates correlated randomness $(\vec{u}, \vec{v})$ and $(\Delta, \vec{w})$ where $\vec{w} - \vec{v} = \vec{u}\Delta$. Our next stepping stone is an ideal functionality that we call subspace VOLE, which produces correlations satisfying $W - V = UG_\mathcal{C}\,\text{diag}(\vec{\Delta})$. Here, $G_\mathcal{C}$ is the generator matrix for a linear code $\mathcal{C}$. Note that $\Delta$-OT (a.k.a. correlated OT) is a special case of subspace VOLE, as is the correlation used by PaXoS [PRTY20]. Our $\Delta$-OT works over any field of polynomial size, so it can encode the inputs for arithmetic garbling [BMR16]. Finally, we hash the subspace VOLE using a correlation robust (CR) hash to build random $\binom{N}{1}$, a correlation $(x, m_x)$ and $(m_0, \dots, m_{N-1})$ where the $m_y$ are all random. These may used directly, or to encode lookup tables representing multiple small-secret $\binom{2}{1}$-OTs [KK13].

We generalize OOS to construct a consistency checking protocol that achieves maliciously secure subspace VOLE, albeit with a selective abort attack. However, while proving our protocol secure, we found flaws (Sect. 4.1) in the major existing works on consistency checks for OT extension. This is minor for OOS — just a flaw in their proof — and a special case of our new proof shows that OOS is still secure. We found two attacks on KOS which show that it is not always as secure as claimed, though it's still secure enough in practice. We leave to future research the problem of finding a sound proof of security for KOS. However, PSS's flaw is more severe, as we found a practical attack that can break their $\binom{256}{1}$-OT extension at $\lambda = 128$ security in time $2^{34}$ with probability $2^{-8}$.

The final step, going from correlated randomness (i.e. subspace VOLE) to extended OTs, requires a CR hash function. For malicious security, a mechanism is needed to stop the receiver from causing a collision between CR hash inputs. [GKWY20] solve this with a tweakable CR (TCR) hash, using a tweak to stop these collisions. TCR hashes are more expensive than plain CR hashes, so Endemic OT [MR19] instead prevent the receiver from controlling the base OTs, proving that it is secure to forgo tweaks in this case. However, their proof assumes stronger properties of the consistency checking protocol than are provided by real consistency checks, allowing us to find an attack on their OT extension (see Sect. 5). We follow [CT21] in using a universal hash to prevent collisions, only using the tweak to improve the security of the TCR hash. We optimize their technique by sending the universal hash in parallel with the consistency check — our new proof shows that the receiver has few remaining choices once it learns the universal hash.

We implemented SoftSpokenOT for $\binom{2}{1}$-OT in the libOTe [Rin] library. When tested with a 1Gbps bandwidth limit, our protocol has almost a 5× speedup over IKNP with $k = 5$, resulting from a 5× reduction in communication. The only case where SoftSpokenOT was suboptimal among the tested configurations was in the WAN setting, where it took second place to Silver. However, the assumptions needed by SoftSpokenOT are much more conservative than those used by Silver.

## 1.2 Technical Overview

SoftSpokenOT is a generalization of the classic oblivious transfer extension of IKNP, which at its core is based on what can be viewed as a protocol for $\mathbb{F}_2$-VOLE. This VOLE protocol starts by using a PRG to extend $\binom{2}{1}$-OT to message size $\ell$. The base OT sender, $P_S$, gets random strings $\vec{m}_0, \vec{m}_1$ and the receiver, $P_R$, gets its choice bit $b \in \mathbb{F}_2$ and its chosen message $\vec{m}_b$. $P_S$ then computes $\vec{u} = \vec{m}_0 \oplus \vec{m}_1$ and $\vec{v} = \vec{m}_1 = 0\,\vec{m}_0 \oplus 1\,\vec{m}_1$, while $P_R$ computes $\Delta = 1 \oplus b$, and $w = \vec{m}_b = \Delta \vec{m}_0 \oplus (1 \oplus \Delta)\vec{m}_1$.[1] Then $\vec{w} \oplus \vec{v} = \Delta \vec{m}_0 \oplus \Delta \vec{m}_1 = \Delta \vec{u}$, which is a VOLE correlation: $P_S$ gets a vector $\vec{u} \in \mathbb{F}_2^{\ell}$ and $P_R$ gets a scalar $\Delta \in \mathbb{F}_2$, and they learn secret shares $\vec{v}, \vec{w}$ of the product. While $\vec{u}$ was chosen by the protocol, it possible to derandomize $\vec{u}$ to be any chosen vector. If $P_S$ wants to use $\vec{u}'$ instead, it can send $\bar{u} = \vec{u} \oplus \vec{u}'$ to $P_R$, who updates its share to be $\vec{w}' = \vec{w} \oplus \Delta \bar{u}$. This preserves the VOLE correlation, $\vec{w}' \oplus \vec{v} = \Delta \vec{u} \oplus \Delta \bar{u} = \Delta \vec{u}'$, while hiding $\vec{u}'$.

The next step of the IKNP protocol is to stack $\lambda$ of these $\mathbb{F}_2$-VOLEs side by side, while sending $\lambda \cdot \ell$ bits to derandomize the $\vec{u}$ vectors to all be the same. That is, for the $i$th VOLE, they get a correlation $W_{\cdot i} \oplus V_{\cdot i} = \Delta_i \vec{u}$, where $V_{\cdot i}$ means the $i$th column of a matrix $V$. In matrix notation, this is an outer product: $W \oplus V = \vec{u}\overleftarrow{\Delta}$, where $\overleftarrow{\Delta}$ is the row vector of all the $\Delta_i$. Then looking at the $j$th row gives $W_{j\cdot} \oplus V_{j\cdot} = u_j \overleftarrow{\Delta}$, which make $u_j$ the choice bit of a $\Delta$-OT. That is, $P_R$ has learned $\overleftarrow{m}_{j0} = W_{j\cdot}$ and $\overleftarrow{m}_{j1} = W_{j\cdot} \oplus \overleftarrow{\Delta}$, while $P_S$ has its choice bit $u_j$ and $\overleftarrow{m}_{u_j} = V_{j\cdot}$, the corresponding message. Notice that this is a correlated OT, but now the OT sender is $P_R$ and the OT receiver is $P_S$ — they have been reversed from what they were for the base OTs. Hashing the $\overleftarrow{m}_{jx}$ then turns them into uncorrelated OT messages.

SoftSpokenOT instead bases the OT extension on a $\mathbb{F}_{2^k}$-VOLE, where $\vec{u}$ is restricted to taking values in $\mathbb{F}_2$. We now only need $\lambda/k$ of these VOLEs to get the $\lambda$ bits per OT needed to make the hash secure. Derandomizing $\vec{u}$ for each OT then only needs $\lambda/k$ *bits* per OT, as for each VOLE the

[1] Note that this is backwards from the usual description of IKNP — it's more usual to set $\Delta$ to be the $b$, the index of the message known to $P_R$. A key insight in SoftSpokenOT is that the unknown base OT message is the most important.

elements of $\vec{u}$ are in $\mathbb{F}_2$, reducing a major bottleneck of IKNP. Instead of $\binom{2}{1}$-OT, our $\mathbb{F}_{2^k}$-VOLE is based on $\binom{2^k}{2^k-1}$-OT, which can be instantiated using a well known protocol [BGI17] based on a punctured PRF; see Sect. 6 for details.

In $\binom{2^k}{2^k-1}$-OT a random function $F\colon \mathbb{F}_{2^k} \to \mathbb{F}_2^\ell$ is known to $P_S$, while $P_R$ has a random point $\Delta$ and the restriction $F^*$ of $F$ to $\mathbb{F}_{2^k} \setminus \{\Delta\}$. The earlier equations for the vectors $\vec{u}$, $\vec{v}$, and $\vec{w}$ were chosen to be suggestive of their generalizations:

$$\begin{aligned}
\vec{u} &= F(0) \oplus F(1) & &\implies \vec{u} = \bigoplus_{x \in \mathbb{F}_{2^k}} F(x) \\
\vec{v} &= 0\,F(0) \oplus 1\,F(1) & &\implies \vec{v} = \bigoplus_{x \in \mathbb{F}_{2^k}} xF(x) \\
\vec{w} &= \Delta F^*(0) \oplus (1 \oplus \Delta)F^*(1) & &\implies \vec{w} = \bigoplus_{x \in \mathbb{F}_{2^k}} (x \oplus \Delta)F^*(x).
\end{aligned}$$

Notice that the formula for $\vec{w}$ multiplies $F^*(\Delta)$ by 0, which is good because $F(\Delta)$ is unknown to $P_R$. Therefore, $\vec{w} \oplus \vec{v} = \bigoplus_x \Delta F(x) = \Delta \vec{u}$.

Reducing communication by a factor of $k$ comes at the expense of increasing computation by a factor of $2^k/k$. While there are now only $\lambda/k$ VOLES, they each require both parties to evaluate $F$ at every point (except the one that $P_R$ does not know) in a finite field of size $2^k$.

# 2 Preliminaries

## 2.1 Notation

We start counting at zero, and the set $[N]$ is $\{0, 1, \ldots, N-1\}$. The finite field with $p$ elements is written as $\mathbb{F}_p$, the vector space of dimension $n$ as $\mathbb{F}_p^n$, and set of all $m \times n$ matrices as $\mathbb{F}_p^{m \times n}$. The vectors themselves are written with an arrow, as $\vec{x}$, while matrices are capital letters $M$. Row vectors are written with a backwards arrow instead: $\overleftarrow{x}$. The componentwise product of vectors is $\vec{x} \odot \vec{y} = \begin{bmatrix} x_0 y_0 & \cdots & x_{n-1} y_{n-1} \end{bmatrix}^\top$. Diagonal matrices are notated $\operatorname{diag}(\vec{x}) = \begin{bmatrix} x_0 & & \\ & \ddots & \\ & & x_{n-1} \end{bmatrix}$, which makes $\vec{x} \odot \vec{y} = \operatorname{diag}(\vec{x})\vec{y}$. The $i$th row of a matrix $M$ is $M_{i\cdot}$, while the $j$th column is $M_{\cdot j}$. The first $r$ rows of $M$ are $M_{[r]\cdot}$, and the first $c$ columns are $M_{\cdot[c]}$.

There are two finite fields we will usually work with: the subfield $\mathbb{F}_p$, and its extension field $\mathbb{F}_q$, where $q = p^k$. Usually $p$ will be prime, but that is not necessary. In a few places we will equivocate between $\mathbb{F}_q$, $\mathbb{F}_p^k$, and $[q]$, using the obvious bijections between them.

**Linear Codes.** Let $\mathcal{C}$ be a $[n_\mathcal{C}, k_\mathcal{C}, d_\mathcal{C}]$ linear code, that is, $\mathcal{C}$ is a $k_\mathcal{C}$-dimensional subspace of $\mathbb{F}_p^{n_\mathcal{C}}$ with minimum distance $d_\mathcal{C} = \min_{\overleftarrow{y} \in \mathcal{C} \setminus \{0\}} \|\overleftarrow{y}\|_0$, where $\|\overleftarrow{y}\|_0$ is the Hamming weight of $\overleftarrow{y}$. For a matrix $A$, we similarly let the Hamming weight $\|A\|_0$ be the number of nonzero columns of $A$. Let $G_\mathcal{C} \in \mathbb{F}_p^{k_\mathcal{C} \times n_\mathcal{C}}$ be the generator matrix of $\mathcal{C}$. We follow the convention that the messages and code words are row vectors, so a row vector $\overleftarrow{x}$ encodes to the codeword $\overleftarrow{x} G_\mathcal{C} \in \mathcal{C}$. The rows of the generator matrix must form a basis of $\mathcal{C}$, which can be completed into a basis $T_\mathcal{C}$ of $\mathbb{F}_p^{n_\mathcal{C}}$; that is, the first $k_\mathcal{C}$ rows of $T_\mathcal{C}$ are $G_\mathcal{C}$. Then $T_\mathcal{C}$ has an inverse $T_\mathcal{C}^{-1}$, the last $n_\mathcal{C} - k_\mathcal{C}$ columns of which form a parity check matrix for $\mathcal{C}$.

There are two specific codes that come up most frequently. There is the trivial code, $\mathbb{F}_p^n$, where all vectors are code words and $G_{\mathbb{F}_p^n} = T_{\mathbb{F}_p^n} = \mathbb{1}_n$. There is also the repetition code, $\mathsf{Rep}(\mathbb{F}_p^n)$, which consists of all vectors where all elements are the same. Its generator matrix is $G_{\mathsf{Rep}(\mathbb{F}_p^n)} = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}$.

**Algorithms.** We use pseudocode for our constructions. In many cases there will be two similar algorithms side by side (e.g. sender and receiver, or real and ideal), and we use whitespace to align matching lines. Sampling a value $x$ uniformly at random in a set $X$ is written as $x \xleftarrow{\$} X$.

## 2.2 Universal Hashes

We make extensive use of universal hashes [CW79], essentially as a more efficient replacement for a uniformly random matrix. We depend on the extra structure of the hash function being linear, so we give definitions specialized to that case.

**Definition 2.1.** *A family of matrices $\mathcal{R} \subseteq \mathbb{F}_q^{m \times n}$ is a* linear $\epsilon$-almost universal family *if, for all nonzero $\vec{x} \in \mathbb{F}_q^n$, $\Pr_{R \xleftarrow{\$} \mathcal{R}}\left[R\vec{x} = 0\right] \leq \epsilon$.*

**Definition 2.2.** *A family of matrices $\mathcal{R} \subseteq \mathbb{F}_q^{m \times n}$ is* linear $\epsilon$-almost uniform family *if, for all nonzero $\vec{x} \in \mathbb{F}_q^n$ and all $\vec{y} \in \mathbb{F}_q^m$, $\Pr_{R \xleftarrow{\$} \mathcal{R}}\left[R\vec{x} = \vec{y}\right] \leq \epsilon$.*

For characteristic 2, this is equivalent to being $\epsilon$-almost XOR-universal. Clearly, a family that is $\epsilon$-almost uniform is also $\epsilon$-almost universal. In Appx. F.1, we prove two composition properties of universal hashes.

**Proposition 2.3.** *Let $\mathcal{R}$ and $\mathcal{R}'$ be $\epsilon$ and $\epsilon'$-almost universal families, respectively. Then $R'R$ for $R \in \mathcal{R}, R' \in \mathcal{R}'$ is a $(\epsilon + \epsilon')$-universal family.*

**Proposition 2.4.** *Let $\mathcal{R}$ and $\mathcal{R}'$ be $\epsilon$-almost uniform families. Then $[R\ R']$ for $R \in \mathcal{R}, R' \in \mathcal{R}'$ is a $\epsilon$-uniform family.*

## 2.3 Ideal Functionalities

The protocols in this paper are analyzed in the Simplified UC model of [CCL15], so whenever an ideal functionality takes inputs or outputs, the adversary is implicitly notified and allowed to delay or block delivery of the message. The functionalities deal with three entities: the sender $P_S$, the receiver $P_R$, and the adversary $\mathcal{A}$. Instead of the usual event-driven style (essentially a state machine driven by the messages), we use a blocking call syntax for our ideal functionalities, where it stops and waits to receive a message. While we will not need to receive multiple messages at once, it would be consistent to use multiple parallel threads of execution, with syntax like $\boxed{\text{recv. } x \text{ from } P_S \,\Big\|\, \text{recv. } y \text{ from } P_R}$. We omit the "operation labels" identifying the messages, instead relying on the variable names and message order to show which send corresponds to each receive. We assume the protocol messages themselves are delivered over an authenticated channel.

All of our functionalities are for different kinds of random input VOLE or OT, meaning that the protocol pseudorandomly chooses the inputs of each party. Essentially, the functionalities just generate correlated randomness. Using random VOLE or OT, the parties can still choose their inputs using derandomization, if necessary.[2] However, we cannot guarantee that a corrupted participant does not exercise partial control over the outputs of the protocols. For this reason, we use the endemic security notion of [MR19], where any corrupted participants get to choose their protocol outputs, then the remaining honest parties receive random outputs, subject to the correlation. One difference, however, is that in our ideal functionalities an honest OT receiver doesn't get to choose its choice bits. Instead, all protocol inputs are random for honest parties.[3]

[2] See [MR19] for details on derandomizing OT messages.

[3] This is similar to the pseudorandom correlation generators (PCGs) used in [BCG+19b] to build Silent OT. In fact, the small field VOLE constructed in Sect. 3.1 can be viewed as a PCG.

```
$\mathcal{F}^{N,\ell,\mathcal{L}}_{\mathsf{OT\text{-}1}}$
for $i \in [\ell]$:
  if $P_S$ is corrupted:
    recv. $F_i \in \big(\{0,1\}^\lambda\big)^{[N]}$ from $\mathcal{A}$
  else:
    $F_i \xleftarrow{\$} \big(\{0,1\}^\lambda\big)^{[N]}$
  if $P_R$ is corrupted:
    recv. $x_i^* \in [N]$ from $\mathcal{A}$
    recv. $F_i^* \in \{0,1\}^\lambda$ from $\mathcal{A}$
    $F_i(x_i^*) := F_i^*$
  else:
    $x_i^* \xleftarrow{\$} [N]$
    $F_i^* := F_i(x_i^*)$
send $\{F_i\}_{i\in[\ell]}$ to $P_S$
$\mathsf{Send/Abort}\big(\{x_i^*\}_{i\in[\ell]}, \{F_i^*\}_{i\in[\ell]}, \mathcal{L}\big)$
```

```
$\mathcal{F}^{N,\ell,\mathcal{L}}_{\mathsf{OT\text{-}\bar{1}}}$
for $i \in [\ell]$:
  if $P_S$ is corrupted:
    recv. $F_i \in \big(\{0,1\}^\lambda\big)^{[N]}$ from $\mathcal{A}$
  else:
    $F_i \xleftarrow{\$} \big(\{0,1\}^\lambda\big)^{[N]}$
  if $P_R$ is corrupted:
    recv. $x_i^* \in [N]$ from $\mathcal{A}$
    recv. $F_i^* \in \big(\{0,1\}^\lambda\big)^{[N]\setminus\{x_i^*\}}$ from $\mathcal{A}$
    $F_i(x) := F_i^*(x), \forall x \in [N] \setminus \{x_i^*\}$
  else:
    $x_i^* \xleftarrow{\$} [N]$
    $F_i^*(x) := F_i(x), \forall x \in [N] \setminus \{x_i^*\}$
send $\{F_i\}_{i\in[\ell]}$ to $P_S$
$\mathsf{Send/Abort}\big(\{x_i^*\}_{i\in[\ell]}, \{F_i^*\}_{i\in[\ell]}, \mathcal{L}\big)$
```

Figure 2: Ideal functionalities for a batch of $\ell$ endemic OTs, with $\binom{N}{1}$-OT on the left and $\binom{N}{N-1}$-OT on the right. Differences are highlighted .

```
$\mathcal{F}^{p,q,\mathcal{C},\ell,\mathcal{L}}_{\mathsf{VOLE}}$
if $P_S$ is corrupted:
  recv. $U \in \mathbb{F}_p^{\ell\times k_\mathcal{C}}, V \in \mathbb{F}_q^{\ell\times n_\mathcal{C}}$ from $\mathcal{A}$
else:
  $U \xleftarrow{\$} \mathbb{F}_p^{\ell\times k_\mathcal{C}}, V \xleftarrow{\$} \mathbb{F}_q^{\ell\times n_\mathcal{C}}$
if $P_R$ is corrupted:
  recv. $\overleftarrow{\Delta} \in \mathbb{F}_q^{n_\mathcal{C}}, W \in \mathbb{F}_q^{\ell\times n_\mathcal{C}}$ from $\mathcal{A}$
  $V := -UG_\mathcal{C}\,\mathrm{diag}(\overleftarrow{\Delta}) + W$
else:
  $\overleftarrow{\Delta} \xleftarrow{\$} \mathbb{F}_q^{n_\mathcal{C}}$
  $W := UG_\mathcal{C}\,\mathrm{diag}(\overleftarrow{\Delta}) + V$
send $U, V$ to $P_S$
$\mathsf{Send/Abort}(\overleftarrow{\Delta}, W, \mathcal{L})$
```

Figure 3: Ideal functionality for endemic subspace VOLE. $\mathcal{C}$ is a linear code.

```
$\mathsf{Send/Abort}(x \in X, y \in Y, \mathcal{L} \subseteq 2^X)$:
  if $P_S$ is malicious:
    recv. $L \in \mathcal{L}$ from $P_S$
    if $x \notin L$:
      send "check failed" to $P_R$
      abort
  send $x, y$ to $P_R$
```

Figure 4: Output with leakage function. Sends $x, y$ to $P_R$, after allowing $P_S$ to do a selective abort attack on $x$.

The ideal functionalities for length $\ell$ batches of $\binom{N}{1}$-OTs or $\binom{N}{N-1}$-OTs are presented in Fig. 2. In each OT, the sender $P_S$ gets a random function $F\colon [N] \to \{\mathsf{0},\mathsf{1}\}^\lambda$, which is chosen by the adversary if $P_S$ is corrupted. If $N$ is exponentially large, $F$ should be thought of as an oracle, which will only be evaluated on a subset of $[N]$. The receiver $P_R$ gets a choice element $x^* \in [N]$, as well as $F^*$, which is either the one point $F(x^*)$ for $\binom{N}{1}$-OT, or the restriction of $F$ to every other point $[N] \setminus x^*$ for $\binom{N}{N-1}$-OT. Again, if $P_R$ is corrupted then the adversary gets to choose these values.

In Fig. 3, we present subspace VOLE, a generalized notion of VOLE. Instead of a correlation of vectors $\vec{w} - \vec{v} = \vec{u}\Delta$, where $\vec{u} \in \mathbb{F}_p^\ell$ and $\vec{v} \in \mathbb{F}_q^\ell$ are given to $P_S$, and $\vec{w} \in \mathbb{F}_q^\ell$ and $\Delta \in \mathbb{F}_q$ to $P_R$ [BCGI18], subspace VOLE produces a correlation of matrices $W - V = UG_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta})$, where $U$ gets multiplied by the generator matrix $G_{\mathcal{C}}$ of a linear code $\mathcal{C}$. Subspace VOLE is essentially $n_{\mathcal{C}}$ independent VOLE correlations placed side-by-side, except that the rows of $U$ are required to be code words of $\mathcal{C}$. For $p = q = 2$, this matches the correlation generated internally by existing $\binom{N}{1}$-OT extensions.

**Selective Aborts.** Our base $\binom{N}{N-1}$-OT OT and subspace VOLE protocols achieve malicious security by using a consistency check to enforce honest behavior. However, the consistency checks allow a selective abort attack where $P_S$ can confirm a guess of part of $P_R$'s secret outputs. This is modeled in the ideal functionality using the function $\mathsf{Send/Abort}$ (Fig. 4). Let $x \in X$ be the value subject to the selective abort attack, and $y \in Y$ be the rest of $P_R$'s output. When $P_S$ is malicious, it can guess a subset $L \subseteq X$, and if it is correct (i.e. $x \in L$) then the protocol continues as normal. But if the guess is wrong then $P_R$ is notified of the error, and the protocol aborts.

The subset $L$ that $P_S$ guesses is restricted to being a member of $\mathcal{L}$, for some set of allowed guesses $\mathcal{L} \subseteq 2^X$. It is required to be closed under intersection, and contain the whole set $X$. For VOLE, where $X$ is a vector space, we also require that $L - \overleftarrow{L}_{\text{off}} \in \mathcal{L}$ when $L \in \mathcal{L}$ and $\overleftarrow{L}_{\text{off}} \in X$. We use one main set of allowed guesses, $\operatorname{Affine}(\mathbb{F}_q^n)$. It is the set of all affine subspaces of $\mathbb{F}_q^n$, i.e. all subsets that are defined by zero or more constraints of the form $a_0x_0 + \cdots + a_{n-1}x_{n-1} = a_n$, for constants $a_0, \ldots, a_n \in \mathbb{F}_q$. Since $\mathbb{F}_q$ can be viewed as the vector space $\mathbb{F}_p^k$, we have a superset relationship $\operatorname{Affine}(\mathbb{F}_p^{nk}) \supseteq \operatorname{Affine}(\mathbb{F}_q^n)$. There is also $\{X\}$, the trivial guess set, which only allows a malicious $P_S$ to guess that $x \in X$. This guess is trivially true, and so leaks no information at all.

**Pre-committed Inputs.** Our malicious OT extension protocol uses a universal hash to stop $P_R$ from causing collisions between two distinct extended OTs, which is sent in parallel with the VOLE consistency check for efficiency. However, the universal hash must be chosen *after* $P_R$ (who acts as the VOLE *sender*) picks its VOLE outputs $U, V$ and its guess $L$. In Fig. 5, we modify the VOLE functionality to notify the VOLE receiver once $U, V, L$ are almost fixed — unfortunately, the consistency check still allows $U, V, L$ to vary somewhat. Specifically, $U$ may have polynomially many options (which can be computationally hard to find), $L$ can get shifted by an offset $\overleftarrow{L}_{\text{off}}$, and $V$ can depend on the part of $\overleftarrow{\Delta}$ that is guessed.

