SoftSpokenOT: Quieter OT Extension From Small-Field Silent VOLE in the Minicrypt Model

Lawrence Roy*

Oregon State University

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 λ bits of communication for each OT, SoftSpokenOT only needs λ/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 $5\times$ 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 and improving 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).

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 λ bits for every extended random OT. Recent works under the heading of Silent OT [BCGI18, BCG+19b, SGRR19, 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,

* Address: ldr709@gmail.com. Supported by a DoE CSGF Fellowship.

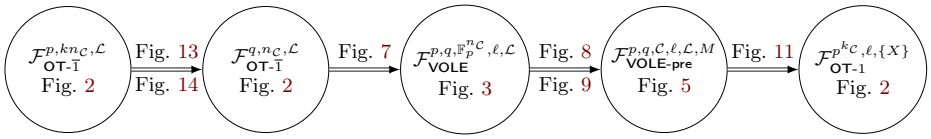


Fig. 1: Sequence of ideal functionalities and protocols used for OT extension. Here $q = p^k$ is the size of the small field VOLE, and $\mathcal{L} = \mathrm{Affine}(\mathbb{F}_p^{knc})$ is the set of allowed selective abort attacks against the base OT receiver. Protocols below the arrows are consistency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

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 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. [CCG18] (hereafter, CCG) generalized OOS to work over larger fields, which have better linear codes. This allowed for fewer base OTs, but required more communication per extended OT.

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 λ/k bits, compared to IKNP's λ 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 2λ 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 ℓ OTs with sublinear communication $\Theta\left(\frac{\lambda \cdot \ell}{\log(\ell)}\right)$, 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 $W - V = UG_{\mathcal{C}} \operatorname{diag}(\vec{\Delta})$. Here, $G_{\mathcal{C}}$ is the generator matrix for a linear code $\mathcal{C}$. Note that Δ-OT (a.k.a. correlated OT) is a special case of subspace VOLE, as is the correlation used by PaXoS [PRTY20]. Our Δ-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 $(m_0, \ldots, m_{N-1})$ where the m_y are all random. These may be 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 three existing works on consistency checks for OT extension. For OOS this is minor — 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}.

There is an existing work on OT extension consistency checking that we did not find to be flawed. CCG base their proof on [CDD+16]'s careful analysis of consistency checking for homomorphic commitments. CCG's check is similar to ours in that it works over any field. However, similarly to [CDD+16] and unlike CCG we use universal hashing to compress the random challenge of the consistency check. Additionally, we prove a tighter concrete security bound than either work, which halves the number of rows that must be consistency checked.

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 the full version). We follow [CT21] in using a universal hash to prevent collisions, only using the tweak to improve the concrete security of the TCR hash. We optimize their technique by sending the universal hash in parallel with our new consistency check — our 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 ℓ. 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 λ 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 ith VOLE, they get a correlation $W_{\cdot i} \oplus V_{\cdot i} = \Delta_i\vec{u}$, where $V_{\cdot i}$ means the ith column of a matrix V. In matrix notation, this is an outer product: $W \oplus V = \vec{u}\vec{\Delta}$, where $\vec{\Delta}$ is the row vector of all the Δ_i. Then looking at the jth row gives $W_{j\cdot} \oplus V_{j\cdot} = u_j\vec{\Delta}$, which make u_j the choice bit of a Δ-OT. That is, P_R has learned $\tilde{m}_{j0} = W_{j\cdot}$ and $\tilde{m}_{j1} = W_{j\cdot} \oplus \vec{\Delta}$, while P_S has its choice bit u_j and $\tilde{m}_{u_j} = V_{j\cdot}$, 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 $\tilde{m}_{jx}$ 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 λ/k of these VOLEs to get the λ bits per OT needed to make the hash secure. Derandomizing $\vec{u}$ for each OT then only needs λ/k *bits* per OT, as for each VOLE the 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 Δ 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:

$$\vec{u} = F(0) \oplus F(1) \qquad \Longrightarrow \vec{u} = \bigoplus_{x \in \mathbb{F}_{2^k}} F(x)$$

[1] Note that this is backwards from the usual description of IKNP — it's more usual to set Δ 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.

$$\vec{v} = 0\,F(0) \oplus 1\,F(1) \qquad \implies \vec{v} = \bigoplus_{x \in \mathbb{F}_{2^k}} xF(x)$$

$$\vec{w} = \Delta F^*(0) \oplus (1 \oplus \Delta)F^*(1) \implies \vec{w} = \bigoplus_{x \in \mathbb{F}_{2^k}} (x \oplus \Delta)F^*(x).$$

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 λ/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 $\vec{x} \odot \vec{y} = [x_0 y_0 \ \cdots \ x_{n-1}y_{n-1}]^\top$. 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 ith row of a matrix M is $M_{i\cdot}$, while the jth 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. The trivial code, $\mathbb{F}_p^n$, has all vectors as code words. That is, $G_{\mathbb{F}_p^n} = T_{\mathbb{F}_p^n} = \mathbb{1}_n$, where $\mathbb{1}_n$ is the $n \times n$ identity matrix. The repetition code, $\mathsf{Rep}(\mathbb{F}_p^n)$, 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 ϵ-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 ϵ-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 ϵ-almost XOR-universal. Clearly, a family that is ϵ-almost uniform is also ϵ-almost universal. We use two composition properties of universal hashes.

