Actively Secure Arithmetic Computation and VOLE with Constant Computational Overhead*

Benny Applebaum[1][0000−0003−4792−369X] and Niv Konstantini[1]

Tel Aviv University, Tel Aviv, Israel
bennyap@post.tau.ac.il, NivKonst@gmail.com

Abstract. We study the complexity of two-party secure arithmetic computation where the goal is to evaluate an arithmetic circuit over a finite field $\mathbb{F}$ in the presence of an active (aka malicious) adversary. In the passive setting, Applebaum et al. (Crypto 2017) constructed a protocol that only makes a *constant* (amortized) number of field operations per gate. This protocol uses the underlying field $\mathbb{F}$ as a black box, makes black-box use of (standard) oblivious transfer, and its security is based on arithmetic analogs of well-studied cryptographic assumptions. We present an actively-secure variant of this protocol that achieves, for the first time, all the above features. The protocol relies on the same assumptions and adds only a minor overhead in computation and communication.

Along the way, we construct a highly-efficient Vector Oblivious Linear Evaluation (VOLE) protocol and present several practical and theoretical optimizations, as well as a prototype implementation. Our most efficient variant can achieve an asymptotic rate of 1/4 (i.e., for vectors of length w we send roughly $4w$ elements of $\mathbb{F}$), which is only slightly worse than the passively-secure protocol whose rate is 1/3. The protocol seems to be practically competitive over fast networks, even for relatively small fields $\mathbb{F}$ and relatively short vectors. Specifically, our VOLE protocol has 3 rounds, and even for 10K-long vectors, it has an amortized cost per entry of less than 4 OT's and less than 300 arithmetic operations. Most of these operations (about 200) can be pre-processed locally in an offline non-interactive phase. (Better constants can be obtained for longer vectors.) Some of our optimizations rely on a novel intractability assumption regarding the non-malleability of noisy linear codes, that may be of independent interest.

Our technical approach employs two new ingredients. First, we present a new information-theoretic construction of Conditional Disclosure of Secrets (CDS) and show how to use it in order to immunize the VOLE protocol of Applebaum et al. against active adversaries. Second, by using elementary properties of low-degree polynomials, we show that, for some simple arithmetic functionalities, one can easily upgrade Yao's garbled-circuit protocol to the active setting with a minor overhead while preserving the round complexity.

Keywords: Foundations · Protocols · Secure Computation.

* Supported by the Israel Science Foundation grant no. 2805/21. The full version of this paper appears in [10].

1 Introduction

Secure multiparty protocols (MPC) allow a set of parties to jointly compute a function over their inputs while keeping those inputs private. In many situations, the underlying sensitive data is *numerical*, and the computation can be naturally expressed as a sequence of arithmetic operations such as addition, subtraction, and multiplication.[1] This calls for *secure arithmetic computation*, namely secure computation of functions defined by arithmetic operations. It is convenient to represent such a function by an *arithmetic circuit*, which is similar to a standard Boolean circuit except that gates are labeled by addition, subtraction, or multiplication. It is typically sufficient to consider such circuits that evaluate the operations over a large *finite field* $\mathbb{F}$, since arithmetic computations over the integers or (bounded precision) reals can be reduced to this case. Computing over finite fields (as opposed to integers or reals) can also be a feature, as it is useful for applications in threshold cryptography (see, e.g., [31]).

It is always possible to reduce arithmetic computation to Boolean computation by implementing each arithmetic operation by Boolean circuit. However, this approach leads to a large blow-up in the circuit size.[2] Thus we strive for "purely arithmetic" solutions that avoid such an emulation. Specifically, following [5], we strive for a protocol that achieves a *constant computational overhead*. That is, we would like to securely evaluate any arithmetic circuit C over any finite field $\mathbb{F}$, with a computational cost (on a RAM machine) that is only a constant times bigger than the cost of performing $|C|$ field operations with no security at all. Here we make the standard convention of viewing the size of C also as a security parameter, namely the view of any adversary running in time $\mathsf{poly}(|C|)$ can be simulated up to a negligible error in $|C|$.

1.1 ADINZ: Constant Overhead with Passive Security

In [5] (hereafter referred to as ADINZ) it was shown that it is possible to realize 2-party arithmetic MPC with constant computational overhead in the presence of a *passive* (aka semi-honest) adversary. Specifically, ADINZ introduced the *Vector Oblivious Linear Evaluation* (VOLE) functionality in which the sender holds a pair of vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}^w$ and the receiver holds a scalar $x \in \mathbb{F}$ and gets as an output the vector $x\boldsymbol{a} + \boldsymbol{b}$, and showed how to (1) realize VOLE of width w with complexity of $O(w)$ and (2) how to use VOLE to realize *batch Oblivious Linear Evaluation* (batch-OLE) of length n with complexity $O(n)$. The latter functionality takes a pair of vectors $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{F}^n$ from the sender and a vector $\boldsymbol{y} \in \mathbb{F}^n$ from the receiver and delivers to the receiver the vector $\boldsymbol{y} \odot \boldsymbol{c} + \boldsymbol{d}$ where $\odot$ stands for entry-wise multiplication. Both the VOLE functionality and batch-OT functionality naturally extend the Oblivious Linear Evaluation

[1] More complex numerical computations can typically be efficiently reduced to these simple ones, e.g., by using suitable low-degree approximations.

[2] For example, for the case of finite fields with n-bit elements, the size of the best known Boolean multiplication circuits is $\omega(n \log n)$.

(OLE) functionality [44,40] that corresponds to the case where $w = 1$ or $n = 1$. Moreover, OLE, VOLE, and batch-OLE can be viewed as the arithmetic versions of oblivious transfer (OT), string-OT, and batch-OT, respectively. Indeed, just like in the binary setting, securely computing an ℓ-size arithmetic circuit reduces via an arithmetic-GMW construction [40] to the task of securely computing batch-OLE of length ℓ. Based on this reduction, ADINZ constructed a constant-overhead MPC protocol for general arithmetic circuits.

The security of the ADINZ protocols is based on arithmetic analogs of well-studied cryptographic assumptions. Concretely, for the VOLE protocol, it suffices to assume the existence of a linear-time computable "code" over $\mathbb{F}$ for which noisy codewords are pseudorandom. Since a conservative choice of *constant-rate* noise suffices, one can instantiate this LPN-type assumption based on an arithmetic variant of Alekhnovich's assumption [2] or based on the codes of Druk and Ishai [25]. The batch-OLE protocol is based on an arithmetic version of a $\mathbf{NC^0}$ polynomial-stretch PRG [38,3,11,9]. (See [53] for security analysis of these two assumptions in the arithmetic setting.)

The ADINZ protocols also enjoy several useful properties. They make only black-box access of the field $\mathbb{F}$ and their arithmetic complexity (the number of field operations) grows linearly with the circuit size of the underlying functionality and is *independent* of the size of $\mathbb{F}$.[3] In addition, all protocols make a *black-box* use of a standard OT channel. In fact, in the hybrid-OT model, they achieve information-theoretic privacy against a corrupted VOLE/batch-OLE sender.[4] As advocated in [39,40], designing protocols in the OT-hybrid model yield several advantages such as native pre-processing [14], simple amortization via OT-extension [14,34,12], and the ability to rely on different concrete implementations (including UC-secure ones) under a variety of computational or physical assumptions. Moreover, black-box usage is typically a necessary condition for obtaining practical efficiency. Indeed, the VOLE protocol of ADINZ makes only light-weight linear-algebraic operations and operates in a constant number of rounds, and a prototype implementation appeared in [5]. It should be mentioned that in the past few years the VOLE primitive has turned out to be an important building block with numerous applications such as secure computation of linear algebraic computations [43], round-efficient secure arithmetic computation via arithmetic garbling [8], secure keyword search and set intersection [26,30], zero-knowledge proofs for arithmetic circuits [17,19,23,51,13], and non-interactive secure computation [19,23]. (See [40,5,17] and references therein.)

1.2 Actively Secure Arithmetic MPC with Constant Overhead?

Unfortunately, the ADINZ protocols are only passively secure, and, as we will later see, an active adversary can completely break the privacy of both protocols (the VOLE protocol and the batch-OLE). One can probably construct

[3] The protocol additionally uses standard "bit-operations" whose complexity is dominated by the field operations.

[4] We mention that the aforementioned assumptions are not known to imply OT.

an actively-secure protocol by combining ADINZ with constant-overhead arithmetic zero-knowledge proofs via an arithmetic version of the GMW-compiler [32]. The elegant work of Bootle et al. [15] provides such a zero-knowledge protocol. However, this approach inherently makes a non-BB use of the underlying OT protocol. Also, the protocol of [15] has a super-constant round complexity.

For the special case of VOLE, the breakthrough results of Boyle et al. [17,18] yield an actively-secure realization with constant overhead. However, their protocols are based on strong LPN-type assumptions with *sub-constant* noise rate. The protocol can be based on OT in a black-box way [18] at the expense of further strengthening the underlying intractability assumption to a *leaky LPN* assumption and by making additional use of *correlation robust hash functions*.

To summarize, to the best of our knowledge, it is currently unknown how to realize batch-OT (or general arithmetic MPC) with active security, constant overhead, and black-box access to OT, regardless of the underlying assumption. For VOLE, the only known constructions either make a non-BB use of OT or rely on relatively strong intractability assumptions such as leaky-LPN with sub-constant noise and correlation robust hash functions. Our goal in this work is to avoid these limitations and derive an actively-secure arithmetic MPC protocol with constant overhead that enjoys all the features of the ADINZ protocol.

2 Our Contribution

We resolve the above question in the affirmative by presenting actively-secure variants of the ADINZ protocols for VOLE, batch-OLE, and general arithmetic secure computation, that enjoy all the additional features and are based on the same assumptions. While our main focus is theoretical, we also present several practical optimizations to the VOLE protocol that make use of less conservative intractability assumptions. Details follow.

2.1 The VOLE protocols

Just like [5], we rely on the existence of *fast pseudorandom matrix* $M \in \mathbb{F}^{m \times k}$ where $m > k$ is a fixed polynomial in k (say $m = k^3$). Here "fast" means the mapping $u \mapsto M \cdot u$ can be computed by making only $O(m)$ arithmetic operations, and "pseudorandom" means that if we take a random vector in the image of M, and add a random μ-sparse noise vector to it, the resulting vector is computationally indistinguishable from a truly random vector over $\mathbb{F}^m$. The noise rate μ can be taken to be a constant, e.g., $1/4$. There are several candidates for such fast pseudorandom matrices (see the discussion after Assumption 3). We prove the following theorem.

Theorem 1 (informal). *Based on the fast pseudorandom matrix assumption, the VOLE functionality of width w can be realized with active security in the OT-hybrid model with arithmetic complexity of $O(w)$ and with perfect security against an active adversary that corrupts the Sender and computational security against the Receiver.*

This protocol (and all the protocols constructed in this work) makes black-box use of the underlying field and is therefore fully arithmetic in the sense of [40]. (One can also derive the stronger form of arithmetic MPC of [4] by instantiating the OT channel with an "arithmetic OT"). In addition, all the protocols that are constructed in this work admit a straight-line black-box simulator. In the 2-party setting, the existence of such simulators implies that the protocol is UC-secure as follows from [41, Theorem 1.5] and [42].

As already mentioned, Theorem 1 is the first to achieve constant overhead and black-box dependency in the OT based on a conservative constant-rate noise LPN-type assumption. Moreover, to the best of our knowledge, this is the first construction that achieves constant overhead and statistical Sender security in the OT-hybrid model, regardless of the underlying assumption,

Remark 1 (realizing the OT-channels with constant overhead). For the sake of communication/ computational complexity, we charge every bit that is passed over the OT channel as a single bit/bit-operation. As already observed in [38,5], when each OT message is sufficiently long compared to the security parameter (which is the case in our protocols), such OT-channels can be realized securely based on an *arbitrary* OT protocol with the aid of a linear-time computable linear-stretch PRG. The existence of the latter follows from the binary version of the fast pseudorandom matrix assumption (see [7,38,5]). In particular, by using any UC-secure OT protocol in the CRS setting (e.g., [46]), we derive UC-secure implementations of our protocols in the standard model. (See See the full version for more details.)