To address these difficulties, we identify the possible input choices with witnesses $w_{\text{pre}}$, and have $\mathcal{A}$ output a witness checker, i.e. an implicitly defined set $\mathcal{W}_{\text{pre}}$ of valid witnesses. Then we require $U$, $V$, and $L$ to be fixed in terms of $w_{\text{pre}}$, using functions $U_{\text{pre}}(w_{\text{pre}})$, $V_{\text{pre}}(w_{\text{pre}}, \overleftarrow{\Delta})$, and $L_{\text{pre}}(w_{\text{pre}})$. We require a polynomial upper bound $M \geq |\mathcal{W}_{\text{pre}}|$ on the number of witnesses. Additionally, so that the correctness check for $V_{\text{pre}}$ does not leak any information, for all $\overleftarrow{\Delta}$ we require that $\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}} \in L_{\text{pre}}(w_{\text{pre}})$ implies $V = V_{\text{pre}}(w_{\text{pre}}, \overleftarrow{\Delta})$.

These changes are behind "if $P_S$ is malicious" checks, so in the semi-honest case $\mathcal{F}_{\mathsf{VOLE}}$ is a equivalent to $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}$. For malicious security, $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}$ gives the adversary less power than $\mathcal{F}_{\mathsf{VOLE}}$ because it forces some of the choices to be made early, so any protocol for $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}$ is also a protocol

```
$\mathcal{F}_{\mathsf{VOLE\text{-}pre}}^{p,q,\mathcal{C},\ell,\mathcal{L},M}$
if $P_S$ is malicious:
  recv. $\mathcal{W}_{\mathrm{pre}} \subseteq \{0,1\}^*$ from $\mathcal{A}$
  recv. $U_{\mathrm{pre}} \colon \mathcal{W}_{\mathrm{pre}} \to \mathbb{F}_p^{\ell \times k_{\mathcal{C}}}$ from $\mathcal{A}$
  recv. $V_{\mathrm{pre}} \colon \mathcal{W}_{\mathrm{pre}} \times \mathbb{F}_q^{n_{\mathcal{C}}} \to \mathbb{F}_q^{\ell \times n_{\mathcal{C}}}$ from $\mathcal{A}$
  recv. $L_{\mathrm{pre}} \colon \mathcal{W}_{\mathrm{pre}} \to \mathcal{L}$ from $\mathcal{A}$
send "commit" to $P_R$
run $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathcal{C},\ell,\mathcal{L}}$
instead of Send/Abort:
  if $P_S$ is malicious:
    recv. $w_{\mathrm{pre}} \in \mathcal{W}_{\mathrm{pre}}, \overleftarrow{L}_{\mathrm{off}} \in \mathbb{F}_q^{n_{\mathcal{C}}}$ from $\mathcal{A}$
    if $U \neq U_{\mathrm{pre}}(w_{\mathrm{pre}}) \vee V \neq V_{\mathrm{pre}}(w_{\mathrm{pre}}, \overleftarrow{\Delta}) \vee \overleftarrow{\Delta} + \overleftarrow{L}_{\mathrm{off}} \notin L_{\mathrm{pre}}(w_{\mathrm{pre}})$
      send "check failed" to $P_R$
      abort
  send $\overleftarrow{\Delta}, W$ to $P_R$
```

Figure 5: Modification of Fig. 3 to get an ideal functionality for subspace VOLE with a pre-commitment notification. We make two additional requirements on $\mathcal{A}$. There must be a polynomial upper bound $M \geq |\mathcal{W}_{\mathrm{pre}}|$ on the number of input choices $P_S$ has. And, for all $\overleftarrow{\Delta}$, $\overleftarrow{\Delta} + \overleftarrow{L}_{\mathrm{off}} \in L_{\mathrm{pre}}(w_{\mathrm{pre}})$ must imply $V = V_{\mathrm{pre}}(w_{\mathrm{pre}}, \overleftarrow{\Delta})$, to ensure that checking $V_{\mathrm{pre}}$ does not make the selective abort any more powerful.

for $\mathcal{F}_{\mathsf{VOLE}}$.

## 2.4 Correlation Robust Hashes

The final step of OT extension is to hash the output from the subspace VOLE. This requires a security assumption on the hash function $H$. We generalize the notion of a tweakable correlation robust (TCR) hash function [GKWY20] to our setting. While this definition will most likely be used with $p = 2$ for efficiency, there are extra theoretical difficulties associated with $p > 2$.

**Definition 2.5.** *A function $H \in \mathbb{F}_q^{n_{\mathcal{C}}} \times \mathcal{T} \to \{0,1\}^\lambda$ is a $(p, q, \mathcal{C}, \mathcal{T}, \mathcal{L})$-*TCR hash *if the oracles given in Fig. 6 are indistinguishable.*[4] *Formally, for any PPT adversary $\mathcal{A}$ that does not call* QUERY *twice on the same input $(\overleftarrow{x}, \overleftarrow{y}, \tau)$,*

$$\mathsf{Adv}_{\mathrm{TCR}} = \left| \Pr\left[\mathcal{A}^{\mathsf{TCR\text{-}real}^{H,p,q,\mathcal{C},\mathcal{L}}}() = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{TCR\text{-}ideal}^{H,p,q,\mathcal{C},\mathcal{L}}}() = 1\right] \right| \leq \mathrm{negl}.$$

Our definition is quite similar to the TCR of [GKWY20] in the special case where $\mathcal{C}$ is the repetition code. However, we explicitly include selective abort attacks in the TCR definition, while they require that the hash be secure for any distribution for $\overleftarrow{\Delta}$ with sufficient min-entropy. Their definition has issues when instantiated from idealized primitives such as random oracles, because, when the TCR is used for OT extension, the distribution for $\overleftarrow{\Delta}$ would have to depend on these primitives [CT21]. In the standard model, their definition is impossible to instantiate: $H(\overleftarrow{\Delta}, 0)$ must be random by TCR security, yet restricting $\overleftarrow{\Delta}$ so that the first bit of $H(\overleftarrow{\Delta}, 0)$ is zero only reduces

[4] Note that we do not consider multi-instance security. In fact, there is a generic attack: given $N$ instances, the attacker chooses an $L$ that contains $\overleftarrow{\Delta}$ with probability $1/N$, then brute forces $\overleftarrow{\Delta}$ for instances where $\overleftarrow{\Delta} \in L$. Thus, it is $N$-times cheaper to brute force attack $H$ for $N$ instances than to target a single one.

```
TCR-real$^{H,p,q,\mathcal{C},\mathcal{L}}$
$\overleftarrow{\Delta} \xleftarrow{\$} \mathbb{F}_q^{nc}$
QUERY($\overleftarrow{x} \in \mathbb{F}_p^{kc} \setminus \{0\}, \overleftarrow{y} \in \mathbb{F}_q^{nc}, \tau \in \mathcal{T}$):
    return $H(\overleftarrow{x} G_{\mathcal{C}} \odot \overleftarrow{\Delta} + \overleftarrow{y}, \tau)$

LEAK($L \in \mathcal{L}$):
    abort if $\overleftarrow{\Delta} \notin L$
```

(a) Real world.

```
TCR-ideal$^{H,p,q,\mathcal{C},\mathcal{L}}$
$\overleftarrow{\Delta} \xleftarrow{\$} \mathbb{F}_q^{nc}$
QUERY($\overleftarrow{x} \in \mathbb{F}_p^{kc} \setminus \{0\}, \overleftarrow{y} \in \mathbb{F}_q^{nc}, \tau \in \mathcal{T}$):
    $z \xleftarrow{\$} \{0,1\}^{\lambda}$
    return $z$
LEAK($L \in \mathcal{L}$):
    abort if $\overleftarrow{\Delta} \notin L$
```

(b) Ideal world.

Figure 6: Oracles for TCR definition. Calls to QUERY must not be repeated on the same input.

the min-entropy by approximately one bit and allows an efficient distinguisher. [CT21] fix the former issue with a definition TCR* that only applies to the ideal model, while ours allows the possibility of standard model constructions.

We now give two hash constructions, which we prove secure in Appx. A. Correlation robust hashes were inspired by random oracles (ROs), so it should be no surprise that a RO is a TCR hash.

**Proposition 2.6.** *A random oracle RO: $\mathbb{F}_q^{nc} \times \{0,1\}^t \to \{0,1\}^{\lambda}$ is a $(p, q, \mathcal{C}, \{0,1\}^t, \mathrm{Affine}(\mathbb{F}_p^{knc}))$-TCR hash, with distinguisher advantage at most $\tau_{max}\left(\mathfrak{q} + \frac{1}{2}\mathfrak{q}'\right)q^{-dc}$. Here, $\tau_{max}$ is the maximum number of times QUERY is called with the same $\tau$, $\mathfrak{q}$ is the number of RO queries made by the distinguisher, and $\mathfrak{q}'$ is the number of calls to QUERY.*

The next construction comes from [GKW+20]. It is the classic $x \mapsto \pi(x) \oplus x$ permutation-based hash function, but it uses an ideal cipher so that the tweak can be the key. Changing keys in a block cipher requires recomputing the round keys, so there is a cost to changing the tweak with this method. It needs a injection $\iota$ to encode its input; when $p = 2$, $\iota$ can be the identity map.

**Proposition 2.7.** *Let Enc: $\{0,1\}^t \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an ideal cipher, and $\iota\colon \mathbb{F}_q^{nc} \to \{0,1\}^{\lambda}$ be an injection. Then $H(\overleftarrow{y}, \tau) = \mathit{Enc}(\tau, \iota(\overleftarrow{y})) \oplus \iota(\overleftarrow{y})$ is a $(p, q, \mathcal{C}, \{0,1\}^t, \mathrm{Affine}(\mathbb{F}_p^{knc}))$-TCR hash. The distinguisher's advantage is at most $\tau_{max}\left((2\mathfrak{q} + \frac{1}{2}\mathfrak{q}')q^{-dc} + \frac{1}{2}\mathfrak{q}'2^{-\lambda}\right)$, with $\mathfrak{q}$ and $\mathfrak{q}'$ as in Thm. 2.6.*

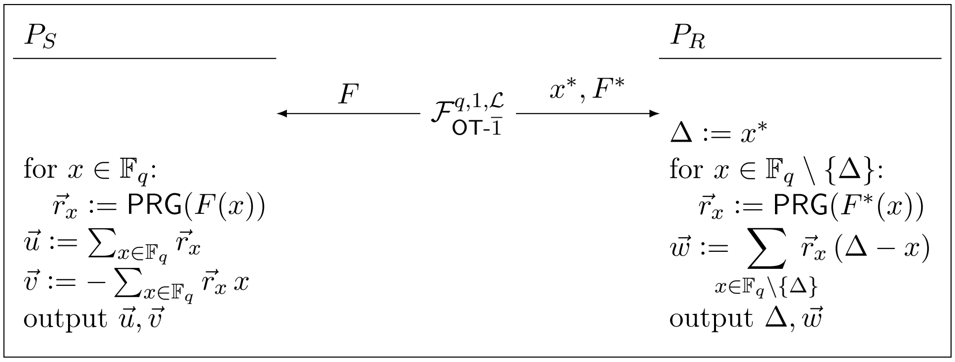
# 3 VOLE

## 3.1 For Small Fields

We already presented our $\mathbb{F}_{2^k}$-VOLE in Sect. 1.2. This VOLE is generalized in Fig. 7 to work over any small field $\mathbb{F}_q$, specifically fields where $q$ is only polynomially large, with $\vec{u}$ taking values in any subfield $\mathbb{F}_p$. It is based on a $\binom{q}{q-1}$-OT, and a pseudorandom generator PRG: $\{0,1\}^{\lambda} \to \mathbb{F}_p^{\ell}$. While this is a VOLE protocol, we analyze it using our subspace VOLE definition by setting $\mathcal{C}$ to be the length one, dimension one code, i.e. $G_{\mathcal{C}} = [1]$. This makes $U$, $V$, and $W$ all become column vectors and $\overleftarrow{\Delta}$ become a scalar.

**Theorem 3.1.** *The VOLE given in Fig. 7 in the $\mathcal{F}_{\mathit{OT}\text{-}\bar{1}}^{q,1,\mathcal{L}}$ hybrid model securely realizes $\mathcal{F}_{\mathit{VOLE}}^{p,q,\mathbb{F}_p,\ell,\mathcal{L}}$, in both the semihonest and malicious models.*

*Proof.* The proof of correctness is simple enough. Notice that the $x = \Delta$ term of the sum for $\vec{w}$



$P_S$

$F \longleftarrow \mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{q,1,\mathcal{L}} \longrightarrow x^*, F^*$

$P_R$

```
for $x \in \mathbb{F}_q$:
    $\vec{r}_x := \mathsf{PRG}(F(x))$
$\vec{u} := \sum_{x \in \mathbb{F}_q} \vec{r}_x$
$\vec{v} := -\sum_{x \in \mathbb{F}_q} \vec{r}_x \, x$
output $\vec{u}, \vec{v}$
```

```
$\Delta := x^*$
for $x \in \mathbb{F}_q \setminus \{\Delta\}$:
    $\vec{r}_x := \mathsf{PRG}(F^*(x))$
$\vec{w} := \sum_{x \in \mathbb{F}_q \setminus \{\Delta\}} \vec{r}_x \, (\Delta - x)$
output $\Delta, \vec{w}$
```

Figure 7: Protocol for small field VOLE. If $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{q,1,\mathcal{L}}$ instead outputs "$\mathsf{check\ failed}$", it should be passed straight through to $P_R$.

would be multiplied by $\Delta - \Delta = 0$, so it makes no difference that it must be excluded because $P_R$ does not know $\vec{r}_\Delta$. Therefore,

$$\vec{w} = \sum_{x \in \mathbb{F}_q \setminus \{\Delta\}} \vec{r}_x \, (\Delta - x) = \sum_{x \in \mathbb{F}_q} \vec{r}_x \, (\Delta - x) = \sum_{x \in \mathbb{F}_q} \vec{r}_x \, \Delta - \sum_{x \in \mathbb{F}_q} \vec{r}_x \, x = \vec{u} \, \Delta + \vec{v}. \tag{1}$$

**Corrupt $P_S$.** After receiving $F$ from $\mathcal{A}$, the simulator will compute $\vec{u}, \vec{v}$ honestly and submit them to $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathbb{F}_p,\ell,\mathcal{L}}$. If $P_S$ is malicious, it will also forward $L \in \mathcal{L}$ to the ideal functionality. In the real world, $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{q,1,\mathcal{L}}$ will generate a random $x^* = \Delta$ and send it to $P_R$, who will compute $\vec{w} = \vec{u} \, \Delta + \vec{v}$ by Eq. (1). In the ideal world, $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathbb{F}_p,\ell,\mathcal{L}}$ will pick $\Delta$ randomly, receive $\vec{u}, \vec{v}$ from the simulator, and compute $\vec{w} = \vec{u} \, \Delta + \vec{v}$. These are identical, implying that these two worlds are indistinguishable and that this case is secure.

**Corrupt $P_R$.** After receiving $F^*, x^*$ from $\mathcal{A}$, the simulator will compute $\Delta = x^*$ and $\vec{w}$ honestly, and submit them to $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathbb{F}_p,\ell,\mathcal{L}}$. We do a hybrid proof, starting from the real world and going to the ideal world.

1. In the real world, $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{q,1,\mathcal{L}}$ sets $F(x) = F^*(x)$ for $x \neq x^*$, generates $F(x^*)$ randomly, and sends them to $P_S$, who will compute $\vec{r}_x = \mathsf{PRG}(F(x))$ and $\vec{u}, \vec{v}$. By Eq. (1), $\vec{v} = \vec{w} - \vec{u} \, \Delta$.
2. Because $F(x^*)$ is only used to compute $\vec{r}_{x^*}$, the security of $\mathsf{PRG}$ implies that $\vec{r}_{x^*}$ can be replaced with a uniformly sampled value.
3. Instead of sampling $\vec{r}_{x^*}$ randomly, sample $\vec{u}$ uniformly at random and set $\vec{r}_{x^*} = \vec{u} - \sum_{x \neq x^*} \vec{r}_x$. This is an identical distribution.
4. We are now at the ideal world, where $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathbb{F}_p,\ell,\mathcal{L}}$ will pick $\vec{u}$ randomly, receive $\Delta, \vec{w}$ from the simulator, and compute $\vec{v} = \vec{w} - \vec{u} \, \Delta$.

If both parties are corrupt then security is trivial, as then the simulator can just forward messages between the corrupted parties. □

**Efficient Computation.** Let $a$ be a generator of $\mathbb{F}_q$ over $\mathbb{F}_p$. For computation, it's convenient to represent $\vec{v}$ as a sequence of $\mathbb{F}_p$ vectors: $\vec{v} = \vec{v}_0 + a\vec{v}_1 + \cdots + a^{k-1}\vec{v}_{k-1}$. Similarly, the index $x$ becomes $x_0 + ax_1 + \cdots + a^{k-1}x_{k-1}$. Naïve computation of $\vec{v}$ using the sum then becomes $\vec{v}_i = \sum_x x_i \vec{r}_x$, but this would require $O(kq)$ vector additions and scalar multiplications over $\mathbb{F}_p$.

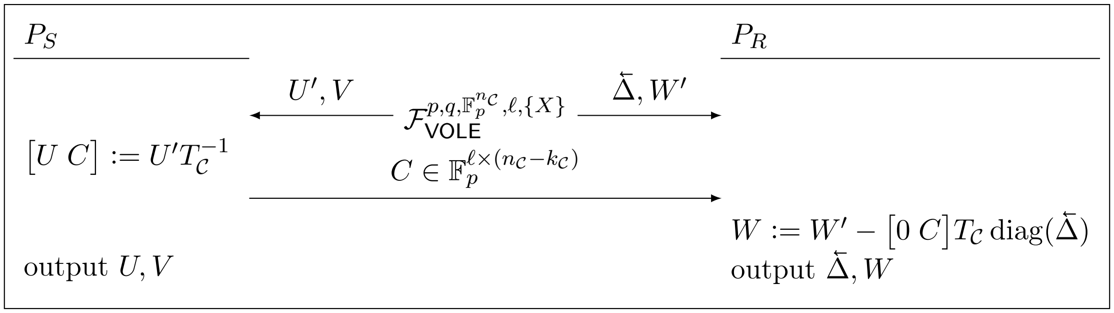

$P_S$
$P_R$
$U', V$
$\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathbb{F}_p^{n_\mathcal{C}},\ell,\{X\}}$
$\overleftarrow{\Delta}, W'$
$\begin{bmatrix}U \; C\end{bmatrix} := U' T_\mathcal{C}^{-1}$
$C \in \mathbb{F}_p^{\ell \times (n_\mathcal{C} - k_\mathcal{C})}$
$W := W' - \begin{bmatrix}0 \; C\end{bmatrix} T_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta})$
output $U, V$
output $\overleftarrow{\Delta}, W$

Figure 8: Protocol for subspace VOLE.

This can be improved to $O(q + \frac{q}{p} + \frac{q}{p^2} + \cdots) = O(q)$ vector additions and no scalar multiplications. For all $x' \in \mathbb{F}_q$ where $x'_0 = 0$, let $\vec{r}'_{x'} = \sum_{x_0 \in \mathbb{F}_p} \vec{r}_{x'+x_0}$, and notice that all $\vec{v}_1, \ldots, \vec{v}_{k-1}$ (and $\vec{u}$) depend only on the $\vec{r}'_{x'}$. Therefore, after computing all $\frac{q}{p}$ vectors $\vec{r}'_{x'}$, the outputs $\vec{v}_1, \ldots, \vec{v}_{k-1}$ can be found by recursion on a smaller problem size. As a byproduct, computing the $\vec{r}'_{x'}$ produces sequences of partial sums $\sum_{x_0 \leq i} \vec{r}_{x'+x_0}$, and adding all of these together then gives $\sum_{x'} \sum_{x_0} (p - x_0) \vec{r}_{x'+x_0} = \vec{v}_0$. $P_R$ can use the same algorithm to compute $\vec{w}$ by just reordering the $\vec{r}_x$ vectors at the start, because $\sum_x \vec{r}_x (\Delta - x) = \sum_x \vec{r}_{x+\Delta}(-x)$.

**Concatenation.** While this does not directly follow directly from the UC theorem, it should be clear that running the protocol Fig. 7 on a batch of $n$ OTs will produce a batch of $n$ VOLEs. The proof trivially generalizes. More precisely, it achieves $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathbb{F}_p^n,\ell,\mathcal{L}}$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{q,n,\mathcal{L}}$ hybrid model, where $\mathbb{F}_p^n$ is the trivial code with $G_{\mathbb{F}_p^n} = \mathbb{1}_n$. This will be the basis for our subspace VOLE.

## 3.2 For Subspaces

For $\binom{2}{1}$-OT extension, the next step would be for $P_S$ to send a correction to make all columns of $U$ be identical, so that each column would use the same set of choice bits. Efficient $\binom{N}{1}$-OT extension protocols like [KK13] instead must correct the rows of $U$ to lie in an arbitrary linear code $\mathcal{C}$, rather than the repetition code. We implement subspace VOLE to handle these more general correlations.

Our protocol for subspace VOLE is presented in Fig. 8. It starts out with a VOLE correlation $W' - V = U' \operatorname{diag}(\Delta)$. Then, $P_S$ divides $U'$ into parts, the message $U \in \mathbb{F}_p^{\ell \times k_\mathcal{C}}$ and the correction syndrome $C \in \mathbb{F}_p^{\ell \times n_\mathcal{C} - k_\mathcal{C}}$, sending the correction to $P_R$. $P_R$ then corrects $W$ to maintain the VOLE correlation property after $P_S$ removes $C$. Unfortunately, $P_S$ can just lie when it sends $C$ to $P_R$, so the protocol only achieves semi-honest security. Since the leakage set $\mathcal{L}$ only matters for malicious security, we simplify by assuming that $\mathcal{L}$ is trivial (i.e. $\{X\}$).

**Theorem 3.2.** *The protocol in Fig. 8 is a semi-honest realization of $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathcal{C},\ell,\{X\}}$ in the $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathbb{F}_p^n,\ell,\{X\}}$ hybrid model.*

*Proof.* First, the protocol outputs correctly satisfy the VOLE correlation:

$$\begin{aligned}
W &= W' - \begin{bmatrix}0 \; C\end{bmatrix} T_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta}) \\
&= V + U' \operatorname{diag}(\overleftarrow{\Delta}) - \begin{bmatrix}0 \; C\end{bmatrix} T_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta}) \\
&= V + \left(\begin{bmatrix}U \; C\end{bmatrix} T_\mathcal{C} - \begin{bmatrix}0 \; C\end{bmatrix} T_\mathcal{C}\right) \operatorname{diag}(\overleftarrow{\Delta}) \\
&= V + U G_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta}).
\end{aligned}$$

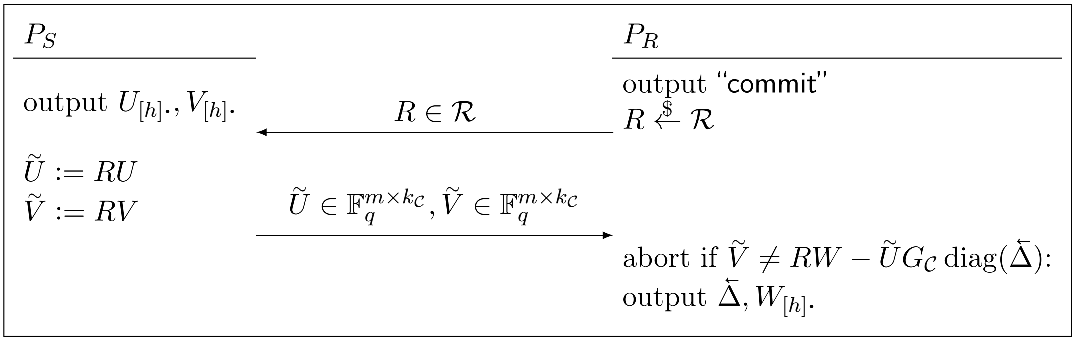

$P_S$
$P_R$
output $U_{[h]\cdot}, V_{[h]\cdot}$
output "commit"
$R \xleftarrow{\$} \mathcal{R}$
$R \in \mathcal{R}$
$\widetilde{U} := RU$
$\widetilde{V} := RV$
$\widetilde{U} \in \mathbb{F}_q^{m \times k_\mathcal{C}}, \widetilde{V} \in \mathbb{F}_q^{m \times k_\mathcal{C}}$
abort if $\widetilde{V} \neq RW - \widetilde{U} G_\mathcal{C} \operatorname{diag}(\overleftharpoon{\Delta})$:
output $\overleftharpoon{\Delta}, W_{[h]\cdot}$

Figure 9: Consistency checking protocol, which should be used with Fig. 8. $\mathcal{R}$ must be a $\epsilon$-universal hash family, where all $R \in \mathcal{R}$ is $\mathbb{F}_p^h$-hiding. The "abort if" means that "check failed" is output if the check fails. If instead of giving $\overleftharpoon{\Delta}, W'$ to $P_R$, the base VOLE outputs "check failed", $P_R$ should continue to play along with the protocol and only output "check failed" when it completes.

