A Note on Low-Communication Secure Multiparty Computation via Circuit Depth-Reduction

Pierre Charbit[1], Geoffroy Couteau[1], Pierre Meyer[2⋆], and Reza Naserasr[1]

[1] Université Paris Cité, CNRS, IRIF, Paris, FRANCE. {charbit,couteau,reza}@irif.fr
[2] Aarhus University, Aarhus, DENMARK. pierre.meyer@cs.au.dk

Abstract. We consider the graph-theoretic problem of removing (few) nodes from a directed acyclic graph in order to reduce its depth. While this problem is intractable in the general case, we provide a variety of algorithms in the case where the graph is that of a circuit of fan-in (at most) two, and explore applications of these algorithms to secure multiparty computation with low communication. Over the past few years, a paradigm for low-communication secure multiparty computation has found success based on decomposing a circuit into low-depth "chunks". This approach was however previously limited to circuits with a "layered" structure. Our graph-theoretic approach extends this paradigm to *all* circuits. In particular, we obtain the following contributions:

- **Fractionally linear-communication MPC in the correlated randomness model.** We provide an N-party protocol for computing an n-input, m-output $\mathbb{F}$-arithmetic circuit with s internal gates (over any basis of binary gates) with communication complexity $(\frac{2}{3}s+n+m)\cdot N\cdot\log|\mathbb{F}|$, which can be improved to $((1+\epsilon)\cdot\frac{2}{5}s+n+m)\cdot N\cdot\log|\mathbb{F}|$ (at the cost of increasing the computational overhead from a small constant factor to a large one). Previously, comparable protocols either used more than $s\cdot N\cdot\log|\mathbb{F}|$ bits of communication, required super-polynomial computation, were restricted to layered circuits, or tolerated a sub-optimal corruption threshold.

- **Sublinear-Communication MPC.** Assuming the existence of N-party Homomorphic Secret Sharing for logarithmic depth circuits (respectively doubly logarithmic depth circuits), we show there exists sublinear-communication secure N-party computation for *all* $\log^{1+o(1)}$-depth (resp. $(\log\log)^{1+o(1)}$-depth) circuits. Previously, this result was limited to $(\mathcal{O}(\log))$-depth (resp. $(\mathcal{O}(\log\log))$-depth) circuits, or to circuits with a specific structure (*e.g.* layered).

- **The $\binom{N}{1}$-OT complexity of MPC.** We introduce the "$\binom{N}{1}$-*OT complexity of MPC*" of a function f, denoted $C_N(f)$, as the number of oracle calls required to securely compute f in the $\binom{N}{1}$-OT hybrid model. We establish the following upper bound: for every $N\geq 2$, $C_N(f)\leq(1+g(N))\cdot\frac{2|f|}{5}$, where $g(N)$ is an explicit vanishing function.

We also obtain additional contributions to reducing the amount of bootstrapping for fully homomorphic encryption, and to other types of sublinear-communication MPC protocols such as those based on correlated symmetric private information retrieval.

⋆ Most of this work was done while P. Meyer was a PhD student jointly at Reichman University, Herzliya, ISRAEL and Université Paris Cité, CNRS, IRIF, Paris, FRANCE.

Table of Contents

1 Introduction

Secure multiparty computation (MPC) [Yao86, GMW87] enables mutually distrusting parties to jointly compute a function on their private inputs, while revealing nothing beyond the function's output. The seminal protocols of the 1980s [Yao86, GMW87, BGW88, CCD88] all require an amount of communication which scales *linearly* with the computational complexity of the function: given a fan-in two circuit representation of the function, the parties are required to communicate for every (non-linear) gate of the circuit. In this paper, we propose to achieve MPC with less communication than the circuit-size by limiting interaction to only a few key gates, which we identify using a graph-theoretic algorithm. Specifically, the goal is to find a small set of nodes in the underlying graph whose removal yields a low-depth directed acyclic graph (DAG). The idea is to reduce the secure computation of an arbitrary circuit to that of a circuit of logarithmic, doubly-logarithmic, or even constant depth for which, generally speaking, there are known protocols using circuit-independent communication.

This problem is motivated by the fact that, aside from protocols based on fully homomorphic encryption, all known ways to achieve sublinear-communication secure computation are restricted to low-depth circuits, and can only currently be extended to deeper circuit if they have a specific "layered" structure (meaning the circuits' gates are partitioned into ordered layers, and each wire only connects one layer to the next).

1.1 Background

This paper proposes to solve the cryptographic problem of securely computing *any* circuit with low communication by reducing it to a graph-theoretic problem. We recall in section 1.1.1 what is known about low-communication secure "general-purpose" computation (including protocols restricted to layered circuits or low-depth circuits, but excluding special-purpose protocols such as *Private Information Retrieval*), and in section 1.1.2 the literature on the graph-theoretic problem to which we reduce the secure computation task.

1.1.1 Sublinear-communication secure multiparty computation. We start by surveying sublinear-communication secure computation protocols, but restrict ourselves to those using a polynomial amount of computation[3] and tolerating the optimal number of corruptions. Because the amount of computation typically grows exponentially (or even doubly exponentially) with the depth of the computation, most of these protocols only support computation up to a certain depth. Throughout this section, s denotes the size of the circuit, d denotes its depth, N denotes the number of parties, and $\mathbb{F}$ denotes the field over which the circuit's arithmetic is performed.

In the correlated randomness model, $d = \log\log s$. Ishai, Kushilevitz, Meldgaard, Orlandi, and Paskin-Cherniavsky [IKM+13] showed how to securely compute any function with a polynomial-size look-up table using circuit-independent communication (only proportional to input and output size). Couteau [Cou19] extended this approach to any log log-depth (boolean or arithmetic) circuit. In turn, this yields a protocol for securely computing any size-s *layered* circuit using $\mathcal{O}(s/\log\log s \cdot N \cdot \log|\mathbb{F}|)$ bits of communication.

From fully homomorphic encryption (FHE), any d. Gentry [Gen09] provided the first construction of *fully homomorphic encryption* from ideal lattices. This powerful primitive, which was later instantiated under a standard variant of the learning with errors assumption (LWE) [BV11, BGV12], allows for a computation to be performed on encrypted data. FHE was later shown to yield asymptotically optimal-communication secure multiparty computation (in the computational setting) [DFH12, AJL+12].

From homomorphic secret sharing (HSS), $d = \mathcal{O}(\log s)$. Boyle, Gilboa, and Ishai [BGI16a] built two-party *homomorphic secret-sharing* (HSS) for branching programs from the decisional Diffie-Hellman assumption (DDH). This yields two-party protocols for securely computing $\mathcal{O}(\log s)$-depth circuits using circuit-independent communication, or arbitrarily deep layered boolean circuits with communication $\mathcal{O}(s/\log s \cdot N \cdot \log|\mathbb{F}|)$. This template for HSS for branching programs was later instantiated under the decision composite residuosity assumption (DCR) [FGJS17, OSY21, RS21], learning with errors (LWE) with a polynomial noise-to-modulus ratio [BKS19], and assumptions based on class groups of imaginary quadratic fields [ADOS22]. We note that some of the above HSS schemes [BGI16a, FGJS17, BKS19] have a noticeable error in correctness, but this has no bearing on the application to low-communication two-party computation. The DDH-based approach only supports boolean circuits, while the LWE- and DCR-based approaches work in any field.[4]

From homomorphic secret sharing (HSS), $d = \log\log s - \log\log\log s$. There exists HSS supporting circuits of depth $(\log\log s - \log\log\log s)$:[5] in the two-party setting from the quasi-polynomial learning parity with noise assumption (LPN) [CM21], in the four-party setting assuming DCR and constant-depth pseudorandom generators (PRG) [COS+22, BCM23], and in the general multiparty

[3] If we lift this restriction, Beaver, Feigenbaum, Kilian, and Rogoway [BFKR91] showed that any function from $\{0,1\}^n$ to $\{0,1\}$ can be securely computed in the presence of up to t corruptions by $N = \mathcal{O}(t \cdot n/\log n)$ computationally unbounded parties (which the protocol indeed requires to use an exponential amount of computation) using $\mathsf{poly}(n+N)$ communication. In addition, the protocol of [IKM+13], which we state as a result for securely computing functions with a polynomial-size look-up table, can be applied to general functions if the parties are computationally unbounded (and have access to a doubly exponential amount of correlated randomness).

[4] These HSS schemes support bounded-integer computation, which translates to low-communication secure computation over arbitrary fields using hybrid encryption.

[5] Note that this class is somewhat related to logarithmic-depth branching programs.

setting assuming sparse LPN [DIJL23]. In turn, this yields 2-, 4-, or N-party computation of $(\log\log s - \log\log\log s)$-depth circuits with circuit-independent communication, or that of arbitrarily deep layered circuits with communication $\mathcal{O}(s/\log\log s \cdot N \cdot \log|\mathbb{F}|)$. All these approaches support arbitrary fields $\mathbb{F}$. The scheme based on super-polynomial LPN [CM21] is a "single-circuit" scheme (it requires the circuit's topology to be known at input-sharing time) and the scheme based on sparse LPN [DIJL23] has a noticeable error in correctness, but here again these limitations have no bearing on the application to low-communication secure multiparty computation.

