A Note on the Communication Complexity of Multiparty Computation in the Correlated Randomness Model

Geoffroy Couteau

A Note on the Communication Complexity of Multiparty Computation in the Correlated Randomness Model

Geoffroy Couteau

CNRS, IRIF, Paris-Diderot University, France
couteau@irif.fr

Abstract. Secure multiparty computation (MPC) addresses the challenge of evaluating functions on secret inputs without compromising their privacy. A central question in multiparty computation is to understand the amount of communication needed to securely evaluate a circuit of size s. In this work, we revisit this fundamental question in the setting of information-theoretically secure MPC in the correlated randomness model, where a trusted dealer distributes correlated random coins, independent of the inputs, to all parties before the start of the protocol. This setting is of strong theoretical interest, and has led to the most practically efficient MPC protocols known to date.
While it is known that protocols with optimal communication (proportional to input plus output size) can be obtained from the LWE assumption, and that protocols with sublinear communication $o(s)$ can be obtained from the DDH assumption, the question of constructing protocols with $o(s)$ communication remains wide open for the important case of information-theoretic MPC in the correlated randomness model; all known protocols in this model require $O(s)$ communication in the online phase.
In this work, we exhibit the first generic multiparty computation protocol in the correlated randomness model with communication sublinear in the circuit size, for a large class of circuits. More precisely, we show the following: any size-s *layered* circuit (whose nodes can be partitioned into layers so that any edge connects adjacent layers) can be evaluated with $O(s/\log\log s)$ communication. Our results holds for both boolean and arithmetic circuits, in the honest-but-curious setting, and do not assume honest majority. For boolean circuits, we extend our results to handle malicious corruption.

Keywords. multiparty computation, correlated randomness model, information-theoretic security, sublinear communication

1 Introduction

Secure multiparty computation (MPC) allows n players with inputs $(x_1, \cdots, x_n)$ to jointly evaluate a function f, while leaking no information on their own input beyond the output of the function. It is a fundamental problem in cryptography, which has received a considerable attention since its introduction in the seminal works of Yao [Yao86], and Goldreich, Micali, and Wigderson [GMW87b, GMW87a] (GMW). One of the core questions in secure multiparty computation is to understand the amount of communication needed to securely compute a function. For almost three decades after the protocols of Yao and GMW, all known constructions of secure computation protocols required a communication proportional to the circuit size of the function, and understanding whether this was inherent was a major open problem.

Secure Computation with Sublinear Communication. In 2009, this situation changed with the introduction by Gentry of the first fully-homomorphic encryption scheme [Gen09] (FHE), which led to secure computation protocols with communication independent of the size of the function (proportional only to its input size and its output size), under (a circular-security variant of) the LWE assumption. This resolved the long-standing open problem of designing MPC protocols with optimal (asymptotic) communication, although only under a specific assumption. More recently, the circuit-size barrier was broken again under the DDH assumption in [BGI16], for a large class of structured circuits[1] and in the two-party case. However, while these results are of strong theoretical interest, they require expensive computations.

[1] The work of [BGI16] considered, as we will do in this work, boolean circuits which can be divided into layers such as any edge connects adjacent layers. Such circuits are called *layered boolean circuits*.

Secure Computation in the Correlated Randomness Model. While secure computation (with no honest majority) is known to require computational assumptions, it was observed in several works (e.g. [IPS08, DPSZ12]) that executing a pre-computation phase independent of the inputs to the protocol, during which correlated random bits are distributed to the parties, allows to make the online phase both information-theoretically secure and significantly more efficient, by removing any expensive cryptographic operation from the online computation phase. These observations led to the development of increasingly efficient secure computation protocols in the correlated randomness model, e.g. [KOS16, DNNR17], which are currently considered the most practical secure computation protocols. Yet, unlike computationally secure protocols, all known unconditionally secure protocols in the correlated randomness model (with computation and storage polynomial in the circuit size) require communication proportional to the circuit size of the function. Therefore, the major question of understanding the communication required for multiparty computation remains wide open for the important case of MPC in the correlated randomness model, which captures the best candidates for practical secure computation. This is the question we address in this work: must MPC protocols in the correlated randomness model inherently use a communication linear in the size of the circuit? Or, in other words, can we get the best of both worlds: unconditional security with high practical efficiency, and sublinear communication?

On the Communication of Secure Computation in the Correlated Randomness Model. A partial answer to this question was given in [IKM+13], where the authors designed a *one-time truth-table* protocol, which allows to evaluate any function $f : \{0,1\}^n \mapsto \{0,1\}^m$ with unconditional security in the correlated randomness model, with optimal communication $O(n+m)$. However, this protocol requires storing an exponential number (in n) of correlated random bits (polynomial in the size of the entire truth-table of f), which makes it practical only for boolean functions with very small inputs. Furthermore, it was argued in [IKM+13] that reducing the amount of correlated random coins from exponential to polynomial (in the input size) for any function f is unlikely to be feasible, as it would imply an unexpected breakthrough for long-standing open problems related to private information retrieval.

While this negative result does not rule out a sublinear-communication protocol with small storage for circuits, this observation and the fact that all known protocols (with polynomial storage) have communication proportional to the circuit size s of the function have been seen as indications that breaking the circuit-size barrier for multiparty computation in the correlated randomness model might be non-trivial. For instance, it was mentioned in [DZ13] that "the results and evidence we know suggest that getting constant overhead [over the circuit size of the function] is the goal we can realistically hope to achieve". More recently in [DNPR16], the authors mentioned that "whether we can have constant round protocols and/or communication complexity much smaller than the size of the circuit and still be efficient (polynomial-time) in the circuit size of the function is a long-standing open problem".

In [DNPR16], the authors made progresses toward understanding why existing protocols have been stuck at the circuit-size barrier, by identifying a property shared by all known efficient protocols in the correlated randomness model, which states (informally) that they evaluate the function in a "gate-by-gate" fashion, and require communication for every multiplication gate. They demonstrated that all protocols following this approach (with passive security and dishonest majority) must inherently have communication proportional to the circuit size of the function. They concluded that improving the communication complexity of secure computation in the correlated randomness model requires a fundamentally new approach, and mentioned that the main question left open in their work is to find out whether their bound does hold for any protocol which is efficient in the circuit size of the function. This is the problem we address in this work.

1.1 Our Contribution

In this paper, we construct for the first time protocols with polynomial storage and communication sublinear in the circuit size, for a large class of circuit. Perhaps surprisingly, our results turn out to be relatively simple to obtain; it appears however that this simple solution was missed in previous works.

Sublinear Protocol for Structured Circuits. We exhibit a generic secure computation protocol in the correlated randomness model, with communication sublinear in the circuit size. More specifically, we consider *layered boolean circuits* (LBC), whose nodes can be arranged into layers so that any edge connects adjacent layers. We prove the following: for any N, there is an unconditionally secure N-party protocol that evaluates an arbitrary LBC of size s with n inputs and m outputs, with total communication

$$O\left(n + N \cdot \left(m + \frac{s}{\log\log s}\right)\right),$$

sublinear in the size of the circuit, and polynomial storage $O(s^2/\log\log s)$, in the correlated randomness model against semi-honest adversaries, with dishonest majority. While this requires an arguably large storage, it can be reduced to being only slightly superlinear in s, namely

$$O\left(s \cdot \frac{2^{(\log s)^{1/c}}}{\log\log s}\right),$$

at the cost of increasing the communication to $O(n + N \cdot (m + c \cdot s/\log\log s))$ (for an arbitrary $c = o(\log\log n)$). Our protocol enjoys perfect security, computational complexity $O(s \log s/\log\log s + n + m)$, and round complexity $d/\log\log s$, where d is the depth of the circuit. All the constants involved are very small (in fact equal to one, up to low order terms), and the computation involves solely searching lookup tables.

Extensions. We generalize our result to secure evaluation of arbitrary *layered arithmetic circuits* (LAC) over any (possibly exponentially large) field $\mathbb{F}$, by relying on a connection between MPC with correlated randomness and the classical notion of private simultaneous message protocols [FKN94]. The resulting protocol for arithmetic circuits has costs comparable to the boolean version. Furthermore, we show that all our results can be extended to the stronger function-independent preprocessing model, where only a bound on the size of the circuit is known in the preprocessing phase, and that the communication can be improved for "tall and narrow" circuits. Eventually, using the techniques of [DNNR17, KOR+17], our protocols directly extend to the malicious setting for boolean circuits, at an additive cost of $N \cdot \kappa$ bits of communication (for some statistical security parameter κ), and a $O(\kappa)$ overhead in computation and correlated randomness (more advanced techniques from [DNNR17] can be used to make this overhead constant).

Static vs Adaptive Setting. While we focus for simplicity on the static setting in this work, where the adversary decides before the protocol which parties to corrupt, our protocols can be proven to also satisfy adaptive security in a relatively straightforward way. Indeed, when it must reveal the input of a party which is being corrupted by the adversary, the simulator of our main protocol (and its variants) can easily explain the view of the adversary as being consistent with any input of its choice, by choosing the preprocessing material in an appropriate way. As the view of the adversary will always consist of values perfectly masked by random coins generated in the preprocessing phase, there will always be a choice of preprocessing material which "unmask" the values known to the adversary to any value chosen by the simulator.

1.2 Our Method

Perhaps surprisingly, our method does not depart significantly from existing techniques in secure computation. Our starting point is the one-time truth-table (OTTT) protocol of [IKM+13], which has optimal communication but requires an exponential amount of data. It has been observed in several works that using OTTT as an internal component in secure protocols can be used to reduce their communication. For example, it was suggested to use OTTT to securely compute S-boxes in AES in [DNNR17, KOR+17], as they can be efficiently represented as small lookup-tables. More recently, the work of [DKS+17] developped methods to automatically create tradeoffs between communication and computation in secure protocols, by relying on a compiler that transforms high-level descriptions of a function into a lookup-table-based representation of the function. All these works rely on the fact that, for functions that can be broken into small interconnected lookup-tables, the protocol of [IKM+13] can be used to save some communication.

Dividing Layered Boolean Circuits into Local Functions. In this work, we show that this intuition can in fact be extended to *arbitrary* layered boolean circuit of size s, and that the savings obtained this way lead to a protocol with $o(s)$ communication. Our protocol builds upon a variant of the result of [IKM+13], which states that every function can be securely evaluated in the correlated randomness model with perfect security, optimal communication, and exponential storage. Our variant relies on the observation that when evaluating *local functions*, where each output bit depends on a number c of input bits, we can reduce the storage cost of the protocol of [IKM+13] from being exponential in the input size to being only exponential in the locality parameter c. Indeed, consider the task of securely evaluating a function with n input bits, and m output bits. The protocol of [IKM+13] (called OTTT, for one-time truth table) requires the parties to store shares of (a shifted version of) the truth table of the function, which has size $m \cdot 2^n$, exponential in the input size. When the function is c-local, however, there is a better solution: the parties can store shares of (shifted variants of) truth tables corresponding to each function mapping c input bits to a given output bit, for a total storage cost of $m \cdot 2^c$. Some care must be taken, as doing straightfoward parallel repetitions of the OTTT protocol for each subfunction would increase the communication from $O(n)$ to $O(c \cdot m)$; we show that carefully avoiding redundancies in the secret-shared representation of the input allows to bring this cost back to $O(n)$. We formally state this result in a lemma, which we call *core lemma*.