Optimizations: VOLE1 and VOLE2. Motivated by the rich applications of VOLE, we present several optimizations for the protocol. At the extreme, we present a VOLE protocol (VOLE1) that can achieve an asymptotic rate of 1/4 (i.e., the communication is dominated by sending roughly $4w$ elements of $\mathbb{F}$), which is only slightly worse than the passively-secure protocol whose rate is 1/3. Assuming that the OT consumes 2 rounds, VOLE1 has 3 rounds of computation. The protocol is provably secure against a computationally-unbounded sender, provably secure against a passive receiver, but only heuristically secure against an active receiver. That is, we conjecture that it achieves security against an active receiver, but do not have a security reduction to a clean intractability assumption. As a compromise, we introduce another protocol VOLE2 that slightly downgrades the asymptotic rate of 1/5, has 6 rounds, but can be proved secure based on a new, yet plausible, intractability assumption.[5]

The Correlated Noisy-Codeword Hardness Assumption. Our assumption intuitively asserts that given a random noisy codeword $\boldsymbol{c}$ sampled from a code T with noise pattern $\boldsymbol{e}$, it is hard to efficiently generate a new noisy codeword $\boldsymbol{d}$ whose noise pattern $\boldsymbol{e}'$ is non-trivially correlated with the noise pattern $\boldsymbol{e}$ in the

[5] In fact, even our most conservative protocol (VOLE3) that proves Theorem 1 has an assymptotic rate of 1/5 and its amortized computational complexity is roughly the same. However, VOLE3 achieves this only over significantly longer vectors.

following sense. The new noise vector $\boldsymbol{e}'$ is "far" from being a scalar multiple of $\boldsymbol{e}$ but it agrees with the original noise vector $\boldsymbol{e}$ with respect to the set of non-noisy coordinates $I = \{i : \boldsymbol{e}_i = 0\}$ of $\boldsymbol{e}$, i.e., $\boldsymbol{d}[I]$ is in the span of $T[I]$. Observe that such a noisy codeword can be generated by sampling a vector in the column span of $(T|\boldsymbol{c})$ and then modifying ℓ entries with the hope that all these entries fall out of the set of clean coordinates I. Such an attack succeeds with probability μ^ℓ, and our assumption states that one cannot do much better than this. (See Section 6 for a formal statement.) We believe that this assumption may be of independent interest and provide some evidence towards its validity in the full version [10].

Implementation and concrete complexity. The computational overhead of VOLE1 and VOLE2 are essentially the same. In the full version [10], we analyze the concrete complexity of these protocols when instantiated with the same building blocks that were used in the passive setting of [5], suggest several practical optimizations, and present an implementation of the protocols. We show that the computational overhead compared to the passive version is minor (less than 20%). Furthermore, even for relatively short vectors of length 10000, our protocols have an amortized cost per VOLE entry of fewer than 4 OTs and less than 300 arithmetic operations (additions and multiplications). We use novel techniques (e.g., sparse LU-decomposition) to push most of these operations (about 200) to a *non-interactive* offline phase that can be pre-processed locally based only on the public parameters and local random tape. As a result, this preprocessing can be applied even before each party knows who will be her partner for the computation. The protocol is also very cheap for the receiver and requires in the online phase less than 10 arithmetic operations and 4 OT's per VOLE entry. (The sender's online amortized computation consists of 4 OT's and less than 80 arithmetic operations per entry.) Such a receiver-efficient protocol is especially useful for applications like VOLE-based zero-knowledge proofs (e.g., [23]) in which the verifier plays the receiver and the prover plays the VOLE sender.

We believe that our protocol may be practically competitive over fast networks even for relatively small fields $\mathbb{F}$ and relatively short vectors. (Think for example, of an arithmetic zero-knowledge for a circuit that contains few 10K's gates, which, by [23], translates into a single VOLE of comparable length.) When comparing our protocols to the alternative compressed-VOLE-based solution [17,18], we see that the latter achieves a better rate of $1/2$, but its amortization point seems to "kick in" only for longer vectors (due to the use of an "internal-MPC" protocol for securely realizing "short VOLE-correlations"). Thus, this approach is in a sense complementary to ours, and it will be interesting to study the combination of the two.[6] We also expect that additional optimizations

[6] The current implementations of the compressed-VOLE-based solution are either restricted to the binary field [18] or achieve passive security [48] and so we cannot compare the actual performance of our implementation against a compressed-VOLE-based implementation. Indeed, to the best of our knowledge, it seems that our work provides the first implementation of actively-secure VOLE over large fields.

of our implementation and the underlying building blocks will further improve the computational cost.

2.2 The batch-OLE protocol

The ADINZ [5] protocol for batch-OLE is based on the existence of *pseudorandom generator* (PRG) $G : \mathbb{F}^k \to \mathbb{F}^n$ with polynomial stretch, e.g., $n = k^2$, that is computable by a constant-depth ($\mathbf{NC^0}$) arithmetic circuit.[7] Candidate constructions are studied in [5,53]. We prove the following theorem.

Theorem 2 (informal). *Assuming the existence of an* $\mathbf{NC^0}$ *arithmetic PRG with polynomial stretch, the batch-OLE functionality of length n can be realized with active security in the VOLE-hybrid model with arithmetic complexity of $O(n)$ and by making a single call to an ideal k-length $O(n/k)$-width VOLE where the batch-OLE receiver (resp., sender) plays the role of the VOLE receiver (resp., sender). The protocol is perfectly secure against an active adversary that corrupts the receiver and computationally secure against an adversary that actively corrupts the sender. Moreover, the protocol has a constant number of rounds.*

Here the k-length $O(n/k)$-width VOLE functionality consists of k copies of $O(n/k)$-width VOLE. The protocol has 4 rounds (counting VOLE as a 2-round protocol). In [27] it is shown that the task of securely computing an arithmetic circuit C with active security reduces to batch-OLE of length $O(|C|)$ with constant computational overhead while preserving information-theoretic security. Combining this with Theorems 2 and 1, we derive the following corollary.

Corollary 1. *Assuming a fast pseudorandom matrix and an* $\mathbf{NC^0}$ *PRG with polynomial stretch, any two-party functionality that is computable by an arithmetic circuit C can be realized with arithmetic complexity of $O(|C|)$ in the OT-hybrid model while providing information-theoretic security for one party and computational security for the other party. The protocol makes black-box use of the underlying field.*

The corollary extends to any constant number of parties via standard reductions (e.g., [27]).

2.3 Technical Overview of the VOLE protocols

We briefly present some of the main technical ideas behind our constructions starting with the VOLE protocols.

[7] The exact level of stretch is not important since one can transform a given PRG with a polynomial stretch of $n = k^c$ for some $c > 1$, to a PRG with a stretch of $n = k^{c'}$ for an arbitrary constant $c' > c$ while increasing the depth of the circuit by a constant factor (see, e.g., [3]).

The passive-VOLE protocol. The VOLE protocol of ADINZ17 is based on a protocol for "reverse VOLE" (RVOLE) functionality in which Bob holds a vector $\boldsymbol{a} \in \mathbb{F}^w$, Alice holds a vector $\boldsymbol{b} \in \mathbb{F}^w$ and a scalar $x \in \mathbb{F}$ and the goal is to deliver the value of $x\boldsymbol{a} + \boldsymbol{b}$ to Bob. Roughly, Bob sends to Alice an encryption $\boldsymbol{c} = E(\boldsymbol{a}) \in \mathbb{F}^m$ of $\boldsymbol{a}$ which is based on the fast pseudorandom matrix. (E also depends on some random field elements that are omitted for simplicity.) This encryption is "almost-linear" and so Alice can homomorphically compute a new ciphertext $\boldsymbol{d} = E(x\boldsymbol{a} + \boldsymbol{b}) \in \mathbb{F}^m$ by applying linear operations over $\boldsymbol{c}$. However, this ciphertext cannot be sent to Bob since it leaks information about x and $\boldsymbol{b}$. In particular, if Bob sees a coordinate of $\boldsymbol{d}$ that was "noisy" in the original ciphertext $\boldsymbol{c}$ he can efficiently extract the private input of Alice. (See the full version [10].) To fix the problem, we let Bob read only the entries of $\boldsymbol{d}$ for which $\boldsymbol{c}$ is non-noisy.[8] Of course, Bob has to hide these locations, and so, for each entry $i \in [m]$, the parties invoke a standard 1-out-of-2 (or even all-or-nothing [47]) OT-channel where Alice sends the pair $(\boldsymbol{d}_i, \perp)$ and Bob's selection bit determines whether to read $\boldsymbol{d}_i$ or to receive a $\perp$ symbol. While in the passive setting Bob can be trusted to read only the clean locations, an actively corrupt Bob can simply read all the entries of the vector $\boldsymbol{d}$, and completely recover Alice's input.

Securing the protocol against active Bob via CDS. Let us denote by T the matrix that corresponds to the linear part of the encryption E, i.e., T maps a plaintext of length w (and a vector of k random field elements) to a vector of length m whose noisy version corresponds to a ciphertext. Our first observation is that the above protocol remains secure if and only if the entries $I \subset [m]$ that Bob reads in the OTs satisfy the following condition: (*) The ciphertext $\boldsymbol{c}$ restricted to I is in the span of $T[I]$, the sub-matrix of T whose rows are indexed by I. (As a sanity check, observe that when Bob is honest the set I of "clean" coordinates satisfies the condition.) At this point, it is natural to try and extend the protocol with some form of zero-knowledge proof in which Bob proves that I satisfies the above condition. However, since I corresponds to Bob's input to the OTs (specifically, Bob's "selection bits") such a proof system seems to lead to a non-BB use of the OTs. To avoid this complication, we take an alternative route and make use of a special-tailored Conditional Disclosure of Secret (CDS) Protocol [28].

Roughly speaking, in such a protocol Alice chooses a random secret s and, for each index $i \in [m]$, Alice sends over the ith OT-call a pair of field elements $(\boldsymbol{d}_i, \boldsymbol{z}_i)$, and Bob has to choose whether to learn $\boldsymbol{d}_i$ or $\boldsymbol{z}_i$. For a selection vector I, Bob learns the vectors $(\boldsymbol{d}_i)_{i \in I}$ and $(\boldsymbol{z}_i)_{i \notin I}$. By design, the latter vector reveals the secret s if and only if I satisfies the (*) condition. Thus the CDS protocol effectively limits the query access to the OT's, and turns it into so-

[8] This information suffices to recover the plaintext $x\boldsymbol{a} + \boldsymbol{b}$ since the encryption internally employs a suitable error-correcting code. Indeed, [5] show how to combine a fast pseudorandom matrix with a linear-time error-correcting code and derive a linear-time encodable code that is pseudorandom under random noise but can be decoded in linear-time in the presence of random erasures.

called a *generalized OT* (GOT) [35,50,49,29].[9] Below, we present such a CDS protocol that achieves a constant computational overhead for both the sender and the receiver and information-theoretic security. Given such a protocol, we can immunize the RVOLE protocol by letting Alice re-encrypt her ciphertext $\boldsymbol{d}$ under the secret s. This approach yields only computational security since the key s, which is a single field element, is shorter than the vector $\boldsymbol{d}$. (A CDS with longer secrets would lead to a super-constant computational overhead.) To achieve information-theoretic security, we note that it suffices to use s to encrypt the scalar x of the RVOLE protocol. That is, Alice invokes the modified RVOLE protocol (with the CDS mechanism) over the inputs $x + s$ and $\boldsymbol{b}$. We show that if Bob's selection strategy I satisfies the (*) condition, we can extract his inputs and perfectly simulate his view based on $x\boldsymbol{a} + \boldsymbol{b}$. If Bob's strategy I does not satisfy (*), he learns the vector $\boldsymbol{b}$ but x remains completely hidden, and we can perfectly simulate his view by sending $\boldsymbol{a} = 0^w$ to the ideal RVOLE functionality. We refer to the resulting protocol as the *modified-RVOLE* protocol (See Section 5).