Other approaches, $d = \log\log s - \log\log\log s$. Boyle, Couteau, and Meyer [BCM22] introduced *correlated symmetric private information retrieval* (correlated SPIR), and instantiated it under LPN plus any of the following assumptions: DDH, the quadratic residuosity assumption (QR), DCR, or poly-modulus LWE. This primitive yields secure two-party computation of doubly logarithmic-depth[6] boolean circuits using $\mathcal{O}(n+m+\sqrt{s}\cdot\mathsf{poly}(\lambda)\cdot(n+m)^{2/3})$ bits of communication, as well as secure two-party computation of *synchronous*[7] boolean circuits using communication $\mathcal{O}(s/\log\log s + d^{1/3}\cdot s^{2(1+\epsilon)/3})$ (which is $\mathcal{O}(s/\log\log s)$ provided $d = o(s^{1-\epsilon}/\mathsf{poly}(\lambda))$, which is to say the circuit is not too "tall and skinny"). Boyle, Couteau, and Meyer [BCM23] later provided a framework, based on 2- or 4-party HSS and correlated SPIR, to extended these protocols to the 3- or 5-party settings. Finally, Abram, Roy, and Scholl [ARS24] extended this approach to the general multiparty setting by instead only relying on a form of two-party HSS with *succinct share size*, and instantiated it under a DDH-like assumption in class groups, DCR, or LWE with a super-polynomial noise-to-modulus ratio: for general layered boolean circuits with communication $\mathcal{O}(s/\log\log s\cdot N)$, or even $\mathcal{O}(s/\log s\cdot N)$ if the layered boolean circuit is wide enough. All these approaches are restricted to the boolean setting.

1.1.2 DAG depth-reduction.

The problem of reducing a DAG's depth by removing nodes can be expressed as a hitting set problem: given a directed graph $G = (V, E)$, we are looking for a set of vertices which "hits" (*i.e.* contains a vertex from) each k-path in G. Ultimately, we will present a family of secure computation protocols whose communication complexity scales with the size of such a hitting set in a well-chosen graph, closely related to a circuit representation of the function to be securely computed. Towards upper bounding the communication complexity of securely computing *any* circuit, we are interested in upper bounding the size of the smallest k-path hitting set for very n-vertex DAG of in-degree at most two.

For every $k \geq 2$, the optimisation version of the directed k-path hitting set problem was shown to be NP-complete for general DAGs by Paindavoine and Vialla [PV16]. For $k = 1$, the problem coincides with the vertex cover problem[8] which was shown to be NP-complete in general DAGs by Naumann [Nau09].

The analogue problem on undirected graphs (which is also NP-complete in its optimisation version) has been studied by Brešar, Kardoš, Katrenič, and Semanišin [BKKS11] and by Jianhua [Tu22]. Since every directed k-path in a DAG corresponds to a k-path in the DAG's underlying (undirected) graph, establishing an upper bound in the undirected case (*i.e.* establishing an upper bound on the smallest k-path hitting set of the worst 2-degenerate graph on n vertices) also applies to the directed case. For $k = 3$, which in the undirected setting corresponds to a bound on the *dissociation number*, applying the result of Brešar, Kardoš, Katrenič, and Semanišin [BKKS11, Corollary 10] yields a bound[9] of $2n/3$. For all $k \geq 2$, [BKKS11, Theorem 4] yields a bound[9] of $(\frac{3}{5} + \frac{2}{5k})\cdot n$.

Finally, we mention that this problem is related to the *bootstrap(ping) problem* [LP13, PV16, BLMZ17]. Most constructions of fully homomorphic encryption are based on LWE and follow the same blueprint. Each ciphertext is associated with some noise, which grows at each homomorphic operation. Typically, the noise grows additively for linear gates and multiplicatively for non-linear gates. Once the noise reaches a certain limit, decryption is no longer possible (at least not without significant

[6] The complexity we provide here is for when the depth is at most $(\log\log s)/4$, not $(\log\log s - \log\log\log s)$; this is done in order to simplify the expression and absorb some negligible terms.

[7] A *synchronous* circuit is a layered circuit whose input gates are all in the first layer.

[8] We note there is some inconsistency in the literature in how the vertex cover problem is extended to directed graphs.

[9] To obtain these bounds, observe that the undirected underlying graphs of fan-in two DAGs have average degree no more than 4. We do not formally state these results (which are direct corollaries of [BKKS11]) in the body however, as we provide improved bounds in Section 4.2.

error in correctness). The key technique to manage this noise growth, due to Gentry [Gen09], is called "bootstrapping". By using an encryption of the secret key (which requires circular-security) one can homomorphically run the decryption procedure on the noisy ciphertext in order to produce a fresh ciphertext, encrypting the same value under the same key, but with a reset noise level. This bootstrapping operation is computationally heavy however, which motivates the question to try and reduce the number of times it is required for computing an arbitrary circuit. The *bootstrap problem*, formalised by Benhamouda, Lepoint, Mathieu, and Zhou [BLMZ17], asks to minimise the number of bootstraps required to compute a circuit. Consider a DAG whose vertices all have in-degree exactly 0 or 2 and are coloured in one of three colours: nodes of in-degree 0 are white (corresponding to inputs to the computation), and the other nodes are either blue (corresponding to additions) or red (corresponding to multiplications). In graph theory terms, the bootstrap problem for maximum noise level L asks to find a small set of *marked* vertices such that every path containing $L+1$ red nodes must also contain at least one marked vertex. In other words, the bootstrap problem asks to reduce the *multiplicative* depth of a circuit by removing a few nodes.

1.2 Overview of our Results

We present a framework for achieving low-communication MPC for *all* circuits, sometimes allowing for a restriction on the depth, but never relying on a specific circuit topology (such as layered circuits). The secure computation protocol is based on the following pebbling game on the circuit's underlying DAG:

1. A node u may be pebbled if it is a source node, or if every length-k directed path (the length is counted in vertices) ending in u contains a node which has already been pebbled.
2. Eventually, every node corresponding to an output gate must be pebbled (this includes every sink).

At a high level, the protocol has the parties compute some "masked version" (secret shares or ciphertexts) of the value of each pebbled gate. The parties start with the inputs, then iteratively compute shares of the value of each pebbled gate using the fact that each one can be computed as a depth-k circuit of already computed masked values. Finally, the parties "open up" the masked output values. Typically, the communication required to generate the masked inputs should scale only with the input size, the cost of computing the intermediary masked values should scale only with the number of pebbled gates, and the cost of opening the masked outputs should scale only with the output size. Instantiating this framework requires two key ingredients:

1. An algorithm to find a good pebbling of the circuit's underlying DAG.
2. A low-communication protocol to compute a masked version of the output of a depth-k circuit, given masked version of the inputs.

Solving the pebbling game. Our approach is to consider the circuit's underlying DAG, remove the input and output nodes, and then identify a directed k-path hitting set of the resulting DAG. Removing all these nodes yields a DAG of depth $k-1$.

Main Theorem 1 (Depth-reduction algorithms for fan-in two circuits)**.** *Let G be an in-degree two, depth-d DAG on n vertices. Then*

$$h_k(G) \leq \begin{cases} \lfloor 2n/3 \rfloor & \text{if } k = 2 \quad (1a) \\ \frac{2n}{5} \cdot \left(1 + \frac{3/2}{k}\right) & \text{for any } 1 \leq k < d \quad (1b) \\ \mathcal{O}\left(n \cdot \left(1 - \frac{\log k}{\log d}\right)\right) & \text{for any } 1 \leq k < d \quad (1c) \end{cases}$$

where $h_k(G)$ is the size of the smallest k-path hitting set of G. Furthermore this bound is constructive. Note that the last expression is particularly interesting in the $k = d^{1-o(1)}$ regime, as it yields a hitting set of size $o(n)$.

Main Corollary 1 (Circuit depth-reduction from k-path hitting set)**.** *Let $\mathcal{R}$ be a finite ring. Let C be a fan-in two, depth-d, n-input, m-output circuit with s gates*[10]*. Let (k, M) be any of the following combinations of parameters:*

$$\begin{cases} M = m + \lfloor 2(s-m)/3 \rfloor & \text{and} \quad k = 2 \\ M = m + \frac{2(s-m)}{5} \cdot \left(1 + \frac{3/2}{k}\right) & \text{and} \quad 1 \leq k < d \\ M = m + \mathcal{O}\left((s-m) \cdot \left(1 - \frac{\log k}{\log d}\right)\right) & \text{and} \quad 1 \leq k < d \end{cases}$$

There exists a sequence of M fan-in two, depth-k, single-output circuits $C_1, \ldots, C_M$ such that:

$$\forall \vec{x} \in \mathcal{R}^n,\ C(\vec{x}) = C_M\left(\vec{x}, (C_j(\vec{x}, (C_i(\vec{x}, \ldots))_{i<j}))_{j<M}\right) .$$

(Or in algorithmic form, if

1. *$y_1 \leftarrow C_1(\vec{x})$*
2. *For $i = 1 \ldots M$, $y_i \leftarrow C_i(\vec{x}, (y_1, \ldots, y_{i-1}))$*

then $C(\vec{x}) = C_M(\vec{x}, (y_i)_{i=1}^M)$.)

Instantiating the framework. We demonstrate the usefulness of main theorem 1 by showing how each of the three expressions has applications to low-communication secure multiparty computation: the parameters of eq. (1a) are useful to obtain *fractionally linear-communication MPC in the correlated randomness model*, those of eq. (1b) are best suited for *1-out-of-2^{2^k} OT-complexity of 2PC*, and finally the parameters of eq. (1c) are best applied to *sublinear-communication secure computation low-depth circuits*.

Corollary 1 (MPC protocols through the depth-reduction lens).

1. **Information-theoretic MPC in the correlated randomness model.** *Let C be an n-input, m-output, $\mathbb{F}$-arithmetic circuit with s non-output computation gates (over any basis of binary gates). There exists an N-party protocol for perfectly securely computing C in the correlated randomness model in the presence of a semi-honest adversary corrupting up to $N-1$ parties. The protocol uses the following resources (in bits):*
   - Total communication:
   $$\left(\frac{2}{3}s + n + m\right) \cdot N \cdot \log|\mathbb{F}|$$
   - Correlated randomness per party: $\mathcal{O}(s+n) \cdot \log|\mathbb{F}|$ *bits of function-dependent correlated randomness per party (or alternatively $B^{\mathcal{O}(1)} \cdot \log|\mathbb{F}|$ bits of function-independent correlated randomness, where B is some* a priori *bound on $s+n+m$),*
   - Local computation per party: $\mathcal{O}(s \cdot N \cdot \log^2|\mathbb{F}|)$.