Proposition 2.3. *Let $\mathcal{R}$ and $\mathcal{R}'$ be ϵ and ϵ'-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 ϵ-almost uniform families. Then $[R\ R']$ for $R \in \mathcal{R}, R' \in \mathcal{R}'$ is a ϵ-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 $\boxed{\text{recv. } x \text{ from } P_S \,\|\, \text{recv. } y \text{ from } P_R}$. 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,

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

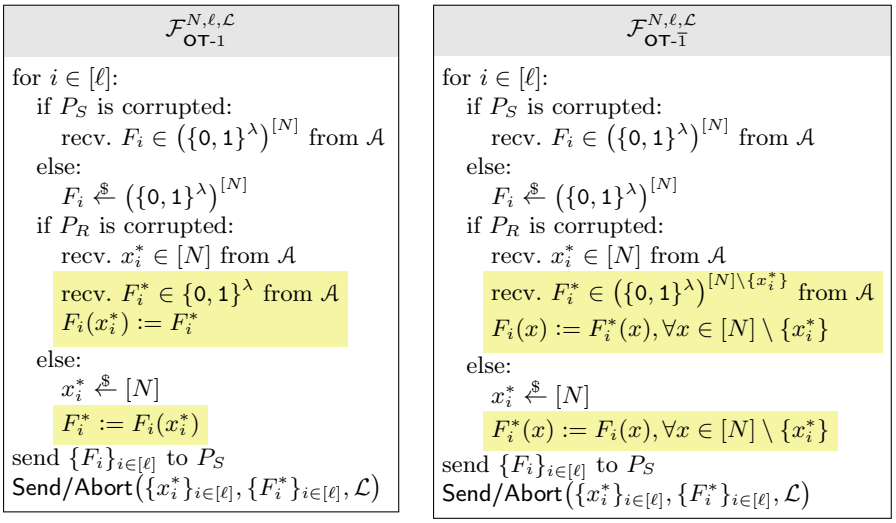


Fig. 2: Ideal functionalities for a batch of ℓ endemic OTs, with $\binom{N}{1}$-OT on the left and $\binom{N}{N-1}$-OT on the right. Differences are highlighted.

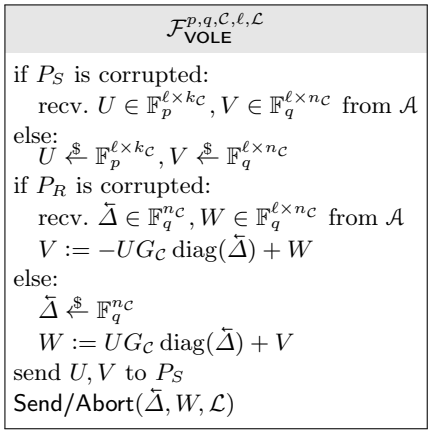


Fig. 3: Ideal functionality for endemic subspace VOLE. $\mathcal{C}$ is a linear code.

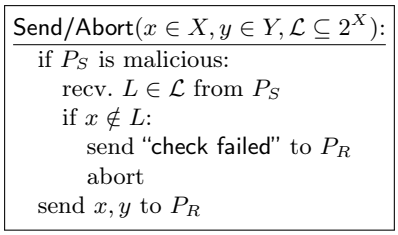


Fig. 4: Output with leakage function. Sends x, y to P_R, after allowing P_S to do a selective abort attack on x.

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]

[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.

The ideal functionalities for length ℓ 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 $F\colon [N] \to \{0,1\}^\lambda$, 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 $W - V = UG_{\mathcal{C}} \operatorname{diag}(\vec{\Delta})$, 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 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 $L - \vec{L}_{\text{off}} \in \mathcal{L}$ when $L \in \mathcal{L}$ and $\vec{L}_{\text{off}} \in X$. We use one main set of allowed guesses, $\text{Affine}(\mathbb{F}_q^n)$. 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 $a_0 x_0 + \cdots + a_{n-1} x_{n-1} = a_n$, 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 $\text{Affine}(\mathbb{F}_p^{nk}) \supseteq \text{Affine}(\mathbb{F}_q^n)$. 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 $\vec{L}_{\text{off}}$, and V can depend on the part of $\vec{\Delta}$ that is guessed.

$\mathcal{F}^{p,q,\mathcal{C},\ell,\mathcal{L},M}_{\mathsf{VOLE\text{-}pre}}$

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 $\mathcal{F}^{p,q,\mathcal{C},\ell,\mathcal{L}}_{\mathsf{VOLE}}$
instead of Send/Abort:
  if P_S is malicious:
    recv. $w_{\mathrm{pre}} \in \mathcal{W}_{\mathrm{pre}}, \check{L}_{\mathrm{off}} \in \mathbb{F}_q^{n_{\mathcal{C}}}$ from $\mathcal{A}$
    if $U \neq U_{\mathrm{pre}}(w_{\mathrm{pre}}) \vee V \neq V_{\mathrm{pre}}(w_{\mathrm{pre}}, \check{\Delta}) \vee \check{\Delta} + \check{L}_{\mathrm{off}} \notin L_{\mathrm{pre}}(w_{\mathrm{pre}})$
      send "check failed" to P_R
      abort
  send $\check{\Delta}, W$ to P_R

Fig. 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 $\check{\Delta}$, $\check{\Delta} + \check{L}_{\mathrm{off}} \in L_{\mathrm{pre}}(w_{\mathrm{pre}})$ must imply $V = V_{\mathrm{pre}}(w_{\mathrm{pre}}, \check{\Delta})$, to ensure that checking V_{pre} does not make the selective abort any more powerful.