Constructing the CDS. Our construction of the CDS is linear-algebraic in nature. As a starting point, we employ the following standard fact: Fix a matrix T and a vector $\boldsymbol{c}$. Suppose that we "encrypt" a secret s by sampling a random row vector $\boldsymbol{z}$ in the co-kernel of T, and publishing the "ciphertext" $s + \langle \boldsymbol{z}, \boldsymbol{c} \rangle$. If $\boldsymbol{c}$ is spanned by T the ciphertext equals to s, on the other hand, if $\boldsymbol{c}$ is *not* spanned by T, the ciphertext information theoretically hides s. Thus, *linear independence* is translated into *secrecy*. We can extend this idea to the CDS setting where we wish to reveal the secret iff $\boldsymbol{c}[I]$ is in the span of $T[I]$ for a subset I. To do this we reveal, for each $i \notin I$, the ith entry of our randomizer, $\boldsymbol{z}_i$, and send the ciphertext $s + \langle \boldsymbol{z}, \boldsymbol{c} \rangle$ in the clear. Given this information, one can map the ciphertext to $s + \langle \boldsymbol{z}[I], \boldsymbol{c}[I] \rangle$ which is decryptable if and only if $\boldsymbol{c}[I] \in \text{span}(T[I])$. While the above construction achieves privacy and correctness, it is not clear whether it achieves constant computational overhead. Indeed, we do not know how to sample a random vector in the co-kernel of T in linear time. Fortunately, there is a simple fix. To achieve a linear-time construction, we uniformly sample $\boldsymbol{z}$ from the entire space (without limiting to the co-kernel) and append to the CDS the value $\boldsymbol{z} \cdot T$ as a public value. Since right multiplication in T can be done in linear time, we can also left-multiply by T in linear time (following the "generalized transposition principle" [16,38]) and so this variant can be realized with constant computational overhead. It is not hard to show that correctness and privacy still hold. (See the full version [10] for a formal definition of CDS and for details about the construction.)

[9] GOT allows Bob to retrieve a subset of the messages of Alice that are "authorized" according to some predicate P. Previous constructions were either based on decomposable randomized encoding (aka private-simultaneous messages protocols) [35] or on secret-sharing [50,49,29]. We generalize these approaches by using CDS which is strictly weaker than both primitives.

Securing the protocol against active Alice? We move on and consider an actively-corrupted Alice. Clearly, even if Alice deviates from the protocol and does not compute the vector $\boldsymbol{d}$ properly, her view is still simulatable since all that she sees is a semantically-secure ciphertext $\boldsymbol{c}$. However, such misbehavior may lead Bob to abort and it is not fully clear how to simulate this case. Specifically, let us assume that Alice misbehaves and generates a vector $\boldsymbol{d} \notin \mathsf{colspan}\left(T|\boldsymbol{c}\right)$. The simulator detects this and can send an "abort" to the ideal functionality. However, in the real execution, Bob aborts only if his I-partial view is inconsistent, namely, if $\boldsymbol{d}[I] \notin \mathsf{colspan}\left((T|\boldsymbol{c})[I]\right)$ where $I = I(\boldsymbol{e})$ is the set of non-noisy coordinates in $\boldsymbol{e}$. To make the problem concrete, consider a malicious Alice that honestly computes $\boldsymbol{d}$ and then adds noise to the first coordinate of $\boldsymbol{d}$. In this case, the above simulator sends an abort, but in the real protocol, Bob aborts only if the first coordinate is in $I(\boldsymbol{e})$ which happens with constant probability $1 - \mu$. We present several solutions to this problem with different levels of efficiency.

1. The first solution is heuristic: We simply assume that the protocol is secure as it is. As evidence, we can prove this statement for the original RVOLE protocol (without the CDS) based on a variant of the Correlated Noisy-Codeword Hardness Assumption. (See the full version [10].) By using a straightforward reduction from VOLE to RVOLE [5], this leads to the VOLE1 protocol. (See Section 5.)
2. In the second solution, we first employ the modified-RVOLE protocol over random vectors $\boldsymbol{a}'$ and $\boldsymbol{b}'$ (this guarantees the ability to perfectly simulate an "abort" event), then use a small sub-protocol in which Alice proves that her CDS secret is independent of Bob's input (based on a simple commitment), and finally, we shift the vectors back to the real inputs vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ by exploiting the linearity of the VOLE functionality. To prove security we still need to extract Alice's input in the event that the protocol does not abort. For this, we rely on the aforementioned "Correlated Noisy-Codeword" intractability assumption. In a nutshell, we show that under this assumption, the simulator who is given a malformed ciphertext $\boldsymbol{d}$ either identifies that Bob would abort in the real execution or successfully extracts an effective input for Alice. Thus the security of the protocol can be based on the Correlated Noisy-Codeword assumption and on the fast pseudorandom matrix assumption. We refer to the resulting protocol as VOLE2 and note that it adds only a minor computational and communication overhead over RVOLE1. (See Section 6.)
3. Finally, our most conservative solution (VOLE3) relies solely on the fast pseudorandom matrix assumption. The starting point is again the modified-RVOLE protocol. We begin by observing that there exist efficient tests that determine whether Alice's ciphertext $\boldsymbol{d}$ is "valid" and whether Alice's vector of CDS messages $\boldsymbol{z}$ is "valid". Furthermore, when $\boldsymbol{d}$ and $\boldsymbol{z}$ are both valid, we can extract a unique effective input for Alice and properly simulate the protocol. We also note that there exists a strategy for Bob that detects (with probability 0.5) whether Alice cheats. Indeed, in the OT phase Bob can toss a coin and ask with probability $1/2$ to receive the vector $\boldsymbol{d}$ (by using

$I = 1^m$) and with probability 1/2 the vector $\boldsymbol{z}$ (by using $I = 0^m$) and check validity. When running in this "detection mode" Bob effectively gives up on the computation and just verifies whether Alice misbehaves or not. Note that Bob's decision is taken only in the OT phase and is hidden from Alice, and so effectively Alice first "commits" to strategy (cheat or not), and only then Bob decides whether to "call her bluff". Furthermore, even when Bob acts as a detector, we can fully simulate his view (since the protocol is actively-secure against any deviation of Bob). We will exploit these observations to obtain a "silent" cut-and-choose version of the protocol. Specifically, we realize W-width VOLE based on many calls to the modified-RVOLE protocol over shorter vectors of width $w \ll W$. Ignoring some technical details, we "sacrifice" a small fraction of these calls for cheating-detection[10], and glue together the remaining copies via a linear-time computable linear exposure-resilient function [20] (also known as perfect deterministic extractors for bit-fixing sources). Such functions can be constructed based on linear-time encodable codes. (See Section 7.)

2.4 Technical Overview of the batch-OLE protocol

The ADINZ passive batch-OLE protocol relies on an arithmetic analog of Beaver's OT extension [14]. Given an arithmetic PRG $G : \mathbb{F}^k \to \mathbb{F}^n$, the idea is to realize a pseudorandom batch-OLE in which Alice holds the vectors $\boldsymbol{c}$ and $\boldsymbol{d}$ of length n, Bob holds a seed of a PRG $\boldsymbol{x}$ of length k and the functionality stretches the seed $\boldsymbol{x}$ to a pseudorandom vector $\boldsymbol{y} = G(\boldsymbol{x})$ of length n, and delivers to Bob the value $\boldsymbol{y} \odot \boldsymbol{c} + \boldsymbol{d}$ where $\odot$ stands for entry-wise multiplication. The latter functionality f_G is realized by using an arithmetic variant of Yao's protocol. Specifically, Alice prepares an arithmetic decomposable affine randomized encoding (DARE) [36,6] (aka arithmetic garbled-circuit) of f_G and sends the "keys" that correspond to her entries. Bob recovers the keys of his inputs by making k calls to VOLE of width $w = O(n/k)$ and recovers the output. When the PRG is computable in $\mathbf{NC^0}$ the protocol can be realized with constant computational overhead. Clearly, the protocol is insecure in the presence of an actively corrupted Alice who can send a malformed encoding that corresponds to a different function. This well-known problem is extensively studied in the binary setting. We note that our concrete setting admits a simple and highly efficient solution.

Specifically, we strongly exploit the following properties: (1) We only care about the case where Bob's input $\boldsymbol{x}$ is chosen at random; (2) When Bob decodes the, possibly malformed, DARE (or the garbled circuit), each output of the computation can be written as a low-degree polynomial whose degree corresponds to

[10] Interestingly, this detection is performed "silently": To test a session Bob just plays this session in a "detection mode". In contrast, in typical cut-and-choose-based solutions, Bob asks Alice to "open" a session. In fact, in our protocol we can even hide from Alice which sessions were tested by Bob.

the degree of f_G which is very small (constant) compared to the field size $|\mathbb{F}|$.[11]
(3) The function f_G is linear in Alice's inputs.

Equipped with these observations, we run an extended variant of the passively-secure protocol in which Alice holds the pair $(\boldsymbol{c}, \boldsymbol{d})$ and another random pair of vectors $(\boldsymbol{c}', \boldsymbol{d}')$ of similar length, and Bob learns $\boldsymbol{y} \odot \boldsymbol{c} + \boldsymbol{d}$ and $\boldsymbol{y} \odot \boldsymbol{c}' + \boldsymbol{d}'$. Let us focus, for simplicity on the first output of f_G. By applying the decoder, Bob learns the values z_1 and z'_1 which are supposedly equal to $\boldsymbol{y}_1 \cdot \boldsymbol{c}_1 + \boldsymbol{d}_1$ and to $\boldsymbol{y}_1 \cdot \boldsymbol{c}'_1 + \boldsymbol{d}'_1$ where where $\boldsymbol{y}_1 = G(\boldsymbol{x})$. Assuming that Alice behaves properly, Bob can now compute the value of $\boldsymbol{y}_1 \cdot L(\boldsymbol{c}_1, \boldsymbol{c}'_1) + L(\boldsymbol{d}_1, \boldsymbol{d}'_1)$ for any linear combination L. The idea is to challenge Alice with a random non-trivial L and ask her to send $c = L(\boldsymbol{c}_1, \boldsymbol{c}'_1)$ and $d = L(\boldsymbol{d}_1, \boldsymbol{d}'_1)$, and let Bob check whether $L(z_1, z'_1) = c\boldsymbol{y}_1 + d$, and abort if the test fails.

First, observe that Alice's additional messages do not leak any information (since $\boldsymbol{c}'_1$ and $\boldsymbol{d}'_1$ mask the values of $\boldsymbol{c}_1$ and $\boldsymbol{d}_1$). Next, by using simple linear-algebraic arguments, we show that if Alice deviates from the protocol she will get caught except with probability $O(D/|\mathbb{F}|)$ where D is the degree of the PRG G. To see this, assume that Alice sends malformed garbled circuits for Bob's first outputs of f_G. This means that Bob computes $z_1 = Q(\boldsymbol{x})$ and $z'_1 = Q'(\boldsymbol{x})$ for some degree-D multivariate polynomials $Q(\cdot)$ and $Q'(\cdot)$ that are not both in the span of $\{G_1(\cdot), 1\}$ (e.g., there are no scalars $\boldsymbol{c}_1, \boldsymbol{d}_1$ for which $Q(\boldsymbol{x}) = \boldsymbol{c}_1 G(\boldsymbol{x}) + \boldsymbol{d}_1$). Consequently, if we take a random linear combination L of $Q(\cdot)$ and $Q'(\cdot)$, the resulting polynomial $L(Q, Q')$ almost surely falls out of the span of $\{G_1(\cdot), 1\}$. In this case, no matter how the scalars c, d are chosen by Alice, the polynomial $L(Q(\cdot), Q'(\cdot))$ will not be equal to the polynomial $cG_1(\cdot) + d$. Since both polynomials are of degree at most D, they will disagree over a random point $\boldsymbol{x}$, except with probability $D/|\mathbb{F}|$, and so Bob will almost surely catch the cheating. (See Section 8 for more details.)

As already mentioned this analysis crucially relies on the low-degree feature of the decoding procedure (property 2) to ensure that Q and Q' are of degree D. To the best of our knowledge, this is the first time that this feature is being employed.

Acknowledgement. We are grateful to Ivan Damgård and Yuval Ishai for early discussions that influenced this work. We also thank YI for explaining various aspects of [17,18]. We thank the reviewers of Eurocrypt2023 for their comments.

3 Preliminaries

3.1 Linear algebraic notations

We define some linear-algebraic notation. Below $\mathbb{F}$ denotes some finite field and $m \in \mathbb{N}$ is a positive integer.

[11] Indeed, here we assume that the field is sufficiently large. In contrast, the VOLE1 and VOLE2 protocols can be realized over small fields as well.