For security, notice that any $U, V, \overleftharpoon{\Delta}, W$ output by the protocol and any $C$ that the adversary eavesdrops on (because the communication is over an authenticated, but not private, channel) corresponds to a unique $U', V, \overleftharpoon{\Delta}, W'$ from the underlying VOLE. Specifically, $U' = \begin{bmatrix} U & C \end{bmatrix} T_\mathcal{C}$ and $W' = W + \begin{bmatrix} 0 & C \end{bmatrix} T_\mathcal{C} \operatorname{diag}(\overleftharpoon{\Delta})$. This implies the adversary does not learn anything new by corrupting either party, as they could already predict what that party knows. They only gain the power to program that the base VOLE's outputs for that party, but the simulator gains the corresponding power to program that party's protocol outputs to match. In more detail, $\mathcal{S}$ should receive from $\mathcal{A}$ the programed base VOLE outputs for the corrupted parties, simulate doing exactly what they would do in the protocol (while sampling a fake $C \xleftarrow{\$} \mathbb{F}_p^{\ell \times (n_\mathcal{C} - k_\mathcal{C})}$ if $P_S$ is honest), and program the protocol outputs to be the result.

In the ideal world, $\mathcal{S}$ generates a uniformly random consistent adversary view $U, V, \overleftharpoon{\Delta}, W$ (together with $U'$ or $W'$ if $P_S$ or $P_R$ was corrupted). In the real world, the underlying VOLE functionality picks $U', V, \overleftharpoon{\Delta}, W'$ uniformly at random subject to the constraints of the VOLE correlation and any outputs programmed by the adversary, and then the adversary gets to see the protocol run. There is a bijection between consistent adversary views and outputs of the underlying VOLE $U', V, \overleftharpoon{\Delta}, W'$, and this bijection implies that these two views are identically distributed. □

# 4 Malicious Security

Our small field VOLE construction in Sect. 3.1 was easily proved maliciously secure. It does not involve any communication, and so there are no opportunities for any of the parties to lie. However, Sect. 3.2 requires $P_S$ to reveal part of $U$, allowing a malicious $P_S$ to lie. Following KOS and OOS, we solve this by introducing a consistency check (Fig. 9) that is run immediately afterwards, to provide a guarantee that if $P_S$ lies then the protocol will either abort or work properly. Then the last few rows of $U$, $V$, and $W$ are thrown away so that the values revealed in the consistency check do not leak anything. This still allows the possibility of selective abort attacks, however.

KOS, OOS, and PSS all compute their consistency checks by multiplying each row of $U$ with a random value — an element of an extension field for KOS or just a vector for OOS and PSS. $V$ and $W$ are also multiplied by random values, in a consistent way. We generalize this idea to work for a large class of linear universal hashes. Any linear $\epsilon$-almost universal hash family $\mathcal{R} \subseteq \mathbb{F}_q^{m \times \ell}$ will work, as long as the following condition is met by every $R \in \mathcal{R}$, to guarantee that throwing away the last few rows of $U$ is sufficient to keep the others hidden.

**Definition 4.1.** *A matrix $R \in \mathbb{F}_q^{m \times \ell}$ is* $\mathbb{F}_p^h$-hiding *if the first $h$ inputs to $R$ will stay hidden when the remaining inputs are secret and uniformly random. More precisely, if $\vec{x} \xleftarrow{\$} \mathbb{F}_p^\ell$ then $R\vec{x}$ must be independently random from $\vec{x}_{[h]}$*

Note that if $R$ is $\mathbb{F}_p^h$-hiding then that it is $\mathbb{F}_q^h$-hiding, so if $R$ is able to keep $U \in \mathbb{F}_p^{\ell \times k_\mathcal{C}}$ hidden then it will keep $V \in \mathbb{F}_q^{\ell \times n_\mathcal{C}}$ hidden as well.

Many useful universal hashes with elements in $\mathbb{F}_p$ satisfy this definition, including hashes based on polynomial evaluation or cyclic redundancy checks. That is, the last $m$ columns of $R$ will span the others, so $R$ will be $\mathbb{F}_p^h$-hiding for $h = \ell - m$. However, this only works if the universal hash is over $\mathbb{F}_p$, rather than $\mathbb{F}_q$, as otherwise there won't be enough entropy in the last $m$ columns to completely hide the other inputs. On the other hand, using a hash over $\mathbb{F}_q$ gives better compression. For a universal hash over $\mathbb{F}_p$, the best possible $\epsilon$ is about $p^{-m}$, while for $\mathbb{F}_q$ it is $q^{-m} = p^{-km}$. We believe that the best approach is to compose two universal hashes, first applying a $\mathbb{F}_p^{\ell - m'}$-hiding hash $R \in \mathcal{R} \subseteq \mathbb{F}_p^{m' \times \ell}$, then further reducing the output down to $m$ entries with a second hash $R' \in \mathcal{R}' \subseteq \mathbb{F}_q^{m \times m'}$ where $m' \geq km$. The composed hash will be $\mathbb{F}_p^{\ell - m'}$-hiding, and will still be universal by Thm. 2.3.

**Remark 4.2.** *$P_S$ outputs $U_{[h]\cdot}, V_{[h]\cdot}$ in the first round, just after sending $C$ and much before the protocol has actually completed. In applications where $U$ will be derandomized immediately (e.g. chosen point OT extension), it is convenient to derandomize $U$ at the same time as sending $C$. The protocol returning early is what allows this within the UC framework.*

**Remark 4.3.** *After sending $C$, $P_S$ will not have many useful options to choose from, so the protocol notifies $P_R$ with "*commit*" (as in $\mathcal{F}_{\textsf{VOLE-pre}}^{p,q,\mathcal{C},h,\mathcal{L},M}$) to indicate that $P_S$'s inputs (mostly) fixed. In Sect. 5, this notification is used to send a second universal hash at the same time as $R$.*

## 4.1 Flaws in Existing Consistency Checks

Given the similarity of Fig. 9 to the KOS, PSS, and OOS consistency checks, it seems natural to adapt their proofs to the subspace VOLE consistency checking protocol. However, it turns out that all three are flawed. We first present the flaw in OOS, because it is most similar to our protocol.

### 4.1.1 Flaw in OOS's Proof.

To get the OOS consistency check, take the protocol in Fig. 9 and set $p = q = 2$ and $R = \begin{bmatrix} X \ \mathbb{1}_m \end{bmatrix}$, where $X \xleftarrow{\$} \mathbb{F}_2^{m \times \ell - m}$ is uniformly random. There are a couple of differences, but these do not affect the consistency check proper. Our sender is their receiver and vice versa, because they are implementing OT extension and we are doing subspace VOLE. And, they send a correction $C$ for the whole of $U'$ at once, instead of just the syndrome, because their OT choice bits are chosen rather than random. To avoid the confusion of introducing a separate set of notations for essentially the same protocol, we ignore these differences and discuss their proof using our notation and protocol. See Appx. B for a discussion using OOS's original language.

Let $[U \ \bar{C}] = U' T_\mathcal{C}^{-1} \oplus \begin{bmatrix} 0 \ C \end{bmatrix}$, so $\bar{C}$ is the error in the correction syndrome $C$ sent by the malicious $P_S$. Similarly, let $\bar{U} = RU \oplus \tilde{U}$ and $\bar{V} = RV \oplus \tilde{V}$ be the errors in the consistency check messages sent by $P_S$. The consistency check then becomes $\bar{V} = [\bar{U} \ \ R\bar{C}] T_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta})$ (see the proof of Thm. 4.5 for details). OOS define a set $E \subseteq [n_\mathcal{C}]$ of column indices $i$ where $\left([\bar{U} \ \ R\bar{C}] T_\mathcal{C}\right)_{\cdot i}$ is nonzero. These are the indices $i$ where $\Delta_i$ will have to be guessed by $P_S$ in order to pass the consistency check. They then attempt to prove that the indices in $E$ will be the only ones that $P_S$ lied about. That is, their simulator tries to correct $U$ to get $P_S$'s real output $U^\star$, so that if $Z = [U^\star \ C] T_\mathcal{C} \oplus U'$ then the indices of all the nonzero columns of $Z$ are in $E$. This would let $\mathcal{S}$ update $V$ accordingly, getting $V^\star = V \oplus Z\,\mathrm{diag}(\vec{\Delta})$, which it could find because $P_S$ must guess $\Delta_i$ for $i \in E$.

The flaw is in their proof that $\mathcal{S}$ can (with high probability, assuming that the check passes) extract $U^\star$. Their technique is to look at $Y = [U\ \bar{C}]T_{\mathcal{C}} = U' \oplus \begin{bmatrix}0\ C\end{bmatrix}T_{\mathcal{C}}$, whose rows would be in $\mathcal{C}$ if $P_S$ were honest, and remove the columns in $E$ to get a punctured matrix $Y_{-E}$. Then they decode the rows of $Y_{-E}$ using the punctured code $\mathcal{C}_{-E}$ to get $U^\star$, since $Y \oplus Z = U^\star G_{\mathcal{C}}$ and $Z_{-E}$ should be 0. For this to work, they need the rows of $Y_{-E}$ to be in $\mathcal{C}_{-E}$. They try to prove this using the following lemma.

**Lemma 4.4** (OOS, Lem. 1)**.** *Let $\mathcal{D}$ be a linear code and $B \in \mathbb{F}_2^{\ell \times n_{\mathcal{D}}}$ be a matrix, where not all rows of $B$ are in $\mathcal{D}$. If $X \xleftarrow{\$} \mathbb{F}_2^{m \times \ell - m}$ and $R = \begin{bmatrix}X\ \mathbb{1}_m\end{bmatrix}$, then the probability that all rows of $RB$ are in $\mathcal{D}$ is at most $2^{-m}$.*

They apply this lemma with $\mathcal{D} = \mathcal{C}_{-E}$ and $B = Y_{-E}$. Note that $RY = [\bar{U}\ \ R\bar{C}]T_{\mathcal{C}} \oplus \widetilde{U}G_{\mathcal{C}}$, so $RY_{-E} = \widetilde{U}G_{\mathcal{C}_{-E}}$ has all rows in $\mathcal{C}_{-E}$. They conclude that with all but negligible probability, all rows of $Y_{-E}$ are in $\mathcal{C}_{-E}$. However, the lemma cannot be used in this way. The lemma requires that $\mathcal{D}$ and $B$ be fixed in advance, *before* $X$ is sampled, yet $\mathcal{C}_{-E}$ and $Y_{-E}$ both depend on $E$. Recall that $E$ is the set of nonzero columns of $[\bar{U}\ \ R\bar{C}]T_{\mathcal{C}}$, which depends on both $R$ directly, and on the consistency check message $\widetilde{U}$ sent by $P_S$ *after* it learns $X$.

While this shows that OOS's proof is wrong, we have not found any attacks that contradict their theorem statement. Additionally, a special case of our new proof (Thm. 4.5) shows that the OOS protocol is still secure, with statistical security only one bit less than was claimed.

### 4.1.2 Attack For PSS's Protocol.

The PSS consistency checking protocol is similar to OOS's, though they only consider Walsh–Hadamard codes, and they generate $R \xleftarrow{\$} \mathbb{F}_2^{m \times \ell}$ using a coin flipping protocol. In Lemma IV.5, they have a similar proof issue to OOS, using Corollary IV.2 on dependent values when the corollary assumes they are independent. However, we focus on a more significant problem, which we summarize here, using our own notation. See Appx. C for a more detailed discussion, using their notation.

The most important difference from OOS is that PSS attempt to compress the consistency check by summing the columns of $\widetilde{V}$ to get $\tilde{v} = \widetilde{V}[1\ \cdots\ 1]^\top$. The consistency check is then that $\tilde{v}$ must equal $\big(RW \oplus \widetilde{U}G_{\mathcal{C}}\,\mathrm{diag}(\vec{\Delta})\big)[1\ \cdots\ 1]^\top = RW[1\ \cdots\ 1]^\top \oplus \widetilde{U}G_{\mathcal{C}}\vec{\Delta}$. Let $\bar{C}$, $\bar{U}$, and $\bar{v}$ be defined analogously to our discussion of OOS. Then the consistency check is $\bar{v} = [\bar{U}\ \ R\bar{C}]T_{\mathcal{C}}\vec{\Delta}$. This means that a malicious receiver only needs to guess XORs of multiple bits from $\vec{\Delta}$, rather than the individual bits themselves.

We used this to create an attack against PSS. Have $P_S$ lie about the bits in $U'$ in length $N$ intervals, where in the first OT it lies about the first $N$ bits of $U'_{0\cdot}$, and in the next OT the second $N$ bits of $U'_{1\cdot}$, and so on. Here, $N$ is a parameter defining the tradeoff between computational cost and attack success rate. Then $[\bar{U}\ \ R\bar{C}]T_{\mathcal{C}}$ will have rows spanned by these $N$ bit intervals, so $[\bar{U}\ \ R\bar{C}]T_{\mathcal{C}}\vec{\Delta}$ only depends on $\lceil \frac{n_{\mathcal{C}}}{N} \rceil$ different values: $\bigoplus_{j=0}^{N-1} \Delta_{Ni+j}$ for $i \in \big[\lceil \frac{n_{\mathcal{C}}}{N} \rceil\big]$. Therefore, the consistency check passes with probability $2^{-\lceil n_{\mathcal{C}}/N \rceil}$, even though we have lied about all $n_{\mathcal{C}}$ bits. Later, having gotten away with these lies, the hashes output by the OT extension can be brute forced to solve for each $N$-bit chunk of $\vec{\Delta}$ individually. This breaks the OT extension in time $\lceil \frac{n_{\mathcal{C}}}{N} \rceil 2^{N-1}$. At the $\lambda = 128$ security level, $n_{\mathcal{C}} = 256$, so by setting $N = 32$ we get an attack with success probability $2^{-8}$ that uses only $2^{34}$ hash evaluations.

### 4.1.3 Flaw in KOS's Proof.

Like with OOS, in this section we will discuss KOS by analogy with our consistency checking protocol Fig. 9. See Appx. D for a more detailed account, using KOS's notation for their protocol.

To get KOS's protocol from ours, start by fixing $p = q = 2$ and $\mathcal{C} = \mathsf{Rep}(\mathbb{F}_2^\lambda)$. Let $\mathcal{R} = \mathbb{F}_2^{\lambda \times \ell}$, which means that $R$ is $\mathbb{F}_2^{\ell-\lambda-\sigma}$-hiding with probability at least $1 - 2^{-\sigma}$. They use a coin flipping protocol to make sure that $P_R$ cannot pick a $R$ that is is not hiding. Let $\alpha$ a primitive element of $\mathbb{F}_{2^\lambda}$, meaning that $\{1, \alpha, \ldots, \alpha^{\lambda-1}\}$ is a basis for $\mathbb{F}_{2^\lambda}$ over $\mathbb{F}_2$. The first half of the consistency check, $\widetilde{U}$, works as normal, except that it gets encoded into a field element $u = \bigoplus_i \widetilde{U}_{i\cdot}\alpha^i = \vec{\alpha}^\top \widetilde{U}$, where $\vec{\alpha} = [1, \alpha, \ldots, \alpha^{\lambda-1}]^\top$. The other half, $\widetilde{V}$, is compressed from $\lambda^2$ bits down to $\lambda$ bits by turning it into a single field element $v = \bigoplus_{ij} \widetilde{V}_{ij}\alpha^{i+j} = \vec{\alpha}^\top \widetilde{V}\vec{\alpha}$. Similarly, let $w = \vec{\alpha}^\top RW\vec{\alpha}$ and $\delta = \overleftharpoon{\Delta}\vec{\alpha}$. Then the consistency check becomes

$$\begin{aligned} v &= \vec{\alpha}^\top RW\vec{\alpha} \oplus \vec{\alpha}^\top \widetilde{U} G_{\mathcal{C}} \operatorname{diag}(\overleftharpoon{\Delta})\vec{\alpha} \\ &= w \oplus uG_{\mathcal{C}} \operatorname{diag}(\overleftharpoon{\Delta})\vec{\alpha} = w \oplus u[1 \ \cdots \ 1] \operatorname{diag}(\overleftharpoon{\Delta})\vec{\alpha} = w \oplus u\delta. \end{aligned}$$

Because $\mathcal{C}$ is a repetition code, $U'$ is supposed to be derandomized so that all columns are identical to $U$. Let $Y = U' \oplus \begin{bmatrix}0 \ C\end{bmatrix} T_{\mathcal{C}}$ be the derandomization of $U'$. Then columns $i$ and $j$ are called *consistent* if they imply the same values of $U$, i.e. if $Y_{\cdot i} = Y_{\cdot j}$. Also let $S_\Delta$ be the set of possible $\Delta$ that cause the consistency check to succeed. KOS's proof of security for malicious $P_S$ depends entirely on their Lemma 1, which states several properties of their consistency check. Most importantly, it implies that for any $u, v$ sent by $P_S$, with probability $1 - 2^{-\lambda}$ there exists $k \in \mathbb{N}$ such that $|S_\Delta| = 2^k$ and $k$ is at most the size of the largest group of consistent columns.

KOS gave no proof for Lemma 1, instead citing the full version of their paper, which has not been made public. However, the authors of KOS were kind enough to give an unpublished draft [KOS21]. Unfortunately, their proof has a similar flaw to OOS's, because they assume that $R$ is sampled after $S_\Delta$ is known.

Unlike OOS, we found a counterexample to show that KOS's Lemma 1 is false, which we call a collision attack. Let the malicious $P_S$ choose $C$ uniformly at random (so $Y$ will also be uniformly random) but still provide an honest $v$ during the consistency check. Because of the correction $P_R$ applies, $W$ will be

$$W = V \oplus (U' \oplus \begin{bmatrix}0 \ C\end{bmatrix} T_{\mathcal{C}}) \operatorname{diag}(\overleftharpoon{\Delta}) = V \oplus Y \operatorname{diag}(\overleftharpoon{\Delta})$$

Let $\overleftharpoon{y} = \vec{\alpha}^\top RY$. The consistency check is then

$$\begin{aligned} v &= \vec{\alpha}^\top RV\vec{\alpha} \oplus \vec{\alpha}^\top RY \operatorname{diag}(\overleftharpoon{\Delta})\vec{\alpha} \oplus uG_{\mathcal{C}} \operatorname{diag}(\overleftharpoon{\Delta})\vec{\alpha} \\ 0 &= (\overleftharpoon{y} \oplus u[1 \ \cdots \ 1]) \operatorname{diag}(\overleftharpoon{\Delta})\vec{\alpha}. \end{aligned}$$

If $u$ is set to be some element $y_i$ of $\overleftharpoon{y}$, the consistency check at least won't depend on $\Delta_i$. Since $Y$ is uniformly random, $\overleftharpoon{y}$ will be as well, so the probability of a collision among the $y_i$ is roughly $\lambda^2 2^{-\lambda-1}$. If there is a collision $y_i = y_j$ and $P_R$ sets $u = y_i$, then $|S_\Delta| = 2^k = 4$. This contradicts KOS's Lemma 1 because $k$ should be at most 1 as no two columns are consistent.

In Appx. D.2 we present (using KOS's notation) a stronger attack against special parameters of KOS. Assuming that a certain MinRank problem always has a solution (and heuristically it should have $2^{\lambda/5}$ solutions on average), the attack succeeds in recovering $\Delta$ with probability $2^{-\frac{3}{5}\lambda}$ using $O(2^{\lambda/5})$ random oracle queries. While this is still not a practical attack, according to KOS's proof of their Theorem 1 an attack with this few random oracle queries should only succeed with probability $O(2^{-\frac{4}{5}\lambda})$.

```
$\mathcal{S}^{p,q,\mathcal{C},\ell}_{\textsf{sub-VOLE-mal-R}}$
recv. $\overleftarrow{\Delta} \in \mathbb{F}_q^{n_\mathcal{C}}, W' \in \mathbb{F}_q^{\ell \times n_\mathcal{C}}$ from $\mathcal{A}$
send $\overleftarrow{\Delta}, W'$ to $P_R$
$C \xleftarrow{\$} \mathbb{F}_p^{\ell \times (n_\mathcal{C} - k_\mathcal{C})}$
send $C$ to $P_R$
$W := W' - \begin{bmatrix} 0 \; C \end{bmatrix} T_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta})$
send $\overleftarrow{\Delta}, W_{[h]\cdot}$ to $\mathcal{F}^{p,q,\mathcal{C},h,\mathcal{L},M}_{\textsf{VOLE-pre}}$
recv. $R \in \mathcal{R}$ from $P_R$
$U_\$ \xleftarrow{\$} \mathbb{F}_q^{\ell \times k_\mathcal{C}}$
$\widetilde{U} := R U_\$$
$\widetilde{V} := RW - \widetilde{U} G_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta})$
send $\widetilde{U}, \widetilde{V}$ to $P_R$
```

```
$\mathcal{S}^{p,q,\mathcal{C},\ell}_{\textsf{sub-VOLE-mal-S}}$
recv. $U' \in \mathbb{F}_p^{\ell \times n_\mathcal{C}}, V \in \mathbb{F}_q^{\ell \times n_\mathcal{C}}$ from $\mathcal{A}$
send $U', V$ to $P_S$
recv. $L' \in \mathcal{L}$ from $P_S$:
recv. $C \in \mathbb{F}_p^{\ell \times (n_\mathcal{C} - k_\mathcal{C})}$ from $P_S$
$[U \; \bar{C}] := U' T_\mathcal{C}^{-1} - \begin{bmatrix} 0 \; C \end{bmatrix}$

$R \xleftarrow{\$} \mathcal{R}$
abort if $\operatorname{rank}(R\bar{C}) < \operatorname{rank}(\bar{C})$
find $R^{-1} \in \mathbb{F}_q^{\ell \times m}$ s.t. $R^{-1} R \bar{C} = \bar{C}$
$\mathcal{W}_\text{pre}, U^\star_\text{pre}, V^\star_\text{pre}, L_\text{pre} := \textsf{Precom}(\bar{C}, R, R^{-1})$
send $\mathcal{W}_\text{pre}, U^\star_\text{pre}, V^\star_\text{pre}, L_\text{pre}$ to $\mathcal{F}^{p,q,\mathcal{C},h,\mathcal{L},M}_{\textsf{VOLE-pre}}$
send $R$ to $P_S$

recv. $\widetilde{U} \in \mathbb{F}_q^{m \times k_\mathcal{C}}, \widetilde{V} \in \mathbb{F}_q^{m \times n_\mathcal{C}}$ from $P_S$
$\bar{U} := RU - \widetilde{U}; \quad U^\star := U^\star_\text{pre}(\bar{U})$
$\bar{V} := RV - \widetilde{V}; \quad V^\star := V - R^{-1}\bar{V}$
send ${U^\star}_{[h]\cdot}, {V^\star}_{[h]\cdot}$ to $\mathcal{F}^{p,q,\mathcal{C},h,\mathcal{L},M}_{\textsf{VOLE-pre}}$
find $\overleftarrow{L}_\text{off} \in -L'$ s.t. $\bar{V} = [\bar{U} \;\; R\bar{C}] T_\mathcal{C} \operatorname{diag}(\overleftarrow{L}_\text{off})$
abort if none exist
send $\bar{U}, \overleftarrow{L}_\text{off}$ to $\mathcal{F}^{p,q,\mathcal{C},h,\mathcal{L},M}_{\textsf{VOLE-pre}}$
```

```
$\textsf{Precom}(\bar{C}, R, R^{-1})$:
$\mathcal{W}_\text{pre} := \{\bar{U} \in \mathbb{F}_q^{m \times k_\mathcal{C}} \mid t \geq \| [\bar{U} \;\; R\bar{C}] T_\mathcal{C} \|_0\}$
$U^\star_\text{pre}(\bar{U}) := U - R^{-1}\bar{U}$
$V^\star_\text{pre}(\bar{U}, \overleftarrow{\Delta}) := V + R^{-1}[\bar{U} \;\; R\bar{C}] T_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta})$
$L'_0 := L' - \overleftarrow{\Delta}_0$ for some $\overleftarrow{\Delta}_0 \in L'$
$L_\text{pre}(\bar{U}) := L'_0 \cap \{\overleftarrow{\Delta} \mid 0 = [\bar{U} \;\; R\bar{C}] T_\mathcal{C} \operatorname{diag}(\overleftarrow{\Delta})\}$
return $\mathcal{W}_\text{pre}, U^\star_\text{pre}, V^\star_\text{pre}, L_\text{pre}$
```

Figure 10: Simulators for malicious security of Fig. 8 combined with Fig. 9, for a single corrupt party. $\mathcal{S}^{p,q,\mathcal{C},\ell}_{\textsf{sub-VOLE-mal-R}}$ is for corrupt $P_R$, while $\mathcal{S}^{p,q,\mathcal{C},\ell}_{\textsf{sub-VOLE-mal-S}}$ is for corrupt $P_S$.