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

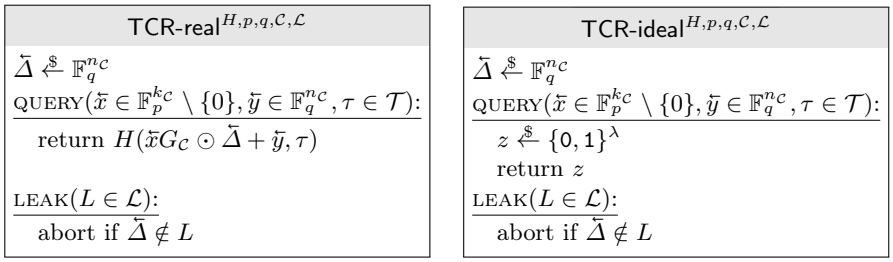
These changes are behind "if P_S is malicious" checks, so in the semi-honest case $\mathcal{F}_{\mathsf{VOLE}}$ is a equivalent to $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}$. For malicious security, $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}$ gives the adversary less power than $\mathcal{F}_{\mathsf{VOLE}}$ 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 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*

[4] Note that we do not consider multi-instance security. In fact, there is a generic attack:



(a) Real world. (b) Ideal world.

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

adversary $\mathcal{A}$ that does not call QUERY *twice on the same input $(\vec{x}, \vec{y}, \tau)$,*

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

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 $\vec{\Delta}$ 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 $\vec{\Delta}$ would have to depend on these primitives [CT21]. In the standard model, their definition is impossible to instantiate: $H(\vec{\Delta}, 0)$ must be random by TCR security, yet restricting $\vec{\Delta}$ so that the first bit of $H(\vec{\Delta}, 0)$ is zero only reduces 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 the full version. 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 $\tau_{max}\left(\mathfrak{q} + \frac{1}{2}\mathfrak{q}'\right)q^{-d_{\mathcal{C}}}$. Here, τ_{max} is the maximum number of times* QUERY *is called with the same τ, $\mathfrak{q}$ 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 ι to encode its input; when $p = 2$, ι can be the identity map.

given N instances, the attacker chooses an L that contains $\vec{\Delta}$ with probability $1/N$, then brute forces $\vec{\Delta}$ for instances where $\vec{\Delta} \in L$. Thus, it is N-times cheaper to brute force attack H for N instances than to target a single one.

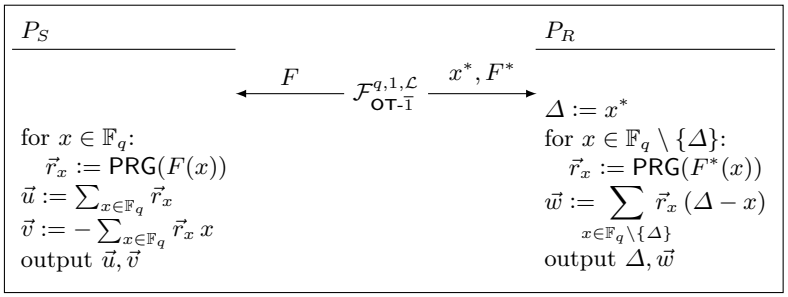


Fig. 7: Protocol for small field VOLE. If $\mathcal{F}_{\text{OT-}\bar{1}}^{q,1,\mathcal{L}}$ instead outputs "check failed", it should be passed straight through to P_R.

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* $H(\breve{y}, \tau) = \mathsf{Enc}(\tau, \iota(\breve{y})) \oplus \iota(\breve{y})$ *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 Prop. 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 $\mathsf{PRG}\colon \{0,1\}^\lambda \to \mathbb{F}_p^\ell$. 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* $\mathcal{F}_{\text{OT-}\bar{1}}^{q,1,\mathcal{L}}$ *hybrid model securely realizes* $\mathcal{F}_{\text{VOLE}}^{p,q,\mathbb{F}_p,\ell,\mathcal{L}}$*, 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 $\vec{w}$ 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,

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

Corrupt P_S*.* After receiving F from $\mathcal{A}$, the simulator will compute $\vec{u}, \vec{v}$ honestly and submit them to $\mathcal{F}_{\text{VOLE}}^{p,q,\mathbb{F}_p,\ell,\mathcal{L}}$. If P_S is malicious, it will also forward $L \in \mathcal{L}$ to the ideal functionality. In the real world, $\mathcal{F}_{\text{OT-}\bar{1}}^{q,1,\mathcal{L}}$ 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, $\mathcal{F}_{\text{VOLE}}^{p,q,\mathbb{F}_p,\ell,\mathcal{L}}$ will pick Δ 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 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 $\mathbb{F}_p$.

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 $\sum_{x'} \sum_{x_0} (p - x_0)\vec{r}_{x'+x_0} = \vec{v}_0$. $\bar{P}_R$ 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.

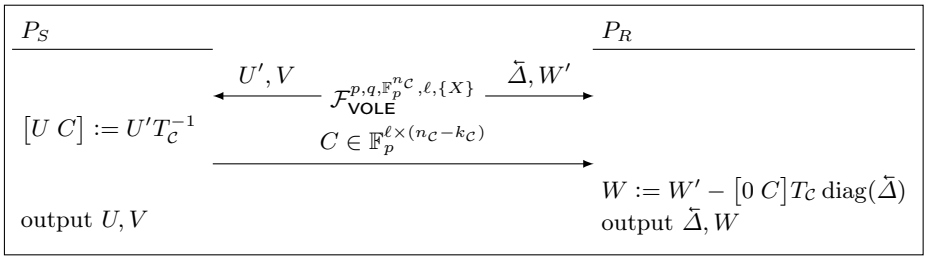


Fig. 8: Protocol for subspace VOLE.