Given the core lemma, our result is obtained by breaking an arbitrary layered circuit into chunks, each chunk containing some number k of consecutive layers. We observe that, as the underlying directed graph of the circuit has indegree 2, each value associated to the last layer of a chunk can be computed as a function of at most 2^k values on the last layer of the previous chunk. Therefore, computing all the values on the last layer of a chunk can be reduced to evaluating a 2^k-local function of the values on the last layer of the previous chunk. Using the core lemma, this can be done using $O(w \cdot 2^{2^k})$ bits of preprocessing material, where w is the width of the input layer, with a communication proportional to w only. If the circuit has size s, width w, and depth d, this means that the circuit can be securely evaluated in a chunk-by-chunk fashion, with total communication $O((d/k) \cdot w) = O(s/k)$, using $O((d/k) \cdot 2^{2^k})$ bits of correlated randomness; setting $k \leftarrow \log\log s$ gives the claimed result.[2]

Extending the Result to Arithmetic Circuit. The above method breaks down in the case of arithmetic circuits over large order fields. While we can decompose an arbitrary LBC into polynomial-size truth tables (by breaking it into interconnected functions operating on logarithmically many inputs), this is not true anymore for arithmetic circuit over fields of exponential size, where even a function with a single input will have an exponential-size truth table. We nevertheless obtain a comparable result for arithmetic circuit, building upon a relation with the notion of private simultaneous message (PSM) protocols [FKN94], which establishes that PSM protocols with some additional decomposability property can be used to build two-party secure computation protocols in the correlated randomness model. This link was indirectly established in [BIKK14], where a connection was drawn both between PSM and PIR, and between MPC in the correlated randomness and PIR. Building upon a recent PSM protocol of [LVW17] for multivariate polynomial evaluation, we get an arithmetic analogue of the protocol of [IKM+13], which relies on the representation of arithmetic functions as multivariate polynomials. From this protocol, we derive a new version of our core lemma, tailored to the arithmetic setting, which directly leads to a secure computation protocol with communication $O(s/\log\log s)$ for layered arithmetic circuits over arbitrary fields.

We note that, while lookup-table-based secure computation protocols for boolean circuits have been investigated, the extension of this approach to the arithmetic setting was (to our knowledge) never observed before. As a minor side contribution of independent interest, we further observe that our generalization to the arithmetic setting does in fact also provide some improvement over the original TinyTable protocol [DNNR17] in the boolean setting: by replacing the lookup-table-based representation of boolean gates by a multivariate-polynomial-based representation, we show that the storage requirement of their protocol can be reduced by 25%.

[2] We assume $w \cdot d = O(s)$ in this high level explanation for simplicity only, this is not a necessary condition in the actual construction.

On the Lower Bounds of [IKM$^+$13, DNPR16]. It should be noted that our protocols do not follow the standard gate-by-gate design of unconditionally secure protocols in the correlated randomness model, hence our result does not contradict the lower bound of [DNPR16]. Moreover, our results apply only to circuits, while the implausibility result of [IKM$^+$13] assumes the existence of a low-storage protocols for evaluating *any function*, which our results do not provide. Therefore, they do not lead to unexpected breakthroughs for information-theoretic private information retrieval.

1.3 Related Work

The possibility of securely computing functions given access to a source of correlated random coins was first studied in the work of Beaver for the (MPC-complete) oblivious-transfer functionality in [Bea95], and later generalized to the *commodity-based* model, where multiple servers generate correlated random coins in a honest majority setting in [Bea97]. The study of multiparty computation in the *preprocessing model*, where the correlated-randomness coin-generation phase is implemented with a computationally secure MPC protocol, was initiated in [Kil88, Bea92, IPS08]. These works started a rich line of work on increasingly efficient MPC protocols in the preprocessing model [IPS09, BDOZ11, NNOB12, DPSZ12, DZ13, DLT14, LOS14, FKOS15, BLN$^+$15, DZ16, KOS16, DNNR17].

The quest for secure multiparty computation protocols with low-communication was initiated in [BFKR91], which gave a protocol with optimal communication, albeit with exponential computation and only for a number of party linear in the input size. An optimal communication protocol with exponential complexity was also given in [NN01]. The work of [BI05] gives a low-communication protocol for constant-depth circuit, for a number of parties polylogarithmic in the circuit size. The breakthrough result of Gentry [Gen09] led to optimal communication protocols in the computational setting [DFH12, AJL$^+$12] under the LWE assumption.[3] More recently, computationally secure MPC protocols with sublinear communication were achieved from the DDH assumption in [BGI16].

The study of low-communication protocols in the correlated randomness model was initiated in [IKM$^+$13], where a protocol with optimal communication and exponential storage complexity was presented. The same paper showed that improving the storage requirement for all functions would imply a breakthrough in information-theoretic PIR. The work of [BIKK14] reduces the storage requirement for functions with n inputs to $2^{O(\sqrt{n})}$, at the cost of increasing the communication complexity to $2^{O(\sqrt{n})}$. The work of [BIKO12] leads to low-communication protocols in the correlated randomness model for the special case of depth-2 circuits with a layer of OR gates and a layer of gates computing a sum modulo m, for composite m. All known protocols for evaluating arbitrary circuits in the correlated randomness model (with polynomial computation and storage) use communication linear in the circuit size. This limitation was formally studied recently in [DNPR16], where it was shown that it is inherent in the setting of gate-by-gate protocols.

The idea of using truth-table representation to reduce the communication of secure computation protocols first arose in [CDv88], and was developped in [IKM$^+$13]. It was later used implicitely in [KK13], to construct one-out-of-two oblivious transfer for short string from one-out-of-N oblivious transfer, and in the works of [DNNR17, KOR$^+$17, DKS$^+$17] to evaluate circuits with an appropriate structure.

On the Relation to [DNNR17]. At a late stage of our work on this paper, it was brought to our attention that the main techniques underlying the proof of our core lemma – informally, breaking a function into interconnected truth-tables, representing each outgoing wires from a table with secret-shared values, and carefully avoiding all redundancies for wires which are used by several tables – are already implicitely present in [DNNR17]. Indeed, [DNNR17] already explored the possibility of breaking a circuit into small interconnected truth table, avoiding redundancies in the secret-shared representation of the values associated to each wire, and envisionned the possibility of generalizing this to larger tables. However, it appears that the authors of [DNNR17] have overlooked the surprising potential consequences of these techniques, which we explore in this paper. Therefore, our work can be seen as indentifying and abstracting out the technical ideas underlying our main

[3] More precisely, the protocol needs to assume the circular security of an LWE-based encryption scheme; alternatively, it can be based on the LWE assumption only, but the communication will grow with the depth of the circuit.

result (as well as providing additional contributions, such as the extension to the arithmetic setting), but while the core lemma is new to our work, we cannot (and do not) claim the novelty of the techniques used in its proof, which should be credited to [DNNR17]. Still, we believe that our result remains interesting and surprising, and that it deserves to be explicitely presented.

1.4 On the Practical Efficiency of our Protocols

In spite of its theoretical nature, our result can in fact lead to concrete efficiency improvements for secure multiparty computation. We focus for simplicity on the case of two-party computation, and argue that our protocols can lead to improved efficiency, for useful types of computation. The state-of-the-art protocol for secure two-party computation in the correlated randomness model is, to our knowledge, the protocol of [DNNR17] (in both the passive setting and the active setting), which also relies on an OTTT-based evaluation of a boolean circuit. In the online phase, the protocol of [DNNR17] communicates 2 bits per AND gate (one from each player), and no bit at all for XOR and NOT gates (we note that our protocols can be readily adapted to allow for free XOR and NOT gates as well).

Concrete Efficiency. Using our protocol with $k = 2$, we get a two-party protocol which communicates on average a *single bit* per AND gate, improving over the protocol of [DNNR17] by 50% in both the passive and the active setting, for arbitrary layered circuits. This comes at the cost of storing 8 times more preprocessed data (a factor $2^{2^k}/k = 8$ for $k = 2$), and a factor $2^k/k = 2$ in computation (which comes from the need to search four-times larger lookup-tables). As noted in [DNNR17], the limiting factor in a concrete implementation of TinyTable is the bandwidth, hence we expect that an implementation of our protocol would result in concrete improvements over [DNNR17] in the speed of the online phase.

On the Generality of Layered Boolean Circuits. Unlike [DNNR17], however, our construction is restricted to layered boolean circuits. While this is a large class of circuits, and getting improved secure computation protocols for this class was already seen as an interesting goal in previous papers [BGI16], one might wonder whether this class captures *useful* circuits, ones that arise naturally in some applications. We argue that it is the case, by providing a (non-exhaustive) list of types of circuits that are well-suited for our protocols. We stress that this list is only for illustration purpose; many more examples can be found.

- *FFT circuit.* The circuit for the fast Fourier transform, which is used in signal processing and integer multiplication, and the circuit for permutation networks [Wak68], which allow to compute arbitrary permutations of the input, have the exact same structure and are layered. For these circuits, which occur naturally in many applications, our protocol leads to an online communication of $O(n \log n / \log \log n)$ instead of $O(n \log n)$.
- *Symmetric crypto primitives.* It was already observed previously that any computation involving large truth tables, such as block ciphers (e.g. AES), have the appropriate structure to be evaluated efficiently with our approach. More generally, algorithms that proceed in sequences of low-complexity rounds, where each round requires only the state of the previous round (and the input), are naturally "layered by blocks", which suffices for our result to apply. This structure is common to many primitives in symmetric cryptography.
- *Circuits for problems with a dynamic-programming algorithm.* Dynamic programming algorithm naturally proceed in stages, such that the computation at each stage depends on a (usually small) state of values stored after the previous stage. Such dynamic programming algorithms arise for example in various useful types of distance measures used in genetic computation, such as the Smith-Waterman distance [SW81], or the Levenshtein distance [Lev66] and its variants (LCS, weighted Levenshtein distance, etc). Privacy-preserving genomic computations are an important application of secure computation, hence the secure computation of the aforementioned measures (which are among the fundamental building blocks of computational biology) has been considered at length (see e.g. [AKD03, SPO+06, JKS08, HEKM11, ALSZ13, CKL15]). The natural circuit for computing Levenshtein and Smith-Waterman distances have size $O(n^2 \log n)$, but can be

computed with online communication $O(n^2)$ with our protocol (the $\log n$ shaving comes from the high locality of dynamic programming algorithms; our result leads to better sublinearity guarantee for very local computations).