Selective matrix-vector entries. For a vector $\boldsymbol{d} \in \mathbb{F}^m$ and a 0-1 vector $I = (I_1, \ldots, I_m) \in \{0,1\}^m$, we define the vector $\boldsymbol{d}[I] \in \mathbb{F}^m$ such that its ith entry is d_i if $I_i = 1$ and 0 otherwise. This notation can be used for both row and column vectors, and can be naturally extended to matrices as follows. For a $m \times k$ matrix M whose rows are denoted by $M_1, \ldots, M_m \in \mathbb{F}^k$, and a binary vector $I \in \{0,1\}^m$, we let $M[I] \in \mathbb{F}^{m \times k}$ denote the matrix whose ith row is M_i if $I_i = 1$ and 0^k otherwise. Note that this operator is linear and can be written in the following matrix form:

$$\boldsymbol{d}[I] = \begin{pmatrix} I_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & I_m \end{pmatrix} \cdot \boldsymbol{d} \qquad M[I] = \begin{pmatrix} I_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & I_m \end{pmatrix} \cdot M$$

Vector of clean coordinates. Let $\boldsymbol{e} \in \mathbb{F}^m$ denote a vector (typically viewed as a noise vector). We define the vector $I(\boldsymbol{e}) \in \{0,1\}^m$ such that its ith entry is 1 iff $\boldsymbol{e}_i = 0$. Note that by our notations $\boldsymbol{e}[I(\boldsymbol{e})] = 0^m$.

Matrix-vector concatenation. Let M be a matrix of dimensions $m \times k$ and a vector $\boldsymbol{c} \in \mathbb{F}^m$. We define $M|\boldsymbol{c}$ to be the result matrix that is obtained by concatenating the column vector $\boldsymbol{c}$ to the matrix M from the right side.

Bernoulli vector. Let $\mathsf{BER}^m(p)$ for real number $p \in [0,1]$ be the distribution of binary vectors $I = (I_1, ..., I_m)$ of length m with i.i.d entries such that for any i: I_i takes a value of 1 with probability p.

Family of finite fields. We always assume that our functionalities are implicitly parameterized by a family of finite fields whose size may grow with the security parameter. Throughout the paper, we fix this family $\mathbb{F} = \{\mathbb{F}_k\}_{k \in \mathbb{N}}$ and assume that it is *efficiently computable*, that is, one should be able to compute all field operations in $\mathsf{poly}(k)$ time (including the ability to add/subtract/multiply/divide and to sample a random field element). Note that this requirement implies that $|\mathbb{F}_k| \leq 2^{\mathsf{poly}(k)}$ and so field elements can be represented by $\mathsf{poly}(k)$-bit strings. In fact, for our protocols, we only need black-box access to the field operations, and the ability to send field elements either directly or over an OT channel. By default, we also assume that the field is sufficiently large, e.g., exponentially large in the security parameter. For sufficiently large width parameter w (e.g., cubic in k), our protocols for width-w VOLE require $O(w)$ field operations, and at most $O(w \log |\mathbb{F}_k|)$ Boolean operations. Accordingly, the overall complexity is dominated by the arithmetic complexity $O(w)$ which is optimal. Indeed, even in an insecure implementation, w arithmetic operations are needed for w-width VOLE.

It should be mentioned that the assumption regarding the field size is mainly needed for achieving linear-time efficiency and most of our protocols (or close variants of them) remain secure even when the field is of small constant-size (the error is always negligible in the security parameter). See Remark 2.

4 The ADINZ protocol

The ADINZ [5] protocol for VOLE is based on a gadget ("encoder") that allows fast encoding and decoding under erasures but semantically hides the encoded messages in the presence of noise. This gadget is mainly based on a public matrix $M \in \mathbb{F}_k^{m \times k}$ with the following (LPN-style) pseudorandomness property: If we take a random vector in the image of M, and add a sparse noise to it, the resulting vector is computationally indistinguishable from a truly random vector over $\mathbb{F}_k^{m(k)}$. The noise distribution that is being used in the ADINZ protocol corresponds to an additive noise vector $\boldsymbol{e} \in \mathbb{F}_k^{m(k)}$ where each coordinate of $\boldsymbol{e}$ is assigned independently with the value of zero with probability $1-\mu$ and with a uniformly chosen non-zero element from $\mathbb{F}_k$ with probability μ. We let $\mathcal{D}(\mathbb{F}_k)_\mu^m$ denote the corresponding noise distributions for such vectors of length m. For concreteness,the reader may think of μ as a small constant, say $1/4$, however μ can also be chosen so that it tends to 0 when the security parameter k tends to infinity. The properties of the ADINZ gadget are summarized in the following assumption.

Assumption 3. *(**Fast pseudorandom matrix**) There exists a noise rate $\mu = \mu(k) < 1/2$ and an efficient randomized algorithm $\mathcal{M}$ that given a security parameter 1^k and a fields family representation $\mathbb{F}$, samples a $m \times k$ ($m = O(k^3)$) matrix M over $\mathbb{F}_k$ such that the following holds:*

1. *(Linear-time computation) The mapping $f_M : \boldsymbol{r} \to M\boldsymbol{r}$ can be computed in time that linear in the output length m, i.e., by performing $O(m)$ arithmetic operations.*
2. *(Noisy-codeword is pseudorandom) The following ensembles are computationally indistinguishable:*

$$\left\{(M, M\boldsymbol{r} + \boldsymbol{e})\right\}_{k\in\mathbb{N}} \approx_c \left\{(M, \boldsymbol{u})\right\}_{k\in\mathbb{N}}$$

   where $M \leftarrow \mathcal{M}(1^k, \mathbb{F}), \boldsymbol{r} \leftarrow \mathbb{F}_k^k, \boldsymbol{e} \leftarrow \mathcal{D}(\mathbb{F}_k)_\mu^m$ and $\boldsymbol{u} \leftarrow \mathbb{F}_k^m$.
3. *(Linear independence) If we sample $M \leftarrow \mathcal{M}(1^k, \mathbb{F})$ and keep each of the first $u = O(k \log^2 k)$ rows independently with probability $1-\mu$ (and remove all other rows), then, except with negligible probability in k, the resulting matrix has full rank of k.*

Concrete instantiations of this matrix-ensemble $\mathcal{M}$ (e.g., based on sparse matrices or on the Druk-Ishai ensemble [25]) are discussed in [5]. The ADINZ encoder also makes use of a (non-cryptographic) linear error correcting code $\mathsf{Ecc} : \mathbb{F}_k^w \to \mathbb{F}_k^v$ which encodes vectors of length w into vectors of length v over the field $\mathbb{F}_k$, with constant rate R and linear time encoding and decoding, such that decoding is possible with high success probability from a constant fraction of erasures μ' which is slightly larger than the noise rate μ. (For $\mu = \frac{1}{4}$ we can take $\mu' = \frac{1}{3}$). Such codes are known to exist and can be efficiently constructed given a black-box access to $\mathbb{F}_k$. The code Ecc and the matrix M are combined together into the so-called protocol's encoder:

ADINZ Protocol's encoder. Given $k \in \mathbb{N}$, $m = O(k^3)$, $w = O(k^3)$, $\boldsymbol{r} \in \mathbb{F}_k^k$ and $\boldsymbol{a} \in \mathbb{F}_k^w$, let M be a $m \times k$ fast pseudorandom matrix. We define the encoding gadget $E_{\boldsymbol{r}}(\boldsymbol{a})$ to be:

$$E_{\boldsymbol{r}}(\boldsymbol{a}) = M \cdot \boldsymbol{r} + 0^u \circ \mathsf{Ecc}(\boldsymbol{a})$$

where $\mathsf{Ecc} : \mathbb{F}_k^w \to \mathbb{F}_k^v$, $u = 2k \log^2 k$, $v = m - u$ and $\circ$ denotes concatenation (so $0^u \circ \mathsf{Ecc}(\boldsymbol{a})$ is a vector of length m). Equivalently, for an information vector $\boldsymbol{a} \in \mathbb{F}_k^w$ and randomness vector $\boldsymbol{r} \in \mathbb{F}_k^k$, we can write the encoder as

$$E_{\boldsymbol{r}}(\boldsymbol{a}) = T \cdot \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{a} \end{pmatrix},$$

where the encoder matrix is

$$T = \left(\begin{array}{c|c} M_{m \times k} & \begin{array}{c} \mathbf{0}_{u \times w} \\ \\ \mathsf{Ecc}_{v \times w} \end{array} \end{array} \right) \tag{1}$$

and $\mathsf{Ecc}_{v \times w} \in \mathbb{F}_k^{v \times w}$ is the generating matrix of the error correcting code. By exploiting Assumption 3 and the features of the error correcting code, the encoder E satisfies the following properties:

1. (Fast and Linear) The mapping $E_{\boldsymbol{r}}(\boldsymbol{a})$ can be computed by making only $O(m)$ arithmetic operations. Moreover, it is a linear function of $\boldsymbol{r}$ and $\boldsymbol{a}$ and so $E_{\boldsymbol{r}}(\boldsymbol{a}) + E_{\boldsymbol{r}'}(\boldsymbol{a}') = E_{\boldsymbol{r}+\boldsymbol{r}'}(\boldsymbol{a} + \boldsymbol{a}')$.
2. (Hiding under errors) For any message $\boldsymbol{a} \in \mathbb{F}_k^w$ and $\boldsymbol{r} \leftarrow \mathbb{F}_k^k$,$\boldsymbol{e} \leftarrow \mathcal{D}(\mathbb{F}_k)_m^\mu$ the vector $E_{\boldsymbol{r}}(\boldsymbol{a}) + \boldsymbol{e}$ is pseudorandom. Namely: for any ensemble $\{\boldsymbol{a}_k\}_{k \in \mathbb{N}}$ the following ensembles are computationally indistinguishable:

   $$\left\{ \left(M, E_{\boldsymbol{r}}(\boldsymbol{a}_k) + \boldsymbol{e}\right) \right\}_{k \in \mathbb{N}} \approx_c \left\{ (M, \boldsymbol{u}) \right\}_{k \in \mathbb{N}}$$

   where $M \leftarrow \mathcal{M}(1^k, \mathbb{F}), \boldsymbol{r} \leftarrow \mathbb{F}_k^k, \boldsymbol{e} \leftarrow \mathcal{D}(\mathbb{F}_k)_\mu^m$ and $\boldsymbol{u} \leftarrow \mathbb{F}_k^m$. In particular, a noisy codeword computationally "hides" $\boldsymbol{a}$.
3. (Fast decoding under erasures) Given a random vector $I \leftarrow \mathsf{BER}^m(1-\mu)$ and a code $\boldsymbol{d}[I] = E_{\boldsymbol{r}}(\boldsymbol{a})[I]$ (i.e. each coordinate is erased independently with probability μ) it is possible to recover the vector $\boldsymbol{a}$, with negligible error probability, by making only $O(m)$ arithmetic operations. We first recover $\boldsymbol{r}$ by solving the linear system $\boldsymbol{d}_{\text{top}}[I_{\text{top}}] = M_{\text{top}}[I_{\text{top}}]\boldsymbol{r}$ (where "top" means top u coordinates) via Gaussian elimination in $O(m)$ arithmetic operations. By Assumption 3 (property 3) the system is likely to have a unique solution. Then we compute $M[I_{\text{bot}}]\boldsymbol{r}$ in time $O(m)$, subtract from $\boldsymbol{d}[I_{\text{bot}}]$ to get the vector $\mathsf{Ecc}(\boldsymbol{a}))[I_{\text{bot}}]$ and recover $\boldsymbol{a}$ by erasure decoding in time $O(m)$.

Remark 2 (On the choice of parameters). Some of the above requirements are tailored to achieve a VOLE of width w with an asymptotic computational complexity of $O(w)$ field operations. This includes the choice of the values of m, w, and u, the requirements for the "fast" computation of E and "fast" decoding under erasures, and the assumption that the field size is exponential in the security

parameter. All these requirements can be waived without affecting the security of the protocols. (Assuming that the pseudorandomness assumption holds.) In particular, for concrete settings, it may be better to set these parameters differently as done in our implementations (See the full version [10]).

In the full version [10] we show that the ADINZ protcol is vulnerable against actively corrupt receiver and prove that it is secure against an active sender under a new intractability assumption. (These parts will not be used in our subsequent protocols.)