Our protocol for subspace VOLE is presented in Fig. 8. It starts out with a VOLE correlation $W' - V = U'\,\mathrm{diag}(\Delta)$. 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:

$$\begin{aligned} W &= W' - [0\ C]T_\mathcal{C}\,\mathrm{diag}(\vec{\Delta}) \\ &= V + U'\,\mathrm{diag}(\vec{\Delta}) - [0\ C]T_\mathcal{C}\,\mathrm{diag}(\vec{\Delta}) \\ &= V + \left([U\ C]T_\mathcal{C} - [0\ C]T_\mathcal{C}\right)\mathrm{diag}(\vec{\Delta}) \\ &= V + UG_\mathcal{C}\,\mathrm{diag}(\vec{\Delta}). \end{aligned}$$

For security, notice that any $U, V, \vec{\Delta}, W$ 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 $U', V, \vec{\Delta}, W'$ from the underlying VOLE. Specifically, $U' = [U\ C]T_\mathcal{C}$ and $W' = W + [0\ C]T_\mathcal{C}\,\mathrm{diag}(\vec{\Delta})$. 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 $U, V, \vec{\Delta}, W$ (together with U' or W' if P_S or P_R was corrupted). In the real world, the underlying VOLE functionality picks $U', V, \vec{\Delta}, W'$ uniformly at random subject to the constraints of the VOLE correlation and any outputs

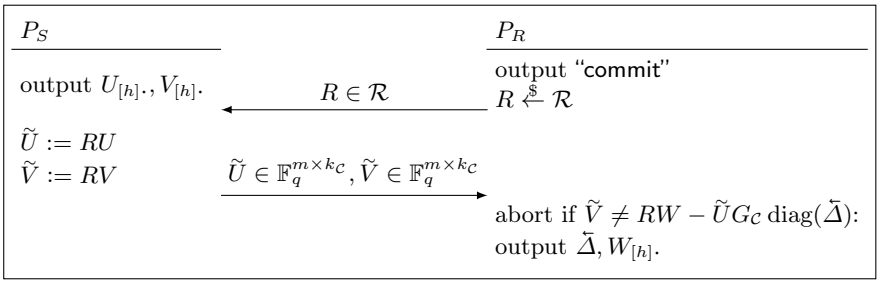


Fig. 9: Consistency checking protocol, which should be used with Fig. 8. $\mathcal{R}$ must be a ϵ-universal hash family, where all $R \in \mathcal{R}$ is $\mathbb{F}_p^h$-hiding. The "abort if" means that "check failed" is output if the check fails. If instead of giving $\vec{\Delta}, W'$ to P_R, the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

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 $U', V, \vec{\Delta}, W'$, 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, PSS, and CCG 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 follow [CDD+16] in generalizing this to use linear universal hashes. Any linear ϵ-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 $R \in \mathcal{R}$, which guarantees 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 $U \in \mathbb{F}_p^{\ell \times kc}$ hidden then it will keep $V \in \mathbb{F}_q^{\ell \times nc}$ 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 ϵ 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 Prop. 2.3.

Remark 4.2. P_S outputs $U_{[h]}, V_{[h]}$. 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 $\mathcal{F}_{\text{VOLE-pre}}^{p,q,\mathcal{C},h,\mathcal{L},M}$) 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. To avoid three separate sets of notations for very similar protocols, we discuss their protocols and proofs using our notation. See the full version for a more detailed discussion of these flaws, using their original notation.

We first present the flaw in OOS, because it is most similar to our protocol.

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.

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 $\bar{V} = [\bar{U}\ \ R\bar{C}] T_{\mathcal{C}} \operatorname{diag}(\tilde{\Delta})$ (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 Δ_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 Δ_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 $RY = [\bar{U}\ \ R\bar{C}]T_{\mathcal{C}} \oplus \tilde{U}G_{\mathcal{C}}$, so $RY_{-E} = \tilde{U}G_{\mathcal{C}_{-E}}$ 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 $\tilde{U}$ 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.

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.

The most important difference from OOS is that PSS attempt to compress the consistency check by summing the columns of $\hat{V}$ to get $\tilde{v} = \hat{V}[1\ \cdots\ 1]^\top$. The consistency check is then that $\tilde{v}$ must equal $\big(RW \oplus \tilde{U}G_{\mathcal{C}}\,\mathrm{diag}(\vec{\Delta})\big)[1\ \cdots\ 1]^\top = RW[1\ \cdots\ 1]^\top \oplus \tilde{U}G_{\mathcal{C}}\vec{\Delta}$. 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 $U'_{0,}$, and in the next OT the second N bits of $U'_{1,}$, 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 $i \in \left[\lceil \frac{n_{\mathcal{C}}}{N} \rceil\right]$. 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 $\check{\Delta}$ 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.

Flaw in KOS's Proof. To turn out consistentcy check into KOS's, 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 an R that is is not hiding. Let α 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, $\tilde{U}$, works as normal, except that it gets encoded into a field element $u = \bigoplus_i \tilde{U}_i.\alpha^i = \vec{\alpha}^\top \tilde{U}$, where $\vec{\alpha} = [1, \alpha, \ldots, \alpha^{\lambda-1}]^\top$. The other half, $\tilde{V}$, is compressed from λ^2 bits down to λ bits by turning it into a single field element $v = \bigoplus_{ij} \tilde{V}_{ij}\alpha^{i+j} = \vec{\alpha}^\top \tilde{V} \vec{\alpha}$. Similarly, let $w = \vec{\alpha}^\top RW\vec{\alpha}$ and $\delta = \check{\Delta}\vec{\alpha}$. Then the consistency check becomes

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

Because $\mathcal{C}$ is a repetition code, U' is supposed to be derandomized so that all columns are identical to U. Let $Y = U' \oplus \left[0 \ C\right] T_{\mathcal{C}}$ 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_Δ be the set of possible Δ 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_Δ 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

$$W = V \oplus (U' \oplus \left[0 \ C\right] T_{\mathcal{C}}) \operatorname{diag}(\check{\Delta}) = V \oplus Y \operatorname{diag}(\check{\Delta})$$

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

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

If u is set to be some element y_i of $\tilde{y}$, the consistency check at least won't depend on Δ_i. Since Y is uniformly random, $\tilde{y}$ 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 the full version 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 Δ 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 $O(2^{-\frac{4}{5}\lambda})$.