### 4.2 Our New Proof

The biggest hurdle in the proof is the case where $P_S$ is malicious, i.e. proving that the consistency check works. If $P_S$ lies when it sends $C$, then it will have to guess some entries of $\Delta$, but which entries depends on what $\widetilde{U}$ it decides to send. As with OOS's flawed proof, $P_S$ does not have to make up its mind until after seeing $R$, and generally speaking universal hashes are only strong when used on data that was chosen independently of the hash. We need to find some property that only depends on $C$ and $R$ so that we can show that it holds (with high probability) based on $C$ being independent of $R$, then use it to prove security.

The property we found was that $R$ should preserve all the lies in $C$. More precisely, if $\bar{C}$ is the difference between the honest $C$ and the one $P_S$ sent, then $R\bar{C}$ and $\bar{C}$ should have the same row space.[5] The idea is that, if $R$ were the identity, the consistency check would clearly ensure that whatever incorrect value $C$ that $P_S$ provides, it can still guess matrices $U, V$ that make the VOLE correlation hold. Although $R$ is not the identity matrix, the check still ensures that the VOLE correlation holds for $\widetilde{U}, \widetilde{V}$. The lie preserving property of $R$ then shows that they contain enough information to correct the whole of $U$ and $V$ so that they do satisfy the VOLE correlation.

**Theorem 4.5.** *The subspace VOLE protocol in Fig. 8 combined with the consistency checking protocol in Fig. 9 is a maliciously secure implementation of* $\mathcal{F}^{p,q,\mathcal{C},h,\mathcal{L},M}_{\textsf{VOLE-pre}}$ *if* $\mathcal{L} \supseteq \operatorname{Affine}(\mathbb{F}_q^{n_\mathcal{C}})$*, assuming that* $\mathcal{R} \subseteq \mathbb{F}_q^{m \times \ell}$ *is a* $\epsilon$*-almost universal family where all* $R$ *are* $\mathbb{F}_p^h$*-hiding. The distinguisher has*

[5] This fails if there are too many lies; however the VOLE would likely abort anyway.

*advantage at most* $\frac{\epsilon q}{q-1} + q^{-t-1}$*, where* $t = \frac{d_\mathcal{C}}{1+\sqrt{1+\frac{d_\mathcal{C}}{n_\mathcal{C}}-\frac{1}{n_\mathcal{C}^2}}} \geq \frac{d_\mathcal{C}}{2}$ *and* $M = n_\mathcal{C}(d_\mathcal{C} - t)$*.*

Note: when instantiated as in OOS, $\epsilon = 2^{-m}$ and $q = 2$, so our proof shows that OOS has only 1 bit less statistical security than was claimed. The $q^{-t-1}$ term only matters for the pre-commitment property, which OOS does not consider.

*Proof.* There are four cases, depending on which parties are corrupted. If both parties are corrupted then the real protocol can be simulated trivially, by ignoring the ideal functionality and just passing messages between the corrupted parties. If both players are honest, the situation is very similar to the semi-honest protocol (Thm. 3.2). The only difference is the additional two rounds, which can be simulated by picking a random $R \in \mathcal{R}$, as well as sampling fake $P_S$ values $U_\$ \xleftarrow{\$} \mathbb{F}_p^{\ell \times k_\mathcal{C}}$ and $V_\$ \xleftarrow{\$} \mathbb{F}_p^{\ell \times n_\mathcal{C}}$ and simulating the third round as $\tilde{U} = RU_\$, \tilde{V} = RV_\$$. Since both parties are honest, $U$ and $V$ are uniformly random, and so Thm. 4.1 guarantees that these fakes are indistinguishable from the real consistency check.

The situation is similar when only $P_R$ is corrupted (simulator in Fig. 10, top left). Following the same principle as for the semi-honest protocol, $\mathcal{S}$ starts by performing the computations that an honest $P_R$ would, while randomly sampling a fake syndrome $C$ to send. To simulate the consistency check, after receiving $R$, the simulator fakes $\tilde{U}$ like in the honest–honest case, then solves for $\tilde{V}$ as the only possibility that will pass the consistency check. The real protocol and the simulation are indistinguishable because the honesty of $P_S$ implies that the consistency check will always pass, so the formula for $\tilde{V}$ must always hold, and $P_R$ cannot tell that $\tilde{U}$ was generated from the fake $U_\$$ because $R$ is $\mathbb{F}_p^h$-hiding.

The most interesting case is when $P_S$ is corrupt. We present a hybrid proof, starting with the real world, where the real protocol gets executed using the underlying ideal functionality $\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathbb{F}_p^{n_\mathcal{C}},\ell,\mathcal{L}}$, and work towards the ideal world, where the simulator (Fig. 10, right) liaises between the corrupted sender and the desired ideal functionality $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}^{p,q,\mathcal{C},h,\mathcal{L},M}$.

1. Compute what $P_S$'s honest output would be, and the difference between the honest syndrome and the one $P_S$ provided: $[U\ \bar{C}] = U' T_\mathcal{C}^{-1} - \begin{bmatrix}0\ C\end{bmatrix}$. Add a check after $P_S$ sends $\tilde{U}$ and $\tilde{V}$, where if $\operatorname{rank}(R\bar{C}) < \operatorname{rank}(\bar{C})$, "check failed" is sent to $P_R$ and the protocol aborts. The environment's advantage for this step is the probability that this abort triggers and the protocol would not have aborted anyway. We bound this probability using the following lemma.

   **Lemma 4.6.** *Let* $\mathcal{R} \subseteq \mathbb{F}_q^{m \times n}$ *be a linear* $\epsilon$*-almost universal family, and let* $A$ *be any matrix in* $\mathbb{F}_q^{n \times l}$*. Then,* $\mathbb{E}_{R \xleftarrow{\$} \mathcal{R}}\left[q^{\operatorname{rank}(A)-\operatorname{rank}(RA)} - 1\right] \leq \epsilon(q^{\operatorname{rank}(A)} - 1)$*.*

   *Proof.* By the rank–nullity theorem, $R$ defines an isomorphism $\mathbb{F}_q^n / \ker(R) \cong \operatorname{colspace}(R)$. Its restriction to $\operatorname{colspace}(A)$ gives an isomorphism $\operatorname{colspace}(A) / \ker(R) \cong \operatorname{colspace}(RA)$. Therefore,

$$\begin{aligned} \operatorname{rank}(RA) &= \dim\big(\operatorname{colspace}(RA)\big) \\ &= \dim(\operatorname{colspace}(A)) - \dim(\operatorname{colspace}(A) \cap \ker(R)) \\ &= \operatorname{rank}(A) - \dim(\operatorname{colspace}(A) \cap \ker(R)). \end{aligned}$$

   We then want to bound the expected value of $X = q^{\dim(\operatorname{colspace}(A) \cap \ker(R))} - 1 = |\operatorname{colspace}(A) \cap \ker(R) \setminus \{0\}|$. That is, $X$ is the number of nonzero $v \in \operatorname{colspace}(A)$ such that $Rv = 0$. By Thm. 2.1, for any particular $v \neq 0$ the probability that $Rv = 0$ is at most $\epsilon$. Since $X$ is the sum of $|\operatorname{colspace}(A) \setminus \{0\}| = q^{\operatorname{rank}(A)} - 1$ indicator random variables, we get $\mathbb{E}[X] \leq \epsilon(q^{\operatorname{rank}(A)} - 1)$. □

For the real protocol to not abort, $\widetilde{V} = RW - \widetilde{U} G_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta})$ must hold. Because $P_R$ is uncorrupted, $\overleftarrow{\Delta}$ is sampled uniformly in $\mathbb{F}_q^{n_{\mathcal{C}}}$ and $W'$ is computed as $U' \operatorname{diag}(\overleftarrow{\Delta}) + V$. Therefore,

$$\begin{aligned} W = W' - \big[0\ C\big] T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) &= (U' - \big[0\ C\big] T_{\mathcal{C}}) \operatorname{diag}(\overleftarrow{\Delta}) + V \\ &= [U\ \bar{C}] T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) + V. \end{aligned}$$

Let $\bar{U} = RU - \widetilde{U}$ and $\bar{V} = RV - \widetilde{V}$ be the differences between the honest consistency check messages and the ones sent by $P_S$. Then the consistency check is equivalent to $-\bar{V} = [\bar{U}\ \ R\bar{C}] T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta})$. Next, we need to bound

$$\begin{aligned} P &= \Pr\big[\text{abort} \wedge \text{check passes}\big] \\ &= \Pr\big[\operatorname{rank}(R\bar{C}) < \operatorname{rank}(\bar{C}) \wedge -\bar{V} = [\bar{U}\ \ R\bar{C}] T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta})\big]. \end{aligned}$$

Triggering this condition requires guessing $[\bar{U}\ \ R\bar{C}] T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta})$, i.e. guessing $\Delta_i$ for every nonzero column $\big([\bar{U}\ \ R\bar{C}] T_{\mathcal{C}}\big)_{\cdot i}$. Let $N = \|\,[\bar{U}\ \ R\bar{C}] T_{\mathcal{C}}\|_0$ be the number of these nonzero columns. A lower bound for $N$ is $\operatorname{rank}([\bar{U}\ \ R\bar{C}] T_{\mathcal{C}})$, because every zero column does not contribute to the rank. $T_{\mathcal{C}}$ is invertible, so multiplying by it does not change the rank. Adding extra columns only increases rank, so $\operatorname{rank}([\bar{U}\ \ R\bar{C}]) \geq \operatorname{rank}(R\bar{C})$. Up until the consistency check, the behavior of $P_R$ has been independent of $\overleftarrow{\Delta}$, and $N$ is also independent of $\overleftarrow{\Delta}$, so $\Pr\big[\text{check} \mid N\big] \leq q^{-N}$. Let $r = \operatorname{rank}(\bar{C}) - \operatorname{rank}(R\bar{C})$, so $N \geq \operatorname{rank}(\bar{C}) - r$. Then $P \leq \mathbb{E}\big[q^{-\operatorname{rank}(\bar{C})+r}\, \mathbb{1}_{r\geq 1}\big]$, since the added abort occurs exactly when $r \geq 1$, and expectation of conditional probability is marginal probability.

Now, apply Thm. 4.6 to $\bar{C}$ to get $\mathbb{E}[q^r - 1] \leq \epsilon(q^{\operatorname{rank}(\bar{C})} - 1)$. If $r \geq 1$ then $\frac{q^r}{q^r-1} \leq \frac{q}{q-1}$. Multiply both sides by $q^r - 1$ to get

$$q^r\, \mathbb{1}_{r\geq 1} \leq \frac{q}{q-1}(q^r - 1).$$

$$P \leq \mathbb{E}\big[q^{-\operatorname{rank}(\bar{C})+r}\, \mathbb{1}_{r\geq 1}\big] \leq \epsilon \frac{q}{q-1} \frac{(q^{\operatorname{rank}(\bar{C})} - 1)}{q^{\operatorname{rank}(\bar{C})}} \leq \epsilon \frac{q}{q-1}$$

2. After checking that $\operatorname{rank}(R\bar{C}) = \operatorname{rank}(\bar{C})$, find $R^{-1} \in \mathbb{F}_q^{\ell\times m}$ such that $R^{-1}R\bar{C} = \bar{C}$. To do this, find the reduced row echelon forms $F = AR\bar{C}$ and $F' = B\bar{C}$ of $R\bar{C}$ and $\bar{C}$, where $A \in \mathbb{F}_q^{m\times m}$ and $B \in \mathbb{F}_p^{\ell\times\ell}$ are invertible matrices. Because they have the same rank, $R\bar{C}$ and $\bar{C}$ must have the same row space. The uniqueness of reduced row echelon forms implies that all nonzero rows of $F$ and $F'$ will be identical, so

$$F' = \begin{bmatrix} F \\ 0 \end{bmatrix} \quad \text{and} \quad \bar{C} = B^{-1}F' = B^{-1} \begin{bmatrix} \mathbb{1}_m \\ 0 \end{bmatrix} F = B^{-1} \begin{bmatrix} \mathbb{1}_m \\ 0 \end{bmatrix} AR\bar{C},$$

   which gives a formula for $R^{-1}$.

   Correct $P_S$'s VOLE correlation as $U^\star = U - R^{-1}\bar{U}$ and $V^\star = V - R^{-1}\bar{V}$. Then, assuming that the consistency check passes,

$$\begin{aligned} W &= [U\ \bar{C}] T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) + V \\ &= \big[(U^\star + R^{-1}\bar{U})\ \ \bar{C}\big] T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) + V^\star + R^{-1}\bar{V} \\ &= U^\star G_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) + V^\star + R^{-1} \Big([\bar{U}\ \ R\bar{C}] T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) + \bar{V}\Big) \\ &= U^\star G_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) + V^\star. \end{aligned}$$

3. Let $\mathcal{W}_{\text{pre}} = \mathbb{F}_q^{m \times k_{\mathcal{C}}}$, then run $\mathsf{Precom}(\bar{C}, R, R^{-1})$ to get the pre-commitment functions $U^\star_{\text{pre}}, V^\star_{\text{pre}}, L_{\text{pre}}$ as in the simulator, as well as $\overleftarrow{\Delta}_0 \in L'$ and $L'_0 = L' - \overleftarrow{\Delta}_0$. Also, find some $\overleftarrow{L}_{\text{off}} \in -L'$ where $\bar{V} = [\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{L}_{\text{off}})$. Replace the underlying guess $\overleftarrow{\Delta} \in L'$ and the consistency check $-\bar{V} = [\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta})$ with $\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}} \in L_{\text{pre}}(\bar{U})$. When such an $\overleftarrow{L}_{\text{off}}$ exists, we need to show that this is equivalent to the consistency check. $L'_0$ is the linear subspace obtained by shifting $L'$ to go through the origin, so $\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}} \in L'_0$ if and only if $\overleftarrow{\Delta} \in L'$ because $\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}}$ is the difference of two elements of the affine subspace $L'$. When $\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}} \in L'_0$, we have that $\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}} \in L_{\text{pre}}(\bar{U})$ is equivalent to $0 = [\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}})$, which equals $[\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) + \bar{V}$. The latter being zero is the consistency check.

   We must also show that if the consistency check would pass, then a solution $\overleftarrow{L}_{\text{off}}$ must exist. Assume that there exists some $\overleftarrow{\Delta}_1 \in L'$ that would pass the consistency check, i.e. $-\bar{V} = [\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}_1)$. Then $-\overleftarrow{\Delta}_1 \in -L'$ is a valid solution for $\overleftarrow{L}_{\text{off}}$.

4. Factor out the sampling of $\Delta$, the computation of $W_{[h]\cdot} = U^\star{}_{[h]\cdot} \operatorname{diag}(\overleftarrow{\Delta}) + V^\star{}_{[h]\cdot}$, and the selective abort attack $\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}} \in L_{\text{pre}}(\bar{U})$ into the ideal functionality $\mathcal{F}^{p,q,\mathcal{C},h,\mathcal{L},M}_{\mathsf{VOLE\text{-}pre}}$. The ideal functionality also includes an abort if $U^\star \neq U^\star_{\text{pre}}(\bar{U})$ or $V^\star \neq V^\star_{\text{pre}}(\bar{U}, \overleftarrow{\Delta})$, and we must show that neither will occur. The former cannot occur because that is exactly how $U^\star$ is calculated. For the latter, when the consistency check passes we have

$$V^\star_{\text{pre}}(\bar{U}, \overleftarrow{\Delta}) = V + R^{-1}[\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} \operatorname{diag}(\overleftarrow{\Delta}) = V - R^{-1}\bar{V} = V^\star.$$

5. We are now almost at the ideal world. We just need to change $\mathcal{W}_{\text{pre}}$ to be $\{\bar{U} \in \mathbb{F}_q^{m \times k_{\mathcal{C}}} \mid t \geq \| [\bar{U} \;\; R\bar{C}]T_{\mathcal{C}}\|_0\}$, as in the simulator, and show that $|\mathcal{W}_{\text{pre}}| \leq M$. Changing $\mathcal{W}_{\text{pre}}$ is only detectable if $\bar{U} \notin \mathcal{W}_{\text{pre}}$ and the consistency check still passes. Then the adversary must guess $\| [\bar{U} \;\; R\bar{C}]T_{\mathcal{C}}\|_0 \geq t+1$ entries of $\overleftarrow{\Delta}$, which has negligible probability $q^{-t-1}$. We just need to choose $t$ to be as large as possible while keeping $M$ small.

   Finding a $\bar{U}$ such that $[\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} = \bar{U}G_{\mathcal{C}} + [0 \; R\bar{C}]T_{\mathcal{C}}$ has few nonzero columns is equivalent to a bounded distance decoding problem over $\mathbb{F}_{q^m}$. That is, interpreting each column as an element of $\mathbb{F}_{q^m}$, $\bar{U}G_{\mathcal{C}}$ must be a code word close to $-[0 \; R\bar{C}]T_{\mathcal{C}}$ in Hamming weight. The simplest choice would be to set $t$ to be the decoding radius $\lfloor \frac{d_{\mathcal{C}}-1}{2} \rfloor$ of $\mathcal{C}$, guaranteeing that there is at most a single element of $\mathcal{W}_{\text{pre}}$. To get a tighter bound, we use the Cassuto–Bruck list decoding bound [CB04], which implies $M \leq n_{\mathcal{C}}(d_{\mathcal{C}} - t)$ when $t = \frac{d_{\mathcal{C}}}{1+\sqrt{1+\frac{d_{\mathcal{C}}}{n_{\mathcal{C}}}-\frac{1}{n_{\mathcal{C}}^2}}}$. □

**Optimizations.** There are a couple ways that the communication complexity of Fig. 9 can be improved. First, if the universal hash $R$ contains a lot of entropy, a seed $s \in \{\mathsf{0}, \mathsf{1}\}^\lambda$ may be sent instead, so $R = \mathsf{PRG}(s)$. The only place the randomness of $R$ was used was to upper bound the probability that $\operatorname{rank}(R\bar{C}) < \operatorname{rank}(\bar{C})$. $\bar{C}$ cannot depend on $s$, so if using a PRG changed this probability more than negligibly then there would be an attack against the PRG.

A second optimization is to hash $\widetilde{V}$ with a local random oracle $\mathsf{Hash}$ before sending it, because all that's needed is an equality check. The simulator (in the malicious $P_S$ case) could then extract $\widetilde{V}$ from its hash, then continue as usual. Interestingly, for concrete security it would be fine even if $\mathsf{Hash}$ were just an arbitrary collision resistant hash. Looking at just $\bar{C}$ and $\widetilde{U}$, the simulator can see which entries of $\overleftarrow{\Delta}$ are being guessed, though not what the guesses are. By looping through a random subset of $2^\sigma$ possible guesses (and for the usual setting of $\sigma = 40$ this is quite feasible), $\mathcal{S}$ can find the preimage of $\mathsf{Hash}(\widetilde{V})$ often enough to only give the distinguisher an additional advantage of $2^{-\sigma}$.

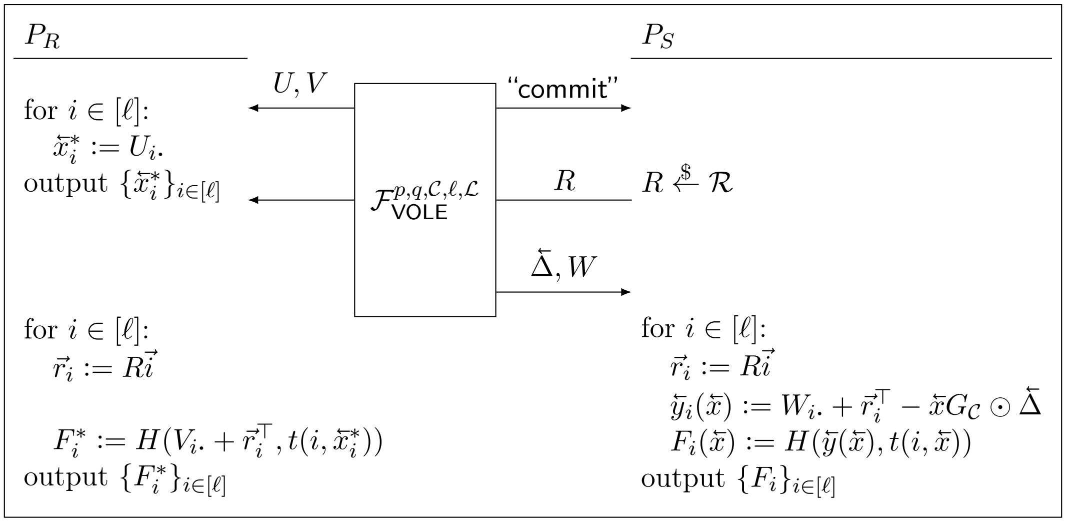

$P_R$
$P_S$
$U, V$
"commit"
for $i \in [\ell]$:
$\overleftarrow{x}_i^* := U_{i\cdot}$
output $\{\overleftarrow{x}_i^*\}_{i \in [\ell]}$
$\mathcal{F}_{\mathsf{VOLE}}^{p,q,\mathcal{C},\ell,\mathcal{L}}$
$R$
$R \xleftarrow{\$} \mathcal{R}$
$\overleftarrow{\Delta}, W$
for $i \in [\ell]$:
$\vec{r}_i := R\vec{i}$
$F_i^* := H(V_{i\cdot} + \vec{r}_i^{\top}, t(i, \overleftarrow{x}_i^*))$
output $\{F_i^*\}_{i \in [\ell]}$
for $i \in [\ell]$:
$\vec{r}_i := R\vec{i}$
$\overleftarrow{y}_i(\overleftarrow{x}) := W_{i\cdot} + \vec{r}_i^{\top} - \overleftarrow{x} G_{\mathcal{C}} \odot \overleftarrow{\Delta}$
$F_i(\overleftarrow{x}) := H(\overleftarrow{y}(\overleftarrow{x}), t(i, \overleftarrow{x}))$
output $\{F_i\}_{i \in [\ell]}$

Figure 11: $\binom{p^{k_{\mathcal{C}}}}{1}$-OT extension protocol. Note that the parties for the base VOLE are swapped, with $P_S$ (instead of $P_R$) getting $\overleftarrow{\Delta}$. If $P_S$ receives "check failed" from the VOLE then the protocol is aborted immediately. For semi-honest security, the "commit" and $R$ steps are skipped, and $\vec{r}_i := 0$.

## 5 OT Extension

Now that we have constructed subspace VOLE, it is time to go back to our original goal: OT extension. Like previous OT extensions, we hash our correlated randomness in order to get random OTs. For malicious security, our protocol (Fig. 11) follows [CT21] in using a universal hash to avoid collisions between extended OTs, avoiding the need for a TCR hash. However, a TCR hash allows for better concrete security (at they expense of performance) by reducing $\tau_{\max}$; we allow an arbitrary function $t(i, \overleftarrow{x})$ to control how many different hashes use the same tweak. Unlike [CT21], our analysis allows $R$ to be sent in parallel with the VOLE protocol, saving a round of communication.

For generality, we allow any finite field, but we expect that $p = 2$ will be most efficient in almost all cases. We equivocate between the choices $U_{i\cdot}$ in $\mathbb{F}_p^{k_{\mathcal{C}}}$ from the VOLE, and the choices $x_i^*$ in $[p^{k_{\mathcal{C}}}]$ expected for OT. This can be thought of as writing $x_i^*$ in base $p$.