2. **Sublinear-communication MPC from homomorphic secret-sharing.** *Assuming the existence of N-party homomorphic secret-sharing supporting logarithmic depth (respectively doubly logarithmic depth circuits) $\mathbb{F}$-arithmetic fan-in two circuits, there exists sublinear-communication secure N-party computation for* all $\log^{1+o(1)}$*-depth (resp.* $(\log\log)^{1+o(1)}$*-depth) circuits.*

3. **The 1-out-of-M-OT complexity of 2PC.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function which can be computed by a boolean circuit with s computation gates (over any basis of binary gates). For every $M \geq 2$, there exists a two-party protocol for computing C with passive security in the 1-out-of-M-OT hybrid model; this protocol makes $\frac{2}{5}\left(1 + \frac{3/2}{\lceil \log\log M \rceil + 1}\right) \cdot s$ calls to the oblivious transfer functionality.*

[10] In order to simplify the expressions of M, we assume none of the input gates are also outputs.

We note that the protocols of Corollary 1 can be extended to the malicious setting. The results for sublinear-communication from HSS also apply to correlated SPIR: Assuming the existence of correlated SPIR, there exists a sublinear-communication secure two-party protocol for $(\log\log)^{1+o(1)}$-depth circuits. Finally our depth-reduction algorithms (notably those in the parameter range of eq. (1b)) provide upper bounds on the number of bootstraps required to achieve fully homomorphic encryption while tolerating a small constant maximum noise level: Given an FHE scheme tolerating a maximum noise level of L, only $\frac{2}{5}s \cdot (1 + \frac{1.5}{L+1})$ bootstraps are required to homomorphically evaluate a size-s circuit (over an arbitrary basis of binary gates).

Concrete efficiency. We conclude this section by comparing our fractionally linear-communication protocol to the previously best known linear-communication protocols in the correlated randomness model, in the two-party setting. The comparison is summarized in Table 1. When compared with the "Tiny Tables" protocol of Damgaard, Nielsen, Nielsen, and Ranelluci [DNNR17], we save a factor 1.5 in communication at the cost of a factor 4 in computation. The savings in communication can be asymptotically increased to a factor 2.5, but only by losing a very large factor in computation.

Previous Works		
	"GMW with OT correlations" [GMW87, Bea92]	"Tiny Tables" [DNNR17]
Communication	$s_\times + n + m$	$s + n + m$
Correlated Randomness	$4s_\times + n$	$3s + n$
Function-Independent CR?	Yes	No
Computation	$s^{\mathcal{O}(1)}$	$s^{\mathcal{O}(1)}$
This Work		
	Colouring-based	FVS-based (Any $k \geq 2$)
Communication	$\frac{2}{3}s + n + m$	$\frac{2}{5}(1 + \frac{3/2}{k}) \cdot s + n + m$
Correlated Randomness	$10s + n$ †	$2^{2^k} \cdot s + n$ †
Function-Independent CR?	No †	No †
Computation	$s^{\mathcal{O}(1)}$	$\mathcal{O}(2^{2^k}) \cdot s^{\mathcal{O}(1)}$

† The correlated randomness can be made *function-independent* at the cost of increasing it to B^{2^k} bits, where B is some *a priori* bound on $n + m + s$.

Table 1: Resources in field elements per party for computing a fan-in two arithmetic circuit with n inputs, m outputs, and s computation gates ($s_\times$ of which are multiplicative, and $s - s_\times$ linear), in the two-party setting. In the last column, k is not necessarily a constant. Note that the GMW protocol coincides with an extension of our protocol to $k = 1$ (up to free addition), as a "1-path hitting set" is the set of all vertices.

1.3 Technical overview

We refer to Section 2 for definitions of the graph-theoretical notions discussed here. We consider three families of depth-reduction algorithms in this paper. Recall our goal is to constructively give an upper bound for $h_k(G)$ where the DAG G has n nodes and is of in-degree-2.

Colouring-based depth-reduction. Observe that 2-paths are exactly (directed) edges, so the hitting set we are looking for is exactly a vertex cover. A DAG of in-degree two is 3-colourable, and

furthermore a 3-colouring can be found in polynomial time by the following greedy algorithm: colour all sources with the same colour, then iteratively colour nodes in topological order with any of the three colours not already used by its parents. Since each node has at most two parents, the algorithm never gets "stuck" and only ever needs three colours.

Consider the 3-partition of the nodes obtained by this colouring. The union of the two smallest partition must have size at most $\lfloor 2n/3 \rfloor$ (or they would not be the smallest), and they must be a vertex cover of the graph (or there would be an edge whose two endpoints are of the third colour, which would violate the proper 3-colouring property).

FVS-based depth-reduction. Our second family of depth-reduction algorithms is based on the following approach:

1. First, remove vertices from the in-degree-2 DAG so the resulting graph lies in some graph class $\mathcal{G}$.
2. Then, use a special-purpose depth-reduction algorithm for class $\mathcal{G}$.

Specifically, we consider the class $\mathcal{G}$ of all directed forests. This is motivated by the fact that directed trees (which in particular can be seen as layered graphs) admit very efficient depth-reduction algorithms. More specifically, we can reduce the depth of a directed forest to $k-1$ while removing only a 1-in-k fraction of its vertices. Indeed, while the forest still has depth at least k we can iteratively remove the $(k-1)^{\text{th}}$ ancestor Ⓤ of the deepest leaf Ⓥ. Whenever we remove a node in this fashion, we are guaranteed we will never need to remove any of the $k-1$ nodes of the path from Ⓤ (excluded) to Ⓥ (included). We say these $k-1$ nodes have been "saved by Ⓤ". Because each node can be saved by at most one node's removal, when the algorithm terminates we are guaranteed to have removed only a 1-in-k fraction of the nodes (if we removed ℓ nodes, then we saved $\ell \cdot (k-1)$; because $\ell + \ell \cdot (k-1) \leq n$, necessarily $\ell \leq \lfloor n/k \rfloor$).

The first step is handled by identifying a feedback vertex set (FVS) of the in-degree-2 DAG's underlying undirected graph. Fortunately, while the problem of finding a small FVS is NP-complete in general graphs, it is known that every 2-degenerate graph (that is, one whose edges can be oriented as an in-degree-2 DAG) admits an FVS of size $\lfloor 2n/5 \rfloor$ [BDBS14].

By combining these two steps, every n-vertex, in-degree-2 DAG has a k-path hitting set of size $\lfloor 2n/5 \rfloor + \lfloor \frac{n - \lfloor 2n/5 \rfloor}{k} \rfloor$ (which is at most $\frac{2n}{5} \cdot (1 + \frac{3/2}{k})$).

Valiant's [Val77] edge-partitioning-based depth-reduction. The third algorithm is one proposed by Valiant [Val77], and which we simply apply to secure multiparty computation[11]. For completeness, we provide here a sketch of how this algorithm works. This DAG depth-reduction algorithm is based on deleting edges. In an in-degree-2 DAG however, the number of edges is within a factor 2 of the number of vertices, so the problem is essentially the same as the variant we consider. Valiant's algorithm starts by partitioning the vertices in d layers, according to their depth in the DAG. Note that because we do not assume the DAG itself has a layered structure, the edges are allowed to join any pair of layers. The edges are partitioned as $X_1 \sqcup \cdots \sqcup X_{\log d}$ as follows: an edge connecting two vertices in layers L_1 and L_2 is placed in partition X_i where i is the most significant bit in which the binary representations of L_1 and L_2 differ. We provide a visual representation of this partition in Figure 1: any edge "crossing the blue line" is in X_1; any edge not in X_1 but "crossing a red line" is in X_2; any edge not in $X_1 \cup X_2$ but "crossing a green line" is in X_3. The key observation is that whenever we iteratively remove one of the partitions, the depth of the DAG is reduced by a factor 2. Removing the $\log(d/k)$ smallest partitions (whose union has size at most $(\log(d/k)) \cdot \frac{\#\text{edges}}{\log d}$) therefore yields a depth-k DAG. It follows that every in-degree-2, n-vertex DAG of depth d admits a $(k+1)$-path hitting set of size $\mathcal{O}(n \cdot (1 - \frac{\log k}{\log d}))$.

[11] We note that Valiant's algorithm was previously applied in essentially the same way to other—yet incomparable—notions of depth-reduction and pebbling games, in the study of memory-hard functions [AB16, ABH17].

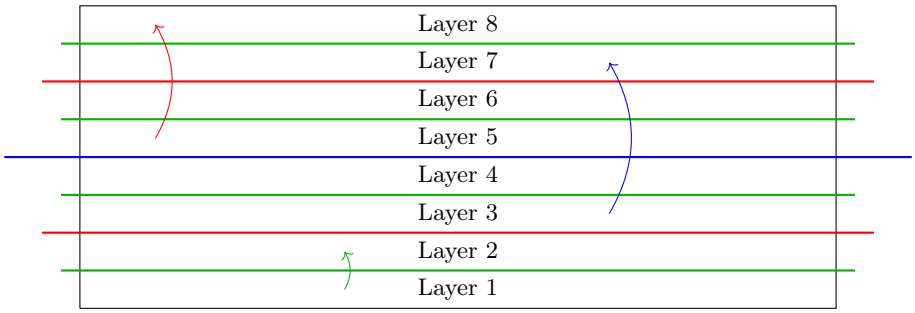

Fig. 1: Visual representation of the edge partitioning mechanism in Valiant's [Val77] depth-reduction algorithm.