4.2 Our New Proof

The biggest hurdle in the proof is the case where P_S is malicious. If P_S lies when it sends C, then it will have to guess some entries of Δ, but which entries depends on what $\tilde{U}$ 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 $\tilde{U}, \tilde{V}$. 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.

The proof of [CDD+16] is based on a similar lie-preserving property, but they analyze this property independently from the consistency check. This leads to a bound of $\Theta(\sqrt{\epsilon})$ on the distinguisher's advantage. We instead consider these events together, because the distinguisher only succeeds when it violates the property *and* passes the consistency check. The product of these event's probabilities is smaller than either individual probability, so we prove a much smaller distinguisher advantage bound of $\Theta(\epsilon)$.

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* $\mathcal{F}_{\mathsf{VOLE\text{-}pre}}^{p,q,\mathcal{C},h,\mathcal{L},M}$ *if* $\mathcal{L} \supseteq \mathrm{Affine}(\mathbb{F}_q^{nc})$*, assuming that* $\mathcal{R} \subseteq \mathbb{F}_q^{m \times \ell}$ *is a* ϵ*-almost universal family where all* R *are* $\mathbb{F}_p^h$*-hiding. The distinguisher has advantage at most*

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

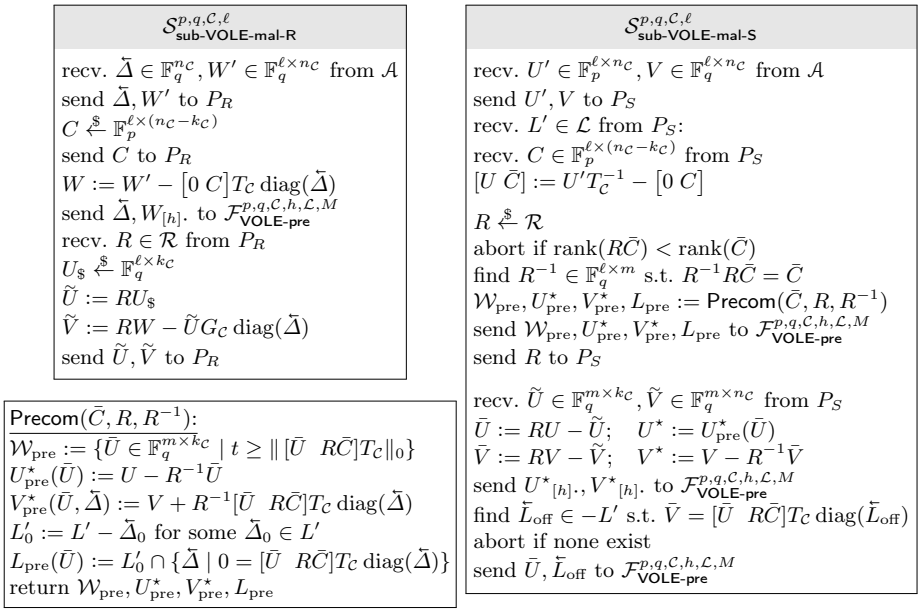


Fig. 10: Simulators for malicious security of Fig. 8 combined with Fig. 9, for a single corrupt party. $\mathcal{S}^{p,q,\mathcal{C},\ell}_{\text{sub-VOLE-mal-R}}$ is for corrupt P_R, while $\mathcal{S}^{p,q,\mathcal{C},\ell}_{\text{sub-VOLE-mal-S}}$ is for corrupt P_S.

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

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 Def. 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 ϵ-almost universal family, and let A be any matrix in $\mathbb{F}_q^{n\times l}$. Then, $\mathbb{E}_{R\xleftarrow{\$}\mathcal{R}}\left[q^{\operatorname{rank}(A)-\operatorname{rank}(RA)}-1\right] \le \epsilon(q^{\operatorname{rank}(A)}-1)$.*

   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 Def. 2.1, for any particular $v\neq 0$ the probability that $Rv = 0$ is at most ϵ. Since X is the sum of $|\operatorname{colspace}(A)\setminus\{0\}| = q^{\operatorname{rank}(A)}-1$ indicator random variables, we get $\mathbb{E}[X] \le \epsilon(q^{\operatorname{rank}(A)}-1)$.

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

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

   Let $\bar{U} = RU - \tilde{U}$ and $\bar{V} = RV - \tilde{V}$ 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 $[\bar{U}\ \ R\bar{C}]T_{\mathcal{C}}\,\mathrm{diag}(\overleftarrow{\Delta})$, i.e. guessing Δ_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 $\mathrm{rank}([\bar{U}\ \ R\bar{C}]T_{\mathcal{C}})$, 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 $\mathrm{rank}([\bar{U}\ \ R\bar{C}]) \geq \mathrm{rank}(R\bar{C})$. 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 $r = \mathrm{rank}(\bar{C}) - \mathrm{rank}(R\bar{C})$, so $N \geq \mathrm{rank}(\bar{C}) - r$. Then $P \leq \mathbb{E}\big[q^{-\,\mathrm{rank}(\bar{C})+r}\,\mathbb{1}_{r\geq 1}\big]$, since the added abort occurs exactly when $r \geq 1$, and expectation of conditional probability is marginal probability.