1.5 On Implementing the Correlated Randomness Model

It is well known that the distribution of correlated random coins in the preprocessing phase can be implemented by any generic MPC protocol. However, in our setting, generic approaches would require a communication superlinear in the circuit size. We note that, for the specific case of generating random shares of correlated strings, there are better (theoretical) solutions: under the learning with error assumption, or under (variants of) the decisional Diffie-Hellman assumption in the two-party case, the preprocessing phase of our protocols can be implemented with *constant* communication $\mathsf{poly}(\lambda)$ (where λ is a security parameter), independent of the size of the circuit, resulting in protocols with sublinear total communication $O(s/\log\log s + \mathsf{poly}(\lambda))$, and information-theoretically secure online phase.

We briefly sketch how the preprocessing phase can be implemented with constant communication. The main technical tool is a primitive known as *homomorphic secret sharing* [BGI16] (HSS); the idea of using HSS to the implement preprocessing phase of MPC protocols was suggested in [BGI17,BCG$^+$17]. Informally, an HSS scheme for a class of functions F allows to secretly share an input x between several parties, such that given its share, each party can *locally* compute an additive share of $f(x)$, for any $f \in F$. Given an HSS scheme for all circuits, the preprocessing phase can be implemented as follows: we assume without loss of generality that the trusted dealer first samples a long random string x, computes $f(x)$ for some specified function f, and distributes random additive shares of $f(x)$ to the parties (e.g. in our protocol, f would output $\approx s/\log\log s$ shifted truth-tables). To implement this preprocessing phase, the parties jointly and securely construct, using a general purpose MPC protocol, an homomorphic secret sharing of a random PRF key K. Then, all parties locally evaluate the function f' that takes some counter c, generates pseudorandom coins x from this counter using the PRF with key K (e.g. by computing $\mathrm{PRF}(K, c)$, $\mathrm{PRF}(K, c+1)$, and so on), and returns $f(x)$. This way, with no further communication except for a one-time generation of the sharing of K (which takes communication $\mathsf{poly}(\lambda)$, independently of s), the parties obtain correlated (pseudo) random coins. An HSS scheme for all functions (and a PRF) can be constructed under the LWE assumption [BGI15, JRS17]. With a more involved construction, a protocol can also be obtained from DDH: under the DDH assumption, there exists an approximately-correct HSS scheme for NC_1 [BGI16], in the two-party setting. Noting that the preprocessing function is parallelizable (in NC_0) and that there exists PRFs in NC_1 under the DDH assumption, we can implement the previous strategy from DDH. The correlated random coins obtained this way are not all correct, but the approximately-correct HSS scheme of [BGI16] allows the parties to make the error probability arbitrarily small, and to detect when an output is erroneous. By setting the error parameter so that, with overwhelming probability, a small (constant) number of correlated random coins will be erroneous, and by introducing some redundancy in the coins generated this way, the parties can simply reveal to each other which correlated coins are susceptible to be erroneous (indicating the position of erroneous bits only requires $O(\log s)$ communication), and locally delete them. To prove security in spite of this small leakage, we need to rely on slightly leakage-resilient PRF and HSS, which can both be constructed from DDH-based primitives using standard approaches. We refer the reader to the full version [BCG$^+$18] of [BCG$^+$17] for a detailed overview of this approach.

1.6 Organization

Section 2 introduces our notations, and recalls standard preliminaries on circuits. In section 3, we summarize the contributions of this paper in the form of a list of theorems, formally state the core lemma on which these theorems are based, and prove it. Section 4 builds upon the core lemma; it introduces our main protocol and several variants, and proves its security. In Section 5, we discuss the extension of our protocols to the malicious setting. Eventually, Section 6 lists some questions left open by our work, that we believe to be of interest for future works.

2 Preliminaries

Notations. Let k be an integer. We let $\{0,1\}^k$ denote the set of bitstrings of length k. For two strings (x, y) in $\{0,1\}^k$, we denote by $x \oplus y$ their bitwise xor. Given a subset S of $[k]$, $x[S]$ denotes the subsequence of the bits of x with indices from S. We use bold letters to denote vector; for a vector $\boldsymbol{x} = (x_1, \cdots, x_N)$, $\boldsymbol{x}[S]$ denotes the vector $(x_1[S], \cdots, x_N[S])$. For a matrix M, we denote $M|_{i,j}$ its entry (i, j).

2.1 Circuits

Boolean Circuits. A boolean circuit C with n inputs and m outputs is a directed acyclic graph with two types of nodes:

- The *input nodes* are labelled according to variables $\{x_1, \cdots, x_n\}$;
- The *gates* are labelled according to a base B of boolean functions.

In this work, we will focus on boolean circuits with indegree two (hence, B contains boolean functions with domain $\{0,1\}$ or $\{0,1\}^2$). C contains m gates with no children, which are called *output gates*. If there is a path between two nodes (v, v'), we say that v is an *ancestor* of v'. The *size* $\mathsf{size}(C)$ of C is the number of its nodes; its *depth* $\mathsf{depth}(C)$ is the length of the longest path from an input node to an output gate. The *width* of a circuit $C = (V, E)$ is defined as $\mathsf{width}(C) = \max_{1 \leq i \leq \mathsf{depth}(C)} \#\{v \in V \mid (0 \leq \mathsf{depth}(v) \leq i) \wedge (\exists w, (v, w) \in E \wedge \mathsf{depth}(w) > i)\}$.

Layered Boolean Circuits. In this work, we will consider a special type of boolean circuits, called *layered boolean circuits* (LBC). An LBC is a boolean circuit C whose nodes can be partitioned into $d = \mathsf{depth}(C)$ layers $(L_1, \cdots, L_d)$, such that any edge (u, v) of C satisfies $u \in L_i$ and $v \in L_{i+1}$ for some $i \leq d-1$. Note that the width of a layered boolean circuit is also the maximal number of non-output gates contained in any single layer. Evaluating a circuit C on input $x \in \{0,1\}^n$ is done by assigning the bits of x to the variables $\{x_1, \cdots, x_n\}$, and then associating to each gate g of C (seen as a boolean function) the bit obtained by evaluating g on the values associated to its parent nodes. The output of C on input x, denoted $C(x)$, is the bit-string associated to the output gates.

Arithmetic Circuits. We define arithmetic circuits over a field $\mathbb{F}$ comparably to boolean circuits, as directed acyclic graphs with input nodes and arithmetic gates. Input nodes are labeled with variables $\{x_1, \cdots, x_n\}$ over $\mathbb{F}$, and the gates compute negation, addition, or multiplication over $\mathbb{F}$. Note that boolean circuits correspond to the special case of arithmetic circuits over the field $\mathbb{F}_2$; we extend layered boolean circuits to layered arithmetic circuits (LAC) in a similar way.

2.2 One-Time Truth Tables

We recall the one-time truth-table protocol of [IKM+13], which is at the heart of our protocols. It allows multiple parties to jointly evaluate a function $f : X_1 \times X_2 \times \cdots \times X_N \mapsto Z$, by sharing between all parties a scrambled version of the truth table of f. We focus for simplicity on a scenario where all parties receive the same output, but the protocol can be trivially generalized to a setting where the parties receive different outputs. The protocol is represented on Figure 1; it has optimal communication $\sum_i \log|X_i| + N \cdot \log|Z|$, and exponential storage complexity $|Z| \cdot \prod_i |X_i|$ per party.

3 Theorems and Core Lemma

In this section, we formally introduce the theorems which we will prove in this work, state the core lemma from which we will derive them, and prove it.

Network Model. We consider protocols involving N parties communicating over synchronous and authenticated broadcast channel. Note that broadcasts channels can be unconditionally implemented from (insecure) point-to-point channels in the correlated randomness model.

Protocol OTTT

Functionality:

- Public parameters: an N-party functionality $f : X_1 \times X_2 \times \cdots \times X_N \mapsto Z$, where the $(X_i, +)$ and $(Z, +)$ are groups.
- The parties $(P_1, \cdots, P_N)$ hold respective inputs $\boldsymbol{x} = (x_1, \cdots, x_N)$;
- Output: each party P_i learns $z = f(\boldsymbol{x})$.

Preprocessing :

1. Sample $\boldsymbol{r} = (r_1, \cdots, r_N) \xleftarrow{\$} X_1 \times X_2 \times \cdots \times X_N$. Let M denote the truth-table of f permuted with the shifts $\boldsymbol{r}$, *i.e.*, for any $\boldsymbol{x} \in X_1 \times X_2 \times \cdots \times X_N$, $M|_{\boldsymbol{x}+\boldsymbol{r}} = f(\boldsymbol{x})$.
2. Let $(M_i)_{i \leq N}$ be a random (N-out-of-N) secret sharing of M. Output (r_i, M_i) to each party P_i.

Protocol($\boldsymbol{x}$) :

1. Each party P_i with input x_i broadcasts $u_i \leftarrow x_i + r_i$.
2. Each party P_i broadcasts $z_i \leftarrow M_i|_{\boldsymbol{u}}$. All parties reconstruct $z \leftarrow \sum_{i=1}^{N} z_i$.

Fig. 1. Protocol OTTT for evaluating an arbitrary N-party functionality f in the correlated randomness model, against a passively corrupted majority

Functionalities. An N-party functionality $F : X_1 \times X_2 \times \cdots \times X_n \mapsto Z_1 \times Z_2 \times \cdots \times Z_N$ specifies a mapping from the N input of each party to N outputs (one for each party). Such functionalities capture arbitrary non-reactive computation tasks. A useful special case of (randomized) N-party functionalities are *secret sharing functionalities* for functions over an abelian group $(\mathbb{G}, +)$: a protocol computes secret shares of a function $g : \mathbb{G} \mapsto \mathbb{G}$ if it computes the (randomized) N-party functionality which, on input $(x_1, \cdots, x_N) \in \mathbb{G}^N$, outputs N uniformly random group elements $(z_1, \cdots, z_N) \in \mathbb{G}^N$ subject to $\sum_{i=1}^{N} z_i = g(\sum_{i=1}^{N} x_i)$. This captures the situation where the parties hold secret shares of an input to a (deterministic) function, and want to receive secret shares of the output of the function.

3.1 Theorems

Following is a summary of the results that we obtain in the subsequent sections.