2 Preliminaries

2.1 Cryptographic Notions

Definition 1 (Homomorphic Secret Sharing). *An N-party* Homomorphic Secret-Sharing (HSS) *scheme (with additive reconstruction) for a class $\mathcal{F}$ of functions over a finite field $\mathbb{F}$ is a pair of algorithms* $\mathsf{HSS} = (\mathsf{HSS.Share}, \mathsf{HSS.Eval})$ *with the following syntax and properties:*

- $\mathsf{Share}(1^\lambda, x)$*: On input 1^λ (the security parameter) and $x \in \mathbb{F}^{n(\lambda)}$ (the input), the* sharing algorithm Share *outputs N input shares $(x^{(1)}, \ldots, x^{(N)})$.*
- $\mathsf{Eval}(i, f, x^{(i)})$*: On input $i \in [N]$ (the party index), $f \in \mathcal{F}$ (the function to be homomorphically evaluated, implicitly assumed to specify input and output lengths n, m), and $x^{(i)}$ (the i^{th} input share), the* evaluation algorithm Eval *outputs the i^{th} output share $y^{(i)} \in \mathbb{F}^m$.*

- ***Correctness:*** *For any 1^λ, input $x \in \mathbb{F}^{n(\lambda)}$, and any function $f \in \mathcal{F}$,*

$$\Pr\left[y^{(1)} + \cdots + y^{(N)} = f(x) \;:\; \begin{array}{l}(x^{(1)}, \ldots, x^{(N)}) \xleftarrow{\$} \mathsf{HSS.Share}(1^\lambda, x) \\ y^{(i)} \xleftarrow{\$} \mathsf{HSS.Eval}(i, f, x^{(i)}),\ i = 1 \ldots N\end{array}\right] = 1 \ .$$

- ***Security:*** *For every set of corrupted parties $\mathcal{D} \subsetneq [N]$, we consider the following game:*
  1. *The adversary $\mathcal{A}$ sends inputs (x_0, x_1) with $|x_0| = |x_1|$.*
  2. *The challenger picks $b \xleftarrow{\$} \{0,1\}$ and $(x^{(1)}, \cdots, x^{(N)}) \xleftarrow{\$} \mathsf{HSS.Share}(1^\lambda, x_b)$.*
  3. *The adversary outputs a guess $b' \leftarrow \mathcal{A}((x^{(i)})_{i \in \mathcal{D}})$.*

We let $\mathsf{Adv}(1^\lambda, \mathcal{A}, \mathcal{D})$ denote the advantage $|1/2 - \Pr[b = b']|$ of $\mathcal{A}$. The scheme is secure *if for any $\mathcal{D} \subsetneq [N]$ and any PPT adversary $\mathcal{A}$, $\mathsf{Adv}(1^\lambda, \mathcal{A}, \mathcal{D})$ is negligible.*

2.2 Graph-Theoretic Notions

In this paper, we consider only simple graphs, both directed and undirected. For a graph G we let $V(G)$ and $E(G)$ be the sets of vertices and edges of G, respectively. For a subset $U \subseteq V(G)$, we use $G[U]$ to denote the subgraph of G induced by U, *i.e.* the graph $(U, E(G) \cap U^2)$. We sometimes use the acronym *DAG* as shorthand for "directed acyclic graph" and *di-graph* or *digraph* as shorthand for "directed graph". In a DAG, the nodes of in-degree 0 are called *sources* and the nodes of out-degree 0 are called *sinks*. Borrowing from circuit terminology, we allow a subset of a DAG's nodes to be identified as *output nodes*: this can be an arbitrary subset, provided that it contain all sinks and no sources.

(Directed) paths. A (directed path) *in* a (directed) graph is a non-repeating sequence of vertices such that each pair of consecutive vertices form an (arc) edge of the graph. The *length* of a (directed) path is the number of vertices[12] in this sequence. The *depth* of a directed acyclic graph is the length of the longest path in it.

[12] Beware that the literature is inconsistent on whether the length of a path should be counted in vertices or edges.

Feedback Vertex Set. A *feedback vertex set* (FVS) of an undirected graph G with vertex set V is a set $S \subseteq V$ of its vertices such that the graph $G[V \setminus S]$ is a forest. The *FVS decision problem* asks, on input an undirected graph G and a positive integer k, whether G admits an FVS of size at most k.

Vertex Colouring. A *(proper) vertex colouring* of graph $G = (V, E)$ is a function $\pi \colon V \to \mathbb{N}$ which assigns different values (called *colours*) to neighbouring vertices. A *k-vertex-colouring* (or simply a *k-colouring*) is a vertex colouring which uses at most k colours.

Independent set. An *independent set* is a set of vertices in a graph such that no pair of them are adjacent to each other.

Graph Degeneracy. The *degeneracy* [LW70] (*a.k.a.* the coloring number -1 [EH66]) of a graph G is the least integer $k \geq 0$ such that every induced subgraph of G contains a vertex with at most k neighbours. If a graph has degeneracy at most d, we say it is *d-degenerate*. A graph is d-degenerate if and only if it admits a d-elimination ordering [LW70], *i.e.* an ordering of the vertices in which each vertex appears after at most d of its neighbours. A graph is d-degenerate if and only if the edges of G can be oriented to form a directed acyclic graph with in-degree at most d [CE91].

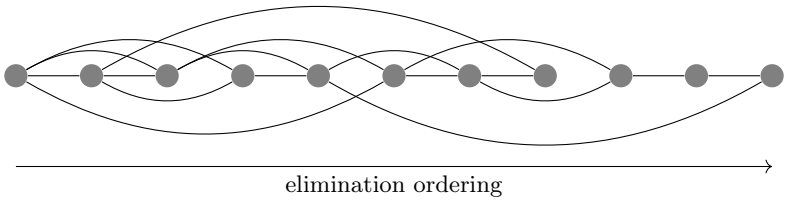

Fig. 2: The nodes of a 2-degenerate graph can be ordered in such a way that each node is preceded (in the ordering) by at most 2 of its neighbours.

3 Depth-Reduction Pebbling Game

The goal of this section is to show how solving the graph-theoretic problem of reducing an in-degree-2 DAG's depth by removing (few) nodes leads to low-communication MPC protocols.

In Section 3.1 we state the graph theoretic problem of *DAG depth reduction*. In Section 3.2 we introduce a "pebbling game" for the gates of a (boolean or arithmetic) circuit, and show it can be solved using DAG depth-reduction. In Section 3.3 we explain how many existing low-communication MPC protocols implicitly rely on finding a good pebbling of the circuit to be securely computed.

3.1 The depth-reduction problem for directed acyclic graphs

Definition 2 (DAG Depth-Reduction).

- **DAG Depth-Reducibility.** *Let $k, \ell \in \mathbb{N}^\star$. We say a directed acyclic graph $G = (V, E)$ is* (k, ℓ)-depth-reducible *if there exists a subset $S \subseteq V$ of size at most ℓ such that $G[V \setminus S]$ has depth at most k (*i.e. *the longest directed path in $G[V \setminus S]$ contains at most k vertices).*
- **The DAG Depth-Reduction Problem.** *Given a directed acyclic graph $G = (V, E)$, the* (k, ℓ)-depth-reduction problem *(denoted (k, ℓ)-DR) asks to find a subset $S \subseteq V$ of size at most ℓ such that $G[V \setminus S]$ has depth at most k (*i.e. *the longest directed path in $G[V \setminus S]$ contains at most k vertices), or output $\perp$ if no such subset exists.*

Remark 1 (Equivalent Problems to DAG Depth-Reduction). The following holds:

1. The (k, ℓ)-DR problem is that of finding a hitting set of size at most ℓ for all directed $(k+1)$-paths in G.
2. The $(1, \ell)$-DR problem is that of finding a vertex cover of size at most ℓ.

We bring the reader's attention to the fact that removing a k-path hitting set from a DAG yields a DAG of depth at most $k - 1$.

3.2 A depth-reduction pebbling abstraction

We outline below a convenient intermediate abstraction to interface between the depth-reduction problem over digraphs and the execution of a secure multiparty computation protocol. We model the distributed computation of a function f, given by a boolean circuit C, as the *Depth-Reduction pebbling game*. In this pebbling game the parties can place a pebble on a node s of the underlying graph G of the circuit if the following condition is met: either the node is an input node, or all directed path of length $k+1$ in G ending at s already contain a pebbled node. Formally,

Definition 3 (DR pebbling game). *Given a digraph $G = (V, E)$ and a depth parameter k, a* depth-reduction *(*DR*) pebbling game* PG *is a list of subsets $S \subset V$ of the nodes of G. A* DR *pebbling game* $\mathsf{PG} = (S_1, \cdots, S_{|\mathsf{PG}|})$ *is* valid *if after executing the commands* $\mathsf{PebbleGates}(S_i)$ *sequentially for $i = 1$ to $|\mathsf{PG}|$, defined in Figure 3,*

- *no command returns an* $\mathsf{invalid}$ *flag,*
- *the termination condition is met (*i.e. *all output nodes are pebbled).*

The cost $\mathsf{cost}(\mathsf{PG})$ *of a valid* DR *pebbling game* PG *is defined as the total number of pebbles placed throughout the game,*

$$\mathsf{cost}(\mathsf{PG}) = \sum_{S \in \mathsf{PG}} |S|$$

and its length $\mathsf{length}(\mathsf{PG}) = |\mathsf{PG}|$ *is the length of the list (*i.e. *the number of subsets).*

The rules of the DR pebbling game are formally described on Figure 3.

DR pebbling rules

Parameters. The game is parameterised by a directed acyclic graph $G = (V, E)$ with in-degree 2, and a depth parameter k.

Initialisation. At the start of the game, a pebble is placed on all input nodes.

Moves. The player can execute the following command:

▷ $\mathsf{PebbleGates}(S)$: given a set $S \subset V$ of gates,
- for every $s \in S$, check that every path in G of length $k+1$ ending at s contains a pebbled node. If not, return the flag $\mathsf{invalid}$.
- Place a pebble on all nodes in S.

Termination. The game terminates once all output nodes have been pebbled.

Fig. 3: The moves and termination condition of a depth-reduction pebbling game.