5 RVOLE Protocol against Actively-Corrupted Receiver

In this section, we construct a protocol for RVOLE, which is actively secure against Bob and passively secure against Alice. The protocol is based on the ADINZ protocol. We will later use this protocol as a building block of an actively secure VOLE protocol of width w over the field family $\mathbb{F}$. Our protocol relies on *CDS for span membership*. Formally, let $f_{T,\boldsymbol{c}} : \{0,1\}^m \to \{0,1\}$ be a predicate that receives a vector $I \in \{0,1\}^m$ and accepts iff $\boldsymbol{c}[I] \in \mathsf{colspan}\left(T[I]\right)$. A CDS for $f_{T,\boldsymbol{c}}$ is a pair of algorithms $\mathsf{Enc}(I,S;R)$ and $\mathsf{Dec}(I,z)$ such that for an input I, secret S and randomizer R the "ciphertext" $\boldsymbol{z} = \mathsf{Enc}(I,S;R)$ perfectly hides S if $f_{T,\boldsymbol{c}}(I) = 0$, and, otherwise, $\mathsf{Dec}(I,z)$ outputs S. In addition, the encoding function can be decomposed to an offline part that does not depend on I and m "online" parts each depending on a single bit of I, i.e., $\mathsf{Enc}(I,S;R) = (\mathsf{Enc}_0(S;R), \mathsf{Enc}_1(I_1,S;R), \ldots, \mathsf{Enc}_k(I_m,S;R))$. We also require that, both $\mathsf{Enc}(I,S;R)$ and Dec, are computable by $O(m)$ arithmetic operations over $\mathbb{F}$. Construction of such a CDS with unconditional information-theoretic security appears in the full version [10].

Protocol 4 (modified RVOLE protocol). *To initialize the protocol Bob samples the matrix $M \leftarrow \mathcal{M}(1^k, \mathbb{F})$ and sends it to Alice.*

1. ***Bob:*** *Given an input $\boldsymbol{a} \in \mathbb{F}_k^w$, Bob samples vectors $\boldsymbol{r} \leftarrow \mathbb{F}_k^k$ and $\boldsymbol{e} \leftarrow \mathcal{D}(\mathbb{F}_k)_\mu^m$, sets $I = I(\boldsymbol{e})$ and sends the vector: $\boldsymbol{c} = E_{\boldsymbol{r}}(\boldsymbol{a}) + \boldsymbol{e}$ to Alice.*
2. ***Alice:*** *Given the inputs $\boldsymbol{b} \in \mathbb{F}_k^w$, $x \in \mathbb{F}_k$, and Bob's message $\boldsymbol{c} \in \mathbb{F}_k^m$, samples a random vector: $\boldsymbol{r}' \leftarrow \mathbb{F}_k^k$ and a field element $x' \leftarrow \mathbb{F}$ and computes the vector $\boldsymbol{d} = x'\boldsymbol{c} + E_{\boldsymbol{r}'}(\boldsymbol{b})$.*
3. ***Alice:*** *Samples randomness R for the span membership CDS and sets the secret $\Delta = x - x'$, and for each $i \in [m]$ Alice computes two possible CDS messages $z_{i,0} = \mathsf{Enc}_i(0,\Delta;R)$ and $z_{i,1} = \mathsf{Enc}_i(1,\Delta;R)$. In addition, Alice computes the CDS offline message by $z_0 = \mathsf{Enc}_0(\Delta;R)$ and sends z_0 to Bob.*
4. ***Alice and Bob:*** *Invoke m-batch OT where the ith entry of Alice is the pair*

$$(z_{i,1}, d_i) \qquad \text{and} \qquad z_{i,0}$$

and Bob uses the vector I as its selection vector.
5. ***Bob:***

– *Collects all the z-part of the OT messages into a vector* $\boldsymbol{z} = (z_0, (z_{i,I_i})_{i\in[m]})$*, and applies the CDS decoder to recover the CDS secret* $\Delta = \mathsf{Dec}(I, \boldsymbol{z})$*. If decoding fails Bob aborts.*
– *If* $\boldsymbol{d}[I]$ *is not in* $\mathsf{colspan}\left((T|\boldsymbol{c})[I]\right)$*, Bob aborts. Otherwise, Bob employs the decoding-under-erasures property of the gadget E (property 3), computes the vector* $\boldsymbol{v}'$ *(supposedly* $x'\boldsymbol{a}+\boldsymbol{b}$*), shifts it by* $\Delta\boldsymbol{a}$ *and outputs the result* $\boldsymbol{v} = \boldsymbol{v}' + \Delta\boldsymbol{a}$ *(supposedly,* $x\boldsymbol{a}+\boldsymbol{b}$*).*

The original ADINZ protocol is obtained by removing the blue parts and setting $x' = x$ and outputting $\boldsymbol{v}'$. As always, we assume the existence of an ideal m-batch OT channel. Through the analysis of Protocol 4, we assume that all the protocol's length parameters: m, w, v and u are polynomial functions of the security parameter k.

Remark 3 (About the set-up). In this protocol (and all the subsequent ones) the set-up step in which the matrix is sampled can be done once and for all. This is reflected in the security proofs which work even if the simulators receive M as an external input.

Lemma 1. *Under Assumption 3, the Protocol 4 realizes the RVOLE functionality of width* w *over* $\mathbb{F}$ *in the OT-hybrid model with arithmetic complexity of* $O(w)$ *(ignoring the initialization cost) and with the following guarantees:*

1. *Computational security against a passive adversary that corrupts Alice.*
2. *Computational privacy against an active adversary that corrupts Alice.*
3. *Perfect security against a passive adversary that corrupts Bob.*
4. *Perfect security against an active adversary that corrupts Bob and deviates from the protocol.*

Proof (sketch). The complexity bound follows from the complexity of the ADINZ encoder and from the complexity of the CDS encoder and decoder. One can easily verify that correctness holds when both parties are honest and that Alice's only incoming message is pseudorandom and is therefore simulatable regardless of Alice's behavior. This implies items 1 and 2. To simulate Bob, we collect $\boldsymbol{c}$ and I as chosen by (a possibly malicious) Bob, and check if $\boldsymbol{c}[I]$ is in $\mathsf{colspan}\left(T[I]\right)$. If the check passes we can extract Bob's effective input $\boldsymbol{a}'$ by solving the linear system $\boldsymbol{c}[I] = T[I]\cdot\begin{pmatrix}\boldsymbol{r}\\ \boldsymbol{a}'\end{pmatrix}$, and if the check fails we set $\boldsymbol{a}'$ to be the all-zero vector. Given $\boldsymbol{v} = x\boldsymbol{a}' + \boldsymbol{b}$ from the ideal functionality, we generate the CDS message of Alice just like in the real protocol by using some $\hat{x}$ and $\hat{\boldsymbol{b}}$ that are consistent with the output $\boldsymbol{v}$. It can be shown that this simulator perfectly emulates the real distribution. (See the full version [10] for details.) □

Some comments are in place:

1. (Computationally-unbounded Bob) The information-theoretic security against Bob holds even if the ideal OT channel is replaced with an OT protocol that provides statistical privacy for the sender (e.g., [45,1]).

2. (Full security against active Alice) We do not know if Protocol 4 provides full security against an actively corrupt Alice and leave this as an open question. It seems reasonable to assume that the protocol achieves full security. Under this assumption, one can plug Protocol 4 to the standard RVOLE-to-VOLE transformation [5] and derive an actively-secure VOLE protocol. We refer to this protocol as *VOLE1*.
3. (Concrete communication complexity of CDS) Our concrete CDS communicates $n+1 = k+w+1$ field elements in the offline message z_0 and leaves the 1-messages $z_{i,1}$ empty. Accordingly, each of the OT messages is just a single field element. Moreover, by resorting to computationally-private CDS (and exploiting PRGs), we can use an economic variant of the CDS in which each of the 0-messages, $z_1, \ldots, z_m$, is of length k independently of the field size. As a result, the total communication complexity of the OT messages can be reduced to $m \log |\mathbb{F}_k| + m \cdot k$. Furthermore, this can be done while keeping the computational complexity linear.[12]
4. (On the achievable rate) Based on the aforementioned optimized CDS, Protocol 4 communicates m field elements from Bob to Alice, $n+1 = k+w$ field elements from Alice to Bob in the offline CDS message, and m field elements plus $O(mk)$ bits over the OT-channel. Overall, the number of field elements that are communicated is $2m+n+1 = 2(u+v)+w+k+1 = (2v+w)(1+o(1))$ where the last equality holds since $k = o(w)$ and $u = o(v)$. Recall that v is the length of the code produced by Ecc, which needs to be at least approximately $\frac{1}{1-\mu}w$ to allow successful decoding of w field elements values from a noisy codeword with a fraction of μ random erasures. Therefore, the communication rate of the protocol, measured as the length of the protocol's output w, divided by the communication complexity, approaches to:

$$\frac{w}{2v+w} = \frac{1-\mu}{3-\mu}.$$

If Assumption 3 holds for any constant error rate $\mu > 0$ then we can obtain a rate approaching $\frac{1}{3} - \varepsilon$ for any constant $\varepsilon > 0$. Furthermore, by choosing a non-constant erasure fraction of $\mu = \frac{1}{f(k)}$ for $f(k)$ that tends to infinity with k (for example $f(k) = \frac{1}{\log k}$) we get an asymptotic rate of $1/3$, namely, in order to realize an RVOLE functionally of size w by our protocol $3w$ fields elements should be communicated (where w and k tend to infinity).[13] The reduction to VOLE increases the communication by w additional field elements and so the rate of the VOLE1 protocol is $\frac{1-\mu}{2(2-\mu)}$ which approaches to $1/4$ for a small noise rate. Recall that the communication rate of the passively-secure ADINZ VOLE protocol approaches $1/3$.

[12] The computational complexity and communication complexity of batch-OT are measured as the total bit-length of the sent messages; see the full version [10] for a justification for this convention.

[13] In the context of binary codes, LPN-style assumptions with sub-constant μ are quite standard.

6 Actively-Secure VOLE under Correlated Noisy-Codewords

In this section, we realize the VOLE functionality directly while achieving active security against both Alice and Bob. For this, we introduce an additional, new, "Correlated Noisy-Codeword" intractability assumption. We will also have to slightly modify the parameters of the ADINZ encoding matrix. Recall that the ADINZ encoding matrix T is defined as follows:

$$T = \left(\begin{array}{c|c} & \mathbf{0}_{u \times w} \\ M_{m \times k} & \\ & \mathsf{Ecc}_{v \times w} \end{array} \right),$$

where $m = \Omega(k^3)$, $u = \Omega(k \log^2 k)$ and $v = O(m)$. For technical reasons we will need to slightly strengthen the linear-independence requirements of the matrix T as follows. Except with negligible probability over the choice of M and Ecc it must hold that: (a) If we sample a random subset of the first u rows of M by taking each row independently with probability $1/\log^{1.5} k$ then the resulting matrix has full rank (all the columns are linearly independent); (b) The error-correcting code Ecc can correct up to $O(\log^{1.1} k)$ *errors* and, as before, can recover from say $1.2\mu v$ arbitrary erasures. (The constant 1.2 can be replaced with any constant larger than 1.)

6.1 The Correlated Noisy-Codeword Hardness Assumption

The following intractability assumption intuitively asserts that given a noisy codeword $\boldsymbol{c} = T\boldsymbol{v} + \boldsymbol{e}$ of T, it is hard to efficiently generate a new noisy codeword $\boldsymbol{d} = T\boldsymbol{v}' + \boldsymbol{e}'$ whose noise is non-trivially correlated with $\boldsymbol{e}$ in the following sense. The new noise vector $\boldsymbol{e}'$ agrees with the original noise vector $\boldsymbol{e}$ with respect to the set of non-noisy coordinates $I = I(\boldsymbol{e})$, i.e., $\boldsymbol{d}[I] \in \mathsf{colspan}\left(T[I]\right)$, but $\boldsymbol{e}'$ is "far" from being a scalar multiple of $\boldsymbol{e}$. That is, $\rho\left(\boldsymbol{d}, \mathsf{colspan}\left(T|\boldsymbol{c}\right)\right) = \ell$ where $\rho\left(\boldsymbol{d}, S\right)$ is the minimal Hamming distance between a vector $\boldsymbol{d}$ and a set of vectors $S \subset \mathbb{F}^m$. Observe that such a noisy codeword can be generated by sampling a vector in the column span of $(T|\boldsymbol{c})$ and then modifying ℓ entries with the hope that all these entries fall out of the set of clean coordinates I. Such an attack succeeds with probability μ^ℓ, the following assumption states that this is essentially the best that one can hope for up to polynomial speed-ups.[14]

Assumption 5 (Correlated noisy codeword). *For a distribution $\mathcal{T}$ over matrices in $\mathbb{F}_k^{m \times n}$ where $m(k), n(k)$ are some polynomials in the security parameter k, and for a constant noise rate of $\mu < 1/2$, the* Correlated Noisy-Codeword

[14] The concrete formulation that is taken here is chosen for the sake of simplicity. More refined and conservative versions (e.g., that assume better speed-ups and consider sub-constant noise regimes) can be adopted as well.

assumption asserts that for every efficient adversary A^* *there exists some negligible* $\varepsilon(k)$ *and constant* C *such that for every integer* $\ell \leq m$*:*

$$\Pr_{\boldsymbol{d} \leftarrow \mathrm{A}^*(\boldsymbol{c})} \left[\boldsymbol{d}[I] \in \mathsf{colspan}\left(T[I]\right) \quad \wedge \quad \rho\left(\boldsymbol{d}, \mathsf{colspan}\left(T|\boldsymbol{c}\right)\right) = \ell \right] \leq \exp(-C\ell) + \varepsilon(k)$$

where $T \leftarrow \mathcal{T}$ *and* $\boldsymbol{c} = T\boldsymbol{v} + \boldsymbol{e}$ *for* $\boldsymbol{v} \leftarrow \mathbb{F}_k^n$*,* $\boldsymbol{e} \leftarrow \mathcal{D}(\mathbb{F}_k)_\mu^m$ *and* $I = I(\boldsymbol{e})$*.*

For our purposes, it suffices to assume the "super-logarithmic version of Assumption 5" that asserts that for every super-logarithmic function $\ell(k) = \omega(\log k)$ the success probability in the above game is negligible in k. (Note that this variant follows from the above formulation.) We conjecture that every matrix distribution whose noisy codewords are pseudorandom also satisfies this assumption, and provide some evidence for this in the full version [10]. From now, we will always use the super-logarithmic version of the assumption with respect to the distribution $\mathcal{T}$ that corresponds to the ADINZ encoding matrix.