Theorem 1. *For any N-party functionality f represented by a layered (boolean or arithmetic) circuit C of size s with n inputs and m outputs, and for any integer k, there is a perfectly secure protocol which realizes f in the preprocessing model against semi-honest parties, without honest majority, with communication $n + N \cdot (m + \lceil s/k \rceil)$ and storage $n/N + (m + \lceil s/k \rceil) \cdot (2^{2^k} + 1)$.*

In the above theorem, "storage" refers to the number of correlated random coins stored by each party at the end of the preprocessing phase (counted as a number of bits in the boolean case, and as a number of field elements in the arithmetic case). This gives, setting $k = \log \log s$,

Corollary 2. *There is a protocol that perfectly realizes any N-party functionality f (in the function-dependent preprocessing model and against semi-honest parties, without honest majority) represented by a layered (boolean or arithmetic) circuit C of size s with n inputs and m outputs, with communication $O(n + N \cdot (m + s/\log \log s))$ and polynomial storage.*

Building on the same techniques, we can also obtain a comparable result in the stronger *function-independent* correlated randomness model, where the correlated randomness is not allowed to depend on the target functionality (but is only given a bound on its size):

Theorem 3. *For any N-party functionality f represented by a layered (boolean or arithmetic) circuit C of size s with n inputs and m outputs, and for any integer k, there is a perfectly secure protocol which realizes f in the function-independent preprocessing model against semi-honest parties, without honest majority, with communication $n + N \cdot (m + \lceil s/k \rceil)$ and storage $n/N + (m + \lceil s/k \rceil) \cdot (2^{k+2^{2^k}} + 1)$.*

Setting $k = \log\log\log s$ gives us

Corollary 4. *There is a protocol that perfectly realizes any N-party functionality f (in the function-independent preprocessing model and against semi-honest parties, without honest majority) represented by a layered (boolean or arithmetic) circuit C of size s with n inputs and m outputs, with communication $O(n + N \cdot (m + s/\log\log\log s))$ and polynomial storage.*

Finally, we can obtain a stronger sublinearity guarantee for "tall and narrow" layered circuits:

Theorem 5. *For any N-party functionality f represented by a layered (boolean or arithmetic) circuit C of size s and width w with n inputs and m outputs, and for any integer k, there is a perfectly secure protocol which realizes f in the preprocessing model against semi-honest parties, without honest majority, with communication $n + N \cdot (m + \lceil s/k \rceil)$ and storage $n/N + (m + \lceil s/k \rceil) \cdot (2^{w \cdot k} + 1)$.*

For example, setting $k = \sqrt{\log s}$ gives us

Corollary 6. *There is a protocol that perfectly realizes any N-party functionality f (in the preprocessing model and against semi-honest parties, without honest majority) represented by a "tall and narrow" layered (boolean or arithmetic) circuit C of size s and width $w = O(\sqrt{\log s})$ with n inputs and m outputs, with communication $O(n + N \cdot (m + s/\sqrt{\log s}))$ and polynomial storage.*

Alternatively, we get a protocol with communication $O(s/\log s)$ for constant-width circuit (which corresponds to the complexity class SC_0). This can again be generalized to the stronger function-independent correlated randomness model. In the next section, we proceed with the description of our protocol. We first focus on the case of layered boolean circuits, and then discuss our extension to the case of arithmetic circuits.

3.2 Core Lemma

In this section, we state and prove the core lemma which underlies our results.

Definition 7 (Local Function). *A Function $g : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ is* c-local *(for some integer $c \leq n$) if on any input $x \in \mathbb{F}_2^n$, any output bit of $g(x)$ depends on at most c bits from x.*

Lemma 8 (Core Lemma). *For any c-local function $g : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, there is an information-theoretic semi-honest N-party secure computation protocol (with dishonest majority) in the correlated randomness model for computing secret shares of g with total online communication $N \cdot n$ bits, and correlated randomness $m \cdot 2^c + n$ bits per party.*

Before proving Lemma 8, it is instructive to compare its guarantees to the protocol obtained by applying directly the one-time truth-table protocol of [IKM+13] to the N-party functionality computing secret shares of g. Applying the OTTT protocol to the N-party functionality which sums its entries (over $\mathbb{F}_2$) before evaluating g, we get a protocol with total communication $N \cdot n$ and correlated randomness $m \cdot 2^{N \cdot n}$. However, it is straightforward to improve this protocol, by applying the OTTT protocol to the 1-party functionality g, and letting the trusted dealer distribute random shares of the shift r to all parties in the preprocessing phase: in the online phase, each party broadcasts his share of the input x, masked with his share of the shift r; this allows all parties to reconstruct $x + r$. With this modification, the parties need only to store a share of the one-dimensional truth-table of g, of size $m \cdot 2^n$.

Therefore, Lemma 8 can be seen as an generalization of the result of [IKM+13], which shifts the exponential cost of the correlated randomness from the input size to the locality parameter of the function. In the most general case, when $c = n$, we recover the result of [IKM+13] (for the special case of the secret sharing functionalities, and up to an additive factor n); when $c < n$, however, this leads to a protocol which uses a smaller amount of correlated randomness.

Proof. Let $g : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ be a c-local function. Without loss of generality, we assume that each output bit of g depends on exactly c input bits. For $j = 1$ to m, we denote by $S_j \subset [n]$ the size-c subset of the bits of the input on which the j'th output bit depends. We denote by $g_j \leftarrow \mathsf{restrict}(g, j)$

the following function: $g_j : \mathbb{F}_2^c \mapsto \mathbb{F}_2$ is the function which, for any $x \in \mathbb{F}_2^n$, computes the j'th output bit of $g(x)$ when given the appropriate subset $x[S_j]$ of the bits of x as input.

We describe on Figure 2 the protocol Π_{local}, which allows N parties holding shares of an input x to securely compute (in the semi-honest model, with correlated randomness) shares of $g(x)$, for some c-local function g. Below, we prove that Π_{local} satisfies all the properties of Lemma 8. It follows immediately by inspection that the total communication of Π_{local} is $N \cdot n$ bits, and that the amount of preprocessing material stored by each party is $m \cdot 2^c + n$. We now turn our attention to correctness and security.

Protocol Π_{local}

Functionality:

- *Public parameters:* a c-local function $g : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, and the m size-c subsets $S_j \subset [n]$ of the bits of the input on which the j'th output bit of g depends.
- *Input:* the parties $(P_1, \cdots, P_N)$ hold random shares $(x_1, \cdots, x_N)$ of an input x over $\mathbb{F}_2^n$;
- *Output:* the parties output uniformly random shares of $g(x)$.

$\Pi_{\mathsf{local}}.\mathsf{Preprocessing}(g)$:

1. Sample $(r_1, \cdots, r_N) \xleftarrow{\$} \mathbb{F}_2^n \times \cdots \times \mathbb{F}_2^n$. Set $r \leftarrow \sum_{i=1}^N r_i$.
2. For $j = 1$ to m, let $g_j \leftarrow \mathsf{restrict}(g, j)$.
3. Let M_j denote the truth-table of g_j permuted with the shift $r[S_j]$, *i.e.*, for any $y \in \mathbb{F}_2^c$, $M_j|_{y+r[S_j]} = g_j(y)$. Note that M_j is of size 2^c.
4. Let $(M_j^i)_{i \leq N, j \leq m}$ be random (N-out-of-N) secret sharings of the M_j. Output $(r_i, (M_j^i)_{j \leq m})$ to each party P_i for $i = 1$ to N.

$\Pi_{\mathsf{local}}.\mathsf{Protocol}(g, \boldsymbol{x})$:

1. Each party P_i with share x_i broadcasts $u_i \leftarrow x_i + r_i$. Let $u \leftarrow \sum_{i=1}^N u_i$.
2. *Output:* each party P_i outputs, for $j = 1$ to m, $z_{i,j} \leftarrow M_j^i|_{u[S_j]}$.

Fig. 2. Protocol Π_{local} for securely computing secret shares of a function g between N-party, with semi-honest and information-theoretic security in the correlated randomness model.

Claim. The protocol Π_{local} is correct.

Proof: for any $j \in [m]$,

$$\begin{aligned}
\sum_{i=1}^N z_{i,j} &= \sum_{i=1}^N M_j^i|_{u[S_j]} \\
&= M_j|_{u[S_j]} \text{ by definition of the } M_j^i \\
&= M_j|_{\sum_i x_i[S_j] + r_i[S_j]} \text{ by definition of } u \\
&= M_j|_{x[S_j] + r[S_j]} \\
&= g_j(x[S_j]) \text{ by definition of } M_j \\
&= g(x)[j] \text{ by definition of } g_j.
\end{aligned}$$

We now turn our attention to security. We represent on Figure 3 the ideal secret-sharing functionality for g. Note that the functionality explicitely allows the adversary to choose the output of the corrupted parties; this is a standard (and minor) technicality of protocols whose output is secret shared between the parties. An alternative is to let the functionality pick the output of all parties at random; however, to realize this functionality, we would need to add a (simple) resharing step at the end of the protocol Π_{local}, which would add unnecessary communication to the protocol.

Claim. The protocol Π_{local} implements the ideal functionality $\mathcal{F}_{\mathsf{local}}$ with perfect security against a semi-honest corruption of a majority of the parties.

Proof: let $H \subset [N]$ denote the subset of honest parties, and let $C \leftarrow [N] \setminus H$ denote the subset of (passively) corrupted parties; the simulator Sim first sends $(\mathsf{corrupt}, C)$ to $\mathcal{F}_{\mathsf{local}}$ on behalf of the

Ideal Functionality $\mathcal{F}_{\mathsf{local}}$

The functionality is parametrized with the description of a function $g : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, and the identities of and adversary $\mathscr{A}$ and N parties $P_1, \cdots, P_N$. The functionality aborts if it receives any incorrectly formatted message.

1. On input a message $(\mathsf{corrupt}, C)$ with $C \subsetneq [N]$ from $\mathscr{A}$, set $H \leftarrow [N] \setminus C$ and store (H, C).
2. On input a message (input, x_i) from each party P_i for $i \in [N]$, store $z \leftarrow g(\sum_{i \in N} x_i)$ and send ready to $\mathscr{A}$.
3. On input a message $(\mathsf{set\text{-}output}, (z_i)_{i \in C})$ from $\mathscr{A}$, pick $|H|$ uniformly random values $(z_i)_{i \in H} \in (\mathbb{F}_2^n)^{|H|}$ under the constraint $\sum_{i \in H} z_i = z - \sum_{i \in C} z_i$.
4. On input a message (send, R) from $\mathscr{A}$ with $R \in [N]$, send z_i to each party P_i with $i \in R$, and $\perp$ to all other parties, then terminate.

Fig. 3. Ideal Functionality $\mathcal{F}_{\mathsf{local}}$ for the secure computation of secret shares of $g(x)$ on an input $x \in \mathbb{F}_2^n$ shared between N parties.

ideal aversary $\mathscr{A}$. Sim simulates the preprocessing phase by distributing uniformly random coins $(r_i, (M_j^i)_{j \leq m})_{i \in C}$ to all corrupted parties. In the online phase, the simulator picks random u_i in $\mathbb{F}_2^n$ for every $i \in H$, and broascasts them on behalf of the honest parties. When he receives $(u_i)_{i \in C}$, he computes for each $i \in C$ $x_i \leftarrow u_i - r_i$, and $z_i \leftarrow (M_j^i|_{u[S_j]})_{j \leq m} \in \mathbb{F}_2^m$. He sends (input, x_i) on behalf of each corrupted party P_i to the ideal functionality $\mathcal{F}_{\mathsf{local}}$, and wait until he receives ready from $\mathcal{F}_{\mathsf{local}}$. Then, he sends $(\mathsf{set\text{-}output}, (z_i)_{i \in C})$ and (send, R) on behalf of $\mathscr{A}$ to $\mathcal{F}_{\mathsf{local}}$, where R is the set of parties that can obtain the output (which Sim can obtain by observing which corrupted parties aborted early). It is immediate to see that the view of the environment (which consists of the preprocessing material, the u_i, and the outputs of the parties) in the ideal world with Sim is perfecty distributed as its view in the real world. This concludes the proof of the core lemma.