Then, we have the following straightforward lemma that relates the cost of a pebbling game to the k-depth-reducibility of G:

Lemma 1. *Let $G = (V, E)$ be a digraph with m output nodes*[13]*; we denote V_{sources} and V_{outputs} the sets of sources and outputs of G, respectively. Let $G' \coloneqq G[V \setminus (V_{\mathsf{sources}} \sqcup V_{\mathsf{outputs}})]$. If G' is (k, ℓ)-depth-reducible, then there is a valid* DR *pebbling* PG *of G with depth parameter k, cost* $\mathsf{cost}(\mathsf{PG}) \leq \ell + m$*, and whose length $|\mathsf{PG}|$ is the largest number of pebbles on a path from a source to a sink in G.*

Proof. The proof follows from a straightforward greedy procedure. Because G' is (k, ℓ)-depth reducible, there exists a size-ℓ subset V' of the nodes of G' whose removal yields a depth-k DAG. Initially, all sources in G are pebbled. At each round, identify the largest set $S \subseteq V' \cup V_{\mathsf{outputs}}$ of unpebbled nodes such that $\mathsf{PebbleGates}(S)$ does not return $\mathsf{invalid}$ (i.e., the set of all nodes $s \in V'$ such that every path in G of length $k+1$ ending at s contains a pebbled node), and pebble this set. Iterate until all nodes in V_{outputs} have been pebbled. It follows from the definition of (k, ℓ)-depth-reducibility terminates with all the sinks coloured. Indeed, assume toward contradiction that a node $s \in V_{\mathsf{outputs}}$

[13] We refer to Section 2.2 for what we mean by "output nodes of a DAG".

was not pebbled by this procedure. Then there must be an ancestor a of s (at distance at most k of s) which was not pebbled by this procedure (otherwise, s would have been pebbled). Repeating this argument from the node a, we arrive after at most $\mathsf{depth}(G)$ steps at an input node, which is pebbled by assumption, reaching a contradiction. Furthermore, all nodes pebbled in this process are nodes from $V' \cup V_{\mathsf{outputs}}$, hence $\mathsf{cost}(\mathsf{PG}) \leq |V' \cup V_{\mathsf{outputs}}| \leq \ell + m$. This concludes the proof. □

3.2.1 Example: a DR pebbling of layered graphs. We say that an in-degree-2 directed acyclic graph is *layered* [GJ11] if its nodes can be divided into layers, such that any edge only connects adjacent layers. Let $G = (V, E)$ be a layered in-degree-2 directed acyclic graph with $|V| = s$ nodes, and let k be an arbitrary parameter. Let n be the number of sources and m be the number of sinks in G. Let d be the depth of G.

Claim. There exists a valid DR pebbling PG of G with $|\mathsf{PG}| \leq \lceil d/(k+1) \rceil + 1$ and $\mathsf{cost}(\mathsf{PG}) \leq m + \lceil s/k \rceil$.

Equivalently, the proof below shows that layered graphs of size s with m outputs are $(k-1, m+\lceil s/k \rceil)$-depth reducible for any k.

Proof. First, observe that d is necessarily equal to the number of layers in G. Second, divide G into $\lceil d/k \rceil$ chunks of k consecutive layers (written from top to bottom). Let $i \in [1, k]$ be an index such that the total number of nodes in the i-th layers of all chunks is at most $\lceil s/k \rceil$ (such an index i exists by the pigeonhole principle).
The pebbling strategy proceed in $\lceil d/k \rceil + 1$ rounds: in the j-th round, place pebbles on all nodes in the i-th layer of the j-th chunk, as well as on all unpebbled sinks above the i-th layer of the j-th chunk. Observe that when $j = 1$, all nodes in the i-th layer of the first chunk, and all sinks above this layer, have depth at most $i \leq k$, hence every path of length k ending at a node of the i-layer must contain an source (and length-k paths can only exist when $i = k$), which is pebbled.
When $j > 1$, all k-ancestors of the i-th layer of the j-th chunk are on the i-th layer of the $(j-1)$-th chunk (because the graph is layered), hence they have been pebbled at the round $j - 1$. Furthermore, each path of length k ending on an unpebbled sink above the j-th chunk contains a node on the i-th layer of the ℓ-th chunk, for some $\ell < j$, which was pebbled in one of the previous rounds.
Eventually, in the $(\lceil d/k \rceil + 1)$'s round (the last round), it only remains to place a pebble on the remaining unpebbled sinks (if any) situated after the i-th layer of the last chunk, and all length-k path ending in these nodes contain a pebbled node from the i-layer of the last chunk. Therefore, all moves are valid, and all sinks are pebbled at the end of the DR pebbling game. In total, the game places pebbles on the i-th layer of each chunk (at most $\lceil s/k \rceil$ pebbles in total) plus a pebble on each sink (at most m additional pebbled), hence $\mathsf{cost}(\mathsf{PG}) \leq m + \lceil s/k \rceil$. This concludes the proof. □

3.3 Recasting MPC protocols through the lens of DR pebblings

The DR pebbling game abstraction allows to recast several existing low-communication MPC protocols in a unified way which isolates their cryptographic component from the graph algorithm that they implicitly rely on to traverse the circuit of the function. The literature contains several protocols which require less communication than the size of the circuit, for a suitable class of "well-structured" circuits. For example, [BGI16a, Def 4.6] introduces the notion of circuits over branching programs, while [Cou19, CM21] use layered circuits, and [BCM22, BCM23] use synchronous circuits.
All these protocols (and others) evaluate the circuit by distributively and iteratively computing the values carried on a subset of intermediate nodes in a hidden fashion (the values are either shared [BGI16a, Cou19], masked [BCM22], or encrypted [Gen09, BV11, BGV12]. Computing the cost of these protocols (in terms of communication, computation, and storage overhead) can be abstracted out as follows:

- the cryptographic mechanism introduced in the protocol induces a (storage, communication, and computation) cost for each intermediate node whose value is (securely) computed, and
- the final cost of running the protocol is derived by summing the costs of computing the intermediate nodes, where the nodes are selected using a suitable valid DR pebbling of the graph of the circuit.

Cast in this language, there is nothing mysterious in the restriction to circuit classes such as layered or synchronous circuit: it simply comes from the fact that these are classes of circuits whose underlying graph G is (k, ℓ)-depth-reducible with non-trivial parameters (k, ℓ), which in turns implies by Lemma 1 the existence of a good DR pebbling game on G; see also Section 3.2.1 for an explicit description of an efficient pebbling strategy for layered graph. It also follows immediately that all of these results can be extended to larger classes of circuits provided that we can find sufficiently non-trivial DR pebbling games for their underlying family of graphs.
For the sake of concreteness, and to facilitate the statements of the corollaries which we obtain in this paper, we work out explicitly the abstraction on a few illustrative examples below. We stress that we do not prove new results on secure computation in what follows: rather, for a list of existing protocols that were originally described over a restricted class of circuits, we observe that the protocols work identically over general circuits but that their efficiency depends on the existence of an efficient DR pebbling. Our lemmas and corollaries simply reformulate the existing results in this setting.

3.3.1 Information-theoretic secure computation in the correlated randomness model. In [Cou19], Couteau introduced the first information-theoretic secure computation protocol in the correlated randomness model which achieves sublinear communication complexity $O(s/\log\log s)$ for all layered circuits of size s, using a polynomial amount of computation and correlated randomness. For simplicity, we focus here on the case of secure computation of boolean circuits with semi-honest security, but the result of [Cou19] extends to arithmetic circuit and to the malicious setting.
On Figure 4, we recall the protocol Π_{corr} of [Cou19]. Rather than focusing on layered circuits, we describe the protocol for an arbitrary circuit C given a suitable pebbling of the underlying graph of C. Let C be a circuit, and let PG be a valid DR pebbling of the graph G of C with depth parameter $k(|C|) = \log\log|C|$.

Protocol Π_{corr}

Parameters. Let C be a boolean circuit whose underlying digraph is $G = (V, E)$. Let $k(|C|) = \log\log|C|$ be a depth parameter, and let PG be a valid DR pebbling of G. The protocol involves N parties $(P_1, \cdots, P_N)$.

Correlated randomness. The trusted dealer proceeds as follows:

- For each node $v \in V$ which is not an output node, the trusted dealer samples a random mask $r_v \xleftarrow{\$} \mathbb{F}_2$. For each output node $v \in V$, the trusted dealer sets $r_v \leftarrow 0$. For all nodes $v \in V$, the dealer samples N random shares $(r_v^{(i)})_{i \leq N}$ of r_v.
- For each set $S \in \mathsf{PG}$, for each $s \in S$, let $V_s \subset V$ denote a set containing one pebbled node from each length-k path ending at s (V_s exists because PG is a valid pebbling). Let $f_s : \{0,1\}^{|V_s|} \mapsto \{0,1\}$ denote the function which computes the value carried by the node s in C from the values carried on all nodes $v \in V_s$. Let M_s denote the truth-table of the function $x \mapsto f_s(x \oplus (r_v)_{v \in V_s})$. The dealer samples N random shares $(M_s^{(i)})_{i \leq N}$ of M_s.
- The dealer sends $(r_v^{(i)})_{v \in V}$ and $(M_s^{(i)})_{s \in S}$ for all sets $S \in \mathsf{PG}$ to each party P_i. The dealer also sends r_v for each output node v to all parties.

Protocol. We assume w.l.o.g that the parties hold shares of all inputs.

- For each input node s and each input shares $(x^{(1)}, \cdots, x^{(N)})$ of an input value x on the node s, each party P_i sends $x^{(i)} \oplus r_s^{(i)}$. All parties reconstruct $x \oplus r_s = \bigoplus_i (x^{(i)} \oplus r_s^{(i)})$. Throughout the protocol the parties will maintain the following invariant: each time they pebble a node v, they will reconstruct $z_v = u_v \oplus r_v$, where u_v is the value carried on this node.
- For each set $S \in \mathsf{PG}$, for each $s \in S$, the parties retrieve the masked values z_v for each $v \in V_s$, which they computed previously. Each party P_i broadcasts $M_s^{(i)}[(z_v)_{v \in V_s}] \oplus r_s^{(i)}$ and all parties reconstruct $\bigoplus_i (M_s^{(i)}[(z_v)_{v \in V_s}] \oplus r_s^{(i)}) = M_s[(z_v)_{v \in V_s}] \oplus r_s = f_s((u_v)_{v \in V_s}) \oplus r_s = u_s \oplus r_s = z_s$.
- Once z_v has been computed for all output nodes v, all parties output the z_v's. Note that $z_v = u_v + r_v$ where u_v is the value carried on the node, and $r_v = 0$ for all output nodes.