**Theorem 5.1.** *The protocol in Fig. 11 achieves $\mathcal{F}_{\mathsf{OT}\text{-}1}^{p^{k_{\mathcal{C}}},\ell,\{X\}}$ with malicious security in the $\mathcal{F}_{\mathsf{VOLE}\text{-}pre}^{p,q,\mathcal{C},\ell,\mathcal{L},M}$ hybrid model, assuming that $H \colon \mathbb{F}_q^{n_{\mathcal{C}}} \times \mathcal{T} \to \{0,1\}^{\lambda}$ is a $(p, q, \mathcal{C}, \mathcal{T}, \mathcal{L})$-TCR hash, and $\mathcal{R} \subseteq \mathbb{F}_q^{n_{\mathcal{C}} \times \lceil \log_q(\ell) \rceil}$ is an $\epsilon$-almost uniform family. The distinguisher advantage is at most $\epsilon M \ell (t_{max} - 1)/2 + \mathsf{Adv}_{\mathrm{TCR}}$, where $t_{max}$ is the maximum number of distict OTs that can have the same tweak under $t$. For the TCR itself, $\tau_{max}$ will be the maximum number of evaluations $F_i(\overleftarrow{x})$ where $t(i, \overleftarrow{x})$ outputs a given tweak. For semi-honest security, $\mathcal{R}$ is unused; instead set $\epsilon = q^{-n_{\mathcal{C}}}$ and $M = 1$.*

*Proof.* See Appx. F.2. □

**The Importance of Universal Hashing.** For malicious security, it is *critical* that tweaking or some other mechanism is used to stop collisions in the input to $H$. This was noted by [GKWY20, MR19], who show that when a malicious receiver can control its own randomness $V$ (as we assume), they can force all $H$ evaluations to be equal between two different extended OTs, causing two different OTs to have the same messages. However, this depends on controlling the seeds used for the underlying IKNP OT extension. Endemic OT use this loophole, giving a protocol where $P_S$

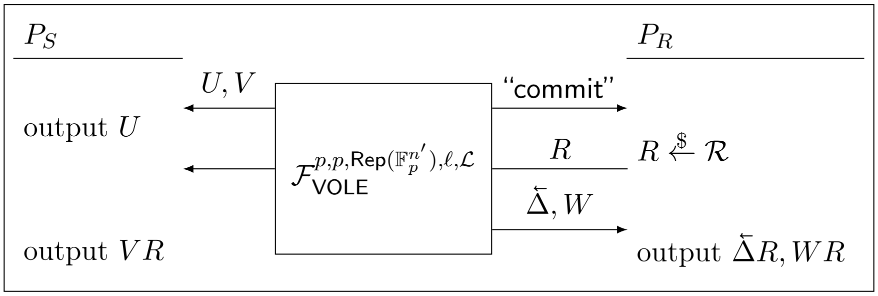

$P_S$
$P_R$
$U, V$
"commit"
output $U$
$\mathcal{F}_{\mathsf{VOLE}}^{p,p,\mathsf{Rep}(\mathbb{F}_p^{n'}),\ell,\mathcal{L}}$
$R$
$R \xleftarrow{\$} \mathcal{R}$
$\overleftarrow{\Delta}, W$
output $VR$
output $\overleftarrow{\Delta}R, WR$

Figure 12: Non-leaky maliciously secure subspace VOLE for the repetition code. If $P_R$ receives "check failed" from the VOLE then the protocol is aborted immediately.

chooses its seeds, and claim to show that it is secure to omit the tweak in this case [MR19, Sect. 5.3]. In Appx. E, we give an attack against their protocol, using only the partial control over $V$ that comes from the correction $C$. When the security parameter is $\lambda = 128$, it should have success probability a constant times $2^{-24}$ on a batch of $10^7$ $\binom{2}{1}$-OTs.

### 5.1 Δ-OT

A common variant of OT extension is Δ-OT (a.k.a. correlated OT), where all OT messages follow the pattern $m_0, m_1 = m_0 \oplus \Delta$. It is useful for authenticated secret sharing and garbled circuits. More generally, over a larger field, it works as $m_x = m_0 + x\Delta$, and is useful for encoding the inputs to arithmetic garbling [BMR16].

Δ-OT works easily as a special case of subspace VOLE where $q = p$ and $\mathcal{C} = \mathsf{Rep}(\mathbb{F}_p^n)$,[6] except which party is called the sender and which the receiver is swapped, like with OT extension. However, in the malicious setting our subspace VOLE allows a selective abort attack, and while for some applications (such as garbling) it may be allowed to leak a few bits for Δ, in others it may not. [BLN+15] solve this problem by multiplying the Δ-OT messages by a uniformly random rectangular matrix, throwing away some of the OT message. With high probability, any correlation among the bits of Δ is also lost, resulting in a non-leaky Δ-OT. In Fig. 12, we generalize this idea to use a universal hash, which can be more computationally efficient than a random matrix.

**Theorem 5.2.** *The protocol in Fig. 12 achieves $\mathcal{F}_{\mathsf{VOLE}}^{p,p,\mathsf{Rep}(\mathbb{F}_p^n),\ell,\{X\}}$ with malicious security in the $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}^{p,p,\mathsf{Rep}(\mathbb{F}_p^{n'}),\ell,\mathrm{Affine}(\mathbb{F}_p^{n'}),M}$ hybrid model, assuming that $\mathcal{R} \subseteq \mathbb{F}_p^{n' \times n}$ is a $\epsilon$-almost uniform family and $n' \geq n$. The advantage is bounded by $\epsilon M(p^n - 1)$.*

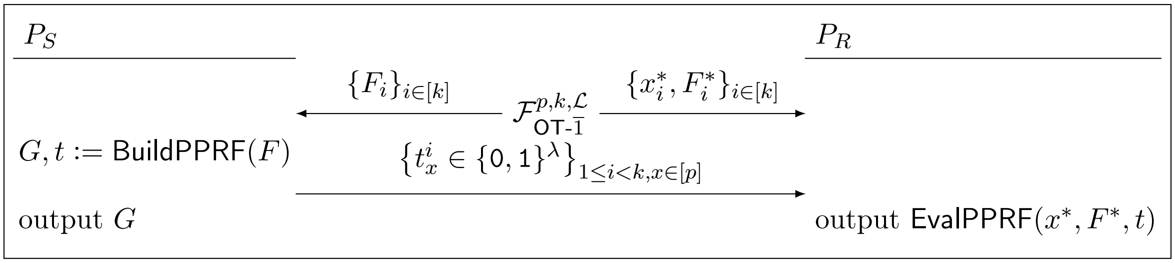
*Proof.* See Appx. F.3. □

Note that if $\mathcal{R}$ has the optimal $\epsilon = p^{-n'}$, such as when it is a uniformly random matrix, the environment's advantage is upper bounded by $Mp^{n-n'}$. Therefore, $n'$ should be set to $n + \log_p(2)\sigma$ for security.

## 6 Base OTs

Our small field VOLE (Fig. 7) is based on $\binom{q}{q-1}$-OT, yet actual base OTs are generally $\binom{2}{1}$-OT. We follow [BGI17] in using a punctured PRF to make a $\binom{2^k}{2^k-1}$-OT from $k$ $\binom{2}{1}$-OTs. Our protocol (see

[6] Note that subspace VOLE with $q = p^k$ and $\mathcal{C} = \mathsf{Rep}(\mathbb{F}_p^n)$ can easily be turned into VOLE for $q = p$ and $\mathcal{C} = \mathsf{Rep}(\mathbb{F}_p^{kn})$, by interpreting $\mathbb{F}_q$ as a vector space over $\mathbb{F}_p$.


$P_S$
$P_R$
$\{F_i\}_{i\in[k]}$
$\mathcal{F}^{p,k,\mathcal{L}}_{\mathsf{OT}\text{-}\bar{1}}$
$\{x_i^*, F_i^*\}_{i\in[k]}$
$G, t := \mathsf{BuildPPRF}(F)$
$\{t_x^i \in \{0,1\}^\lambda\}_{1\le i<k,x\in[p]}$
output $G$
output $\mathsf{EvalPPRF}(x^*, F^*, t)$

```
BuildPPRF(F):
  for $x \in [p]$:
    $s_x^1 := F_0(x)$

  for $i := 1$ to $k-1$:
    for $y \in [p^i]$, $x \in [p]$:
      $s_{py+x}^{i+1} := \mathsf{PRG}_x(s_y^i)$
    for $x \in [p]$:
      $t_x^i := F_i(x) \oplus \bigoplus_{y\in[p^i]} s_{py+x}^{i+1}$

  return $(y \mapsto s_y^k), t$
```

```
EvalPPRF(x*, F*, t):
  for $x \in [p] \setminus \{x_0^*\}$:
    $s_x^{*1} := F_0^*(x)$
  $y_1^* := x_0^*$
  for $i := 1$ to $k-1$:
    for $y \in [p^i] \setminus \{y_i^*\}$, $x \in [p]$:
      $s_{py+x}^{*i+1} := \mathsf{PRG}_x(s_y^{*i})$
    for $x \in [p] \setminus \{x_i^*\}$:
      $s_{py_i^*+x}^{*i+1} := t_x^i \oplus F_i^*(x) \oplus \bigoplus_{y\in[p^i]\setminus\{y_i^*\}} s_{py+x}^{*i+1}$

    $y_{i+1}^* := py_i^* + x_i^*$
  return $y^*, (y \mapsto s_y^{*k-1})$
```

Figure 13: Protocol for $\binom{q}{q-1}$-OT based on $\binom{p}{p-1}$-OT, using a punctured PRF.

Fig. 13) is based on the optimized version in [BCG+19a], which we generalize to make $\binom{p^k}{p^k-1}$-OT from $\binom{p}{p-1}$-OT.

It depends on a $\mathsf{PRG}\colon \{0,1\}^\lambda \to \left(\{0,1\}^\lambda\right)^p$. The $x$th block of $\lambda$ bits from this PRG is written as $\mathsf{PRG}_x(s)$. The PRG is used to create a GGM tree [GGM86]. Starting at the root of the tree, $P_R$ gets $p-1$ of the $p$ children from $\mathcal{F}^{p,k,\mathrm{Affine}(\mathbb{F}_p^k)}_{\mathsf{OT}\text{-}\bar{1}}$, and at every level down the tree the protocol maintains the property that $P_R$ knows all but one of the nodes at that level. Each level $i$ of the tree is numbered from 0 to $p^i - 1$, with the $y$th node in the layer containing the value $s_y^i$. This means that the children of node $s_y^i$ are $s_{py+x}^{i+1} = \mathsf{PRG}_x(s_y^i)$, for $x \in [p]$. $P_S$ computes the whole GGM tree in $\mathsf{BuildPPRF}$, finds the totals $\bigoplus_y s_{py+x}^{i+1}$ for each $y$, and uses the base OTs to send all but one of these totals to $P_R$. Let $y_i^*$ be the index of the node on the active path, the nodes that $P_R$ cannot learn, in layer $i$. Then $P_R$ will know every $s_y^i$, except for $s_{y_i^*}^i$, so it can compute $\bigoplus_{y\neq y_i^*} s_{py+x}^{i+1}$ and take the differences from the totals sent by $P_S$ to find $s_y^{i+1}$ for all $y \neq y^*$. It will then get $p^k - 1$ of the $p^k$ leaf nodes $s_y^k$.

**Theorem 6.1.** *Figure 13 constructs $\mathcal{F}^{q,1,\{X\}}_{\mathsf{OT}\text{-}\bar{1}}$ out of $\mathcal{F}^{p,k,\{X\}}_{\mathsf{OT}\text{-}\bar{1}}$, and is secure in the semi-honest model.*

*Proof.* See Appx. F.4. □

While the protocol only does a single $\binom{q}{q-1}$-OT from a batch of $k$ $\binom{p}{p-1}$-OTs, it should be clear that a batch of $n$ $\binom{q}{q-1}$-OT can be constructed from a batch of $nk$ $\binom{p}{p-1}$-OTs. For $p = 2$, the base $\binom{p}{p-1}$-OTs are just $\binom{2}{1}$-OTs. For $p > 2$, they can be constructed from chosen message $\binom{p}{1}$-OT, by sending just the messages $P_R$ is supposed to see.

```
$P_S$                                                              $P_R$
$\tilde{s}, \tilde{t} := \mathsf{ProvePPRF}(G)$   $\xrightarrow{\tilde{s}, \tilde{t} \in \{0,1\}^{2\lambda}}$
                                                                   abort if $\tilde{s} \neq \mathsf{VerifyPPRF}(y^*, G^*, \tilde{t})$
output $\mathsf{PRG}'_1 \circ G$                                   output $y^*, \mathsf{PRG}'_1 \circ G^*$
```

```
$\mathsf{ProvePPRF}(G)$:                                   $\mathsf{VerifyPPRF}(y^*, G^*, \tilde{t})$:
    for $y \in [q]$:                                           for $y \in [q] \setminus \{y^*\}$:
        $\tilde{s}_y := \mathsf{PRG}'_0(G(y))$                     $\tilde{s}^*_y := \mathsf{PRG}'_0(G^*(y))$
    $\tilde{t} := \bigoplus_{y \in [q]} \tilde{s}_y$           $\tilde{s}^*_{y^*} := \tilde{t} \oplus \bigoplus_{y \in [q] \setminus \{y^*\}} \tilde{s}^*_y$
    $\tilde{s} := \mathsf{Hash}(\tilde{s}_0 \parallel \cdots \parallel \tilde{s}_{q-1})$      $\tilde{s} := \mathsf{Hash}(\tilde{s}^*_0 \parallel \cdots \parallel \tilde{s}^*_{q-1})$
    return $\tilde{s}, \tilde{t}$                              return $\tilde{s}$
```

Figure 14: Consistency checking for $\binom{q}{q-1}$-OT. This makes Fig. 13 maliciously secure.

### 6.1 Consistency Checking

With Fig. 14, we also use the technique of [BCG$^+$19a] for malicious security. We prove a slightly stronger result for their consistency check by showing that the selective abort attack allowed by the check is always in $\mathrm{Affine}(\mathbb{F}_p^k)$. In fact, $P_S$ can only check guesses for the $x_i^*$s individually, not all of them together. As in [BCG$^+$19a], the protocol needs a second PRG, $\mathsf{PRG}' \colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda} \times \{0,1\}^\lambda$, which must be collision resistant in its first output $\mathsf{PRG}'_0$. The consistency check works by giving $P_R$ the total of all $\tilde{s}_y^k = \mathsf{PRG}'_0(s_y^k)$, which it will know all but one of already, so that it can reconstruct them all. $P_S$ also sends a collision resistent hash of all $\tilde{s}_y^k$, so that $P_R$ can verify that every $\tilde{s}_y^k$ it received was correct.

Additionally, to prove that the selective abort attack is always in $\mathrm{Affine}(\mathbb{F}_p^k)$, we also have to assume that PRG is collision resistent for its whole output, so there are no $s \neq s'$ such that $\mathsf{PRG}_x(s) = \mathsf{PRG}_x(s')$ for all $x \in [p]$. This is plausable for reasonable choices of PRG, and is provable in the ideal cipher model. In Appx. F.5 we use these assumptions to prove the following.

**Proposition 6.2.** *The selective abort attack allowed in Fig. 14 will always be in $\mathcal{L} = \mathrm{Affine}(\mathbb{F}_p^k)$.*

**Theorem 6.3.** *Figure 14 (composed with Fig. 13) is a maliciously secure $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{q,1,\mathrm{Affine}(\mathbb{F}_p^k)}$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{p,k,\mathrm{Affine}(\mathbb{F}_p^k)}$ hybrid model.*

## 7 Implementation

We implemented our $\binom{2}{1}$-OT semi-honest and malicious protocols[7] in the libOTe library [Rin], so that we could assess efficiency and parameter choices. We focus only on the case of binary fields ($p = 2$), as for this problem there is no benefit to using a larger $p$. First, we discuss the choices we made in instantiation.

For semi-honest security, our protocol depends on only a PRG and a TCR hash. We instantiate the CR hash using Thm. 2.7 with AES, modeled as an ideal cipher. To keep $\tau_{\max}$ low, we set

[7] Source code is at https://github.com/ldr709/softspoken-implementation.

$t(i, \vec{x}) = \lfloor i/1024 \rfloor$, changing the tweak every 1024 OTs. We also used the hash as a PRG, evaluating it as $H(s, t(0)), H(s \oplus 1, t(1)), \ldots$ for a seed $s$. This allows the same AES round keys to be used across the many different PRG seeds used by OT extension, while AES-CTR would need to store many sets of round keys — too many to fit in L1 cashe.

Malicious security additionally requires a universal hash for Fig. 9. As recommended in Sect. 4, we construct the universal hash in two stages. First, take each block of 64 bits from $\vec{x}$ and interpret it as an element of $\mathbb{F}_{2^{64}}$. These blocks become the coefficients of a polynomial over $\mathbb{F}_{2^{64}}$, which is evaluated at a random point to get $R\vec{x}$. We choose the constant term to always be zero, which makes this a uniform family (not just universal), allowing the use of Thm. 2.4 to sum multiple hashes together. Limiting each hash to $2^{20}$ blocks (each 64-bits long) before switching to the next (generated from a PRG seed) makes this a $2^{-44}$-almost uniform family. The second stage $R'$ of the universal hash further compresses the output in $\mathbb{F}_{2^k}^{64}$ down to only $\mathbb{F}_{2^k}^{\lceil 40/k \rceil}$. We made the simple choice of a uniformly random matrix in $\mathbb{F}_{2^k}^{\lceil 40/k \rceil \times 64}$, which achieves the optimal $\epsilon = 2^{-k \lceil \frac{40}{k} \rceil}$ for a uniform family of this size.

Fig. 11 needs a uniform hash, and we use multiplication over $\mathbb{F}_{2^{128}}$, multiplying each tweak by a 128-bit hash key to get a 128-bit value. Guessing the hash would require guessing this hash key, so it is a $2^{-128}$-almost uniform family.

The punctured PRF (Fig. 14) requires collision resistant primitives $\mathsf{PRG}$, $\mathsf{PRG}'$, and $\mathsf{Hash}$. For $\mathsf{PRG}$, we assume that it is hard to find $s \neq s'$ such that $H(s, 0) = H(s', 0)$ and $H(s, 1) = H(s', 1)$, which is true in the ideal cipher model. For $\mathsf{PRG}'$, which requires collision resistance for its first output on its own, we use Blake2 [ANWW13]. We also use Blake2 for $\mathsf{Hash}$.

### 7.1 Performance Comparison

In Tables 1 and 2, we present benchmarks of our implementation in both the semi-honest and malicious settings, for a variety of communication settings and parameter choices. We also compare to existing OT extensions. All results were measured on an Intel i7-7500U laptop CPU, with the sender and receiver each running on a single thread. The software was compiled with GCC 11.1 with `-O3` and link-time optimizations enabled, and executed on Linux. In the localhost setting, there is no artificial limit on the communication between these threads, though the kernel has overhead in transferring the data, which is why our $k = 2$ is faster than $k = 1$ even in this case. We simulated communicating over a LAN by applying a latency of 1 ms and a 1 Gbps bandwidth limit. For the WAN setting, this becomes 40 ms and 100 Mbps. Base OTs were generated using the EKE-based OT of [MRR21].[8] The choice bits of SoftSpokenOT were derandomized immediately, as were the choice bits for Ferret, to provide the most direct comparison with IKNP and KOS. The choice bits for the Silent OTs were not derandomized, slightly biasing the comparison in their favor.

Although for $k = 1$ our protocol is the same as IKNP in the semi-honest setting, our implementation is significantly faster. This mainly comes from a new implementation of 128 × 128 bit transposition, based on using AVX2 to implement Eklundh's algorithm [TE76]. This gave a 6× speedup for bit transposition, which is a significant factor of IKNP's overall runtime.

In our benchmark, Silver did not perform as well as IKNP in the localhost setting, while [CRR21] found that Silver was nearly 60% faster than IKNP. We attribute this difference to using a lower quality computer, which has less memory bandwidth than the machine used for their benchmark. This is important for Silver's transposed encoding, a memory intensive operation. Compared to Silent OT, we achieve better concrete performance in the localhost and LAN settings, but the

[8] Silent OT needs more than $\lambda$ base OTs, and so as an optimization it generates them using KOS, which needs only $\lambda$ base OTs.

| Protocol | Semi-honest Security | | | | | Malicious Security | | |
|---|---|---|---|---|---|---|---|---|
| | Communication | | Time (ms) | | | Time (ms) | | |
| | KB | bits/OT | localhost | LAN | WAN | localhost | LAN | WAN |
| IKNP [IKNP03] / KOS [KOS15] | 160010 | 128 | 391 | 1725 | 15525 | 443 | 1802 | 15662 |
| SoftSpoken ($k = 1$) | 160009 | 128 | 243 | 1590 | 15420 | 298 | 1637 | 15648 |
| SoftSpoken ($k = 2$) | 80009 | 64 | **210** | 815 | 7730 | **255** | 893 | 7985 |
| SoftSpoken ($k = 3$) | 53759 | 43 | 223 | 568 | 5208 | 322 | 677 | 5419 |
| SoftSpoken ($k = 4$) | 40008 | 32 | 261 | 433 | 3995 | 311 | 530 | 4114 |
| SoftSpoken ($k = 5$) | 32510 | 26 | 337 | **348** | 3271 | 454 | **465** | 3447 |
| SoftSpoken ($k = 6$) | 27509 | 22 | 471 | 488 | 2811 | 588 | 613 | 2985 |
| SoftSpoken ($k = 7$) | 23760 | 19 | 777 | 843 | 2380 | 899 | 966 | 2554 |
| SoftSpoken ($k = 8$) | 20008 | 16 | 1259 | 1314 | 1916 | 1293 | 1322 | **2130** |
| SoftSpoken ($k = 9$) | 18759 | 15 | 2302 | 2338 | 2439 | 2460 | 2457 | 2590 |
| SoftSpoken ($k = 10$) | 16259 | 13 | 3984 | 3983 | 4097 | 4126 | 4132 | 4223 |
| Ferret [YWL+20] | 2976 | 2.38 | 2156 | 2160 | 2825 | 2240 | 2242 | 3108 |
| Silent (Quasi-cyclic) [BCG+19a] | **127** | **0.10** | 7735 | 7736 | 8049 | | | |
| Silent (Silver, weight 5) [CRR21] | 127 | 0.10 | 613 | 613 | **746** | | | |

Table 1: Time and communication required to generate $10^7$ OTs, averaged over 50 runs. The best entry in each column is **bolded**, and the second best is underlined. Communication costs for maliciously secure versions are within 10 KB of the semi-honest ones. The setup costs are included.

Semi-honest Security

| Protocol | Comm. | Time (ms) | | | | | |
|---|---|---|---|---|---|---|---|
| | | localhost | | LAN | | WAN | |
| | KB | $P_R$ | $P_S$ | $P_R$ | $P_S$ | $P_R$ | $P_S$ |
| IKNP [IKNP03] | 4.2 | 27 | 19 | 32 | 21 | 94 | 54 |
| SoftSpoken ($k$ in 1–10) | 8.3–9.8 | 27–29 | 28–30 | 32–44 | 33–45 | 86–101 | 127–142 |
| Silent (Quasi-cyclic) [BCG+19a] | 53.4 | 31 | 33 | 32 | 34 | 102 | 146 |
| Silent (Silver, weight 5) [CRR21] | 53.4 | 28 | 30 | 33 | 35 | 102 | 147 |
| Ferret [YWL+20] | 1166.8 | 65 | 65 | 70 | 65 | 552 | 342 |

Malicious Security

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| KOS [KOS15] | 4.2 | 28 | 28 | 33 | 32 | 105 | 145 |
| SoftSpoken ($k$ in 1–10) | 9.3–16.8 | 27–33 | 28–34 | 32–38 | 32–38 | 100–109 | 141–151 |
| Ferret [YWL+20] | 1175.3 | 73 | 73 | 75 | 73 | 608 | 553 |

Table 2: One-time setup costs for OT protocols in Table 1. SoftSpokenOT protocols have nearly identical setup costs, and so only a range is given.

extremely low communication of Silent OT puts Silver in first place for the WAN setting. We claim another a benefit to our protocol over Silver, since SoftSpokenOT only needs fairly conservative assumptions about well-studied objects like block ciphers, while Silver depends on hardness of LPN for a novel family of codes that has yet to receive much cryptanalysis. More conservative versions of Silent OT, based on either quasi-cyclic codes [BCG+19a] or local linear codes [YWL+20], are slower than SoftSpokenOT across the tested settings.

For malicious security, we use a more efficient universal hash function compared to KOS[9], who require the additional generation of 128 bits from a PRG for every OT as part of the consistency check. We have not benchmarked maliciously secure implementations of Silent OT and Silver, but they likely have very similar performance to the semi-honest case.

[9] The implementation of KOS in libOTe has the CR hashing flaw discussed in Sect. 5.

# References

[ANWW13] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2: Simpler, smaller, fast as MD5. In *ACNS 13*, volume 7954 of *LNCS*, pages 119–135. Springer, June 2013.

[BCG$^+$19a] 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 *ACM CCS*, pages 291–308, 2019.

[BCG$^+$19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, August 2019.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996.

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 163–193. Springer, April 2017.

[BLN$^+$15] Sai Sheshank Burra, Enrique Larraia, Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, Emmanuela Orsini, Peter Scholl, and Nigel P. Smart. High performance multi-party computation for binary circuits based on oblivious transfer. Cryptology ePrint Archive, Report 2015/472, 2015. https://eprint.iacr.org/2015/472.