4 A Sublinear Protocol for Layered Circuits

In this section, we prove Theorem 1, by exhibiting a generic secure multiparty computation protocol in the correlated randomness model against passive corruption of a majority of the parties, for any layered boolean circuit, with sublinear communication in the circuit size s. Informally, the construction proceeds by breaking the layered circuit into chunks, each chunk containing $k = k(s)$ consecutive layers, for some function k. The parties will evaluate the circuit by computing shares of the values carried by the wires leaving a chunk, given as input shares of the values carried by the wires entering the chunk. As a chunk contains k layers and the directed graph of the circuit has indegree 2, this task corresponds to the secure evaluation of (shares of) a 2^k-local function, with (approximately) w inputs and w outputs (where w is the width of the circuit). By the core lemma (Lemma 8), this can be done with communication $O(w)$ and using $O(w \cdot 2^{2^k})$ bits of correlated randomness per party. After d/k chunk evaluations (d is the depth of the circuit), the parties end up with shares of the values the output wires, which they can broadcast to reconstruct the output. The total communication involved is $O(dw/k) = O(s/k)$, with $O(2^{2^k} \cdot s/k)$ bits of correlated randomness per party.

4.1 Construction

Let C be a layered boolean circuit with n inputs and m outputs, of size s and depth $d = d(n)$, with layers $(L_1, \cdots, L_d)$. For $i = 1$ to d, we let w_i denote the width of the layer L_i. We fix an arbitrary ordering of the nodes.

Let k be an integer. We divide C into $d' = \lceil d/k \rceil$ chunks $(\mathsf{ch}_i)_{i \leq d'}$, each chunk containing k consecutive layers (the last chunk contains less layers is k does not divide d). Let $t \in [k]$ be chosen so that the sum of the widths of the t'th layer of each chunk is bounded by $\lceil s/k \rceil$ (such a t necessarily exists, otherwise, we would get a contradiction: $s = \sum_{i=1}^{d} |L_i| = \sum_{i=1}^{k} (\sum_{j=1}^{d/k} |L_{jk+i}|) >$

$\sum_{i=1}^{k} \lceil s/k \rceil \geq s$). For $i = 1$ to d', we denote t_i the index of the t'th layer in ch_i; it holds that $\sum_{i=1}^{d'} w_{t_i} \leq \lceil s/k \rceil$.

For $i = 1$ to d', we let m_i denote the number of output nodes between the layers $L_{t_{i-1}}$ and L_{t_i} ($\sum_i m_i = m$). For any $i \leq d'$, and $j \leq w_{t_i} + m_i$, we denote $\mathsf{n}_{i,j}$ the j'th node of the layer $L_{t_i} \in \mathsf{ch}_i$ if $j \leq w$, and the $(j-w)$'th output node between the layers $L_{t_{i-1}}$ and L_{t_i} otherwise. We associate two sets to each $\mathsf{n}_{i,j}$: we let $A_{i,j}$ denote the set of ancestors of $\mathsf{n}_{i,j}$ which belong to $L_{t_{i-1}}$ ($A_{1,j}$ is empty for all $j \leq w_{t_1} + m_1$), and we let $I_{i,j}$ denote the set of input nodes between the layers $L_{t_{i-1}}$ and L_{t_i} which are ancestors of $\mathsf{n}_{i,j}$. We let $\alpha_{i,j}$ (resp. $\iota_{i,j}$) denote the size of the set $A_{i,j}$ (resp. $I_{i,j}$). We illustrate this construction on Figure 4. Observe that C has indegree 2, which implies that any node $\mathsf{n}_{i,j}$ of the t'th layer of a chunk can have at most 2^k ancestors in the t'th layer of the previous chunk, hence $\alpha_{i,j} + \iota_{i,j} \leq 2^k$.

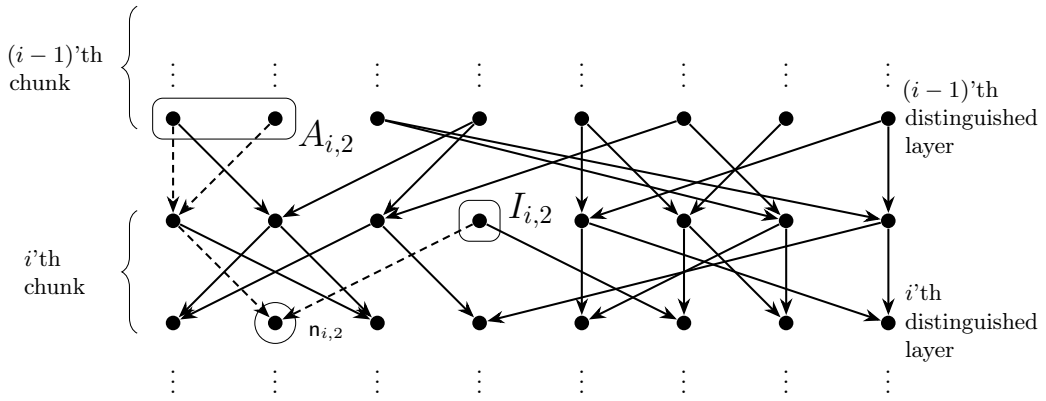

Fig. 4. Illustration of the construction of the sets $(A_{i,j}, I_{i,j})$ for a node $\mathsf{n}_{i,j}$ on a layered directed acyclic graph. The index j is taken equal to 2 on this figure. The dashed edges denote the paths of the graph that end at $\mathsf{n}_{i,2}$.

Our protocol proceeds by evaluating the circuit C on an input $\boldsymbol{x}$ (seen as a size-N vector $(x_1, \cdots, x_N)$ over $\{0,1\}^{n/N}$, where x_j is the input of the party P_j) in a chunk-by-chunk fashion. We say that the parties *evaluate a chunk* i when they compute (shares of) all the values associated to the nodes of the layer L_{t_i}, as well as (shares of) all the values associated to the output nodes between the layers $L_{t_{i-1}}$ and L_{t_i}. Each chunk will be evaluated during a round. We will denote by $y_{i,\ell}$ the bitstring of the shares of the values on L_{t_i} computed by the party P_ℓ in the i'th round, and $y_i = \bigoplus_{\ell=1}^{N} y_{i,\ell}$ the reconstructed value. Similarly, we denote by $z_{i,\ell}$ the bitstring of the shares of the values on the output wires between $L_{t_{i-1}}$ and L_{t_i} computed by the party P_ℓ in the i'th round, and $z_i = \bigoplus_{\ell=1}^{N} z_{i,\ell}$ the reconstructed output string. For simplicity, for any $\ell \leq N$, we denote by $y_{0,\ell}$ an arbitrary dummy string (this is just to simplify the description of the protocol; as the $A_{1,j}$ are empty, these strings will not have any effect on the protocol anyway).

For any $i \leq d'$ and $j \leq w_{t_i} + m_i$, we let $f_{i,j}$ denote the following function: on input the substring $\boldsymbol{x}[I_{i,j}]$ of the input string $\boldsymbol{x}$, and the bitstring $y_{i-1}[A_{i,j}]$ (whose bits form a substring of the values in $L_{t_{i-1}}$), $f_{i,j}$ outputs the value associated to the node $\mathsf{n}_{i,j}$. We let $\delta_i \leftarrow w_{t_i} + m_i$ denote the number of functions $f_{i,j}$ for a fixed i. Finally, we denote by $f_i : \mathbb{F}_2^{w_{t_i}+n} \mapsto \mathbb{F}_2^{\delta_i}$ the following function: on input the string y_{i-1} associated to the distinguished layer of the $(i-1)$'th chunk and the input string $\boldsymbol{x}$, f_i outputs $(f_{i,j}(\boldsymbol{x}[I_{i,j}], y_{i-1}[A_{i,j}]))_{j \leq \delta_i} = (y_i, z_i)$. Observe that, by construction, f_i is a 2^k-local function (the j'th output bit of f_i depends on $\alpha_{i,j} + \iota_{i,j} \leq 2^k$ input bits). The full protocol is represented on Figure 5.

4.2 Proof of Theorem 1

We now argue that the protocol Π_{sub} satisfies all the properties outlined in Theorem 1.

Protocol Π_{sub}

Functionality:

- Public parameters: a layered boolean circuit C of size s and depth d, with n input gates and m output gates, and an integer k.
- The parties $(P_1, \cdots, P_N)$ hold respective inputs $\boldsymbol{x} = (x_1, \cdots, x_N)$ of length n/N (we assume inputs of equal length for simplicity, but the protocol can be adapted to inputs of different lengths in a straightfoward way);
- Output: all the parties learn $C(\boldsymbol{x})$.

$\Pi_{\mathsf{sub}}.\mathsf{Preprocessing}(C)$: for $i = 1$ to $d' = \lceil d/k \rceil$, execute $\Pi_{\mathsf{local}}.\mathsf{Preprocessing}(f_i)$.

$\Pi_{\mathsf{sub}}.\mathsf{Protocol}(C, \boldsymbol{x})$:

- For $i = 1$ to d', all parties execute $\Pi_{\mathsf{local}}.\mathsf{Protocol}(f_i, (y_{i-1}, \boldsymbol{x}))$ (using their shares of y_{i-1} and $\boldsymbol{x}$; note that $y_0 =$ by definition). Each party P_ℓ gets as output $(y_{i,\ell}, z_{i,\ell})$.
- Output: all the parties broadcast the $z_{i,\ell}$ for $i = 1$ to d'. All the parties reconstruct the output $z = (\sum_\ell z_{i,\ell})_{i \leq d'}$.

Fig. 5. Protocol Π_{sub} for evaluating a layered boolean circuit C of size s and depth d, with n input gates and m output gates, in the correlated randomness model against passive corruption of up to $N - 1$ parties.

Correctness. It follows immediately by inspection: by the correctness of Π_{local}, the values $y_{i,\ell}$ computed by the parties form shares of the outputs of the functions $f_{i,j}$ evaluated on the ancestors (in $L_{t_{i-1}}$) of the nodes of layer L_{t_i} (and the ancestors in $L_{t_{i-1}}$ of the output nodes between the layers $L_{t_{i_1}}$ and L_{t_i}), as well as on the input nodes between the layers $L_{t_{i_1}}$ and L_{t_i}. By definitions, those values are exactly the values associated to the output nodes between the layers $L_{t_{i_1}}$ and L_{t_i} and the nodes in the layer L_{t_i}. From there, it immediately follows that the reconstructed outputs $(z_1, \cdots, z_m)$ are correct.

Security. We prove that the protocol Π_{sub} is perfectly secure against an adversary passively corrupting a majority of the parties. The ideal functionality $\mathcal{F}_{\mathsf{sub}}$ that Π_{sub} must realize is straightforward; it is represented on Figure 6. The simulator Sim simply simulates Π_{sub} in the $\mathcal{F}_{\mathsf{local}}$ hybrid model, relying on the simulator for Π_{local} to interface with the real protocol. As Π_{sub} is a simple sequential composition of executions of Π_{local}, security follows immediately.