Fig. 4: An information-theoretic secure computation protocol in the correlated randomness model

Lemma 2. *Let C be an n-input boolean circuit with m output gates, and let PG be a valid DR pebbling of its graph $G = (V, E)$. Then the protocol Π_{corr} of Figure 4 is an information-theoretically secure N-party protocol for computing C in the correlated randomness model. Furthermore, the protocol Π_{corr} has the following efficiency properties:*

- Correlated randomness: *each party receives at most $|V| + \mathsf{cost}(\mathsf{PG}) \cdot 2^{2^k}$ bits of correlated randomness from the trusted dealer.*
- Communication: *the total communication of the protocol is upper bounded by $N \cdot (n + \mathsf{cost}(\mathsf{PG}))$.*
- Computation: *each party performs at most $O(\mathsf{cost}(\mathsf{PG}) \cdot (2^{2^k} + N))$ boolean operations.*

Lemma 2 is a reformulation of [Cou19, Theorem 1] in the setting of general circuits with a DR pebbling game, and the proof of Lemma 2 is a direct adaptation of the analysis in [Cou19]. Due to the choice of $k = \log\log(s)$, note that the amount of correlated randomness and computation remain polynomial. Furthermore, whenever $\mathsf{cost}(\mathsf{PG}) \ll s$, the total communication of the protocol is $\ll N \cdot (n + s)$, *i.e.*, below the "circuit-size barrier". Plugging the efficient valid DR pebbling of layered graphs described in Section 3.2.1 (whose cost is at most $m + \lceil s/\log\log(s) \rceil$) recovers the exact statement of Theorem 1 in [Cou19] for layered boolean circuits.

3.3.2 Secure computation from homomorphic secret sharing. We recall below another approach for sublinear secure computation, which was originally introduced in [BGI16b]. This protocol established HSS as the first known alternative to FHE to obtain sublinear secure computation for an expressive class of circuits. On Figure 5, we recall a simplified variant of the protocol of [BGI16b] for arbitrary circuits with a valid DR pebbling given a statistically correct HSS scheme for the class NC^1, and discuss extensions that rely on weaker forms of HSS afterwards. Let C be a circuit, and let PG be a valid DR pebbling of the graph G of C with depth parameter $k(|C|) = \log|C|$.

Protocol Π_{hss}

Parameters. Let C be a boolean circuit whose underlying digraph is $G = (V, E)$. Let $k(|C|) = \log|C|$ be a depth parameter, and let PG be a valid DR pebbling of G. Let $\sigma : V \mapsto \{0,1\}$ be a function which, on input $v \in V$, returns 0 if v is an output node, and 1 otherwise. The protocol involves 2 parties (P_0, P_1). Let $\mathsf{HSS} = (\mathsf{HSS.Share}, \mathsf{HSS.Eval})$ be an HSS scheme for the class of functions NC^1, and let $\{F_K\}_{K \in \{0,1\}^\lambda}$ be a PRF family in NC^1. Let Π_{Share} denote a secure 2-party protocol which, on input $K_b \in \{0,1\}^\lambda$ from each party P_b, computes $(K_b^{(0)}, K_b^{(1)}) \xleftarrow{\$} \mathsf{HSS.Share}(1^\lambda, K_b)$ for $b = 0, 1$, and outputs $(K_0^{(0)}, K_1^{(0)})$ to P_0 and $(K_0^{(1)}, K_1^{(1)})$ to P_1.

Initialisation. We assume w.l.o.g. that the two parties hold additive shares of all inputs.

- Each party P_b samples $K_b \xleftarrow{\$} \{0,1\}^\lambda$.
- For $b = 0, 1$, the two parties run Π_{Share} on inputs (K_0, K_1). Each party P_b gets $(K_0^{(b)}, K_1^{(b)})$.
- For each input node $v \in V$ and each input shares $(x^{(0)}, x^{(1)})$ of an input value x on the node v, each party P_b broadcasts $v_b \leftarrow F_{K_b}(v) \oplus x^{(b)}$. All parties reconstruct $v^\star = v_0 \oplus v_1 = x \oplus (F_{K_0}(v) \oplus F_{K_1}(v))$ (it helps to view the value $v^\star$ as the pebble on v).

Protocol. Let us denote $\mathsf{PG} = (S_1, \cdots, S_{|\mathsf{PG}|})$. For $i = 1$ to $|\mathsf{PG}|$,

- For each $s \in S_i$, let $V_s \subset V$ denote a set containing one pebbled node in S_{i-1} from each length-k path ending at s (V_s exists because PG is a valid pebbling). Let $f_s : \{0,1\}^{|V_s|} \mapsto \{0,1\}$ denote the function which computes the value carried during the computation (in the circuit C) by the node s from the values carried on all nodes $v \in V_s$. Let us denote $(v_1, \cdots, v_{|V_s|})$ the elements of V_s. For $j = 1$ to $|V_s|$, the parties retrieve the values $v_j^\star$ computed in the previous round. Let $v^\star \leftarrow (v_1^\star, \cdots, v_{|V_s|}^\star)$ Then, the parties define the following function g_s:

$$g_s : \begin{cases} \{0,1\}^{|V_s|} & \mapsto \{0,1\} \\ (K_0, K_1) & \to f_s(v^\star \oplus (F_{K_0}(v_j) \oplus F_{K_1}(v_j))_{j \le |V_s|}) \oplus \sigma(s) \cdot (F_{K_0}(s) \oplus F_{K_1}(s)) \end{cases}$$

Note that $g_s \in \mathsf{NC}^1$ (because the circuit of f_s is a log-depth circuit and the PRF is in NC^1). Furthermore, by construction, $g_s(K_0, K_1) = y \oplus (F_{K_0}(s) \oplus F_{K_1}(s))$ if s is not an output node, where y is the value carried by s during the computation, and $g_s(K_0, K_1) = y$ if s is an output node.

- For each $s \in S_i$, each party P_b computes and sends $s_b \leftarrow \mathsf{HSS.Eval}(b, g_s, (K_0^{(b)}, K_1^{(b)}))$. Both parties reconstruct $s^\star \leftarrow s_0 \oplus s_1$.

OUTPUT. For each output node $v \in V$, the parties output $v^\star$.

Fig. 5: A two-party protocol for C with semi-honest security

Lemma 3. *Let C be a boolean circuit with n input gates and m output gates, and let PG be a valid DR pebbling of its graph $G = (V, E)$. Then the protocol Π_{hss} of Figure 5 is a secure 2-party protocol in the honest-but-curious setting for computing C. Furthermore, the total communication of the protocol Π_{hss} is upper bounded by $2 \cdot (n + \mathsf{cost}(\mathsf{PG})) + \mathsf{poly}(\lambda)$.*

Lemma 3 follows directly from the same analysis as HSS-based secure computation protocols from previous works [BGI16b]. Above, the $\mathsf{poly}(\lambda)$ term refers to the fixed communication cost (independent of n, m, and the circuit size) of the secure 2-party protocol for distributively running $\mathsf{HSS.Share}$ on a pair of λ-bit inputs. Plugging the efficient valid DR pebbling of layered graphs described in Section 3.2.1 (whose cost is at most $m + \lceil s/\log(s) \rceil$) recovers the result of previous works from HSS for NC^1 together with PRFs in NC^1.

Extension 1: Las Vegas HSS. We note that the construction of Lemma 3 is simplified compared to the original construction of [BGI16b] since it assumes a statistically correct HSS scheme for NC^1. While such schemes were later constructed from assumptions such as DCR [OSY21] or class groups [ADOS22], they were not known (without using indistinguishability obfuscation of FHE-style primitives) at the time of [BGI16b]. Instead, [BGI16b] relied on a DDH-based construction of *Las Vegas* HSS, which satisfies a weaker correctness property. A similar, slightly more complex construction works nonetheless given Las Vegas HSS, by letting the functions g_s encode their output with a suitable low-complexity error correcting code. Combining the efficient valid DR pebbling of layered graphs from Section 3.2.1 with this variant recovers the result of [BGI16b].

Extension 2: single-function HSS. Another extension replaces the HSS by *single-function* HSS, a weaker notion where $\mathsf{HSS.Share}$ must specify in advance the circuit to be computed on the shares. One can also modify the previous construction to work with single-function HSS, by initially distributing $\mathsf{HSS{\cdot}Share}(1^\lambda, K_b, (g_s)_{s \in S})$ for all sets $S \in \mathsf{PG}$ (viewing $(g_s)_{s \in S}$ as a single function that takes as input (K_0, K_1) and outputs $(g_s(K_0, K_1))_{s \in S}$). This increases the total communication to $2 \cdot (n + \mathsf{cost}(\mathsf{PG})) + |\mathsf{PG}| \cdot \mathsf{poly}(\lambda)$, which is still sublinear for layered circuits which are not too "tall-and-skinny" (since there, $|\mathsf{PG}| = O(d/k)$ where d is the circuit depth).
Combining the efficient valid DR pebbling of layered graphs from Section 3.2.1 with this variant, and setting $k \leftarrow \log\log s$, almost recovers the result of [CM21] which achieves sublinear 2-party computation from the super-polynomial hardness of LPN, by building single-function HSS for loglog-depth circuits from superpoly-LPN. A minor distinction is that our construction would require a PRF computable in depth loglog. At a high level, the PRF is used in our construction to turn a (possibly non-compact) HSS into a compact HSS (whose share size on input x is $|x| + \mathsf{poly}(\lambda)$) using a hybrid encryption technique. Instead, the work of [CM21] avoids this additional assumption by directly building a compact single-function HSS from the super-polynomial hardness of LPN. Summing up:

Corollary 2. *Let C be a boolean circuit with m output gates, and let PG be a valid DR pebbling of its graph $G = (V, E)$ of depth k. Then:*

- *if $k = \log |C|$ and assuming the hardness of either DCR or DDH, there exists a secure 2-party protocol for C with communication $2 \cdot (n + \mathsf{cost}(\mathsf{PG})) + \mathsf{poly}(\lambda)$, and*
- *i $k = \log\log |C|$ and assuming the super-polynomial hardness of LPN (semi-honest setting) or additionally the existence of collision-resistant hash functions (malicious setting), there exists a secure 2-party protocol for C with communication $2 \cdot (n + \mathsf{cost}(\mathsf{PG})) + |\mathsf{PG}| \cdot \mathsf{poly}(\lambda)$*

In the corollary above, the statement about security in the malicious setting comes from the existence of a communication-preserving semi-honest to malicious compiler given succinct zero-knowledge arguments (via the GMW compiler [GMW86]), which exists assuming collision-resistant hash functions [Kil92]. The latter is implied by either DDH or DCR, but not by the flavour of LPN used in [CM21] (though strong forms of LPN imply CRHF [YZW+19]).