We will make use of the following simple observation whose proof is deferred to the full version [10].

Lemma 2. *There exists a probabilistic polynomial-time algorithm* G *that given the matrix* T *and the vectors* $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{F}_k^m$ *outputs a vector* $\boldsymbol{u} \in \mathbb{F}_k^{n+1}$ *with the following guarantee. Except with negligible probability in* k *over the choice of* $(T, \boldsymbol{c})$ *(which are distributed as in Assumption 5) and the randomness of* G*, if* $\boldsymbol{d}$ *is* ℓ*-close to* $\mathsf{colspan}\left(T|\boldsymbol{c}\right)$ *for* $\ell = O(\log^{1.1} k)$ *then the algorithm outputs* $\boldsymbol{u}$ *such that* $(T|\boldsymbol{c}) \cdot \boldsymbol{u}$ *is* ℓ*-close to* $\boldsymbol{d}$*.*

6.2 The VOLE2 protocol

We present our VOLE protocol and prove security under Assumption 5. The protocol employs an ideal-commitment functionality, aka *commitment channel*, which is a 2-phase ideal functionality of the following form. In the commit phase, the functionality takes an input x from a sender (e.g., a field element), and delivers a commit message to the receiver. At a later phase, the sender can de-commit by sending an "open" message to the functionality which delivers to the receiver the committed message x. Such an ideal commitment channel can be constructed based on OT-channel [21,22] perfect security against the receiver and statistical security against the sender the communication and computational complexity of k OT-calls and k field additions (where k is the statistical security parameter).

Protocol 6 (VOLE2 protocol). *To initialize the protocol Bob samples the matrix* $M \leftarrow \mathcal{M}(1^k, \mathbb{F})$ *and sends it to Alice.*

1. ***Alice and Bob:*** *Hold inputs* $x \in \mathbb{F}_k$ *and* $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_k^w$ *respectively. The parties invoke Protocol 4 for RVOLE where Bob's input is a random vector* $\boldsymbol{a}' \leftarrow \mathbb{F}_k^w$*, and Alice's input is* x *and a random vector* $\boldsymbol{b}' \leftarrow \mathbb{F}_k^w$*. Let* x' *denote the random field element that is being sampled by Alice in the protocol and let* $\Delta_x = x - x'$ *denote the secret that Alice delivers via the CDS.*

2. ***Alice:*** *Sends Δ_x over an ideal commitment channel which delivers a "commit" message to Bob.*
3. ***Bob:*** *If Bob aborts during Protocol 4 then he aborts the entire execution. Otherwise, Bob recovers from the protocol the vector $\boldsymbol{v}'$ (supposedly $x'\boldsymbol{a}'+\boldsymbol{b}'$) and the CDS secret Δ_x (supposedly $x-x'$). Bob sends Δ_x to Alice.*
4. ***Alice:*** *Verifies that Bob's message equals to Δ_x, and aborts if the check fails. If the check passes, Alice decommits by sending an "open" message to the commitment channel which delivers the committed value, Δ_x to Bob.*
5. ***Bob:*** *Verifies that the decommitted value equals to Δ_x, and aborts if the check fails. If the check passes, Bob sends to Alice the vectors $\boldsymbol{v} = \boldsymbol{v}' + \Delta_x\boldsymbol{a}' + \boldsymbol{b}$ (supposedly, $x\boldsymbol{a}' + \boldsymbol{b}' + \boldsymbol{b}$) and $\boldsymbol{\Delta_a} = \boldsymbol{a} - \boldsymbol{a}'$.*
6. ***Alice:*** *Computes the vector $\boldsymbol{w} = x\boldsymbol{\Delta_a} + \boldsymbol{v} - \boldsymbol{b}'$ (supposedly, $x\boldsymbol{a} + \boldsymbol{b}$) and outputs the result.*

In the full version [10] we prove that the protocol is computationally secure against an actively corrupt Alice, and statistically secure against an actively corrupt Bob. By replacing the commitment with k calls to OT, we derive the following lemma.

Lemma 3. *Suppose that Assumptions 3 and 5 hold. Then protocol 6 for honest parties realizes the VOLE functionality of width w over $\mathbb{F}$ with arithmetic complexity of $O(w)$ (ignoring the initialization cost) in the OT-hybrid model. The protocol is statistically-secure against an active sender Bob with negligible deviation error and computationally secure against an active receiver Alice.*

Some comments are in place:

1. (Unbounded sender) Here too, the protocol achieves information-theoretic security against the sender even if the ideal channels are replaced by an OT protocol that provides statistical privacy for the sender and by a commitment scheme that is statistically hiding. (The latter reduces to the former by using the OT-to-Commitment transformation.
2. (Working over small fields) The statistical error is exponentially-small in the bit-length of the field element Δ_x. (This essentially corresponds to the case where Bob guesses the value of Δ_x despite playing dishonestly in the RVOLE protocol in a way that keeps the CDS secret hidden). Thus, when the field is small the error is only $1/|\mathbb{F}_k|$. Nevertheless, even when the field is small, one can easily get a negligible error at a minor cost by randomly padding the element Δ_x to length k.
3. (On the achievable rate of Protocol 6) The communication of Protocol 6 (VOLE2) consists of $(2v+w)(1+o(1))$ field elements in Step 1 (when invoking the RVOLE Protocol), $2k$ field elements to commit (via k OT calls) and to de-commit, and additional $2w+1$ elements. Since $k = o(w)$, the total communication complexity is $(2v+3w)(1+o(1))$. Recall that v is the length of the code produced by Ecc, which needs to be at least approximately $\frac{1}{1-\mu}w$ to allow successful decoding of w field elements values from a noisy codeword with fraction of μ random erasures. Therefore, the communication rate of

the protocol, measured as the length of the protocol's output w, divided by the communication complexity, approaches to

$$\frac{w}{2v+3w} = \frac{1-\mu}{5-3\mu}.$$

If Assumption 3 holds for any constant error rate $\mu > 0$ then the rate of RVOLE2 approaches to $\frac{1}{5} - \varepsilon$ for any constant $\varepsilon > 0$.

4. (Comparison to VOLE1) In terms of communication we pay an amortized cost of an extra field element per each VOLE entry compared to VOLE1, which, in turn, pays an extra field element compared to the passively-secure ADINZ protocol. In terms of computation, VOLE2 has a negligible overhead compared to VOLE1 which consists of a single commitment (for the entire VOLE), and, an amortized cost of $1/R$ field multiplication and $2/R$ field additions per VOLE entry where $R = w/v$ is the rate of the error-correcting code.[15]

7 Actively-Secure VOLE under Fast Pseudorandom Matrix

In this section, we describe a VOLE protocol with full active security based on the modified-RVOLE protocol (Protocol 4). Following the outline in Section 2.3, we begin with some useful observations.

7.1 Useful Observations

More CDS properties. We will make use of the fact that our concrete CDS sends messages only on "zero" inputs. (That is, our CDS is effectively a secret sharing scheme for the negated predicate.) Let $\boldsymbol{z} = (z_0, z_{1,0}, \ldots, z_{m,0})$ be a (possibly malformed) full vector of CDS messages. We say that $\boldsymbol{z}$ is *valid* if for every input I that satisfies the underlying predicate f, the CDS decoder recovers the same secret. We say that $\boldsymbol{z}$ is *honestly generated* if it is generated by invoking the CDS message generator honestly on some random tape and some secret. (By perfect correctness, an honestly generated CDS is always valid.) We assume the existence of an efficient tester $\mathcal{T}$ that rejects every invalid $\boldsymbol{z}$, and accepts every honestly generated $\boldsymbol{z}$. (That is $\mathcal{T}$ is allowed to accept a vector that is not honestly generated as long as it is valid.) Furthermore, if $\mathcal{T}$ accepts then it should be able to recover the secret. Our CDS construction satisfies these properties.

[15] Recall that VOLE1 is obtained by combining the RVOLE-to-VOLE transformation with Protocol 4 for RVOLE. Accordingly, the latter protocol achieves provable active security against the Sender, provable passive security against the Receiver, and heuristic active security against the Receiver.

Closer look at cheating Alice. Fix some strategy A^* for a malicious Alice in Protocol 4. Formally, this is a deterministic mapping that takes Alice's inputs $(x, \boldsymbol{b})$, the public matrix M, and Bob's message $\boldsymbol{c}$ and outputs a CDS message vector $\boldsymbol{z} = (z_0, (z_{i,0})_{i\in[m]})$ and a vector $\boldsymbol{d}$ (to be placed on the 1-inputs of the OT). If either $\boldsymbol{z}$ is invalid or $\boldsymbol{d}$ is invalid in the sense that $\boldsymbol{d} \notin \mathsf{colspan}\left((T|\boldsymbol{c})\right)$, we say that A^* *cheats* on $(x, \boldsymbol{b}, M, \boldsymbol{c})$. In the full version [10] we show that when Alice does not cheat, her "effective input" can be extracted given her OT messages, and use this to prove the following lemma. Let us denote Protocol 4 by Π.

Lemma 4. *There exists a simulator* $\textsc{Sim}'(x, \boldsymbol{b})$ *that makes a black-box use of* A^* *and simulates* Π *whenever* A^* *does not cheat. Formally, for every sequence of inputs* $((x_k, \boldsymbol{b}_k), \boldsymbol{a}_k)$ *it holds that the ensemble*

$$\left[\textsc{Real}_{A^*,\Pi}((x_k, \boldsymbol{b}_k), \boldsymbol{a}_k) \quad | \quad A^* \textit{ doesn't cheat } \right]$$

is computationally indistinguishable from the ensemble

$$\left[\textsc{Ideal}_{\textsc{Sim},f_k}(\boldsymbol{a}_k, \boldsymbol{b}_k) \quad | \quad \textsc{Sim} \textit{ doesn't fail } \right].$$

The next crucial observation is that there exists a strategy for Bob that detects (with probability 0.5) whether Alice cheats. Indeed, as explained in the introduction, in the OT phase Bob can toss a coin and ask with probability 1/2 to receive the vector $\boldsymbol{d}$ (by using $I = 1^m$) and with probability 1/2 the vector $\boldsymbol{z}$ (by using $I = 0^m$) and check validity. When running in this "detection mode" Bob effectively gives up on the computation and just verifies whether Alice misbehaves or not. Note that Bob's decision is taken only in the OT phase and is hidden from Alice, and so effectively Alice first "commits" to strategy (cheat or not), and only then Bob decides whether to "call her bluff". Furthermore, even when Bob acts as a detector, we can fully simulate his view (since the protocol is actively-secure against any deviation of Bob). We will exploit this property to obtain a "silent" cut-and-choose version of the protocol as follows.