Ideal Functionality $\mathcal{F}_{\mathsf{sub}}$

The functionality is parametrized with the description of a circuit $C : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, and the identities of and adversary $\mathscr{A}$ and N parties $P_1, \cdots, P_N$. The functionality aborts if it receives any incorrectly formatted message.

1. On input a message $(\mathsf{corrupt}, C)$ with $C \subsetneq [N]$ from $\mathscr{A}$, set $H \leftarrow [N] \setminus C$ and store (H, C).
2. On input a message (input, x_ℓ) from each party P_ℓ for $\ell \in [N]$, store $z \leftarrow C(x_1, \cdots, x_\ell)$ and send ready to $\mathscr{A}$.
3. On input a message (send, R) from $\mathscr{A}$ with $R \in [N]$, send z to all parties P_ℓ with $\ell \in R$, and $\perp$ to all other parties, then terminate.

Fig. 6. Ideal Functionality $\mathcal{F}_{\mathsf{sub}}$ for the secure computation of secret shares of $g(x)$ on an input $x \in \mathbb{F}_2^n$ shared between N parties.

Complexity. We now analyze the communication, storage, computation, and interaction of the protocol Π_{sub}. We first outline a straightforward optimization: observe that each execution of Π_{local} to evaluate (shares of) the output of one of the f_i operates, in particular, on the input $\boldsymbol{x}$ (whose length is n). Instead of using independent executions of Π_{local}, where the input vector $\boldsymbol{x}$ ends up being re-shared between the parties for each execution, the parties can share it only once in an

"input sharing step", before the execution of the first instance of Π_{local}, and reuse these shares in each execution. With this optimization, the parties exchange n bits in the input sharing step, and $N \cdot (\delta_i)$ bits during the i'th round of the circuit evaluation step, for $i = 1$ to $d' = \lceil d/k \rceil$. Therefore, the total number of bits exchanged is

$$n + N \cdot \sum_{i=1}^{d'} w_{t_{i-1}} + m_{i-1} \leq n + N \cdot (m + \lceil s/k \rceil)$$

(note that the additive factor n would be $n \cdot d'$ without the simple optimization outlined above). The amount of correlated randomness stored by each party can be upper bounded by

$$n/N + \sum_{i=1}^{d'} \sum_{j=1}^{\delta_i} 2^{\alpha_{i,j}+\iota_{i,j}} \leq n/N + \sum_{i=1}^{d'} \sum_{j=1}^{\delta_i} 2^{2^k} \leq n/N + (m + \lceil s/k \rceil) \cdot 2^{2^k},$$

where the first inequality comes from the fact that any node $\mathsf{n}_{i,j}$ of the t'th layer of a chunk can have at most 2^k ancestors in the t'th layer of the previous chunk, which leads to the claimed total storage. Eventually, the round complexity of the protocol is $d' + 1 = O(s/k)$, and the computation performed by each party essentially boils down to performing $m + \lceil s/k \rceil$ searches in lookup tables of size bounded by 2^{2^k}, which takes time $(m + \lceil s/k \rceil) \cdot 2^k$.

4.3 Extension to Layered Arithmetic Circuits

So far, our protocol does not readily extend to arithmetic circuits over (exponentially large) finite fields. The main obstacle toward getting an arithmetic analogue of the protocol Π_{sub} lies in the generalization of the core lemma to the arithmetic setting: our proof of Lemma 8 relies on the fact that we can use the OTTT protocol of [IKM+13] to evaluate functions with a "small enough" truth-table. While in the boolean case, any functionality with c input bits has a truth table of size 2^c, this is not true anymore for arithmetic functionalities over large fields, where even single-input functions have truth table of exponential size. In addition, the standard conversion of arithmetic circuits into boolean circuits would blow up the size too much: any size-s arithmetic circuit can be securely evaluated (in the correlated randomness model) with communication $O(s)$ (counting the number of field elements), but the conversion to a boolean circuit will in general blow up the circuit size by a $\log s$ factor, while our protocol only saves a factor $\log \log s$, and does therefore not lead to a sublinear communication protocol for arithmetic circuits.

Nevertheless, we show that our protocol can be extended to the arithmetic setting, by exhibiting a natural analogue of the OTTT protocol, tailored to arithmetic functions. Our starting point is the recent work of [LVW17], on conditional disclosure of secret and private simultaneous message (PSM) protocols. The authors of [LVW17] build an elegant PSM protocol for multivariate polynomial evaluation. The protocol has the following features: Alice holds an n-variate polynomial P of degree deg, Bob holds a vector of input $\boldsymbol{x} \in \mathbb{F}^n$, and both parties share a common random string. They send a single simultaneous message to a third player, Charlie, with optimal communication (Alice's message has size $O(\binom{n+\mathsf{deg}}{\mathsf{deg}})$, Bob's message has size $O(n)$). This allows Charlie to learn $P(\boldsymbol{x})$, and nothing more. The protocol works as follows:

- The shared randomness is $\boldsymbol{r} \in \mathbb{F}^n$ and a random n-variate degree-deg polynomial R.
- Alice sends $(\boldsymbol{x}', u) \leftarrow (\boldsymbol{x} + \boldsymbol{r}, R(\boldsymbol{x} + \boldsymbol{r}))$.
- Bob sends the polynomial $Q(\boldsymbol{X}) = P(\boldsymbol{X} - \boldsymbol{r}) + R(\boldsymbol{X})$.
- Charlie outputs $Q(\boldsymbol{x}') - u$.

The correctness follows immediately by inspection, and security follows by the argument of [LVW17, Section 5.1]. The above PSM can be readily converted into a 2-player arithmetic analogue of the OTTT protocol, which relies on a multivariate polynomial representation of an arithmetic circuit (instead of a truth table representation). We represent on Figure 7 a variant of the protocol Π_{local}, tailored to the arithmetic setting (over an arbitrary field $\mathbb{F}$). Each party sends n field elements (which is essentially optimal), and stores $O(\binom{n+\mathsf{deg}}{\mathsf{deg}})$ field elements.

Using the above protocol, we immediately get a generalization of Lemma 8:

Protocol POLY

Functionality:

- Public parameters: an arithmetic function $f : \mathbb{F}^n \mapsto \mathbb{F}^m$ of depth k over a finite field $\mathbb{F}$, and the m size-2^k subsets $S_j \in [n]$ of the coordinates of the input on which the j'th coordinate of the output of f depends.
- The parties $(P_1, \cdots, P_N)$ hold additive shares $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)$ of an input $\boldsymbol{x} \in \mathbb{F}^n$;
- Output: the parties output uniformly random shares of $f(\boldsymbol{x})$.

Preprocessing :

1. Sample $(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N) \xleftarrow{\$} \mathbb{F}^n \times \cdots \times \mathbb{F}^n$. Set $\boldsymbol{r} \leftarrow \sum_{i=1}^N \boldsymbol{r}_i$.
2. For $j = 1$ to m, let $f_j \leftarrow \mathsf{restrict}(f, j)$.
3. For $j = 1$ to m, Let $P_j(\boldsymbol{X})$ denote the normal-form of f_j, seen as a 2^k-variate polynomial of degree 2^k over $\mathbb{F}$.
4. For $j = 1$ to m, let $Q_j(\boldsymbol{X}) \leftarrow P(\boldsymbol{X} - \boldsymbol{r}[S_j])$ denote the polynomial P shifted with $\boldsymbol{r}[S_j]$.
5. For $j = 1$ to m, sample $N - 1$ uniformly random degree-2^k n-variate polynomials $(R_j^i(\boldsymbol{X}))_{i \leq N-1}$, and set $R_j^N(\boldsymbol{X}) \leftarrow Q_j(\boldsymbol{X}) + \sum_{i=1}^{N-1} R_j^i(\boldsymbol{X})$. Output $(\boldsymbol{r}_i, (R_j^i(\boldsymbol{X}))_{j \leq m})$ to each party P_i.

Protocol($\boldsymbol{x}$) :

1. Each party P_i with share $\boldsymbol{x}_i$ broadcasts $\boldsymbol{u}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$. Let $\boldsymbol{u} \leftarrow \sum_i \boldsymbol{u}_i$.
2. Each party P_i outputs, for $j = 1$ to m, $z_{i,j} \leftarrow R_j^i(\boldsymbol{u}[S_j])$.

Fig. 7. Protocol POLY for evaluating an arithmetic function f over a finite field $\mathbb{F}$ in the correlated randomness model, against a passively corrupted majority

Lemma 9. *For any depth-k arithmetic circuit $f : \mathbb{F}^n \mapsto \mathbb{F}^m$, there is an information-theoretic semi-honest N-party secure computation protocol (with dishonest majority) in the correlated randomness model for computing secret shares of f with total online communication $N \cdot n$ elements of $\mathbb{F}$, and correlated randomness $m \cdot \binom{2^{k+1}}{2^k} + n \approx m \cdot 2^{2^{k+1}}/\sqrt{\pi 2^k} + n$ elements of $\mathbb{F}$ per party.*

Therefore, we get polynomial storage (in s) by setting $k \leftarrow \log\log s$, as before. This leads to a protocol for arithmetic circuits of size s, with n inputs and m outputs, with polynomial storage and total communication $O(n + N \cdot (m + s/\log\log s))$.

Reducing Storage in TinyTable. While the idea of using a multivariate-polynomial representation instead of a truth-table representation seems relatively natural and is the key to extend the construction to the arithmetic setting, it was not explicitly observed before. Somewhat surprisingly, we observe that even in the original (boolean) setting of the TinyTable paper [DNNR17] (which uses truth-table representation at the gate level, for two-party evaluation of AND gates in boolean circuits), replacing truth-tables by multivariate polynomials in normal form improves the construction: it reduces the storage of the parties by 25%. We sketch this observation below. The TinyTable protocol maintains the following invariant: the parties know masked representation of all inputs to some gate of the circuit, and will compute a masked representation of the output. Typically, for a two-input AND gate, both parties will know $u = x + r$ and $v = y + s$, where x, y are the inputs to the gate, and r, s are random masks. In addition, the parties hold random shares of the truth-table of the function

$$F_{r,s,t} : (u, v) \to (u - r) \cdot (v - s) + t,$$

where t is another fresh random coin. Observe that $F_{r,s,t}(x + r, y + s) = x \cdot y + t$, maintaining the appropriate invariant. In the TinyTable paper, each party knows a share F_0, F_1 of the truth-table of a function of this form, for each AND gate of the circuit, and the output is computed by broadcasting $F_0(u, v), F_1(u, v)$ and reconstructing $w = F_0(u, v) \oplus F_1(u, v)$. This represent a total storage of $4s$ bits per party (and $2s$ bits of communication), where s is the number of AND gates of the circuit.