3.3.3 Further protocols that fit the abstraction. We now list other secure computation protocols whose recasting as protocols computing a circuit through a DR pebbling of its graph captures adequately some of their efficiency properties.

2-party computation from correlated symmetric PIR. The work of [BCM22] achieves sublinear secure computation using a strong form of private information retrieval, called correlated symmetric PIR. In a similar fashion as for the protocols of the previous sections, the protocol of [BCM22] can be described for all circuits given a suitable DR pebbling of their graph with depth parameter $k = c \cdot \log\log|C|$ for an appropriate constant c. The dependency in the parameters of the DR pebbling is slightly more complex, but still translates to a protocol with sublinear communication whenever $\mathsf{cost}(\mathsf{PG}) = o(|C|)$ and the circuit is not too "small and skinny" (which translates to $|\mathsf{PG}|$ being significantly smaller than $\mathsf{cost}(\mathsf{PG})$). The lemma below is the generalization of Corollary 18 in [BCM22] to arbitrary circuits C:

Lemma 4. *Let C be a circuit with n inputs, m outputs, and let PG be a valid DR pebbling of the graph G of C with depth parameter k. Assuming the existence of correlated SPIR, there exists a secure 2-party protocol for C with communication complexity*

$$O\left(n + m + |\mathsf{PG}|^{1/3} \cdot \left(2^{k+2^k} \cdot \mathsf{cost}(\mathsf{PG})\right)^{2/3} \cdot \mathsf{poly}(\lambda) + \mathsf{cost}(\mathsf{PG})\right).$$

Correlated SPIR can be constructed assuming the LPN assumption (with polynomial hardness) and either of QR, DDH, DCR, or LWE [BCM22, BCM23]. This implies secure 2-party computation with the communication complexity outlined above under these combinations of assumptions.

Sublinear multiparty computation. The recent work of [BCM23] introduced an approach for sublinear secure computation which, at a high level, combines correlated SPIR with N-party homomorphic secret sharing to achieve sublinear MPC for $N + 1$ parties. Combining this with existing constructions of 2-party and 4-party HSS, they obtain constructions of sublinear 3-party and 5-party secure computation protocols for layered circuits. Another recent protocol that fits our abstraction is given in the work of [DIJL23], which introduced an N-party homomorphic secret sharing scheme (with imperfect correctness) for any polynomial N and all log log-depth circuits, and derived a sublinear N-party secure computation protocol for all layered graphs. We briefly note that all these approaches also fit our framework, and their protocols can be seen to work identically over any circuit C with a suitable DR pebbling.

FHE-based secure computation with reduced bootstrapping. Until now, we outlined protocols whose *communication complexity* depends on finding a suitable DR pebbling of the graph of the circuit. We now show that this abstraction is also useful beyond this setting. It is well known that fully homomorphic encryption (FHE) [Gen09] implies secure computation of arbitrary function with communication independent of the circuit size. However, evaluating arbitrary circuits involves bootstrapping, which is typically quite expensive.
The work of [BLMZ17] initiated the study of the number of bootstrapping operations required to homomorphically evaluate a circuit. In their abstraction, each FHE ciphertext is associated to a *noise level*, represented as an integer. Evaluating any non-linear gate increase the noise level by 1. When some pre-specified maximum noise level is reach, an expensive bootstrapping must be performed. While one could set the maximum noise level to be above the circuit size to avoid bootstrapping altogether, allowing higher noise levels typically results in much larger ciphertexts and much less efficient homomorphic operations. This suggests the following question studied in [BLMZ17]:

Given a maximum noise level k and a circuit C, how many bootstrappings are required to homomorphically compute C?

Concretely, the model is as follows: fix a maximum noise level k. All inputs are encrypted with respect to a noise level 0. At each gate, the output of the gate becomes encrypted with respect to a noise level equal to the maximum noise levels of its inputs (for linear gates) or the maximum noise level of its inputs plus 1 (non-linear gates). If the noise level of an input to a gate is equal to k, a bootstrapping operation must be performed, which resets the noise level to 0.
Let C be a circuit with n inputs, m outputs, and let PG be a valid DR pebbling of the graph G of C with depth parameter k. The following follows almost immediately from the definition of DR pebblings:

Lemma 5. *The circuit C can be homomorphically evaluated with FHE ciphertexts of maximum noise level k using at most $\mathsf{cost}(\mathsf{PG})$ bootstrapping operations.*

Proof. The proof is straightforward: the homomorphic evaluation runs a bootstrapping evaluation at each gate where a pebble is placed during the game. Because all path of length k ending in a pebbled gate are guaranteed to contain a pebbled node already (which implies that the ciphertext encrypting the output of the node has noise level 0 because a bootstrapping was performed), the noise level of the ciphertexts encrypting the inputs to the gate is at most k. □

Remark 2. The size $\mathsf{cost}(\mathsf{PG})$ of a DR pebbling of the underlying digraph of the circuit C yields an upper bound on the number of bootstrapping operations required to homomorphically evaluate a circuit. We note that, since this measure depends solely of the graph of the circuit, it is agnostic of the type of gates. On the downside, this means that it does not take advantage of the fact that addition gates are typically "for free" in FHE schemes. On the positive sides, it yields an upper bound that holds for homomorphic evaluation of boolean circuits over an arbitrary boolean basis.

We note that our result is not directly comparable to the result of [BLMZ17]: their work showed that closely approximating the minimal number of bootstrapping is NP-hard, and provided a polytime k-approximation of the best solution. However, their result does not provide any bound on the size of the best possible solution. In contrast, we provide an upper bound on the number of bootstrapping required for *any* boolean circuit, as a function of its DR pebbling complexity. Looking ahead, combined with the non-trivial DR pebbling algorithms which we introduce in the next section, this will yield algorithms to homomorphically evaluate low-depth circuits with a sublinear number of bootstrapping operations, and algorithms to homomorphically evaluate any circuit using bootstrapping for a constant fraction of all gates.

The complexity of OT-based secure computation. An 1-out-of-n oblivious transfer (OT) is a protocol that allows a sender holding inputs $(s_1, \cdots, s_n)$ to reveal s_i to a receiver with input $i \in \{1, \cdots, n\}$ without learning i, and without revealing any s_j for $j \neq i$ to the receiver. The seminal GMW protocol [GMW87] showed that any circuit can be securely evaluated (in the 2-party setting) using a 1-out-of-4 oblivious transfer protocol. Informally, the 1-out-of-4 OT is used to let one party obliviously retrieve its share from the truth table (of size 4) of a binary gate (scrambled with the masks held by the other party). This approach generalizes immediately to securely computing circuits with k-ary gates using 1-out-of-2^k oblivious transfer. This suggests the following natural question: given a circuit C and a 1-out-of-n oblivious transfer protocol, how many invocations of the OT are necessary to securely evaluate C?
For $n = 2$, the OT complexity of secure computation was previously studied in [BIKK14]. In this section, we observe that for general values of n, the protocols of [GMW87, DNNR17] immediately yield an information-theoretic protocol given access to 1-out-of-n OT functionality for circuits with n-ary gates. Now, given an efficient DR pebbling with depth parameter k of the graph of a circuit C, observe that the value of each pebbled node v can be computed as a function of the value of at most 2^k pebbled ancestors of v (since all path of length k ending in v must contain a pebbled node, and there are at most 2^k such paths). In turn, this implies that the computation of v from its pebbled ancestors can be viewed as a 2^k-ary gate which, using [GMW87, DNNR17], can be evaluated with one call to a 1-out-of-2^{2^k} OT functionality. Summarising, we have the following lemma:

Lemma 6. *Let C be a circuit with n inputs, m outputs, and let PG be a valid DR pebbling of the graph G of C with depth parameter k. There exists an information-theoretic secure 2-party protocol for C in the $\binom{2^{2^k}}{1}$-OT-hybrid model which makes at most $m + \mathsf{cost}(\mathsf{PG})$ to the OT functionality, and requires no further communication.*

4 Depth-Reduction Algorithms for Fan-in Two Circuits

In this section, we present depth-reduction algorithms for (the underlying DAG of) fan-in 2 circuits.

In Section 4.1, we provide a conservative depth-reduction algorithm for *any* fan-in two circuit which removes only a *sublinear* number of nodes. However, because the reduction in depth is only sub-polynomial, we can only reach (doubly) logarithmic depth if the starting circuit is already shallow.

In Section 4.2 we provide extreme depth-reduction algorithms, reducing *any* in-degree-2 circuit's depth to a constant, while removing a constant fraction of the vertices.

In Section 4.3 we exclude the existence of a "best of both worlds" result, by establishing there are high-depth circuits whose depth cannot be reduced polynomially without removing a linear number of nodes.