[BMR16] Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for Boolean and arithmetic circuits. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 565–577. ACM Press, October 2016.

[CB04] Yuval Cassuto and Jehoshua Bruck. A combinatorial bound on the list size. Technical report, California Institute of Technology, May 2004.

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, LNCS, pages 3–22. Springer, August 2015.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827, pages 502–534, Virtual Event, August 2021. Springer.

[CT21] Yu Long Chen and Stefano Tessaro. Better security-efficiency trade-offs in permutation-based two-party computation. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021*, pages 275–304, 2021.

[CW79] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.

[FLP08] Jean-Charles Faugère, Françoise Levy-dit-Vehel, and Ludovic Perret. Cryptanalysis of minrank. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 280–296. Springer, August 2008.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GKW$^+$20] Chun Guo, Jonathan Katz, Xiao Wang, Chenkai Weng, and Yu Yu. Better concrete security for half-gates garbling (in the multi-instance setting). In *CRYPTO 2020, Part II*, pages 793–822. Springer, August 2020.

[GKWY20] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *2020 IEEE Symposium on Security and Privacy*, pages 825–841, May 2020.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, August 2003.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, 1995.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, pages 54–70. Springer, August 2013.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, August 2015.

[KOS21] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. Unpublished draft of full version, March 2021.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 203–228. Springer, April 2015.

[MR19] Daniel Masny and Peter Rindal. Endemic oblivious transfer. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 309–326. ACM Press, November 2019.

[MRR21] Ian McQuoid, Mike Rosulek, and Lawrence Roy. Batching base oblivious transfers. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021*, pages 281–310, Cham, 2021. Springer International Publishing.

[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*, pages 381–396. Springer, February 2017.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, May 2020.

[PSS17] Arpita Patra, Pratik Sarkar, and Ajith Suresh. Fast actively secure OT extension for short secrets. In *NDSS 2017*. The Internet Society, 2017.

[Rin] Peter Rindal. libOTe: an efficient, portable, and easy to use Oblivious Transfer Library. `https://github.com/osu-crypto/libOTe`.

[TE76] R. E. Twogood and M. P. Ekstrom. An extension of Eklundh's matrix transposition algorithm and its application in digital image processing. *IEEE Transactions on Computers*, C-25(9):950–952, 1976.

[YWL$^+$20] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In *ACM CCS 2020*, pages 1607–1626. ACM Press, November 2020.

# A Correlation Robust Hash Constructions

**Proposition 2.6.** *A random oracle* $\mathsf{RO}\colon \mathbb{F}_q^{nc} \times \{0,1\}^t \to \{0,1\}^\lambda$ *is a* $(p, q, \mathcal{C}, \{0,1\}^t, \mathrm{Affine}(\mathbb{F}_p^{knc}))$*-TCR hash, with distinguisher advantage at most* $\tau_{max}\left(\mathfrak{q} + \frac{1}{2}\mathfrak{q}'\right)q^{-d_\mathcal{C}}$*. Here,* $\tau_{max}$ *is the maximum number of times* QUERY *is called with the same* $\tau$*,* $\mathfrak{q}$ *is the number of RO queries made by the distinguisher, and* $\mathfrak{q}'$ *is the number of calls to* QUERY*.*

*Proof.* We argue indistinguishability, going from the ideal oracle to the real oracle. But first, we must define two bad events, and bound the probability of their occurrence in the ideal world. For both events, we actually use a slightly modified ideal world where the abort in LEAK is delayed to the end. This makes no difference, as it will always give the same output (i.e. "abort") in the end. These bad events are:

1. Adversary runs queries $\mathsf{RO}(\tau, \overleftarrow{u})$ and QUERY$(\overleftarrow{x}, \overleftarrow{y}, \tau)$ such that $\overleftarrow{u} = \overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}$.
2. Adversary runs queries QUERY$(\overleftarrow{x}, \overleftarrow{y}, \tau)$ and QUERY$(\overleftarrow{x}', \overleftarrow{y}', \tau)$ such that $\overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y} = \overleftarrow{x}'G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}'$. Equivalently, $\overleftarrow{y} - \overleftarrow{y}' = (\overleftarrow{x}' - \overleftarrow{x})G_\mathcal{C} \odot \overleftarrow{\Delta}$.

For both events, an equation of the form $\overleftarrow{y} = \overleftarrow{c} \odot \overleftarrow{\Delta}$ must hold, for some non-zero code word $\overleftarrow{c} \in \mathcal{C}$. Since $\|\overleftarrow{c}\|_0 \geq d_\mathcal{C}$, any such equation has probability at most $q^{-d_\mathcal{C}}$. There are at most $\mathfrak{q}\tau_{\max}$ suitable query pairs for the first bad event, and $\mathfrak{q}'\tau_{\max}/2$ suitable pairs for the second. Therefore, the union bound shows the probability of either event occurring is at most $\tau_{\max}\left(\mathfrak{q} + \frac{1}{2}\mathfrak{q}'\right)q^{-d_\mathcal{C}}$.

In QUERY we can replace the sampling of $z \leftarrow \{0,1\}^\lambda$ with its value in the real oracle, $\mathsf{RO}(\overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}, \tau)$. Assuming that the bad events never happens, no $\mathsf{RO}$ query in QUERY will have the same inputs as any other $\mathsf{RO}$ query, from either the adversary (Event 1) or another call to QUERY (Event 2). Therefore this change is indistinguishable. We are now at the real world. □

**Proposition 2.7.** *Let* $\mathsf{Enc}\colon \{0,1\}^t \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ *be an ideal cipher, and* $\iota\colon \mathbb{F}_q^{nc} \to \{0,1\}^\lambda$ *be an injection. Then* $H(\overleftarrow{y}, \tau) = \mathsf{Enc}(\tau, \iota(\overleftarrow{y})) \oplus \iota(\overleftarrow{y})$ *is a* $(p, q, \mathcal{C}, \{0,1\}^t, \mathrm{Affine}(\mathbb{F}_p^{knc}))$*-TCR hash. The distinguisher's advantage is at most* $\tau_{max}\left((2\mathfrak{q} + \frac{1}{2}\mathfrak{q}')q^{-d_\mathcal{C}} + \frac{1}{2}\mathfrak{q}'2^{-\lambda}\right)$*, with* $\mathfrak{q}$ *and* $\mathfrak{q}'$ *as in Thm. 2.6.*

*Proof.* We start by defining four bad events, and prove an upper bound on the probability of their occurrence when the distinguisher is given access to the oracle $\mathsf{TCR\text{-}ideal}^{H,p,q,\mathcal{C},\mathcal{L}}$. As with the $\mathsf{RO}$-based TCR, we modify the ideal world slightly so as to ignore the aborts when bounding the bad events. The first two bad event are essentially the same as for the $\mathsf{RO}$-based TCR hash, while the others come from $\mathsf{Enc}$ having an inverse. Note that if the adversary queries $\mathsf{Enc}$, it is also counted as a $\mathsf{Enc}^{-1}$ query for the purposes of the bad events — all that matters for these events is the relation between $\mathsf{Enc}$ inputs and outputs.

1. Adversary queries $\mathsf{Enc}(\tau, u)$ and QUERY$(\overleftarrow{x}, \overleftarrow{y}, \tau)$ such that $u = \iota(\overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y})$. Equivalently, $\iota^{-1}(u)$ exists and equals $\overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}$.
2. Adversary queries QUERY$(\overleftarrow{x}, \overleftarrow{y}, \tau)$ and QUERY$(\overleftarrow{x}', \overleftarrow{y}', \tau)$ such that $\iota(\overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}) = \iota(\overleftarrow{x}'G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}')$. Equivalently, $\overleftarrow{y} - \overleftarrow{y}' = (\overleftarrow{x}' - \overleftarrow{x})G_\mathcal{C} \odot \overleftarrow{\Delta}$.
3. Adversary queries $\mathsf{Enc}^{-1}(\tau, v)$ and $z =$ QUERY$(\overleftarrow{x}, \overleftarrow{y}, \tau)$ such that $v \oplus z = \iota(\overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y})$. Equivalently, $\iota^{-1}(v \oplus z)$ exists and equals $\overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}$.
4. Adversary queries $z =$ QUERY$(\overleftarrow{x}, \overleftarrow{y}, \tau)$ and $z' =$ QUERY$(\overleftarrow{x}', \overleftarrow{y}', \tau)$ such that $z \oplus z' = \iota(\overleftarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}) \oplus \iota(\overleftarrow{x}'G_\mathcal{C} \odot \overleftarrow{\Delta} + \overleftarrow{y}')$.

Events 1–3 all require $\overleftarrow{c} \odot \overleftarrow{\Delta}$ to take a specific value, for some nonzero codeword $\overleftarrow{c} \in \mathcal{C}$. Each has probability at most $q^{-d_\mathcal{C}}$, for any pair of queries. Event 4 instead requires that $z \oplus z'$ take a particular value, when either $z$ or $z'$ will be a freshly random $\lambda$-bit string. Therefore, it has probability at most $2^{-\lambda}$, for any pair of queries. There are at most $\mathfrak{q}\tau_{\max}$ suitable query pairs for Events 1 and 3, and at most $\mathfrak{q}'\tau_{\max}/2$ suitable pairs for Events 2 and 4. Therefore, a union bound shows that probability of any bad event occurring is it most $\tau_{\max}\left((2\mathfrak{q} + \frac{1}{2}\mathfrak{q}')q^{-d_{\mathcal{C}}} + \frac{1}{2}\mathfrak{q}'2^{-\lambda}\right)$.

Next, we argue indistinguishability. Replacing the ideal oracle with the real oracle replaces the random value $z \leftarrow \{0,1\}^\lambda$ with $z = \mathsf{Enc}(\tau, u) \oplus u$, where $u = \iota(\overleftarrow{x} G_{\mathcal{C}} \odot \overleftarrow{\Delta} + \overleftarrow{y})$. The $\mathsf{Enc}$ call will always return fresh randomness, which will never be revealed again, because it cannot overlap with any $\mathsf{Enc}$ call (Event 1) or other call to QUERY (Event 2). Past $\mathsf{Enc}$ calls also rule out using those same value again, so $z \oplus u$ cannot be the output of another $\mathsf{Enc}$ call, nor can it equal $z' \oplus u'$ for another call to QUERY. But these are exactly what is ruled out by Event 3 and Event 4, respectively. Therefore, once the bad events have been excluded, the real and ideal worlds oracles identically. □

# B OOS Details

This section is written in using the notation of OOS. Please review that paper to familiarize yourself with the notation.

In their proof for security against a malicious receiver, OOS define a set $E \subseteq [n_{\mathcal{C}}]$ of indices of the sender's secret $b$. Passing the consistency check requires that the receiver guess some bits of $b$; $E$ is the set of these bits. $E$ depends on the corrections $\mathbf{u}^j$ sent by the receiver, and it also depends on the consistency check choice bits, which would be computed by an honest receiver as $\mathbf{w}^{(\ell)} = \sum_{i\in[m]} \mathbf{w}_i x_i^{(\ell)} + \mathbf{w}_{m+\ell}$. For example, assume that OOS is used for $\binom{2}{1}$-OT by setting $\mathcal{C}$ to be the repetition code, and that the receiver lies in its correction for a single one of the extended OTs by providing the first half of the correction as if its choice bit was zero, and the second half as if it were one. During the consistency check it could compute the $\mathbf{w}^{(\ell)}$ as if its choice bit were 0, or as if the choice bit were 1. In the former case, $E$ would be the index range $\left[\frac{n_{\mathcal{C}}}{2}+1, n_{\mathcal{C}}\right]$, while in the latter it would be $\left[1, \frac{n_{\mathcal{C}}}{2}\right]$.

The flaw in OOS's proof is in the proof of Proposition 2, which states that the simulator can extract the receiver's choice bits $\mathbf{w}_i$ from the consistency check, such that the encoded choice bits $\mathcal{C}(\mathbf{w}_i)$ agree with correction the receiver sent, except at the indices in $E$. The proposition is quoted below.

> **Proposition 2.** If the check passes then with probability at least $1 - 2^{-s} - 2^{-d_{\mathcal{C}}}$, $\mathcal{S}$ can extract values $\mathbf{w}_i \in \mathbb{F}_2^{k_{\mathcal{C}}}, \mathbf{e}_i \in \mathbb{F}_2^{n_{\mathcal{C}}}$, for $i \in [m]$, such that
>
> 1. $\mathbf{c}_i = \mathcal{C}(\mathbf{w}_i) + \mathbf{e}_i$
> 2. $\mathbf{e}_i[j] = 0$ for all $j \notin E$

Its proof is based on the following lemma.

> **Lemma 1.** Let $\mathcal{C}$ be a linear code of length $n_{\mathcal{C}}$, $m' = m+s$ be an integer and $\mathbf{c}_i \in \mathbb{F}_2^{n_{\mathcal{C}}}$, for $i \in [m']$, such that there exists at least one $j \in [m]$ with $\mathbf{c}_j \notin \mathcal{C}$. Then, if $x_i^{(\ell)} \xleftarrow{\$} \mathbb{F}_2$, we have that:
>
> $$\Pr\Big(\forall \ell \in [s], \sum_{i\in[m]} \mathbf{c}_i \cdot x_i^{(\ell)} + \mathbf{c}_{m+\ell} \in \mathcal{C}\Big) \leq 2^{-s}.$$
>
> *Proof.* . . . □

This seems reasonable enough. The span of the $\mathbf{c}_i$ is not contained in $\mathcal{C}$, so a collection of random vectors in the span are unlikely to all be in $\mathcal{C}$. Notice that this assumes that $\mathcal{C}$ and the $\mathbf{c}_i$ are known *before* the $x_i^{(\ell)}$ are sampled, which matches the lemma statement. Now let's see how this lemma is used:

> ... For a vector $\mathbf{c} \in \mathbb{F}_2^{n_\mathcal{C}}$, define $\mathbf{c}_{-E}$ to be the vector obtained by removing the positions $j \in E$. Let $\mathcal{C}_{-E}$ be the *punctured code* consisting of the codewords $\{\mathbf{c}_{-E} : \mathbf{c} \in \mathcal{C}\}$.
>
> By definition of $E$, we must have $\overline{\mathbf{c}}_{-E}^{(\ell)} = 0$, and because $\mathbf{c}_*^{(\ell)} = \overline{\mathbf{c}}^{(\ell)} + \mathcal{C}(w_*^{(\ell)})$, we also have $(\mathbf{c}_*^{(\ell)})_{-E} \in \mathcal{C}_{-E}$, for every $\ell \in [s]$. Therefore, by applying Lemma 1 with the code $\mathcal{C}_{-E}$ and vectors $(\mathbf{c}_1)_{-E}, \ldots, (\mathbf{c}_{m'})_{-E}$, it holds that for every $i \in [m]$, $(\mathbf{c}_i)_{-E} \in \mathcal{C}_{-E}$, except with probability $\leq 2^{-s}$. ...

Note that the code given to Lemma 1 is not the code $\mathcal{C}$ that is fixed in advance, it is the punctured code $\mathcal{C}_{-E}$, which depends on $E$. Additionally, the punctured vectors $(\mathbf{c}_1)_{-E}, \ldots, (\mathbf{c}_{m'})_{-E}$ also depend on $E$. Here is the problem: $E$ *depends* on the consistency check choice bits $\mathbf{w}^{(\ell)}$ that the receiver sent, *after* receiving the challenge bits $x_i^{(\ell)} \xleftarrow{\$} \mathbb{F}_2$. That is, $E$ can depend on $x_i^{(\ell)} \xleftarrow{\$} \mathbb{F}_2$, so Lemma 1 cannot be applied here.

# C PSS Details

At a high level, PSS's consistency check seems the same as OOS's specialized to Walsh–Hadamard codes $\mathcal{C}_{\mathtt{WH}}^{\kappa}$, which have length $\kappa$ and minimum distance $\kappa/2$. Random bits $w_i^{(l)}$ are sampled for $l \in [\mu]$ and $i \in [m+\mu]$. They choose linear combinations of the OT correlations for a consistency check:

$$\mathbf{a}^{(l)} = \bigoplus_{i=1}^{m+\mu} w_i^{(l)} \mathbf{a}_i \qquad \mathbf{b}^{(l)} = \bigoplus_{i=1}^{m+\mu} w_i^{(l)} \mathbf{b}_i \qquad \mathbf{e}^{(l)} = \bigoplus_{i=1}^{m+\mu} w_i^{(l)} \mathbf{e}_i,$$

where $\mathbf{a}^{(l)}$ is computed by the sender, while $\mathbf{b}^{(l)}$ and $\mathbf{e}^{(l)}$ are computed by the receiver. If the consistency check were $\mathbf{a}^{(l)} = \mathbf{b}^{(l)} \oplus (\mathbf{s} \odot \mathbf{e}^{(l)})$ for all $l$, with the $\mathbf{e}^{(l)}$ required to be code words in $\mathcal{C}_{\mathtt{WH}}^{\kappa}$, then it would work the same as OOS. However, PSS attempt to save communication by only checking the XOR of all bits in each $\mathbf{a}^{(l)}$. That is, the PSS consistency check is

$$\bigoplus_{j=1}^{\kappa} a_j^{(l)} = \bigoplus_{j=1}^{\kappa} b_j^{(l)} \oplus \bigoplus_{j=1}^{\kappa} s_j e_j^{(l)},$$

so that $b^{(l)} = \bigoplus_{j=1}^{\kappa} b_j^{(l)}$ can be sent instead of $\mathbf{b}^{(l)}$. Unfortunately, a malicious receiver can take advantage by making guesses on XORs of several bits from $\mathbf{s}$, rather than having to guess each bit individually.

Have the receiver pick $\mathbf{e}_i$ to have an interval of 1 bits, and the rest be zeros. That is, when $i \leq \lceil \frac{\kappa}{N} \rceil$, set $e_i^j = 1$ if $N(i-1) < j \leq Ni$, and 0 otherwise, where the interval width $N$ is a parameter of the attack. The remainder, where $i > \lceil \frac{\kappa}{N} \rceil$, are all set to be zero. For the consistency check, send all zeros for $\mathbf{e}^{(l)}$, and send the honest value for $b^{(l)}$. The sender's values $\mathbf{a}_i$ equal $\mathbf{b}_i \oplus (\mathbf{s} \odot \mathbf{e}_i)$, so the consistency check becomes

$$\begin{aligned}\bigoplus_{j=1}^{\kappa}\bigoplus_{i=1}^{m+\mu} w_i^{(l)} a_i^j &= \bigoplus_{j=1}^{\kappa}\bigoplus_{i=1}^{m+\mu} w_i^{(l)} b_i^j \\ 0 &= \bigoplus_{j=1}^{\kappa}\bigoplus_{i=1}^{m+\mu} w_i^{(l)} s^j e_i^j \\ 0 &= \bigoplus_{i=1}^{\lceil \kappa/N \rceil} w_i^{(l)} \bigoplus_{j=N(i-1)+1}^{Ni} s^j.\end{aligned}$$

Therefore, the consistency check passes if the XORs of intervals $\bigoplus_{j=N(i-1)+1}^{Ni} s^j$ are all zero. There are $\lceil \frac{\kappa}{N} \rceil$ of these intervals, so this has probability $2^{-\lceil \kappa/N \rceil}$.

If the check passes, we can now break the OT extension with only $\lceil \frac{\kappa}{N} \rceil 2^{N-1}$ hash evaluations. That is, if the sender sends the all zeros message for each of its first $\lceil \frac{\kappa}{N} \rceil$ OT results, the receiver can solve for $\mathbf{s}$ and learn every other OT message. The receiver can do this by using the hash to check guesses of $\mathbf{a}_i$. Since $\mathbf{a}_i = \mathbf{b}_i \oplus (\mathbf{s} \odot \mathbf{e}_i)$ depends on only $N$ bits of $\mathbf{s}$, and the XOR of these bits is known to be zero, there are only $2^{N-1}$ possibilities to check. Repeating this for all $i \le \lceil \frac{\kappa}{N} \rceil$ recovers $\mathbf{s}$.

**Proof Flaws.** How was this attack missed by PSS's proof? There are two major issues in the proof that the attack exploits. First, PSS's Lemma IV.4 proof implicitly assumes that there will not be a linear dependency between different bits of the consistency check, and our attack causes such a linear dependency by forcing the consistency check to only depend on $\mathbf{s}$ through $\bigoplus_{j=N(i-1)+1}^{Ni} s^j$. Second, while proving that their first and second hybrids are indistinguishable they say that the sender's mask for $x_{i,j}$ is $H(i, \mathbf{b}_i \oplus (\mathbf{s} \odot (\mathbf{c}_{r_i} \oplus \mathbf{c}_j)))$, when it is really $H(i, \mathbf{a}_i \oplus (\mathbf{s} \odot \mathbf{c}_j))$. These are only equal when the receiver behaves honestly.

# D KOS Details

The notation of KOS is used for this section, so please review that paper if you need to.

The most important part of KOS's proof of security against a malicious receiver is the behavior of their consistency check. They give that information in the following lemma.

> **Lemma 1.** Let $S_\Delta \subseteq \mathbb{F}_2^\lambda$ be the set of all $\Delta$ for which the correlation check passes, given the view of the receiver. Except with probability $2^{-\lambda}$, there exists $k \in \mathbb{N}$ such that
>
> 1. $|S_\Delta| = 2^k$
> 2. For every $\mathbf{s} \in \{\mathbf{x}^i\}_{i \in [\lambda]}$, let $H_\mathbf{s} = \{i \in [\lambda] \mid \mathbf{s} = \mathbf{x}^i\}$. Then one of the following holds:
>    - For all $i \in H_\mathbf{s}$ and any $\Delta^{(1)}, \Delta^{(2)} \in S_\Delta$, $\Delta_i^{(1)} = \Delta_i^{(2)}$.
>    - $k \le |H_\mathbf{s}|$, and $|\{\Delta_{H_\mathbf{s}}\}_{\Delta \in S_\Delta}| = 2^k$, where $\Delta_{H_\mathbf{s}}$ denotes the vector consisting of the bits $\{\Delta_i\}_{i \in H_\mathbf{s}}$. In other words, $S_\Delta$ restricted to the bits corresponding to $H_\mathbf{s}$ has entropy at least $k$.
>
>    Furthermore, there exists $\hat{\mathbf{s}}$ such that $k \le |H_{\hat{\mathbf{s}}}|$.

*Proof.* See full version.

As of writing, no full version has been made public. However, the authors of KOS were kind enough to provide an unpublished draft of the full version [KOS21]. It contains an attempted proof of Lemma 1. They first observe that Lemma 1 is trivial if $S_\Delta \leq 2$, then define $\Delta^1, \Delta^2, \Delta^3$ to be three distinct elements of $S_\Delta$. They let $\Delta' = \Delta^1 + \Delta^2$ and $\Delta'' = \Delta^1 + \Delta^3$, then derive the following equation from the consistency check.

$$0 = \sum_{j=1}^{\ell} \chi_j \cdot \underbrace{((\mathbf{x}_j * \Delta') \cdot \Delta'' - (\mathbf{x}_j * \Delta'') \cdot \Delta')}_{\tilde{x}_j}.$$

> Since $\{\chi_j\}_{j\in[\ell]}$ are independently random, the equality holds with probability $2^{-\lambda}$ if not all $\{\tilde{x}_j\}_{j\in[\ell]}$ are 0 by the principle of deferred decisions. Hence, we assume that $\tilde{x}_j = 0$ for all $j \in [\ell]$.

The logic is to delay the sampling of $\chi_j$ until this sum is calculated, at which point it's clear that the output will be uniformly random in $\mathbb{F}_{2^\lambda}$ unless every $\tilde{x}_j$ is zero. However, we cannot actually delay the sampling of $\chi_j$ until then, because they are used earlier. Similarly to the problem with $E$ in OOS, $S_\Delta$ depends on which consistency check message $x$ the receiver sends. In an extreme case, if the receiver behaves entirely honestly, then $S_\Delta$ will be all of $\mathbb{F}_2^\lambda$, but if after looking at the $\chi_j$ they decide to send a different $x$ instead, then they will have to guess all of $\Delta$ and so $|S_\Delta|$ will be 1. As differences between arbitrarily selected members of $S_\Delta$, $\Delta'$ and $\Delta''$ also depend on the $\chi_j$, making it impossible to delay the sampling.

### D.1 Collision Attack.

Unlike OOS where we have not managed to find a counterexample to their stated Proposition 2, we have managed to find some (impractical) attacks that break KOS's Lemma 1. The simplest succeeds with probability roughly $\lambda^2 2^{-\lambda-1}$, based on getting a collision in a set of $\lambda$ vectors of length $\lambda$. It works by generating $\mathbf{x}_1, \ldots, \mathbf{x}_{\ell'}$ as uniformly random elements of $\mathbb{F}_2^\lambda$, instead of the monochrome vectors that an honest receiver would generate.