Now, if we view instead $F_{r,s,t}$ as a degree-2 polynomial in two variables, we have $F_{r,s,t} = uv + \alpha u + \beta v + \gamma$ for some appropriate $(\alpha, \beta, \gamma) = (-s, -r, t + rs)$. Observe that to randomly share $F_{r,s,t}$ *viewed as a multivariate polynomial*, it suffices to share additively each of its coefficients randomly; furthermore, the leading coefficient of $F_{r,s,t}$ is always one. Hence, we can improve the TinyTable AND gate evaluation protocol as follows: the parties receive shares $(\alpha_0, \beta_0, \gamma_0)$ and

$(\alpha_1, \beta_1, \gamma_1)$ of (α, β, γ) (this is identical to giving a random degree-one bivariate polynomial R to one party, and $F_{r,s,t} + R$ to the other party; note that R needs only having degree one since it needs not hide the leading coefficient of $F_{r,s,t}$, which is 1). Given public values $u = x + r$ and $v = y + s$, the parties exchange $w_0 = \alpha_0 u + \beta_0 v + \gamma_0$ and $w_1 = \alpha_1 u + \beta_1 v + \gamma_1$, and publicly reconstruct $w = uv + w_0 + w_1$. The communication and computation are essentially the same as in [DNNR17], but the parties must now only store three bits per AND gate, hence $3s$ bits in total, reducing the amount of storage required by the protocol by 25%.

4.4 Further Extensions

We sketch in this section how to extend our protocol to the case of function-independent correlated randomness, and to the case of tall-and-narrow circuits.

Function-Independent Preprocessing. We introduce below a variant of the core lemma, tailored to function-independent preprocessing. Theorem 3 follows immediately from this variant.

Lemma 10. *For any c-local function $g : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, there is an information-theoretic semi-honest N-party secure computation protocol (with dishonest majority) in the* function-independent *correlated randomness model for computing secret shares of g with total online communication $N \cdot n$ bits, and correlated randomness $m \cdot 2^{c+2^c} + n$ bits per party.*

Proof. To prove Lemma 10, we modify Π_{local} as follows: instead of computing shares of the truth table M_j of g_j (which is of size 2^c) permuted with the shift $r[S_j]$, we consider the list $(M_{j,q})_{q \leq 2^{2^c}}$ of all possible truth tables, corresponding to a lexicographic ordering of all possible functions g_j, each table being shifted with the same $r[S_j]$. Each party P_i receives $(r_i, (M^i_{j,q})_q)$, which amounts to $n + 2^c \cdot 2^{2^c}$ bits of correlated randomness. In the online protocol $\Pi_{\mathsf{local}}.\mathsf{Protocol}$, when the functions g_j are revealed, the parties locally drop all unnecessary shares of shifted truth tables, keeping only the one corresponding to g_j. The security analysis immediately follows from the analysis of Π_{local}.

Tall-and-Narrow Circuits. For tall-and-narrow circuits, whose width w is small, the proof follows by observing that in this situation the bound on the size of the sets $A_{i,j}$ and $I_{i,j}$ can be refined to $|A_{i,j}|+|I_{i,j}| \leq w \cdot k$, hence the $f_{i,j}$ have truth tables of size bounded by $2^{w \cdot k}$. Theorem 5 follows immediately.

5 Malicious Setting

In the two-party case, combining our passively secure protocol Π_{sub} with the techniques of [DNNR17] directly implies the existence of a (statistical) unconditionally secure two-party protocol secure against malicious adversaries, with communication $O(n + m + \frac{s}{\log \log s} + \kappa)$ for a layered boolean circuit of size s, where κ is a statistical security parameter. Indeed, the protocol of [DNNR17] has a structure similar to our protocol: it decomposes the circuit into tables, and distributes scrambled version of these tables to the parties in the preprocessing phase. Each gate of the circuit is evaluated using the OTTT protocol to obliviously select the output of the gate from its corresponding scrambled truth-table.

To enhance this protocol to security against malicious adversaries, [DNNR17] uses a simple and natural information-theoretic authentication procedure. Namely, for each entry $b \in \{0, 1\}$ of a table given to the first party (let us call it A), the trusted dealer additionaly generates two κ-bit string (x_0, x_1), hands x_b to A, and (x_0, x_1) to the other player B. This way, when A must send the entry b of the table to B, she can authenticate b by sending it along with x_b. B then retrieves the corresponding value x_b and checks that A honestly opened b; if A is dishonest, she will be caught with probability $1 - 2^{-\kappa}$. Note that the only local computation performed by the parties are searches through lookup table, hence authenticating each entry this way suffices to guarantee security of the entire protocol.

However, directly applying this approach would require transmitting κ bits per output of a table, which would increase the total communication by a factor of κ. To avoid this overhead, the authors

of [DNNR17] observe that it is not necessary to explicitely authenticate each entry sent by a party. Instead, each time A reveals an entry b corresponding to some authentication string x_b, she locally updates a "global MAC key" $\Delta_A \leftarrow \Delta_A \oplus x_b$, where Δ_A is set to 0 at the start of the protocol. Simultaneously, when he receives an entry b from A, B retrieves the pair (x_0, x_1) corresponding to this entry, and locally updates $\Gamma_B \leftarrow \Gamma_B \oplus x_b$, where Γ_B is set to 0 at the start of the protocol. The parties proceed symmetrically, with Δ_B, Γ_A, when B sends an entry to A. At the end of the protocol, A reveals Δ_A and B reveals Δ_B. If $\Delta_B \neq \Gamma_A$, A aborts the protocol; B does the same if $\Delta_A \neq \Gamma_B$. If both checks passed, the parties reconstruct the output. The analysis of [DNNR17] shows that this guarantees that no party can cause its opponent to accept an incorrect output, except with probability $2^{-\kappa}$. It increases the amount of preprocessed material by a factor κ,[4] but only adds 2κ bits to the total communication.

For completeness, we provide a full self-contained description of the maliciously-secure two-party version of our protocol on Figure 8. We refer the reader to Theorem 1 of [DNNR17] for a detailed proof of security against malicious adversaries; it is straightforward to adapt the proof to our protocol (we note that, while [DNNR17] focuses on small tables implementing standard two-input boolean gates, [DNNR17, Section 2.3] already observes that this mechanisms can be directly generalized to protocols evaluating larger tables).

Extension to N Parties. While the work of [DNNR17] focused only on (maliciously secure) two-party computation, it was subsequently observed in [KOR+17] that the techniques used in [DNNR17] can be easily generalized to the multiparty setting, for an arbitrary number N of parties. We refer the reader to [KOR+17] for more details; this directly gives:

Theorem 11. *For any N-party functionality f represented by a layered boolean circuit C of size s with n inputs and m outputs, and for any integer k and statistical security parameter κ, there is a κ-secure protocol which realizes f in the preprocessing model against malicious adversaries with adaptive corruption (of up to $N-1$ parties), with communication $n + N \cdot (m + \lceil s/k \rceil + \kappa)$ and correlated randomness $n/N + (3\kappa + 1) \cdot (m + \lceil s/k \rceil) \cdot (2^{2^k} + 1)$ per party.*

6 Open Questions

While our work shows that a large class of circuits of size s can be securely evaluated in the correlated randomness model using $o(s)$ communication, many questions related to the communication of MPC in the correlated randomness model remain open.

Question 1. Can our protocols be extended to arbitrary non-layered circuits?

It is immediate to extend our protocol to any circuit that is layered "by blocks" of depth c, in the sense that no edge crosses more than c consecutive layers, for any $c = o(\log \log s)$. However, generalizing our result to all circuits remains an interesting open question.

Question 2. Can we achieve better sublinearity for unconditional MPC in the correlated randomness model, in general or for specific circuits?

It is known that some specific functions can be evaluated in the correlated randomness model, with stronger sublinearity guarantees than those obtained in this work. In particular, *matrix multiplication* can be computed with communication linear in the size n^2 of the matrices, while the best known algorithm for multiplying matrices of size n requires $O(n^t)$ communication, with $t \approx 2.3$. The work of [BIKO12] also implies the existence of low-communication protocols in the correlated randomness model, for $N \geq 3$ parties, for specific types of constant-depth circuits. It would be interesting to improve the sublinearity of our work, and to characterize the functions for which better sublinearity can be achieved.

[4] A technique to amortize this overhead, using a linear MAC scheme, is described in [DNNR17]; it applies to our setting as well, and allows to remove this factor κ overhead in the storage complexity, but we focus on the more naive approach in this work for simplicity.

Protocol $\Pi^{\mathsf{mal}}_{\mathsf{sub}}$

Functionality:

- Public parameters: a layered boolean circuit C of size s and depth d, with n input gates and m output gates, and an integer k. We let κ denote a statistical security parameter.
- The parties (P_1, P_2) hold respective inputs $\boldsymbol{x} = (x_1, x_2)$ of length $n/2$.

$\Pi^{\mathsf{mal}}_{\mathsf{sub}}.\mathsf{Preprocessing}$:

- Sample $\boldsymbol{\rho} = (\rho_1, \rho_2) \xleftarrow{\$} (\{0,1\}^{n/2})^2$ and for $i = 1$ to d', sample δ_i bits $\boldsymbol{r}_i = (r_{i,1}, \cdots, r_{i,\delta_i})$ such that $r_{i,j}$ is 0 if (i,j) corresponds to an output gate, and random otherwise (looking ahead, the bits $r_{i,j}$ will be used to mask the output value of the function $f_{i,j}$).
- For $i = 1$ to d', for $j = 1$ to δ_i, let $M_{i,j}$ denote the permuted truth table of $f_{i,j}$ with shifts $(\boldsymbol{\rho}[I_{i,j}], \boldsymbol{r}_{i-1}[A_{i,j}])$ and output masked with $r_{i,j}$, *i.e.*:

$$M_{i,j}|_{(\boldsymbol{x}\oplus\boldsymbol{\rho})[I_{i,j}],(y\oplus\boldsymbol{r}_{i-1})[A_{i,j}]} = f_{i,j}(\boldsymbol{x}[I_{i,j}], y[A_{i,j}]) \oplus r_{i,j}.$$

- For $i = 1$ to d', for $j = 1$ to δ_i, sample a random truth-table $R^1_{i,j}$ over $\{0,1\}^{2^{\alpha_{i,j}+\iota_{i,j}}}$, and let $R^2_{i,j} \leftarrow R^1_{i,j} \oplus M_{i,j}$.
- For $\ell = 1, 2$, for $i = 1$ to d', for $j = 1$ to δ_i, for $q = 1$ to $2^{\alpha_{i,j}+\iota_{i,j}}$, we denote $R^\ell_{i,j,q}$ the q'th entries of $R^\ell_{i,j}$. Sample 2 random κ-bit strings $(s^0_{\ell,i,j,q}, s^1_{\ell,i,j,q})$ (looking ahead, these values will allow to authenticate the value $R^\ell_{i,j,q}$). To simplify notations, we denote by $s'_{\ell,i,j,q}$ the value

$$s'_{\ell,i,j,q} \leftarrow s^b_{3-\ell,i,j,q}, \text{ with } b = R^{3-\ell}_{i,j,q}.$$