7.2 The VOLE3 protocol

Let Π denote Protocol 4 instantiated with width $w(k) = O(k^3)$ and recall that Π makes a call to an $m = m(k)$-batch OT channel. The new protocol (hereafter denoted as VOLE3) realizes VOLE with width $W = W(k)$ (for some value that will be determined later) by making $t = t(k)$ calls to Π, and by "opening" $p = p(k)$ sessions for detecting a potential cheating by Alice. In addition to these parameters, we let $s = s(k) = t(k) - p(k)$ denote the number of remaining "un-opened" sessions, and let $\ell = \ell(k)$ be a leakage parameter. The product $\ell p/t$ should be polynomial in the security parameter k and, for efficiency purposes, s should be $\Omega(t)$. For example, set $t = k$, $p = \ell = k^{0.9}$ and $s = k - k^{0.9}$. Again, we make use of ideal commitments which can be realized based on OT channels.

Linear-time resilient functions. We will need a linear mapping $\mathsf{Ext} : \mathbb{F}^{sw} \to \mathbb{F}^{(1-\beta)sw}$ where $\beta = \beta(k) < 1$ is bounded away from 1, with the following properties: (1) Ext should be computable in linear arithmetic time (i.e., by making $O(sw)$ operations); and (2) The distribution $\mathsf{Ext}(X)$ should be uniform whenever the input X is uniform except for at most $\ell' = \ell(k)w + 1 = o(sw)$ entries that may be arbitrarily fixed. Formally, for every ℓ'-subset $L \subset [sw]$ and fixing $(X_i)_{i\in L} \in \mathbb{F}^{\ell'}$ if $(X_i)_{i\notin L}$ is uniform over $\mathbb{F}^{sw-\ell'}$, the output $\mathsf{Ext}(X_1, \ldots, X_{sw})$ is uniform over $\mathbb{F}^{(1-\beta)sw}$. Such functions are known as *ℓ'-resilient functions* [20] and can also be viewed as perfect deterministic extractors for bit-fixing sources. One can realize such functions, with the desired parameters, based on linear-time encodable error-correcting codes with rate $1 - \beta$ and distance $\ell' + 1$ (see [20] and [24, Theorem 3.1.7]). The width of VOLE3 is taken to be the output length of Ext, i.e., $W(k) = (1 - \beta)sw$.

Protocol 7 (VOLE3 protocol). *Upon initialization, bob samples $M \leftarrow \mathcal{M}(1^k, \mathbb{F})$ and sends it to Alice. The input of Bob is a pair of vectors $\boldsymbol{g}, \boldsymbol{f} \in \mathbb{F}_k^W$ and the input of Alice is $x \in \mathbb{F}_k$.*

1. ***Bob:*** *Invokes t independent parallel sessions of Π with the matrix M as a public parameter, where the jth private input is a random vector $\boldsymbol{a}_j \leftarrow \mathbb{F}_k^w$. For each such session $j \in [t]$, Bob computes the first-round message $\boldsymbol{c}_j$ as in Step 1 of Π, and sends $\boldsymbol{c}_j$. Let I_j denote the (vector representation of the) set of clean coordinates in $\boldsymbol{c}_j$.*
2. ***Alice:*** *Samples $x^* \leftarrow \mathbb{F}_k$. For every $j \in [t]$, Alice sets her inputs for the jth session to be $(x_j, \boldsymbol{b}_j)$ where $x_j = x^*$ and $\boldsymbol{b}_j \leftarrow \mathbb{F}_k^w$, she samples a random tape for the jth session, sends the corresponding offline CDS message $z_{i,0}$ to Bob, and computes the vectors $(\boldsymbol{z}_j, \boldsymbol{d}_j)$ that will be sent in the OT phase of the jth session (by following Steps 2 and 3 of Π). Here we view Alice's input to the m-batch-OT as a pair of m-long vectors.*
3. ***OT-phase:*** *Bob samples a random p-subset $P \subset [t]$ of sessions that will be "opened". For each $j \in [t]$ in parallel, the parties invoke the m-batch OT channel of the jth session. Alice's input is $(\boldsymbol{z}_j, \boldsymbol{d}_j)$. If $j \notin P$ Bob's input is I_j; Otherwise, Bob samples a random bit $\sigma_j \leftarrow \{0, 1\}$ and uses a trivial selection vector $I'_j := \sigma_j^m \in \{0^m, 1^m\}$.*
4. ***Bob:*** *If cheating is detected in one of the "opened copies" $j \in P$ (i.e., if the received vector is invalid), Bob aborts. Otherwise, let $S = [t] \setminus P$ denote the set of unopened copies. For every $j \in S$, Bob recovers the output $\boldsymbol{v}_j \in \mathbb{F}_k^w$ of the jth session just like in Step 5 in Π (hereafter referred to as Π_5). If the output is "abort" Bob sets $\boldsymbol{v}_j = 0^w$.*
5. ***Sub-protocol:*** *To verify that Alice's inputs $(x_j)_{j\in S}$ are all equal, the parties do:*
   - ***Bob:*** *Computes the sum δ of all the last elements of the vectors $(\boldsymbol{a}_j \in \mathbb{F}_k^w)_{j\in S}$, i.e., $\delta := \sum_{j\in S} \boldsymbol{a}_j[w]$ and sends to Alice the pair (S, δ). (Bob challenges Alice to compute the sum $\sum_{j\in S} \boldsymbol{v}_j[w]$.)*
   - ***Alice:*** *Sends $\lambda = x^*\delta + \sum_{j\in S} \boldsymbol{b}_j[w]$ over the ideal commitment channel which delivers a "commit" message to Bob.*
   - ***Bob:*** *Given a "commit" message, sends the value $\lambda' := \sum_{j\in S} \boldsymbol{v}_j[w]$.*

- ***Alice:*** *decommits by sending an "open" message to the commitment channel if $\lambda' = \lambda$, else Alice aborts.*
- ***Bob:*** *Halts with an abort symbol if Alice does not open the commitment or if the decommitment $\lambda \neq \lambda'$, and continues otherwise.*

6. ***Alice:*** *Sends $\Delta = x - x^*$.*
7. ***Bob:*** *Aligns the results of the unopened sessions vectors by setting*

$$\boldsymbol{u}_j := \boldsymbol{v}_j + \Delta \boldsymbol{a}_j, \quad \forall j \in S,$$

concatenates the vectors $(\boldsymbol{a}_j)_{j\in S}$ to a single vector $\boldsymbol{a}_S \in \mathbb{F}_k^{sw}$ and the vectors $(\boldsymbol{u}_j)_{j\in S}$ to a single vector $\boldsymbol{u}_S \in \mathbb{F}_k^{sw}$, and extracts the vectors

$$\boldsymbol{a}' := \mathsf{Ext}(\boldsymbol{a}_S), \quad \boldsymbol{u}' := \mathsf{Ext}(\boldsymbol{u}_S).$$

Bob sends to Alice the vectors $\boldsymbol{\alpha} := \boldsymbol{f} - \boldsymbol{a}'$ and $\boldsymbol{h} := \boldsymbol{u}' + \boldsymbol{g}$.

8. ***Alice:*** *Concatenates $(\boldsymbol{b}_j)_{j\in S}$ to a vector $\boldsymbol{b}_S \in \mathbb{F}_k^{sw}$, extracts $\boldsymbol{b}' := \mathsf{Ext}(\boldsymbol{b}_S)$, and outputs $\boldsymbol{h} + x\boldsymbol{\alpha} - \boldsymbol{b}'$.*

The protocol has an arithmetic complexity of $O(tw) = O(sw) = O(W)$, as required. The communication complexity is dominated by the complexity of Steps 1–3 and Step 7 which is $t \cdot C_{\Pi}(w) + 2W = tO(w) + O(W) = O(W)$ where $C_{\Pi}(w)$ is the communication complexity of Π over width w. (The communication in Steps 4-6 consists of $O(s)$ bits and $O(k)$ field elements for realizing the commitment channel via OT). By employing extractors that shrink their input by a factor of $1 - \beta$ for arbitrarily small constant β (e.g., based on the codes of [33]), we can take $W = (1 - \beta)(1 - o(1))tw$. Recalling that $C_{\Pi}(w)$ approaches to $3w$, the total communication of VOLE3 approaches to $t3w + 2W = 5W/(1 - \beta - o(1))$, i.e., the asymptotic rate approaches to $1/5$. The simulators for Alice and Bob appear in the full version [10], leading to Theorem 1.

8 Batch-OLE

Let $n(k)$ be a polynomial in k and let $G : \mathbb{F}^k \to \mathbb{F}^n$ be a function that can be computed by a constant-depth bounded fan-in arithmetic circuit (aka $\mathbf{NC^0}$ arithmetic circuit). We assume that G is *input-regular* in the sense that each input affects at most $O(n/k)$ outputs. Let $f_G : \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}^k \to \mathbb{F}^n$ be the mapping

$$(\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{x}) \mapsto G(\boldsymbol{x}) \odot \boldsymbol{c} + \boldsymbol{d},$$

where $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{F}^n$, $\boldsymbol{x} \in \mathbb{F}^k$ and $\odot$ stands for entry-wise multiplication. We will be interested in computing the *Generalized Affine* Functionality $\mathcal{F}_G$ which takes the vectors $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{F}^n$ from Alice (the sender) and delivers to the receiver Bob a random vector $\boldsymbol{x} \in \mathbb{F}^k$ together with the outcome of $f_G(\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{x})$. The functionality $\mathcal{F}_G$ is corruption-aware and it allows a malicious Bob to choose $\boldsymbol{x}$ arbitrarily. Throughout, we assume, wlog, that, for every $i \in [n]$, the output of $G_i(\cdot)$ is a non-constant function (i.e., for some x, x', it holds that $G_i(x) \neq G_i(x')$).

Realizing $\mathcal{F}_G$. In the passive setting, it is easy to realize $\mathcal{F}_G$ by using an arithmetic variant of Yao's protocol [52] where the garbled circuit is replaced with fully-decomposable randomized encoding (DARE). (See the full version [10] for background on DARE; for now, the reader can think of DARE as arithmetic garbled circuit.) In fact, this protocol also provides active security against the receiver Bob. We will show how to cheaply upgrade the protocol and provide active security against the sender as well. Our protocol employs DARE for $f = f_G$. By [37,8] there exists a DARE $\hat{f}$ which can be encoded and decoded by an $O(n)$-size arithmetic circuit. Since $\hat{f}$ is decomposable we can write it as

$$\hat{f}(\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{d}; \boldsymbol{r}) = (\hat{f}_0(\boldsymbol{c}, \boldsymbol{d}; \boldsymbol{r}), (\hat{f}_i(x_i; \boldsymbol{r}))_{i\in[k]}).$$

(That is, we collapse the $\boldsymbol{c}$-dependent outputs and the $\boldsymbol{d}$-dependent outputs to a single "block".) Furthermore, for every $i \in [k]$ the function $\hat{f}_i(x_i; \boldsymbol{r})$ can be written as $x_i\boldsymbol{a}_i + \boldsymbol{b_i}$ where the vectors $(\boldsymbol{a}_i, \boldsymbol{b}_i)$ are sampled by a "key-sampling" mapping $K_i : \boldsymbol{r} \mapsto (\boldsymbol{a}_i, \boldsymbol{b_i})$. By padding, we may assume that the key generation functions $K_1, \ldots, K_k$ have uniform output length of w, and since G is an input-regular $\mathbf{NC^0}$ function, it holds that $w = O(n/k)$. Moreover, recall that given $(\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{d})$ we can collectively compute $\hat{f}_0(\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{d}), (K_i(\boldsymbol{r}))_{i\in[k]}$ by $O(n)$ arithmetic operations. We denote the randomness complexity of the DARE by ρ. We realize $\mathcal{F}_G$ by Protocol 8 which employs an ideal batch-VOLE of width $2w$ and length k (that is, k parallel calls to VOLE of width $2w$), and performs 2 additional rounds of interaction. In the full version [10] we prove the following lemma.