Now, apply Lem. 4.6 to $\bar{C}$ to get $\mathbb{E}[q^r - 1] \leq \epsilon(q^{\mathrm{rank}(\bar{C})} - 1)$. 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

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

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

2. After checking that $\mathrm{rank}(R\bar{C}) = \mathrm{rank}(\bar{C})$, 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,

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

3. Let $\mathcal{W}_{\mathrm{pre}} = \mathbb{F}_q^{m\times kc}$, then run $\mathsf{Precom}(\bar{C}, R, R^{-1})$ to get the pre-commitment functions $U^\star_{\mathrm{pre}}, V^\star_{\mathrm{pre}}, L_{\mathrm{pre}}$ as in the simulator, as well as $\overleftarrow{\Delta}_0 \in L'$ and $L'_0 = L' - \overleftarrow{\Delta}_0$. Also, find some $\overleftarrow{L}_{\mathrm{off}} \in -L'$ where $\bar{V} = [\bar{U}\ \ R\bar{C}]T_{\mathcal{C}}\,\mathrm{diag}(\overleftarrow{L}_{\mathrm{off}})$. Replace the underlying guess $\overleftarrow{\Delta} \in L'$ and the consistency check $-\bar{V} = [\bar{U}\ \ R\bar{C}]T_{\mathcal{C}}\,\mathrm{diag}(\overleftarrow{\Delta})$ with $\overleftarrow{\Delta} + \overleftarrow{L}_{\mathrm{off}} \in L_{\mathrm{pre}}(\bar{U})$. When such an $\overleftarrow{L}_{\mathrm{off}}$ 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 $\overleftarrow{\Delta} + \overleftarrow{L}_{\mathrm{off}} \in L'_0$ if and only if $\overleftarrow{\Delta} \in L'$

because $\vec{\Delta} + \vec{L}_{\text{off}}$ is the difference of two elements of the affine subspace L'. When $\vec{\Delta} + \vec{L}_{\text{off}} \in L'_0$, we have that $\vec{\Delta} + \vec{L}_{\text{off}} \in L_{\text{pre}}(\bar{U})$ is equivalent to $0 = [\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} \operatorname{diag}(\vec{\Delta} + \vec{L}_{\text{off}})$, which equals $[\bar{U} \;\; R\bar{C}]T_{\mathcal{C}} \operatorname{diag}(\vec{\Delta}) + \bar{V}$. The latter being zero is the consistency check.

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

4. Factor out the sampling of Δ, the computation of $W_{[h]} . = U^\star{}_{[h]} . \operatorname{diag}(\vec{\Delta}) + V^\star{}_{[h]} .$, and the selective abort attack $\vec{\Delta} + \vec{L}_{\text{off}} \in L_{\text{pre}}(\bar{U})$ 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 $V^\star \neq V^\star_{\text{pre}}(\bar{U}, \vec{\Delta})$, 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

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

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 $\vec{\Delta}$, 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 $s \in \{0,1\}^\lambda$ 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 $\tilde{V}$ with a local random oracle Hash before sending it, because all that's needed is an equality check. The simulator (in the malicious P_S case) could then extract $\tilde{V}$ from its hash, then continue as usual. Interestingly, for concrete security it would be fine even if Hash were just an arbitrary collision resistant hash. Looking at just $\bar{C}$ and $\tilde{U}$, the simulator can see which entries of $\vec{\Delta}$ are being guessed, though not what the guesses are. By

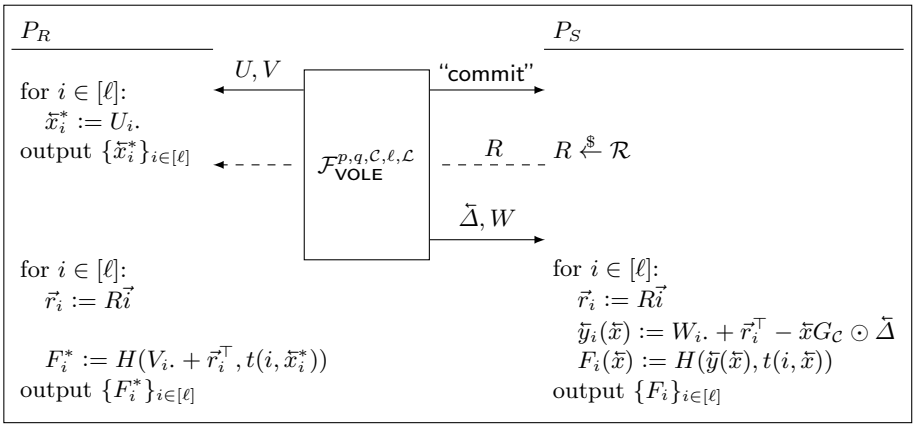


Fig. 11: $\binom{p^{kc}}{1}$-OT extension protocol. Note that the parties for the base VOLE are swapped, with P_S (instead of P_R) getting $\vec{\Delta}$. If P_S receives "check failed" from the VOLE then the protocol is aborted immediately. For semi-honest security, the "commit" and R steps are skipped,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

looping through a random subset of 2^σ possible guesses (and for the usual setting of $\sigma = 40$ this is quite feasible), $\mathcal{S}$ can find the preimage of $\mathsf{Hash}(\tilde{V})$ often enough to only give the distinguisher an additional advantage of $2^{-\sigma}$.

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 $\tau_{\max}$, which is the maximum number of queries $H(\vec{y}, \tau)$ on the same tweak τ. We allow an arbitrary function $t(i, \vec{x})$ 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 $U_i.$ in $\mathbb{F}_p^{kc}$ from the VOLE, and the choices x^*_i in $\left[p^{kc}\right]$ expected for OT. This can be thought of as writing x^*_i in base p.