- For $\ell = 1, 2$, output to P_ℓ

$$\left(\rho_\ell, (R^\ell_{i,j})_{i\le d', j\le\delta_i}, \left(s^0_{\ell,i,j,q}, s^1_{\ell,i,j,q}, s'_{\ell,i,j,q}\right)_{i,j,q}\right).$$

$\Pi^{\mathsf{mal}}_{\mathsf{sub}}.\mathsf{Protocol}(\boldsymbol{x})$:

- Initialization: for $\ell = 1, 2$, P_ℓ sets $\Delta_\ell = \Gamma_\ell = 0^\kappa$.
- Input Sharing: for $\ell = 1, 2$, P_ℓ broadcasts $u_\ell \leftarrow x_\ell \oplus \rho_\ell$. Let $\boldsymbol{u} \leftarrow (u_1, u_2)$. Set v_0 to be an arbitrary dummy string.
- Circuit Evaluation: for $i = 1$ to d',
  - For $\ell = 1, 2$, P_ℓ sets

$$\boldsymbol{v}_{i,\ell} \leftarrow \left(R^\ell_{i,1}|_{\boldsymbol{u}[I_{i,1}],v_{i-1}[A_{i,1}]}, \cdots, R^\ell_{i,\delta_i}|_{\boldsymbol{u}[I_{i,\delta_i}],v_{i-1}[A_{i,\delta_i}]}\right).$$

  - For $\ell = 1, 2$, P_ℓ broadcasts $\boldsymbol{v}_{i,\ell}$; let $\boldsymbol{v}_i \leftarrow \bigoplus_{\ell=1}^N \boldsymbol{v}_{i,\ell}$.
  - For $\ell = 1, 2$, P_ℓ sets $q_{i,j}$ to be the string $(\boldsymbol{u}[I_{i,j}], \boldsymbol{v}_{i-1}[A_{i,j}])$, and sets

$$\Delta_\ell \leftarrow \Delta_\ell \oplus s'_{3-\ell,i,j,q_{i,j}}, \Gamma_\ell \leftarrow \Gamma_\ell \oplus s^{\boldsymbol{v}_{i,3-\ell,j}}_{\ell,i,j,q_{i,j}}.$$

- Verification of all opened bits: for $\ell = 1, 2$, P_ℓ sends Δ_ℓ to $P_{3-\ell}$, and $P_{3-\ell}$ checks that $\Gamma_{3-\ell} = \Delta_\ell$.
- Output: For $\ell = 1, 2$, P_ℓ broadcasts the δ_i-bit string $v_{i,\ell,j}$ for every $i \le d'$ and $j > w_{t_i}$. All the parties reconstruct $\boldsymbol{z} = (z_1, \cdots, z_m) = (\bigoplus_\ell v_{i,\ell,j})_{i\le d', j>w_{t_i}}$.

Fig. 8. Two-party protocol $\Pi^{\mathsf{mal}}_{\mathsf{sub}}$ for evaluating a layered boolean circuit C of size s and depth d, with n input gates and m output gates, in the correlated randomness model against active corruption of one of the parties.

Question 3. Can we achieve sublinear communication and linear storage at the same time?

Our protocols only achieve slightly superlinear storage; in the regime where the $1/\log\log s$ factor would give non-trivial communication savings, this implies that a rather large storage is required. Protocols for specific functions, such as matrix multiplication, achieve both sublinearity and linear storage, but the question remains open for more general functions.

Acknowledgements. We thank Yuval Ishai for helpful comments and pointers, Benny Applebaum for detailed comments regarding the content and the structure of this paper, and Elaine Shi for pointing out an inaccurate account of a previous lower bound on the storage of uncondionally secure computation. Work supported by ERC grant 724307 (project PREP-CRYPTO).

References

AJL+12. G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT 2012*, *LNCS* 7237, pages 483–501. Springer, April 2012.

AKD03. M. J. Atallah, F. Kerschbaum, and W. Du. Secure and private sequence comparisons. In *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, pages 39–44. ACM, 2003.

ALSZ13. G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *ACM CCS 13*, pages 535–548. ACM Press, November 2013.

BCG+17. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, and M. Orrù. Homomorphic secret sharing: Optimizations and applications. In *ACM CCS 17*, pages 2105–2122. ACM Press, 2017.

BCG+18. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, and M. Orrù. Homomorphic secret sharing: Optimizations and applications. Cryptology ePrint Archive, Report 2018/419, 2018. `https://eprint.iacr.org/2018/419`.

BDOZ11. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT 2011*, *LNCS* 6632, pages 169–188. Springer, May 2011.

Bea92. D. Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO'91*, *LNCS* 576, pages 420–432. Springer, August 1992.

Bea95. D. Beaver. Precomputing oblivious transfer. In *CRYPTO'95*, *LNCS* 963, pages 97–109. Springer, August 1995.

Bea97. D. Beaver. Commodity-based cryptography (extended abstract). In *29th ACM STOC*, pages 446–455. ACM Press, May 1997.

BFKR91. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In *CRYPTO'90*, *LNCS* 537, pages 62–76. Springer, August 1991.

BGI15. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. LNCS, pages 337–367. Springer, 2015.

BGI16. E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO 2016, Part I*, LNCS, pages 509–539. Springer, August 2016.

BGI17. E. Boyle, N. Gilboa, and Y. Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. LNCS, pages 163–193. Springer, 2017.

BI05. O. Barkol and Y. Ishai. Secure computation of constant-depth circuits with applications to database search problems. In *CRYPTO 2005*, *LNCS* 3621, pages 395–411. Springer, August 2005.

BIKK14. A. Beimel, Y. Ishai, R. Kumaresan, and E. Kushilevitz. On the cryptographic complexity of the worst functions. In *TCC 2014*, *LNCS* 8349, pages 317–342. Springer, February 2014.

BIKO12. A. Beimel, Y. Ishai, E. Kushilevitz, and I. Orlov. Share conversion and private information retrieval. In *Computational Complexity (CCC), 2012 IEEE 27th Annual Conference on*, pages 258–268. IEEE, 2012.

BLN+15. S. S. Burra, E. Larraia, J. B. Nielsen, P. S. Nordholt, C. Orlandi, E. Orsini, P. Scholl, and N. P. Smart. High performance multi-party computation for binary circuits based on oblivious transfer. Cryptology ePrint Archive, Report 2015/472, 2015. `http://eprint.iacr.org/2015/472`.

CDv88. D. Chaum, I. Damgård, and J. van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *CRYPTO'87*, *LNCS* 293, pages 87–119. Springer, August 1988.

CKL15. J. H. Cheon, M. Kim, and K. E. Lauter. Homomorphic computation of edit distance. In *FC 2015 Workshops*, LNCS, pages 194–212. Springer, 2015.

DFH12. I. Damgård, S. Faust, and C. Hazay. Secure two-party computation with low communication. In *TCC 2012*, *LNCS* 7194, pages 54–74. Springer, March 2012.

DKS+17. G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, S. Zeitouni, and M. Zohner. Pushing the communication barrier in secure computation using lookup tables. In *Network and Distributed System Security Symposium (NDSS'17). The Internet Society*, 2017.

DLT14. I. Damgård, R. Lauritsen, and T. Toft. An empirical study and some improvements of the MiniMac protocol for secure computation. In *SCN 14*, LNCS, pages 398–415. Springer, 2014.

DNNR17. I. Damgård, J. B. Nielsen, M. Nielsen, and S. Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. LNCS, pages 167–187. Springer, 2017.

DNPR16. I. Damgård, J. B. Nielsen, A. Polychroniadou, and M. Raskin. On the communication required for unconditionally secure multiplication. In *CRYPTO 2016, Part II*, LNCS, pages 459–488. Springer, August 2016.

DPSZ12. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO 2012*, *LNCS* 7417, pages 643–662. Springer, August 2012.

DZ13. I. Damgård and S. Zakarias. Constant-overhead secure computation of Boolean circuits using preprocessing. In *TCC 2013*, *LNCS* 7785, pages 621–641. Springer, March 2013.

DZ16. I. Damgård and R. W. Zakarias. Fast oblivious AES A dedicated application of the MiniMac protocol. In *AFRICACRYPT 16*, LNCS, pages 245–264. Springer, 2016.

FKN94. U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation (extended abstract). In *26th ACM STOC*, pages 554–563. ACM Press, May 1994.

FKOS15. T. K. Frederiksen, M. Keller, E. Orsini, and P. Scholl. A unified approach to MPC with preprocessing using OT. LNCS, pages 711–735. Springer, December 2015.

Gen09. C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

GMW87a. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

GMW87b. O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO'86*, *LNCS* 263, pages 171–185. Springer, August 1987.

HEKM11. Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, pages 331–335, 2011.

IKM+13. Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC 2013*, *LNCS* 7785, pages 600–620. Springer, March 2013.

IPS08. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO 2008*, *LNCS* 5157, pages 572–591. Springer, August 2008.

IPS09. Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In *TCC 2009*, *LNCS* 5444, pages 294–314. Springer, March 2009.

JKS08. S. Jha, L. Kruger, and V. Shmatikov. Towards practical privacy for genomic computation. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 216–230. IEEE, 2008.

JRS17. A. Jain, P. M. R. Rasmussen, and A. Sahai. Threshold fully homomorphic encryption. Cryptology ePrint Archive, Report 2017/257, 2017. `http://eprint.iacr.org/2017/257`.

Kil88. J. Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

KK13. V. Kolesnikov and R. Kumaresan. Improved OT extension for transferring short secrets. In *CRYPTO 2013, Part II*, *LNCS* 8043, pages 54–70. Springer, August 2013.

KOR+17. M. Keller, E. Orsini, D. Rotaru, P. Scholl, E. Soria-Vazquez, and S. Vivek. Faster secure multi-party computation of AES and DES using lookup tables. In *ACNS 17*, LNCS, pages 229–249. Springer, 2017.

KOS16. M. Keller, E. Orsini, and P. Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In *ACM CCS 16*, pages 830–842. ACM Press, 2016.

Lev66. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, pages 707–710, 1966.

LOS14. E. Larraia, E. Orsini, and N. P. Smart. Dishonest majority multi-party computation for binary circuits. In *CRYPTO 2014, Part II*, LNCS, pages 495–512. Springer, August 2014.

LVW17. T. Liu, V. Vaikuntanathan, and H. Wee. Conditional disclosure of secrets via non-linear reconstruction. LNCS, pages 758–790. Springer, 2017.

NN01. M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *33rd ACM STOC*, pages 590–599. ACM Press, July 2001.

NNOB12. J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In *CRYPTO 2012*, *LNCS* 7417, pages 681–700. Springer, August 2012.

SPO$^+$06. D. Szajda, M. Pohl, J. Owen, B. G. Lawson, and V. Richmond. Toward a practical data privacy scheme for a distributed implementation of the smith-waterman genome sequence comparison algorithm. In *NDSS*, 2006.

SW81. T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

Wak68. A. Waksman. A permutation network. *Journal of the ACM (JACM)*, 15(1):159–163, 1968.

Yao86. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.