Lemma 5. *Protocol 8 realizes $\mathcal{F}_G$ with information-theoretic active security in a constant number of rounds by making a single call to an ideal k-length $O(n/k)$-width VOLE, communicating $O(n)$ field elements, and performing $O(n)$ arithmetic operations. The protocol is perfectly secure against an active adversary that corrupts Bob (receiver) and statistically secure against an adversary that actively corrupts Alice (the sender) where the statistical deviation is $O(D/|\mathbb{F}|) = \mathsf{negl}(k)$ where $D = O(1)$ is the degree of G.*

Protocol 8 (GA protocol). *Let X denote the uniform distribution over $\mathbb{F}^k$. Given an input $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{F}^n$ for Alice and an empty input for Bob, the protocol proceeds as follows.*

1. ***Alice:** selects randomness $\boldsymbol{r} \leftarrow \mathbb{F}^\rho$ for the encoding, and sets $\boldsymbol{v}_0 = \hat{f}_0(\boldsymbol{c}, \boldsymbol{d}; \boldsymbol{r})$ and $(\boldsymbol{a}_i, \boldsymbol{b}_i) = K_i(\boldsymbol{r})$ for every $i \in [k]$. In addition, Alice samples random $\boldsymbol{c}', \boldsymbol{d}' \in \mathbb{F}^n$ and $\boldsymbol{r}' \leftarrow \mathbb{F}^\rho$, and sets $\boldsymbol{v}'_0 = \hat{f}_0(\boldsymbol{c}', \boldsymbol{d}'; \boldsymbol{r})$ and $(\boldsymbol{a}'_i, \boldsymbol{b}'_i) = K_i(\boldsymbol{r}')$ for every $i \in [k]$.*
2. *The parties invoke k-length of $O(n/k)$-width VOLE as follows. Bob samples $\boldsymbol{x} \leftarrow X$, and plays the role of the receiver with the input $\boldsymbol{x} = (x_1, \ldots, x_k)$. Alice plays the role of the sender and sets her inputs to be the $2w$-length vectors $(\boldsymbol{a}_i \circ \boldsymbol{a}'_i)$ and $(\boldsymbol{b}_i \circ \boldsymbol{b}_i)$ for every $i \in [k]$ where $\circ$ denotes concatenation. For every $i \in [k]$, the functionality delivers to Bob the vectors $\boldsymbol{v}_i = x_i\boldsymbol{a}_i + \boldsymbol{b}_i$, $\boldsymbol{v}'_i = x_i\boldsymbol{a}'_i + \boldsymbol{b}'_i$.*
   In addition, Alice sends to Bob the vectors $\boldsymbol{v}_0$ and $\boldsymbol{v}'_0$.

3. ***Bob:*** *decodes the vectors* $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{F}^n$ *by setting* $\boldsymbol{z} = \mathsf{Dec}((\boldsymbol{v}_i)_{0 \le i \le k})$ *and* $\boldsymbol{z}' = \mathsf{Dec}((\boldsymbol{v}'_i)_{0 \le i \le k})$ *where* Dec *is the decoder of the DARE. In addition, Bob sends to Alice a random non-zero field element* $\alpha \leftarrow \mathbb{F}^*$.
4. ***Alice:*** *sends to Bob the* n*-length vectors* $\boldsymbol{\gamma} = \boldsymbol{c} + \alpha \boldsymbol{c}'$ *and* $\boldsymbol{\delta} = \boldsymbol{d} + \alpha \boldsymbol{d}'$.
5. ***Bob:*** *outputs* $(\boldsymbol{x}, \boldsymbol{z})$ *if* $\boldsymbol{z} + \alpha \boldsymbol{z}' = G(\boldsymbol{x}) \odot \boldsymbol{\gamma} + \boldsymbol{\delta}$*, and, otherwise, aborts.*

Constructing batch-OLE. One can easily construct batch-OLE of length n based on a single call to the $\mathcal{F}_G$ functionality where G is a PRG (see [5]). If $G : \mathbb{F}^k \to \mathbb{F}^n$ is an $\mathbf{NC^0}$ PRG with, say quadratic stretch $n = k^2$, we can further realize $\mathcal{F}_G$ using the above protocol with constant computational overhead. Note that the protocol assumes that G is input-regular. This requirement is satisfied by natural PRG candidates in $\mathbf{NC^0}$, and, in fact, it can be fully waived (see the full version [10]).

On the concrete complexity of the protocol. The complexity of the protocol is dominated by the parameters of the PRG G and the cost of the DARE for f_G. Assuming that $G : \mathbb{F}^k \to \mathbb{F}^n$ is a d-local regular PRG whose DARE can be computed and decoded by T arithmetic operations, Protocol 8 has a complexity of $2T$ plus $4n$ additions and $4n$ multiplications. The value of T depends on the locality d and the exact choice of the predicate that computes G_i, and deserves further study. Recall that Protocol 8 essentially realizes "pseudrandom" batch-OLE. This can be upgraded to a "standard" batch-OLE with an additional overhead of $2n$ multiplications and $5n$ additions.

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44987-6_8
2. Alekhnovich, M.: More on average case vs approximation complexity. In: 44th FOCS. pp. 298–307. IEEE Computer Society Press (Oct 2003). https://doi.org/10.1109/SFCS.2003.1238204
3. Applebaum, B.: Cryptographic hardness of random local functions - survey. Comput. Complex. **25**(3), 667–722 (2016). https://doi.org/10.1007/s00037-015-0121-8
4. Applebaum, B., Avron, J., Brzuska, C.: Arithmetic cryptography. J. ACM **64**(2), 10:1–10:74 (2017). https://doi.org/10.1145/3046675
5. Applebaum, B., Damgård, I., Ishai, Y., Nielsen, M., Zichron, L.: Secure arithmetic computation with constant computational overhead. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 223–254. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_8
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in nc^0. SIAM J. Comput. **36**(4), 845–888 (2006). https://doi.org/10.1137/S0097539705446950
7. Applebaum, B., Ishai, Y., Kushilevitz, E.: On pseudorandom generators with linear stretch in nc^0. Comput. Complex. **17**(1), 38–69 (2008). https://doi.org/10.1007/s00037-007-0237-6
8. Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. SIAM J. Comput. **43**(2), 905–929 (2014). https://doi.org/10.1137/120875193

9. Applebaum, B., Kachlon, E.: Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In: Zuckerman, D. (ed.) 60th FOCS. pp. 171–179. IEEE Computer Society Press (Nov 2019). https://doi.org/10.1109/FOCS.2019.00020
10. Applebaum, B., Konstantini, N.: Actively secure arithmetic computation and vole with constant computational overhead. Cryptology ePrint Archive, Paper 2023/270 (2023), https://eprint.iacr.org/2023/270, https://eprint.iacr.org/2023/270
11. Applebaum, B., Lovett, S.: Algebraic attacks against random local functions and their countermeasures. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC. pp. 1087–1100. ACM Press (Jun 2016). https://doi.org/10.1145/2897518.2897554
12. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 673–701. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_26
13. Baum, C., Malozemoff, A.J., Rosen, M.B., Scholl, P.: Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 92–122. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_4
14. Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: 28th ACM STOC. pp. 479–488. ACM Press (May 1996). https://doi.org/10.1145/237814.237996
15. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 336–365. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70700-6_12
16. Bordewijk, J.L.: Inter-reciprocity applied to electrical networks. Applied Scientific Research, Section A **6**(1), 1–74 (1957)
17. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 896–912. ACM Press (Oct 2018). https://doi.org/10.1145/3243734.3243868
18. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 291–308. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3354255
19. Chase, M., Dodis, Y., Ishai, Y., Kraschewski, D., Liu, T., Ostrovsky, R., Vaikuntanathan, V.: Reusable non-interactive secure computation. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 462–488. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_15
20. Chor, B., Goldreich, O., Håstad, J., Friedman, J., Rudich, S., Smolensky, R.: The bit extraction problem of t-resilient functions (preliminary version). In: 26th FOCS. pp. 396–407. IEEE Computer Society Press (Oct 1985). https://doi.org/10.1109/SFCS.1985.55
21. Crépeau, C.: Equivalence between two flavours of oblivious transfers. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 350–354. Springer, Heidelberg (Aug 1988). https://doi.org/10.1007/3-540-48184-2_30
22. Crépeau, C., Kilian, J.: Weakening security assumptions and oblivious transfer (abstract). In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 2–7. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34799-2_1

23. Dittmer, S., Ishai, Y., Ostrovsky, R.: Line-point zero knowledge and its applications. In: Tessaro, S. (ed.) ITC 2021. LIPIcs, vol. 199, pp. 5:1–5:24. Schloss Dagstuhl (2021). https://doi.org/10.4230/LIPIcs.ITC.2021.5
24. Druk, E.: Linear Time Encodable Codes and Cryptography. Master's thesis, Technion (2013)
25. Druk, E., Ishai, Y.: Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In: Naor, M. (ed.) ITCS 2014. pp. 169–182. ACM (Jan 2014). https://doi.org/10.1145/2554797.2554815
26. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (Feb 2005). https://doi.org/10.1007/978-3-540-30576-7_17
27. Genkin, D., Ishai, Y., Prabhakaran, M., Sahai, A., Tromer, E.: Circuits resilient to additive attacks with applications to secure computation. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 495–504. ACM Press (May / Jun 2014). https://doi.org/10.1145/2591796.2591861
28. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: 30th ACM STOC. pp. 151–160. ACM Press (May 1998). https://doi.org/10.1145/276698.276723
29. Ghosh, S., Nielsen, J.B., Nilges, T.: Maliciously secure oblivious linear function evaluation with constant overhead. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 629–659. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70694-8_22
30. Ghosh, S., Nilges, T.: An algebraic approach to maliciously secure private set intersection. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 154–185. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17659-4_6
31. Gilboa, N.: Two party RSA key generation. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 116–129. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_8
32. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987). https://doi.org/10.1145/28395.28420
33. Guruswami, V., Indyk, P.: Linear-time encodable/decodable codes with near-optimal rate. IEEE Trans. Inf. Theory **51**(10), 3393–3400 (2005). https://doi.org/10.1109/TIT.2005.855587
34. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_9
35. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: Fifth Israel Symposium on Theory of Computing and Systems, ISTCS 1997, Ramat-Gan, Israel, June 17-19, 1997, Proceedings. pp. 174–184. IEEE Computer Society (1997). https://doi.org/10.1109/ISTCS.1997.595170
36. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st FOCS. pp. 294–304. IEEE Computer Society Press (Nov 2000). https://doi.org/10.1109/SFCS.2000.892118
37. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Ruiz, F.T., Bueno, R.M., Hennessy,

M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (Jul 2002). https://doi.org/10.1007/3-540-45465-9_22
38. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 433–442. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374438
39. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_32
40. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 294–314. Springer, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5_18
41. Kushilevitz, E., Lindell, Y., Rabin, T.: Information-theoretically secure protocols and security under composition. SIAM J. Comput. **39**(5), 2090–2112 (2010). https://doi.org/10.1137/090755886, https://doi.org/10.1137/090755886
42. Lindell, Y.: General composition and universal composability in secure multi-party computation. In: 44th FOCS. pp. 394–403. IEEE Computer Society Press (Oct 2003). https://doi.org/10.1109/SFCS.2003.1238213
43. Mohassel, P., Weinreb, E.: Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 481–496. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_27
44. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: 31st ACM STOC. pp. 245–254. ACM Press (May 1999). https://doi.org/10.1145/301250.301312
45. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA. pp. 448–457. ACM-SIAM (Jan 2001)
46. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_31
47. Rabin, M.O.: How to exchange secrets with oblivious transfer. IACR Cryptol. ePrint Arch. p. 187 (2005), http://eprint.iacr.org/2005/187
48. Schoppmann, P., Gascón, A., Reichert, L., Raykova, M.: Distributed vector-OLE: Improved constructions and implementation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 1055–1072. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3363228
49. Shankar, B., Srinathan, K., Rangan, C.P.: Alternative protocols for generalized oblivious transfer. In: Rao, S., Chatterjee, M., Jayanti, P., Murthy, C.S.R., Saha, S.K. (eds.) Distributed Computing and Networking, 9th International Conference, ICDCN 2008, Kolkata, India, January 5-8, 2008. Lecture Notes in Computer Science, vol. 4904, pp. 304–309. Springer (2008). https://doi.org/10.1007/978-3-540-77444-0_31
50. Tassa, T.: Generalized oblivious transfer by secret sharing. Des. Codes Cryptogr. **58**(1), 11–21 (2011). https://doi.org/10.1007/s10623-010-9378-8
51. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In: 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021. pp. 1074–1091. IEEE (2021). https://doi.org/10.1109/SP40001.2021.00056

52. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986). https://doi.org/10.1109/SFCS.1986.25
53. Zichron, L.: Locally computable arithmetic pseudorandom generators. Master's thesis, Tel Aviv University (2017), available from Applebaum's home page.