Theorem 5.1. *The protocol in Fig. 11 achieves $\mathcal{F}_{\mathsf{OT}\text{-}1}^{p^{kc},\ell,\{X\}}$ with malicious security in the $\mathcal{F}_{\mathsf{VOLE}\text{-}\mathsf{pre}}^{p,q,\mathcal{C},\ell,\mathcal{L},M}$ hybrid model, assuming that $H\colon \mathbb{F}_q^{nc} \times \mathcal{T} \to \{0,1\}^\lambda$ is a $(p, q, \mathcal{C}, \mathcal{T}, \mathcal{L})$-TCR hash, and $\mathcal{R} \subseteq \mathbb{F}_q^{nc \times \lceil \log_q(\ell) \rceil}$ is an ϵ-almost uniform family. The distinguisher advantage is at most $\epsilon M \ell(t_{max} - 1)/2 + \mathsf{Adv}_{\mathrm{TCR}}$, where t_{max}*

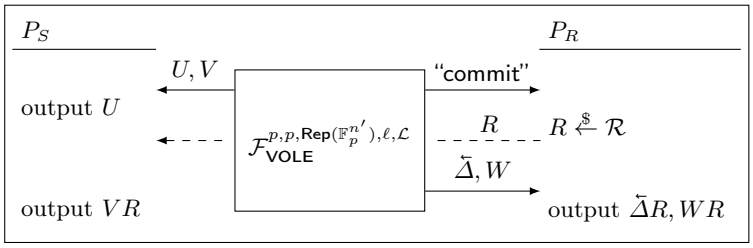


Fig. 12: Non-leaky maliciously secure subspace VOLE for the repetition code. If P_R receives "check failed" from the VOLE then the protocol is aborted immediately.

is the maximum number of distict OTs that can have the same tweak under t. For the TCR itself, τ_{max} will be the maximum number of evaluations $F_i(\tilde{x})$ where $t(i, \tilde{x})$ outputs a given tweak. For semi-honest security, $\mathcal{R}$ is unused; instead set $\epsilon = q^{-nc}$ and $M = 1$.

Proof. See the full version of this work.

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 ϵ-almost uniform family and $n' \geq n$. The advantage is bounded by $\epsilon M(p^n - 1)$.*

Proof. See the full version of this work.

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] 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 $\mathbb{F}_p$.

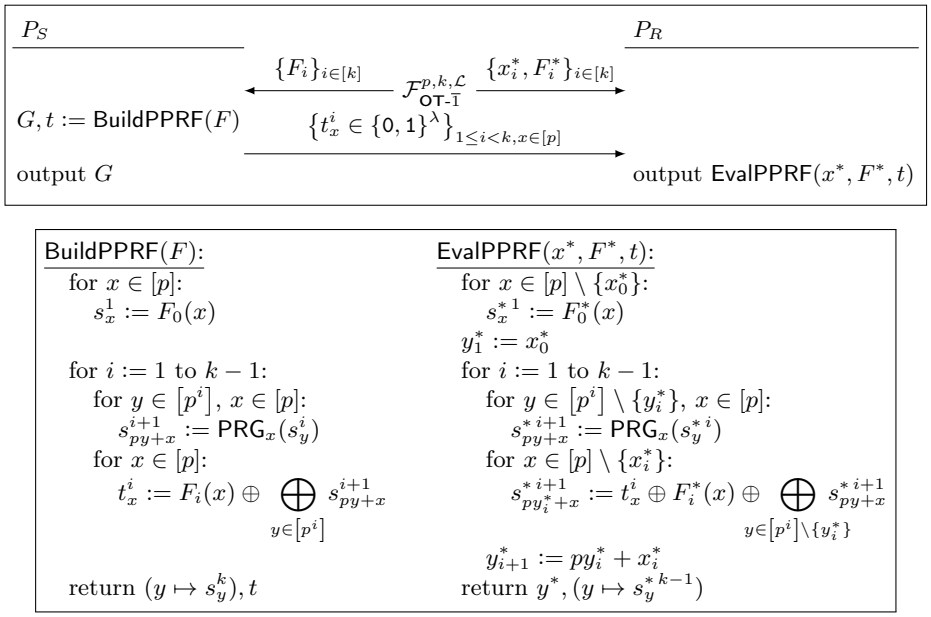


Fig. 13: Protocol for $\binom{q}{q-1}$-OT based on $\binom{p}{p-1}$-OT, using a punctured PRF.

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 efficiently construct $\binom{N}{N-1}$-OT. Our protocol (see Fig. 13) is based on the optimized version in [SGRR19], which generates $\binom{p^k}{p^k-1}$-OT from k $\binom{p}{p-1}$-OTs.

It depends on a $\mathsf{PRG}\colon \{0,1\}^\lambda \to \left(\{0,1\}^\lambda\right)^p$. The xth block of λ 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 $\mathcal{F}_{\mathsf{OT}\text{-}\bar{1}}^{p,k,\mathrm{Affine}(\mathbb{F}_p^k)}$, 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 yth 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 $t_x^i = \bigoplus_y s_{py+x}^{i+1}$ for each x, and uses the ith base OT to send all but one of these totals to P_R. Let y_i^* be the index of the node on the active path in layer i, i.e., the layer i node that P_R cannot learn. Then P_R will know every s_y^i except for $s_{y_i^*}^i$, so it can compute $s_{py_i^*+x}^{i+1} = t_x^i \oplus \bigoplus_{y\neq y_i^*} s_{py+x}^{i+1}$. Thus, it learns s_y^{i+1} for all $y \neq y_{i+1}^*$. In the full version, we prove that the leaves becomes the messages of an $\binom{p^k}{p^k-1}$-OT.