Similarly to how an honest receiver would compute its consistency check message as $x = \sum_j x_j \cdot \chi_j$, for each column $i$ let $\bar{x}_i = \sum_j x_j^i \cdot \chi_j$. For an honest receiver every $\mathbf{x}^i$ is the same, so every $\bar{x}_i$ will be exactly $x$. Similarly, instead of just computing $t = \sum_j \mathbf{t}_j \cdot \chi_j$ and $q = \sum_j \mathbf{q}_j \cdot \chi_j$, find $\bar{t}_i = \sum_j t_j^i \cdot \chi_j$ and $\bar{q}_i = \sum_j q_j^i \cdot \chi_j$. Since $q_j^i = t_j^i + x_j^i \cdot \Delta_i$, we have $\bar{q}_i = \bar{t}_i + \bar{x}_i \cdot \Delta_i$.

Let $\alpha$ be the generator of $\mathbb{F}_{2^\lambda}$ that is being used by the protocol to represent elements of $\mathbb{F}_2^\lambda$ as field elements in $\mathbb{F}_{2^\lambda}$. That is, a vector $\mathbf{v}$ becomes the field element $v_1 + \alpha \cdot v_2 + \cdots + \alpha^{\lambda-1} \cdot v_\lambda$. Then $t = \bar{t}_1 + \alpha \cdot \bar{t}_2 + \cdots + \alpha^{\lambda-1} \cdot \bar{t}_\lambda$, and similarly for $q$. Assume that the receiver gives the honest value for $t$.[10] Then the consistency check becomes

$$t = q + x \cdot \Delta = \sum_i \bar{q}_i \cdot \alpha^{i-1} + x \cdot \Delta = \sum_i \bar{t}_i \cdot \alpha^{i-1} + \sum_i \bar{x}_i \cdot \alpha^{i-1} \cdot \Delta_i + x \cdot \Delta$$

$$0 = \sum_i \bar{x}_i \cdot \alpha^{i-1} \cdot \Delta_i + x \cdot \Delta = \sum_i (\bar{x}_i + x) \cdot \alpha^{i-1} \cdot \Delta_i.$$

Therefore, if the receiver sets $x = \bar{x}_i$ for some $i$ then they won't have to guess $\Delta_i$. They will only

[10] There seems to be no advantage for the receiver to ever lie about $t$.

have to guess all $\Delta_{i'}$ for which $x \neq \bar{x}_{i'}$.

If there is a collision, so there are $i \neq i'$ such that $\bar{x}_i = \bar{x}_{i'}$, setting $x = \bar{x}_i$ forces $|S_\Delta| \geq 4$ and so $k \geq 2$. The $\bar{x}_i$ will all be uniformly random elements of $\mathbb{F}_{2^\lambda}$, so this happens with probability roughly $\lambda^2 2^{-\lambda-1}$. For sufficiently large $\ell$ each $|H_{\mathbf{x^i}}|$ will be 1, as the columns $\mathbf{x}^i$ of the matrix whose rows are $\mathbf{x}_i$ will be unique. Therefore, this attack contradicts Lemma 1.

### D.2 Subfield Attack.

At least for a special case, a slightly more practical attack seems to be possible. Assume that $\lambda$ is divisible by 20, and that the minimal polynomial of $\alpha$ (the element being used to represent $\mathbb{F}_{2^\lambda}$) can be written in the form $P(\alpha^5)$ for some irreducible polynomial $P$.[11] Then $\mathbb{F}_{2^\lambda}$ has a subfield $\mathbb{F}_{2^{\lambda/5}}$ that is generated by $\alpha^5$. In a subfield attack, a malicious receiver arranges the consistency check so that for any $\Delta \in S_\Delta$ (i.e. any $\Delta$ that allows the consistency check to pass) and for any $s \in \mathbb{F}_{2^{\lambda/5}}$, $s\Delta$ is also in $S_\Delta$. This makes $S_\Delta$ be a vector space over $\mathbb{F}_{2^{\lambda/5}}$. The receiver tries to make the dimension of $S_\Delta$ over $\mathbb{F}_{2^{\lambda/5}}$ as high as possible, which seems to be 2 according to a heuristic analysis. This gives an attack that passes the consistency check with probability $2^{-\frac{3}{5}\lambda}$, and can then recover $\Delta$ using $q = 5 \cdot 2^{\lambda/5}$ queries to the random oracle, contradicting the stated bound at the end of the proof of KOS's Theorem 1, which would only allow a success probability of $O(q2^{-\lambda}) = O(2^{-\frac{4}{5}\lambda})$.

The importance of $\mathbb{F}_{2^{\lambda/5}}$ being generated by $\alpha^5$ is that for any $u \in \mathbb{F}_{2^{\lambda/5}}$ and $y \in \mathbb{F}_{2^\lambda}$, we have $u * v \in \mathbb{F}_{2^{\lambda/5}}$. The proof is that for $u$ to be in the subfield the only nonzero entries in it must be at indices of the form $5i + 1$, so that they correspond to powers of $\alpha^5$. Then $u * v$ will be zero wherever $u$ is zero, and so will also be in the subfield.

To make use of this fact, let the malicious receiver pick the OT corrections to make $x^i_j$ be 1 when $5 \mid i - j$ and 0 otherwise. Also assume that the malicious receiver sends the honest value of $t$ in its consistency check. Let $\delta_j = (\mathbf{x}_j * \Delta) \cdot \alpha^{1-j} = \sum_{i=0}^{(\lambda/5)-1} \alpha^{5i} \cdot \Delta_{5i+j} \in \mathbb{F}_{2^{\lambda/5}}$ for $1 \leq j \leq 5$, and let $\vec{\delta} = [\delta_1 \ \cdots \ \delta_5]^\top$. Also let $\vec{\alpha} = [1 \ \alpha \ \cdots \ \alpha^4]^\top$, so that $\vec{\alpha}^\top \vec{\delta} = \Delta$. Let $\chi'_j = \alpha^{j-1} \sum_i \chi_{5i+j}$. Write $x = \sum_{j=1}^5 x'_j \cdot \alpha^{j-1}$ and $\chi'_j = \sum_{i=1}^5 \chi'_{ji} \cdot \alpha^{i-1} = \vec{\alpha}^\top \vec{\chi}'_j$ for $x'_j, \chi'_{ji} \in \mathbb{F}_{2^{\lambda/5}}$ and $\vec{\chi}'_j = [\chi'_{j1} \ \cdots \ \chi'_{j5}]^\top$. The consistency check then becomes

$$\begin{aligned}
0 &= t + q + x \cdot \Delta \\
&= \sum_j (\mathbf{t}_j + \mathbf{q}_j) \cdot \chi_j + x \cdot \Delta \\
&= \sum_j (\mathbf{x}_j * \Delta) \cdot \chi_j + x \cdot \Delta \\
&= \sum_{j=1}^5 \sum_i \delta_j \cdot \alpha^{j-1} \cdot \chi_{5i+j} + \sum_{j=1}^5 x'_j \cdot \alpha^{j-1} \cdot \vec{\alpha}^\top \vec{\delta} \\
&= \sum_{j=1}^5 \delta_j \cdot \chi'_j + \sum_{j=1}^5 x'_j \cdot \sum_{k=1}^5 \alpha^{j+k-2} \cdot \delta_k \\
&= \vec{\alpha}^\top \left( \mathbf{X} + \sum_{j=1}^5 x'_j \cdot \mathbf{A}_j \right) \vec{\delta}, \qquad (2)
\end{aligned}$$

[11] For any $\lambda$ divisible by 20, a possible $P$ may be found by picking a element of $\mathbb{F}_{2^{\lambda/5}}$ that is not a perfect power of 5, then setting $P$ to be its minimal polynomial. Such an element must exist because $\left|\mathbb{F}^*_{2^{\lambda/5}}\right| = 2^{\lambda/5} - 1 = 16^{\lambda/20} - 1 \equiv 0 \pmod{5}$.

where $\mathbf{X} = [\vec{\chi}'_1 \ \cdots \vec{\chi}'_5]$ and the matrices $A_j$ are:

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & \alpha^5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad A_3 = \begin{bmatrix} 0 & 0 & 0 & \alpha^5 & 0 \\ 0 & 0 & 0 & 0 & \alpha^5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 0 & 0 & \alpha^5 & 0 & 0 \\ 0 & 0 & 0 & \alpha^5 & 0 \\ 0 & 0 & 0 & 0 & \alpha^5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad A_5 = \begin{bmatrix} 0 & \alpha^5 & 0 & 0 & 0 \\ 0 & 0 & \alpha^5 & 0 & 0 \\ 0 & 0 & 0 & \alpha^5 & 0 \\ 0 & 0 & 0 & 0 & \alpha^5 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Notice that in Eq. (2) both the matrix in parenthesis and the $\vec{\delta}$ on the right are in the subfield $\mathbb{F}_{2^{\lambda/5}}$. Since $\alpha$ generates $\mathbb{F}_{2^\lambda}$, which is a degree 5 extension of $\mathbb{F}_{2^{\lambda/5}}$, the only way for the equation to hold is if

$$\left(\mathbf{X} + \sum_{j=1}^{5} x'_j \cdot \mathbf{A}_j\right)\vec{\delta} = 0.$$

$\vec{\delta}$ will be uniformly random because $\Delta$ is, so to maximize the chance of the consistency check succeeding the receiver should minimize the rank of $\mathbf{X} + \sum_{j=1}^{5} x'_j \cdot \mathbf{A}_j$.[12] The question is, how low can it go?

We believe that for most possible $\mathbf{X}$ (which is uniformly random), the minimum rank will be 3. Unfortunately, we only have a heuristic justification for this. The number of rank $\leq 3$ matrices over $\mathbb{F}_{2^{\lambda/5}}$ is at least $2^{\frac{21}{5}\lambda}$, because any matrix of the form $\begin{bmatrix} \mathbb{1}_3 \\ Y \end{bmatrix} Z$ has rank $\leq 3$ for $Y \in \mathbb{F}_{2^{\lambda/5}}^{2\times 3}$ and $Z \in \mathbb{F}_{2^{\lambda/5}}^{3\times 5}$, and this representation is unique. Therefore, a uniformly random matrix will have rank $\leq 3$ with probability at least $2^{\frac{21}{5}\lambda} 2^{-\frac{25}{5}\lambda} = 2^{-\frac{4}{5}\lambda}$. When $\mathbf{X}$ and all the $x'_j$ are chosen uniformly at random then $\mathbf{X} + \sum_{j=1}^{5} x'_j \cdot \mathbf{A}_j$ will be uniformly random. The solver gets to choose all of the $x'_j$ for a total of $2^\lambda$ possibilities, making the expected number of solutions be at least $2^\lambda 2^{-\frac{4}{5}\lambda} = 2^{\lambda/5}$, so if the rank $\leq 3$ matrices are relatively evenly spread over all possible $\mathbf{X}$ then a solution is very likely to exist.

Assuming that a rank 3 solution exists, the receiver should send the corresponding consistency check message $x$. Then $|S_\Delta| = 2^k$ for $k = \frac{2}{5}\lambda$, yet every $H_\mathbf{s}$ has size $\frac{1}{5}\lambda$ because $H_{\mathbf{x}_j} = \{j, 5 + j, \cdots, \lambda - 5 + j\}$. This contradicts Lemma 1.

This also gives an attack on the real KOS protocol. The consistency check will succeed with probability $2^{-\frac{3}{5}\lambda}$. If it does and if the receiver gets to see both of the sender's output OT messages for the first five OTs, then it can do a brute force attack to recover $\Delta$. Each message was protected using only $\frac{1}{5}\lambda$ bits of $\Delta$ because the Hamming weight of each $\mathbf{x}_j$ is $\frac{1}{5}\lambda$. Using $2^{\frac{1}{5}\lambda}$ queries to the random oracle, these bits can be brute forced to learn a part of $\Delta$. Doing this five times reveals all of $\Delta$, and then the receiver can read every OT message.

[12] See [FLP08] for an algorithm to solve the MinRank problem that runs in $O(\lambda)$ time for constant size matrices. It should be fast in practice for this small problem size.

# E Endemic OT Details

In this section, we describe our attack against the OT Extension Protocol With an Ideal Cipher from Endemic OT [MR19, Sect. 5.3]. The flaw results from assuming a stronger guarantee from consistency checking than protocols like KOS or OOS provide. When describing their protocol in Figure 9, they state:

> $\mathsf{R}$ proves in zero knowledge that
>
> $$\forall i \in [m], \exists w \in \mathbb{F}_2^{k_\mathcal{C}} \mid 0 = b \odot (\mathbf{u}_i + \mathbf{t}_i + \mathbf{t}_{1,i} + \mathcal{C}(w))$$
>
> Note: $\mathbf{b} \in \mathbb{F}_2^{k_\mathcal{C}}$ is distributed uniformly in the view of $\mathbf{R}$.
> For example, the proof of KOS for $N = 2$ or OOS otherwise.

This is much stronger that what consistency checking protocols provide. At best, they merely prove that $b \odot (\mathbf{u}_i + \mathbf{t}_i + \mathbf{t}_{1,i} + \mathcal{C}(w))$ was successfully guessed by $\mathsf{R}$.

This opens an avenue for attack. Let $\mathsf{R}$ be corrupted, and behave honestly until $T_0, T_1$ are known. Find the closest pair of rows $\mathbf{t}_i, \mathbf{t}_j$ of $T_0$ in hamming distance (with $i < j$), so $D = \|\mathbf{t}_i + \mathbf{t}_j\|_0$ is as small as possible.[13] Then set $\mathbf{c}_j = \mathbf{t}_i \oplus \mathbf{t}_j$, and all other $\mathbf{c}_k = 0$. Later, for the consistency check, let $w = 0$ for each OT. The receiver will then have to guess $b \odot (\mathbf{t}_i \oplus \mathbf{t}_j)$; have it guess $\mathbf{t}_i \oplus \mathbf{t}_j$, which is correct with probability $2^{-D}$. Then,

$$\mathbf{q}_j = \mathbf{c}_j \cdot \mathbf{b} \oplus \mathbf{t}_j = (\mathbf{t}_i \oplus \mathbf{t}_j) \cdot \mathbf{b} \oplus \mathbf{t}_j = \mathbf{t}_i = \mathbf{c}_i \cdot \mathbf{b} \oplus \mathbf{t}_i = \mathbf{q}_i,$$

so extended OTs $i$ and $j$ will agree on all evaluations, breaking the security of the OT extension.

For each $u$ and $v$, $\|\mathbf{t}_u + \mathbf{t}_v\|_0$ follows a binomial distribution with parameters $n = n_\mathcal{C}$ and $p = \frac{1}{2}$. Given $D$, our attack succeeds with probability $2^{-D}$. In the simplest case of $m = 2$, the success probability is then $\mathbb{E}[2^{-D}] = ((1-p) + p2^{-1})^n = \left(\frac{3}{4}\right)^n \approx 2^{-0.415n}$, which is already considerably higher than is claimed by Endemic OT. For greater $m$, we estimate $D$ by finding some $D_0$ such that the event $D \leq D_0$ has constant probability. For any $D_0$, the event $D \leq D_0$ is a union of the $\binom{m}{2}$ events $\|\mathbf{t}_u + \mathbf{t}_v\|_0 \leq D_0$ for each pair $u < v$. While these events are not independent, we estimate that $\Pr[D \leq D_0]$ will be constant if $\Pr\left[\|\mathbf{t}_u + \mathbf{t}_v\|_0 \leq D_0\right] \geq \binom{m}{2}^{-1}$. We can then choose $D_0$ with the quantile function of the binomial distribution, and estimate that the success probability is around $2^{-D_0}$. We evaluated this for $n = 128$ and several choices of $m$:

| $m$ | $D_0$ |
|---|---|
| $10^6$ | 27 |
| $10^7$ | 24 |
| $10^8$ | 21 |
| $10^9$ | 18 |

# F Extra Proofs

## F.1 Universal Hash Proofs

**Proposition 2.3.** *Let $\mathcal{R}$ and $\mathcal{R}'$ be $\epsilon$ and $\epsilon'$-almost universal families, respectively. Then $R'R$ for $R \in \mathcal{R}, R' \in \mathcal{R}'$ is a $(\epsilon + \epsilon')$-universal family.*

*Proof.* Let $\vec{x} \in \mathbb{F}_q^n$ be nonzero. Except with probability $\epsilon$, $R\vec{x} \neq 0$. If it is nonzero, then with probability at least $1 - \epsilon'$, $R'R\vec{x} \neq 0$. Therefore $R'R\vec{x} \neq 0$ with probability at least $(1-\epsilon)(1-\epsilon') \geq 1 - \epsilon - \epsilon'$, so $R'R$ is $(\epsilon + \epsilon')$-universal. □

[13] This problem is important in the decoding of linear codes. See [MO15] for an efficient algorithm.

**Proposition 2.4.** *Let $\mathcal{R} \subseteq \mathbb{F}_q^{m\times n}$ and $\mathcal{R}' \subseteq \mathbb{F}_q^{m\times n'}$ be $\epsilon$-almost uniform families. Then $[R\ R']$ for $R \in \mathcal{R}, R' \in \mathcal{R}'$ is a $\epsilon$-uniform family.*

*Proof.* Let $\vec{x} = \begin{bmatrix} \vec{x}_0 \\ \vec{x}_1 \end{bmatrix}$ be nonzero, for $\vec{x}_0 \in \mathbb{F}_q^n, \vec{x}_1 \in \mathbb{F}_q^{n'}$. Without loss of generality, assume that $\vec{x}_0$ is nonzero. For $[R\ R']\vec{x}$ to equal $\vec{y}$, we must have $R\vec{x}_0 = \vec{y} + R'\vec{x}_1$. Because $R$ is independent of $\vec{y}$ and $R'$, this has probability at most $\epsilon$ by Thm. 2.2. □

## F.2 OT Extension Proof

**Theorem 5.1.** *The protocol in Fig. 11 achieves $\mathcal{F}_{\mathsf{OT\text{-}1}}^{p^{k_\mathcal{C}},\ell,\{X\}}$ with malicious security in the $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}^{p,q,\mathcal{C},\ell,\mathcal{L},M}$ hybrid model, assuming that $H\colon \mathbb{F}_q^{n_\mathcal{C}} \times \mathcal{T} \to \{0,1\}^\lambda$ is a $(p,q,\mathcal{C},\mathcal{T},\mathcal{L})$-TCR hash, and $\mathcal{R} \subseteq \mathbb{F}_q^{n_\mathcal{C}\times\lceil\log_q(\ell)\rceil}$ is an $\epsilon$-almost uniform family. The distinguisher advantage is at most $\epsilon M\ell(t_{max} - 1)/2 + \mathsf{Adv}_{\mathrm{TCR}}$, where $t_{max}$ is the maximum number of distict OTs that can have the same tweak under $t$. For the TCR itself, $\tau_{max}$ will be the maximum number of evaluations $F_i(\overleftrightarrow{x})$ where $t(i,\overleftrightarrow{x})$ outputs a given tweak. For semi-honest security, $\mathcal{R}$ is unused; instead set $\epsilon = q^{-n_\mathcal{C}}$ and $M = 1$.*

*Proof.* First, we establish correctness, which will be used in most cases of the security proof. Since the base VOLE gives a correlation $W = UG_\mathcal{C}\,\mathrm{diag}(\overleftarrow{\Delta}) + V$,

$$F_i(\overleftrightarrow{x}_i^*) = H(W_{i\cdot} + \vec{r}_i^\top - \overleftrightarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta}, t(i,\overleftrightarrow{x})) = H(V_{i\cdot} + \vec{r}_i^\top, t(i,\overleftrightarrow{x}_i^*)) = F_i^*.$$

Starting with the easiest case, if both parties are corrupted then the simulator can trivially program the whole output, which will be consistent by correctness.

**Corrupt $P_S$.** At the start of the protocol, the simulator sends "$\mathsf{commit}$" to $P_S$. For malicious security, it receives $R$ from $P_S$, while for semi-honest security it generates it randomly and puts it in the transcript instead. The simulator receives $\overleftarrow{\Delta}, W$ from $\mathcal{A}$ and forwards them to $P_S$. It then sets $F_i(\overleftrightarrow{x}) = H(W_{i\cdot} + \vec{r}_i^\top - \overleftrightarrow{x}G_\mathcal{C} \odot \overleftarrow{\Delta}, t(i,\overleftrightarrow{x}))$ and sends it to $\mathcal{F}_{\mathsf{OT\text{-}1}}^{p^{k_\mathcal{C}},\ell,\{X\}}$, for all $i \in [\ell]$. For malicious security, it sends $X$ to the ideal functionality as well. By correctness this works identically to the real protocol.

**Corrupt $P_R$.** This is the first case where the TCR's security is used. For malicious security, $\mathcal{S}$ first receives $\mathcal{W}_{\mathrm{pre}}$, $U_{\mathrm{pre}}$, $V_{\mathrm{pre}}$, and $L_{\mathrm{pre}}$ from $\mathcal{A}$, then samples $R \xleftarrow{\$} \mathcal{R}$ and sends it to $P_R$. In either case, the simulator then receives $U, V$ from $\mathcal{A}$ and forwards them to $P_R$. For malicious security, $\mathcal{S}$ then receives $w_{\mathrm{pre}} \in \mathcal{W}_{\mathrm{pre}}, \overleftarrow{L}_{\mathrm{off}} \in \mathbb{F}_q^{n_\mathcal{C}}$ from $\mathcal{A}$, generates a uniformly random $\overleftarrow{\Delta} \xleftarrow{\$} \mathbb{F}_q^{n_\mathcal{C}}$, and aborts the protocol if the consistency check in $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}^{p,q,\mathcal{C},\ell,\mathcal{L},M}$ would fail. Finally, it sends $x_i^* = U_{i\cdot}$ and $F_i^* = H(V_{i\cdot} + \vec{r}_i^\top, t(i,\overleftrightarrow{x}_i^*))$ to the ideal functionality, for $i \in [\ell]$.

Next, we prove that the real world, where the real protocol is run in the $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}^{p,q,\mathcal{C},\ell,\mathcal{L}}$-hybrid model, is indistinguishable from the ideal world, where the simulator is given access to the ideal functionality $\mathcal{F}_{\mathsf{OT\text{-}1}}^{p^{k_\mathcal{C}},\ell,\{X\}}$. First, there is a bad event that we must show is unlikely. We must show that there are no two distinct OT indices $i < j$ satisfying $V_{i\cdot} + \vec{r}_i^\top = V_{j\cdot} + \vec{r}_j^\top$ that have overlapping tweaks, meaning that there exists $\overleftrightarrow{x}_i$ and $\overleftrightarrow{x}_j$ such that $t(i,\overleftrightarrow{x}_i) = t(j,\overleftrightarrow{x}_j)$. We start with the malicious case. For any such pair, $V_{i\cdot} - V_{j\cdot}^\top = R(\vec{j} - \vec{i})$, so if $V$ were independent of $R$ then this would have probability at most $\epsilon$ because $\mathcal{R}$ is a uniform hash family. Although $V$ is allowed to depend on $R$, it must equal $V_{\mathrm{pre}}(w_{\mathrm{pre}}, \overleftarrow{\Delta})$, and $w_{\mathrm{pre}}$ can only be chosen from $M$ options. There are at most $\ell(t_{\max} - 1)/2$ possible pairs of indices $i, j$ with overlapping tweaks, so by a union bound the probability of the bad event is at most $\epsilon M\ell(t_{\max} - 1)/2$.

However, in the semi-honest case $\vec{r}_i = 0$ for all $i$. If $V$ were uniformly random then we could instead use that $V_{i\cdot} = V_{j\cdot}$ with probability only $q^{-n_\mathcal{C}}$. But $V$ isn't uniformly random because $P_R$ (who is the sender for the VOLE) is corrupted — it's chosen by the adversary. The trick is to prove security in a slightly different hybrid model, where the VOLE is required to output a $V$ such that $V_{i\cdot} \neq V_{j\cdot}$ when $i \neq j$ have overlapping tweaks. Any semi-honest protocol that achieves $\mathcal{F}_{\textsf{VOLE-pre}}$ based on functionalities which do not explicitly depend on who is corrupt (such as a communication channel) also achieves the modified functionality. That is, in semi-honest security, corruption does not give the adversary and environment any new power other than to see more information, and the honest–honest case (where $V$ is guaranteed to be uniformly random) still gives enough information to see whether $V$ has distinct rows. Therefore, if when $P_S$ is corrupt the VOLE outputs a $V$ with repeated rows more often than random, then there is an attack against the honest–honest case of the VOLE.

Next, we present the hybrids.

1. Start from the real world, then rewrite the usage of $H$ and $\overleftarrow{\Delta}$ into oracle calls to $\textsf{TCR-real}^{H,p,q,\mathcal{C},\mathcal{L}}$. That is, use $\textsc{leak}(L_{\text{pre}}(w_{\text{pre}}) - \overleftarrow{L}_{\text{off}})$ to implement the selective abort, instead of checking $\overleftarrow{\Delta} + \overleftarrow{L}_{\text{off}} \in L$ directly, and change $F_i(\overleftarrow{x})$ to be computed as $\textsc{query}(\overleftarrow{x}_i^* - \overleftarrow{x}, V_{i\cdot} + \vec{r}_i^\top, t(i, \overleftarrow{x}))$ where $\overleftarrow{x}_i^* = U_{i\cdot}$. This is the same because

   $$H(W_{i\cdot} + \vec{r}_i^\top - \overleftarrow{x} G_\mathcal{C} \odot \overleftarrow{\Delta}, t(i, \overleftarrow{x})) = H((U_{i\cdot} - \overleftarrow{x}) G_\mathcal{C} \odot \overleftarrow{\Delta} + V_{i\cdot} + \vec{r}_i^\top, t(i, \overleftarrow{x}_i^*)),$$

   by the correctness of the VOLE. This is just refactoring the computation, and so results in no observable difference for the environment.

2. Use TCR security to swap $\textsf{TCR-real}^{H,p,q,\mathcal{C},\mathcal{L}}$ for $\textsf{TCR-ideal}^{H,p,q,\mathcal{C},\mathcal{L}}$. This is allowed because the calls to $\textsc{query}$ are distinct, as otherwise the bad event would trigger.

3. Inline the oracles calls to $\textsf{TCR-ideal}^{H,p,q,\mathcal{C},\mathcal{L}}$. Notice that $\overleftarrow{\Delta}$ is only used for the selective abort attack, as in the simulator. Call the process of sampling $F_i$ and outputting it to $P_S$ the ideal functionality, and call the rest of this hybrid the simulator. We are now at the ideal world.

**Both Honest.** This case has the simplest simulator. The simulator only needs to sample $R \xleftarrow{\$} \mathcal{R}$ and allow $\mathcal{A}$ to eavesdrop on it. We again use a hybrid proof, starting from the real world.

1. Let $\overleftarrow{c} \in \mathcal{C}$ be a non-zero code word. Instead of sampling $W \xleftarrow{\$} \mathbb{F}_q^{\ell \times n_\mathcal{C}}$, sample $sk \xleftarrow{\$} \mathbb{F}_p^{k_\mathcal{C}}$ and $W' \xleftarrow{\$} \mathbb{F}_q^{\ell \times n_\mathcal{C}}$, and set $W_{i\cdot} = W'_{i\cdot} + \overleftarrow{c} \odot sk$ for all $i \in [\ell]$.

2. We now need to show that the functions $F_i^* = H(W_{i\cdot} + \overleftarrow{c} \odot sk + \vec{r}_i^\top - \overleftarrow{x} G_\mathcal{C} \odot \overleftarrow{\Delta}, t(i, \overleftarrow{x}))$ are uniformly random. They can be written in terms of $\textsc{query}$ from $\textsf{TCR-real}^{H,p,q,\mathcal{C},\mathcal{L}}$, using $sk$ instead of $\overleftarrow{\Delta}$ as the TCR key. There will be no duplicate queries, for the same reason as in the case of corrupt $P_R$. By the security of the TCR, all queries will be uniformly random.

3. Forget about $V, W, \overleftarrow{\Delta}$, which are now unused. We are now at the ideal world.

□

### F.3 Δ-OT Extension Proof

**Theorem 5.2.** *The protocol in Fig. 12 achieves $\mathcal{F}_{\textsf{VOLE}}^{p,p,\textsf{Rep}(\mathbb{F}_p^n),\ell,\{X\}}$ with malicious security in the $\mathcal{F}_{\textsf{VOLE-pre}}^{p,p,\textsf{Rep}(\mathbb{F}_p^{n'}),\ell,\mathrm{Affine}(\mathbb{F}_p^{n'}),M}$ hybrid model, assuming that $\mathcal{R} \subseteq \mathbb{F}_p^{n' \times n}$ is a $\epsilon$-almost uniform family and $n' \geq n$. The advantage is bounded by $\epsilon M(p^n - 1)$.*

*Proof.* First, we need to show correctness, which is used for all cases of the proof.

$$WR = UG_{\mathsf{Rep}(\mathbb{F}_p^{n'})}\,\mathrm{diag}(\overleftarrow{\Delta})R + VR = U\overleftarrow{\Delta}R + VR$$

The two cases where $P_R$ is corrupt are trivial, as the simulator can program the protocol output to be $\overleftarrow{\Delta}R, WR$, and $U$ is unchanged. When $P_R$ is honest, we need to use the following lemma, which shows that the entropy in $\overleftarrow{\Delta}$ is enough to make $\overleftarrow{\Delta}R$ uniform.

**Lemma F.1.** *Let $\mathcal{R} \subseteq \mathbb{F}_p^{n' \times n}$ be an $\epsilon$-almost uniform family and let $A \in \mathbb{F}_p^{m \times n'}$ be full rank, where $m \leq n'$. Then*

$$\Pr_{R \xleftarrow{\$} \mathcal{R}}[\mathrm{rank}(AR) < n] \leq \epsilon p^{n'-m}(p^n - 1).$$

*Proof.* Let $X = |\ker(AR) \setminus \{0\}| = |\ker(AR)| - 1$. The statement that $\mathrm{rank}(AR) < n$ is equivalent to $X \geq 1$. Any nonzero $\vec{x} \in \mathbb{F}_p^n$ has probability at most $\epsilon p^{n'-m}$ of being in $\ker(AR)$, because that implies $R\vec{x} \in \ker(A)$, which has size $|\ker(A)| = p^{n'-m}$ by the rank–nullity theorem. Therefore $\mathbb{E}[X] \leq \epsilon p^{n'-m}(p^n - 1)$, and the lemma follows by Markov's inequality. □

If both parties are honest then all that's needed is for $R$ to be full rank, as then all it does is throw away part of the VOLE output. This is true except with probability $\epsilon(p^n - 1)$ by Thm. F.1 with $A = \mathbb{1}_{n'}$. The only interesting case is when only $P_S$ is corrupt. The simulator first receives $\mathcal{W}_{\text{pre}}$, $U_{\text{pre}}$, $V_{\text{pre}}$, and $L_{\text{pre}}$ from $\mathcal{A}$, then samples $R \xleftarrow{\$} \mathcal{R}$ and sends it to $P_S$. The simulator receives $U, V$ from $\mathcal{A}$ and forwards them to $P_S$, and then receives $w_{\text{pre}} \in \mathcal{W}_{\text{pre}}, \overleftarrow{L}_{\text{off}} \in \mathbb{F}_p^{n'}$ from $\mathcal{A}$. The simulator generates a uniformly random $\overleftarrow{\Delta} \xleftarrow{\$} \mathbb{F}_p^{n'}$, and aborts the protocol if the guess in $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}^{p,p,\mathsf{Rep}(\mathbb{F}_p^{n'}),\ell,\mathcal{L},M}$ would fail. Finally, it sends $U, VR$ to the ideal functionality.

For security, we first define a bad event, and bound its probability. Let $L = L_{\text{pre}}(w_{\text{pre}}) - \overleftarrow{L}_{\text{off}}$, so the protocol aborts if $\overleftarrow{\Delta} \notin L$. Because $L \in \mathrm{Affine}(\mathbb{F}_p^{n'})$, there exists a vector $\overleftarrow{\Delta}_0 \in \mathbb{F}_p^{n'}$ and a full rank matrix $A \in \mathbb{F}_p^{m \times n'}$ such that $L = \overleftarrow{\Delta}_0 + \mathrm{rowspace}(A)$, where $m = \dim(L)$. Also, $A$ is independent of $\overleftarrow{L}_{\text{off}}$, because $\overleftarrow{L}_{\text{off}}$ only shifts $L$. The bad event is that $\overleftarrow{\Delta} \in L$ and $\mathrm{rank}(AR) < n$. The former has probability at most $p^{m-n'}$, since $\overleftarrow{\Delta}$ was sampled uniformly and $|L| = p^m$. If $A$ were independent of $R$, the latter probability would be at most $\epsilon p^{n'-m}(p^n - 1)$ by Thm. F.1. Though $\mathcal{A}$ sees $R$ before choosing $L$, we can still use a union bound. $L_{\text{pre}}$ is chosen independently of $R$, and there are at most $M$ possibilities for $L_{\text{pre}}(w_{\text{pre}})$, so the bad event has probability at most $\epsilon M(p^n - 1)$.

Next, the hybrid proof goes from the real world to the ideal world. The only information $P_S$ learns about $\overleftarrow{\Delta}$ is that $\overleftarrow{\Delta} \in L$, so it is equivalent to sample a second $\Delta' \in L$ after the check, let $\Delta'' = \Delta' R$, and subsequently use $\Delta''$ instead of $\Delta R$. Since we are assuming the bat event does not trigger, $\mathrm{rank}(AR)$ has full rank, so it's also equivalent to sample $\Delta'' \xleftarrow{\$} \mathbb{F}_p^n$. Split into the simulator and the ideal functionality, with $\Delta''$ being sampled in the ideal functionality. We are now at the ideal world. □

## F.4 Semi-honest PPRF Proof

**Theorem 6.1.** *Figure 13 constructs $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{q,1,\{X\}}$ out of $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{p,k,\{X\}}$, and is secure in the semi-honest model.*

*Proof.* First, we show correctness, as it is useful for all cases of the proof. We prove by induction that $s_y^i = s_y^{*\,i}$ for all $y \in [p^i] \setminus \{y_i^*\}$. In the base case, $F_0(x) = s_x^1 = s_x^{*\,1} = F_0^*(x)$ by correctness of the first $\binom{p}{p-1}$-OT. For induction, $P_S$ and $P_R$ both compute $s_{py+x}^{i+1}$ in exactly the same way for all

```
$\mathcal{S}^{p,q,\mathcal{L}}_{\mathsf{sub\text{-}PPRF\text{-}S}}$
for $i \in [k]$:
    recv. $F_i \in \left(\{0,1\}^\lambda\right)^{[N]}$ from $\mathcal{A}$
$G, t := \mathsf{BuildPPRF}(F)$
$P_S$ sends $\{t^i_x\}_{1 \le i < k, x \in [p]}$ to $P_R$
send $G$ to $\mathcal{F}^{q,1,\mathcal{L}}_{\mathsf{OT}\text{-}\bar{1}}$
```

(a) Corrupt $P_S$.

```
$\mathcal{S}^{p,q,\mathcal{L}}_{\mathsf{sub\text{-}PPRF\text{-}R}}$
for $i \in [k]$:
    recv. $x^*_i \in [N]$ from $\mathcal{A}$
    recv. $F^*_i \in \left(\{0,1\}^\lambda\right)^{[N]\setminus\{x^*_i\}}$ from $\mathcal{A}$
for $i := 1$ to $k-1$, $x \in [p]$:
    $t^i_x \xleftarrow{\$} \{0,1\}^\lambda$
$y^*, G^* := \mathsf{EvalPPRF}(x^*, F^*, t)$
send $\{t^i_x\}_{1 \le i < k, x \in [p]}$ to $P_R$
send $y^*, G^*$ to $\mathcal{F}^{q,1,\mathcal{L}}_{\mathsf{OT}\text{-}\bar{1}}$
```

(b) Corrupt $P_R$.

Figure 15: Simulators for semi-honest security of Fig. 13. In (a), $P_S$ sending $t$ to $P_R$ means that the adversary is allowed to eavesdrop on a fake message from $P_S$ to $P_R$.

$y \neq y^*_i$. Then for any $x \neq x^*_i$,

$$\begin{aligned}
s^{*\,i+1}_{py^*_i+x} &= t^i_x \oplus F^*_i(x) \oplus \bigoplus_{y \in [p^i]\setminus\{y^*_i\}} s^{*\,i+1}_{py+x} \\
&= F_i(x) \oplus \bigoplus_{y \in [p^i]} s^{i+1}_{py+x} \oplus F_i(x) \oplus \bigoplus_{y \in [p^i]\setminus\{y^*_i\}} s^{i+1}_{py+x} \\
&= s^{i+1}_{py^*_i+x}.
\end{aligned}$$

Therefore, $s^{i+1}_y = s^{*\,i+1}_y$ for all $y \neq y^*_{i+1} = py^*_i + x^*_i$.

When both parties are honest, the simulator can generate the $t^i_x$ uniformly at random and give them to the adversary as a fake eavesdropped message. In the hybrid proof, starting from the real world, first replace the random sampling of all $F_i(x)$ with instead sampling $t^i_x \xleftarrow{\$} \{0,1\}^\lambda$, then setting $F_i(x) = t^i_x \oplus \bigoplus_{y \in [p^i]} s^{i+1}_{py+x}$. These are the same distribution. Next, use correctness to replace $s^{*\,i}_y$ with $s^i_y$ everywhere, so that the $F_i(x)$ are all unused and can be removed. Now the internal leaves of the GGM tree are unused, which makes the $s^k_y$ be all the evaluations of a GGM PRF [GGM86]. Therefore, we can replace them all with uniform randomness. Finally, since the $x^*_i \in [p]$ are all uniformly random, it is equivalent to sample $y \xleftarrow{\$} [p^k]$, then let $x^*_0, \ldots, x^*_{i-1}$ be its expansion in base $p$, in big endian order. We are now at the ideal world.

Next, assume that only $P_S$ is corrupt. The simulator for this case is illustrated in Fig. 15a. By correctness, the output from $P_R$ will be $G^*$, the punctured version of the $G$ computed by a simulator. Also, $y$ will be uniformly random for the same reason as in the honest–honest case. Therefore, the real protocol will be indistinguishable from the simulation.

Finally, we have the case where $P_R$ is corrupt (simulator in Fig. 15b). Going from the real world to the ideal world, we use the following hybrids.

1. In a sequence of hybrids, replace all the $t^i_x$ and $s^i_{y^*_i}$ with uniformly random values. For $i = 1$, $s^1_{y^*_1} = F_0(x^*_{)}$ is already sampled uniformly at random by the ideal functionality $\mathcal{F}^{p,k,\{X\}}_{\mathsf{OT}\text{-}\bar{1}}$. For $i > 1$, first use that $s^{i-1}_{y^*_{i-1}}$ is uniformly random to sample $s^i_{py^*_{i-1}+x} = \mathsf{PRG}_x(s^{i-1}_{y^*_{i-1}})$ uniformly at random for all $x \in [p]$. Then instead of sampling the $s^i_{py^*_{i-1}+x}$ at random, sample $t^i_x \xleftarrow{\$} \{0,1\}^\lambda$,

```
$\mathcal{S}^{p,q,\mathcal{L}}_{\textsf{sub-PPRF-mal-S}}$
for $i \in [k]$:
    recv. $F_i \in \left(\{0,1\}^\lambda\right)^{[N]}$ from $\mathcal{A}$
recv. $L' \in \mathcal{L}$ from $P_S$:
recv. $\left\{t^i_x \in \{0,1\}^\lambda\right\}_{1 \le i < k, x \in [p]}$ from $P_S$
$G(y) := 0, \forall y \in [q]$
$L := \emptyset$
for $\{x^*_i\}_{i \in [k]} \in [p]^k$:
    $y^*, G_{y^*} := \mathsf{EvalPPRF}(x^*, F, t)$
    if $\widetilde{s} = \mathsf{VerifyPPRF}(y^*, G_{y^*}, \widetilde{t})$:
        $G(y) := G_{y^*}(y), \forall y \in [q] \setminus \{y^*\}$
        $L := L \cup \{y^*\}$
send $\mathsf{PRG}'_1 \circ G$ to $\mathcal{F}^{q,1,\mathcal{L}}_{\textsf{OT-}\overline{1}}$
send $L \cap L'$ to $\mathcal{F}^{q,1,\mathcal{L}}_{\textsf{OT-}\overline{1}}$
```

(a) Malicious $P_S$.

```
$\mathcal{S}^{p,q,\mathcal{L}}_{\textsf{sub-PPRF-mal-R}}$
for $i \in [k]$:
    recv. $x^*_i \in [N]$ from $\mathcal{A}$
    recv. $F^*_i \in \left(\{0,1\}^\lambda\right)^{[N] \setminus \{x^*_i\}}$ from $\mathcal{A}$
for $i := 1$ to $k-1$, $x \in [p]$:
    $t^i_x \xleftarrow{\$} \{0,1\}^\lambda$
$\widetilde{t} \xleftarrow{\$} \{0,1\}^{2\lambda}$
$y^*, G^* := \mathsf{EvalPPRF}(x^*, F^*, t)$
$\widetilde{s} := \mathsf{VerifyPPRF}(y^*, G^*, \widetilde{t})$
send $t, \widetilde{s}, \widetilde{t}$ to $P_R$
send $y^*, \mathsf{PRG}'_1 \circ G^*$ to $\mathcal{F}^{q,1,\mathcal{L}}_{\textsf{OT-}\overline{1}}$
```

(b) Malicious $P_R$.

Figure 16: Simulators for malicious security of Fig. 14.

then compute $s^i_{py^*_{i-1}+x}$ as in $\mathsf{EvalPPRF}$, for $x \neq x^*_{i-1}$. Finally, $t^{i-1}_{x^*}$ can be also be sampled randomly, because the underlying ideal functionality samples $F_{i-1}(x^*) \xleftarrow{\$} \{0,1\}^\lambda$. Continue this sequence of hybrids until $i = k$ to get the desired modifications.

2. Notice that $G^*$, the restriction of $P_S$'s output to $x \neq x^*$ is now computed in exactly the same way as $\mathsf{EvalPPRF}$, and that $G(y^*) = s^k_{y^*}$ is uniformly random. Put the computation of the former in the simulator and the latter in the desired ideal functionality $\mathcal{F}^{q,1,\{X\}}_{\textsf{OT-}\overline{1}}$. Also notice that the $t^i_x$ are all sampled uniformly at random and put them in the simulator. This is exactly the same as the ideal world, where the simulator talks to the ideal functionality and generates a fake transcript of the protocol.

□

## F.5 Maliciously Secure PPRF Proofs

**Proposition 6.2.** *The selective abort attack allowed in Fig. 14 will always be in $\mathcal{L} = \mathrm{Affine}(\mathbb{F}^k_p)$. More precisely, the $L$ sent by $\mathcal{S}^{p,q,\mathcal{L}}_{\textsf{sub-PPRF-mal-S}}$ (Fig. 16) will always be in $\mathrm{Affine}(\mathbb{F}^k_p)$.*

*Proof.* This is trivial if $L = \{\}$, so assume that $L$ is not empty. The simulator will then find a preimage for $\widetilde{s} = \mathsf{Hash}(\widetilde{s}_0 \parallel \cdots \parallel \widetilde{s}_{q-1})$. By collision resistance, every time the simulator calls $\mathsf{VerifyPPRF}(y^*, G^*, \widetilde{t})$ for $y \in L$, it finds the same $\widetilde{s}_y$. Next, assume that $L$ contains at least two elements $z$ and $z'$, because any $L$ containing only a single element is trivially in $X$. Then collision resistance of $\mathsf{PRG}'_0$ implies that $G_z(y) = G_{z'}(y)$ for all $y \notin \{z, z'\}$. That is, every $G_{y^*}(y)$ will agree with $G(y)$ on every $y$ they can compute. Use this to prove the following lemma.

**Lemma F.2.** *Let $z, z' \in L$, let $z_1, \ldots, z_k$ and $z'_1, \ldots, z'_k$ be their active paths and $w_0, \ldots, w_{k-1}$ and $w'_0, \ldots w'_{k-1}$ be their base OT choices. Let $j$ be the first index where they differ, so that $z_i \neq z'_i$. Then for any $y^* \in [q]$ with base OT choice bits $x^*_0, \ldots, x^*_{k-1}$ and active path $y^*_1, \ldots, y^*_k$, if $x^*_i = w_i$ except when $i = j$, we have $y^* \in L$.*

*Proof.* The collision resistance of $\mathsf{PRG}$ implies that all $s_y^{*\,i}$ computed by $\mathsf{EvalPPRF}(x^*, F, t)$ that outputs either $z$ or $z'$ will agree. They both miss the nodes of the GGM tree on their common path $z_1, \dots, z_{j-1}$, but every other node in the tree can be computed by at least one of them. Let $s_y^i$ be the seeds in the GGM tree that at least one of them may compute. This implies a correctness property for the $t_x^i$ when $i \geq j-1$: any $t_x^i$ used by either $z$ or $z'$ during evaluation of the PPRF must take its correct value of $F_i(x) \oplus \bigoplus_{y \in [p^i]} s_{py+x}^{i+1}$. Otherwise $z$ (or $z'$) would reconstruct a different $s_{pz_i+x}^{i+1}$ during evaluation, which would not pass the consistency check.

Since the active path of $y^*$ agrees with $z$ on its first $j-1$ nodes, it will find the same seeds $s_y^i$ for $i < j$ and $y \neq y_i^*$. Additionally, $y^*$ agrees with $z$ after its first $j$ nodes, so it only uses correct $t_x^i$ for $i \geq j$. This only leaves the corrections $t_x^{j-1}$ for the node on layer $j$. Because $z$ and $z'$ disagree on layer $j$, so $w_{j-1} \neq w'_{j-1'}$, every $t_x^{j-1}$ must be correct for this layer. Therefore, the evaluation on $y^*$ will get exactly the same seeds $s_y^i$ for any $y \neq y_i^*$, and so $G_{y^*}$ must agree with $G_z$ and $G_{z'}$. Finally, $\widetilde{t}$ must be correct for there to be two evaluations $z$ and $z'$ that pass the consistency check, so the evaluation for $y^*$ will correctly find all the $\widetilde{s}_y$, and so $y^* \in L$. □

In each position $i \in [k]$, either all $y^* \in L$ will have the same $x_i^*$, or there will be at least two different possible $x_i^*$. Let $L' \supseteq L$ be the set of all $y^*$ that match with the $x_i^*$ in the positions where all of $L$ are the same. This allows $y^* \in L'$ to take any value at the positions where at least two $x_i^*$ differ. These are affine constraints so $L' \in \mathrm{Affine}(\mathbb{F}_p^k)$. We need to prove that $L' = L$.

Let $z \in L$ and $y^* \in L'$, where $z \neq y^*$. Then the first place where they differ must be a position that $L'$ does not constrain, so there must be some $z' \in L$ that disagrees at this position. By Thm. F.2, there must be some $z'' \in L$ that is identical to $z$ except at this position, where it agrees with $y^*$ instead. Repeating this process eventually finds an element of $L$ that is exactly the same as $y^*$. Therefore, $L' = L$. □

**Theorem 6.3.** *Figure 14 (composed with Fig. 13) is a maliciously secure $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{q,1,\mathrm{Affine}(\mathbb{F}_p^k)}$ in the $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{p,k,\mathrm{Affine}(\mathbb{F}_p^k)}$ hybrid model.*

*Proof.* If both parties are honest then this is essentially the same as for the semi-honest case. The consistency check messages $\widetilde{s}, \widetilde{t}$ can be simulated as a hash of uniformly random values, and a uniformly random value in $\{0,1\}^{2\lambda}$. By the security of $\mathsf{PRG}'$, the OT outputs $\mathsf{PRG}'_1 \circ G$ will be indistinguishable from uniformly random, as will the values $\widetilde{s}_y$.

**Malicious $P_S$.** The simulator for this case is shown in Fig. 16a. The collision resistance of $\mathsf{Hash}$ and $\mathsf{PRG}'_0$ implies that $G_{y^*}(y) = G(y)$ for all $y \neq y^*$, when $y^* \in L$. Therefore, when the desired ideal functionality computes $P_R$'s output, it will match what $P_R$ would output in the real protocol, assuming that $y^* \in L$. When $y^* \notin L$, $P_R$ never gets to see the output, so this is equivalent. Finally, Thm. 6.2 implies that $L \in \mathrm{Affine}(\mathbb{F}_p^k)$, and the adversary must always provide a $L' \in \mathrm{Affine}(\mathbb{F}_p^k)$, so $L \cap L' \in \mathrm{Affine}(\mathbb{F}_p^k)$ because it is closed under intersection. Therefore, the real protocol is indistinguishable from the simulated protocol using the desired ideal functionality.

**Malicious $P_R$.** Because $P_R$ never sends any messages, malicious security is essentially the same as semi-honest security for this case. The only difference from the semi-honest protocol is the consistency check messages $\widetilde{s}, \widetilde{t}$. By the security of $\mathsf{PRG}'$, these will be indistinguishable from being generated from uniformly random values $\widetilde{s}_y$. Also, the OT outputs $\mathsf{PRG}'_1 \circ G$ will be indistinguishable from uniformly random. Therefore, the protocol is secure in this case. □