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.*

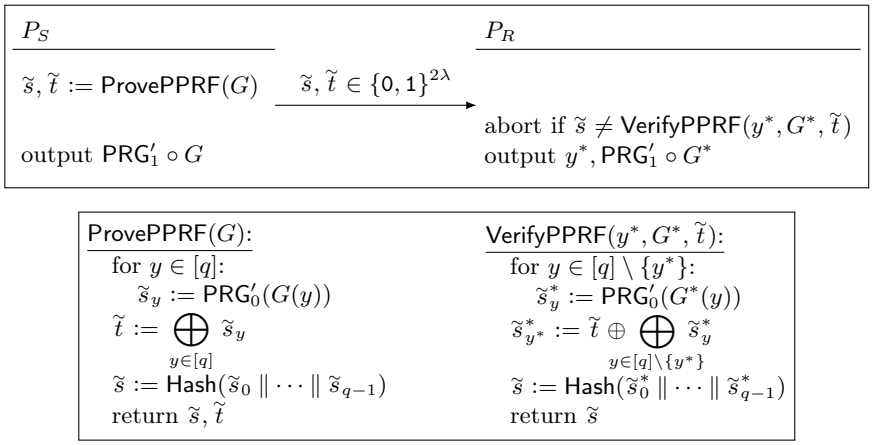


Fig. 14: Consistency checking for $\binom{q}{q-1}$-OT. This makes Fig. 13 maliciously secure.

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.

6.1 Consistency Checking

In Fig. 14, we present the consistency check from the maliciously secure $\binom{2^k}{2^k-1}$-OT of [BCG+19a]. We prove a stronger property of their check, that P_S can only check guesses for the x_i^*s individually, not all of them together, which shows that any possible selective abort attack is in $\mathrm{Affine}(\mathbb{F}_p^k)$. This assumes that that PRG is collision resistant for its whole output, so there are no $s \neq s'$ such that $\mathsf{PRG}_x(s) = \mathsf{PRG}_x(s')$ for all $x \in [p]$. 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 PRG'_0. In the consistency check, P_S sends the total of all $\widetilde{s}_y^k = \mathsf{PRG}'_0(s_y^k)$ so that P_R can reconstruct $\widetilde{s}_{y^*}^k$. P_R then evaluates a collision resistant hash of all the $\widetilde{s}_y^k$ and checks that it matches the hash from P_S. As we prove in the full version, this commits P_S to a single possibility for each $\widetilde{s}_y^k$.

Theorem 6.2. *Figure 14 (composed with Fig. 13) is a maliciously secure* $\mathcal{F}_{\mathsf{OT}\text{-}\overline{1}}^{q,1,\mathrm{Affine}(\mathbb{F}_p^k)}$ *in the* $\mathcal{F}_{\mathsf{OT}\text{-}\overline{1}}^{p,k,\mathrm{Affine}(\mathbb{F}_p^k)}$ *hybrid model.*

7 Implementation

We implemented[7] our $\binom{2}{1}$-OT semi-honest and malicious protocols in the libOTe library [Rin], so that we could assess efficiency and parameter choices. We focus

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

only on the case of binary fields ($p = 2$), as for this problem there is little 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 TCR hash using Prop. 2.7, with AES as the ideal cipher. To keep $\tau_{\max}$ low, we set $t(i, \tilde{x}) = \lfloor i/1024 \rfloor$, 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 cache.

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 Prop. 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 over $\mathbb{F}_2$. The second stage R' of the universal hash is over $\mathbb{F}_{2^k}$. It 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 also needs a uniform hash, and we use $\mathbb{F}_{2^{128}}$-multiplication of the tweak with the hash key.

The punctured PRF (Fig. 14) requires collision resistant primitives PRG, PRG', and Hash. For 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 (see full version). We use Blake2 [ANWW13] for PRG'[8] and 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].[9] The choice bits of SoftSpokenOT were derandomized immediately, as were the choice bits for Ferret, to provide the most

[8] H would also work, assuming that PRG'_0 concatenates two output blocks from H.

[9] Silent OT needs more than λ base OTs, and so as an optimization it generates them using KOS, which needs only λ 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	<u>298</u>	1637	15648
SoftSpoken ($k = 2$)	80009	64	**210**	815	7730	**255**	893	7985
SoftSpoken ($k = 3$)	53759	43	<u>223</u>	568	5208	322	677	5419
SoftSpoken ($k = 4$)	40008	32	261	<u>433</u>	3995	311	<u>530</u>	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	<u>2554</u>
SoftSpoken ($k = 8$)	20008	16	1259	1314	<u>1916</u>	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]	<u>127</u>	<u>0.10</u>	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 <u>underlined</u>. 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.

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 $6\times$ 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 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, 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.

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 *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 *EUROCRYPT 2017, Part II*, 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 *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.

CCG18. Ignacio Cascudo, René Bødker Christensen, and Jaron Skovsted Gundersen. Actively secure OT-extension from q-ary linear codes. In *SCN 18*, volume 11035 of *LNCS*, pages 333–348. Springer, September 2018.

CCL15. Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In *CRYPTO 2015, Part II*, LNCS, pages 3–22. Springer, August 2015.

CDD+16. Ignacio Cascudo, Ivan Damgård, Bernardo David, Nico Döttling, and Jesper Buus Nielsen. Rate-1, linear time and additively homomorphic UC commitments. In *CRYPTO 2016, Part III*, pages 179–207. August, 2016.

CRR21. Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In *CRYPTO 2021, Part III*, pages 502–534. Springer, August 2021.

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.

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 *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, 2021.

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.

SGRR19. Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In *ACM CCS 2019*, pages 1055–1072, November 2019.

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.