In Section 4.4 we list the implications for secure multiparty computation.

4.1 Depth-Reduction of Low-Depth Circuits

Valiant [Val77, Theorem 5.1] (recalled in this section as Theorem 1) established that the depth of *any* circuit can be reduced sub-polynomially (*i.e.* the depth goes from d to $d^{1-o(1)}$, thereby saving the sub-polynomial factor $d^{o(1)}$) by the removal of only a sublinear number of vertices.

Theorem 1 (Subpolynomial depth-reduction for all circuits, Immediate Corollary of [Val77, Theorem 5.1]). *Let G be an in-degree-2, depth-d, n-vertex DAG. For every $k \leq d$, there exists a subset of $\mathcal{O}(n \cdot (1 - \frac{\log k}{\log d}))$ vertices whose removal yields a depth-k DAG.*

Proof. [Val77, Theorem 5.1] states that the smallest in-degree-2, depth-d DAG whose depth cannot be reduced to k by removing ℓ *edges* has order at least $(\ell \cdot \log d)/(\log(d/k))$. It follows that every in-degree-2, depth-d DAG on n vertices can have its depth reduced to k by removing ℓ *edges* if the following inequality holds: $n \leq (\ell \cdot \log d)/(\log(d/k))$. By setting $\ell \leftarrow n \cdot (1 - \frac{\log k}{\log d})$ and noting that removing a giving set of k edges can also be done by removing k nodes, we get the desired result. □

To better understand the trade-off between depth-reduction and number of nodes removed in Theorem 1, it may be instructive to introduce the variable change $\kappa \leftarrow \log(d/k)$ and observe that the theorem can be restated as:

> Let G be an in-degree-2, depth-d, n-vertex DAG. For every $\kappa \leq \log d$, there exists a subset of $\mathcal{O}(n \cdot \kappa / \log d)$ vertices whose removal yields a depth-$(d/2^{\kappa})$ DAG.

It should now be apparent that if we are only willing to remove $o(n)$ nodes, then we need to set $\kappa = o(\log d)$, which means that the depth of the DAG will only be reduced to $d^{1-o(1)}$. If this quantity is to be logarithmic or even doubly logarithmic in n (as is required for some applications of Section 3.3), the result can only be applied to circuits which are already low-depth. We state the result for low-depth circuits in Corollary 3.

Corollary 3 (Depth-reduction of low-depth circuits, Adapted from [Val77, Corollary 5.3]). *Let G be an in-degree-2, n-vertex DAG of depth $\log^{1+o(1)} n$ (resp. $(\log \log n)^{1+o(1)}$). There exists a subset of $o(n)$ vertices whose removal yields a DAG of depth $\mathcal{O}(\log n)$ (resp. $\mathcal{O}(\log \log n)$).*

4.2 Depth-Reduction of General Circuits

In this section we show how removing a constant fraction of the vertices can reduce the depth of an in-degree-2 DAG all the way down to a constant.

4.2.1 Reduction to depth $k = 1$ based on 3-colouring. Our first solution removes a fraction 2/3 of the vertices in order to reduce the depth to 1.

Theorem 2 (Colouring-based depth reduction). *Any in-degree-in 2, n-vertex DAG admits a subset of $\lfloor \frac{2n}{3} \rfloor$ vertices whose removal yields a depth-1 DAG (*i.e. *an independent set).*

Proof. An independent set in a DAG is the same thing as an independent set of the underlying (undirected) graph. Recall that the underlying graph of an in-degree-in 2 DAG is 2-degenerate. A 2-degenerate graph is 3-colourable [Mat68,LW70] and furthermore a 3-colouring can be found greedily in polynomial time: colour vertices following a 2-elimination ordering, always assigning the smallest available colour (by definition of a 2-elimination ordering, whenever we colour a vertex, at most two of its neighbours have already been assigned a colour, so the greedy algorithm will never be stuck and will never need to use more than three colours). The vertices are now partitioned into three colours, and removing the two smallest partitions (this union has size at most $\lfloor 2n/3 \rfloor$) yields an independent set. □

Note that Theorem 2 is tight in the sense that there exist n-vertex in-degree-2 DAGs whose independence number (*i.e.* the size of its maximum independent set) is $n - \lfloor 2n/3 \rfloor$.

4.2.2 Reduction to depth $k \geq 1$ based on Feedback Vertex Set. We now present an alternative solution, which removes a smaller fraction than 2/3 of the vertices, but at the cost of reducing the depth to "only" a constant, not one. The algorithm first removes a feedback vertex set, and then proceeds to remove vertices from the resulting forest.

Depth-reduction of forests. The first observation is that forests can be reduced to depth k by removing a fraction $1/(k+1)$ of its vertices.

Algorithm Depth-Reduction for Forests

On input a directed forest $G = (V, E)$ and an integer k, $\mathsf{DR}_{\mathsf{forest}}(\cdot, \cdot)$ does the following:

1. Let D be the depth of G.
2. If $D \leq k$ return $\varnothing$.
3. Else:
   (a) Let u_0 be a depth-D vertex, and for $i \in [k]$ let u_i be its ancestor at depth $D - i$.
   (b) Return $\{u_k\} \cup \mathsf{DR}_{\mathsf{forest}}(k, V \setminus \{u_i\}_{i \in [0,k]})$.

Fig. 6: Algorithm which, on input an n-vertex directed forest and an integer k, produces a set of at most $\lfloor \frac{n}{k+1} \rfloor$ vertices whose removal yields a DAG of depth at most k.

Lemma 7 (Depth-reduction algorithm for directed forests). *Let $k \in \mathbb{N}^\star$. Every n-vertex directed forest admits a set of $\lfloor \frac{n}{k+1} \rfloor$ vertices whose removal yields a depth-k DAG. Furthermore the (deterministic) algorithm of Figure 6 finds such a set in polynomial time.*

Proof. The fact that $\mathsf{DR}_{\mathsf{forest}}$ runs in polynomial-time follows from inspection. Let us show by induction on $n \in \mathbb{N}^\star$ that for every n-vertex directed forest $G = (V, E)$ and every integer $k \geq 1$, $\mathsf{DR}_{\mathsf{forest}}(G, k)$ outputs a set $S \subseteq V$ of size at most $\lfloor \frac{n}{k+1} \rfloor$ such that $G[V \setminus S]$ has depth at most k.

- *Initialisation:* Let $k \geq 1$. Any graph G with a single vertex has depth 1,therefore $\mathsf{DR}_{\mathsf{forest}}(G, k)$ therefore returns $\varnothing$, and the claim is true.
- *Induction Step:* Let $n \in \mathbb{N}^\star$, and assume the induction hypothesis is true from ranks 1 to n. Let $G = (V, E)$ be an $(n+1)$-vertex forest and let $k \in \mathbb{N}^\star$. If G has depth at most k, then the claim is trivially true. If G has depth more than k, then in particular $n \geq k+1$. Furthermore, by induction hypothesis $\mathsf{DR}_{\mathsf{forest}}(G, k)$ outputs a set of size at most $1 + \lfloor \frac{n-(k+1)}{k+1} \rfloor = \lfloor \frac{n}{k+1} \rfloor$.

□

Note that the algorithm of Figure 6 is optimal in the sense that for n-vertex directed paths, the smallest set whose removal yields a graph of depth at most k has size $\lfloor \frac{n}{k+1} \rfloor$.

From forest depth-reduction to circuit depth-reduction. The second observation is that finding an FVS reduces the depth-reduction problem from in-degree-2 DAGs to directed forests.

Lemma 8 (FVS-based depth reduction). *Let G be an n-vertex DAG, and denote G' the underlying (undirected) graph of G. If G' has a feedback vertex set of size f, then G admits a set of $\lfloor \frac{n-f}{k+1} \rfloor$ vertices whose removal yields a depth-k DAG.*

Proof. By definition of a feedback vertex set, removing a size-f FVS from G' yields an $(n-f)$-vertex forest. Evidently, removing the same size-f vertex set from G yields an $(n-f)$-vertex directed forest. The desired result follows from applying Lemma 7: removing an additional $\lfloor \frac{n-f}{k+1} \rfloor$ vertices from G yields a graph of depth at most k. □

Lemma 9 (A feedback vertex set for all 2-degenerate graphs, [BDBS14, Theorems 2,3]). *Every 2-degenerate (undirected) graph on n vertices admits a feedback vertex set of size at most $\lfloor 2n/5 \rfloor$, and furthermore such an FVS can be found in polynomial time.*

Note that Lemma 9 is tight in the sense that there exist n-vertex graphs whose smallest feedback vertex set has size $\lfloor 2n/5 \rfloor$ [BDBS14, Theorem 4].

Wrapping-up. We are now ready to conclude by combining lemmata 8 and 9.

Theorem 3 (FVS-based depth reduction). *Let $k \geq 1$. Any in-degree-2, n-vertex DAG admits a subset of $\frac{2n}{5} \cdot (1 + \frac{3/2}{k+1})$ vertices whose removal yields a depth-k di-graph.*

Proof. Let G be an in-degree-2, n-vertex DAG. The underlying (undirected) graph of G is 2-degenerate, therefore by combining lemmata 8 and 9, G admits a subset of $\lfloor \frac{2n}{5} \rfloor + \lfloor \frac{n-\lfloor 2n/5 \rfloor}{k+1} \rfloor$ vertices whose removal yields a depth-k graph. Since $(\frac{2n}{5} \cdot (1 + \frac{3/2}{k+1})) - (\lfloor \frac{2n}{5} \rfloor + \lfloor \frac{n-\lfloor 2n/5 \rfloor}{k+1} \rfloor) \geq (\frac{2n}{5} \cdot (1 + \frac{3/2}{k+1})) - (\lfloor \frac{2n}{5} \rfloor + \frac{n-\lfloor 2n/5 \rfloor}{k+1}) = (\frac{2n}{5} - \lfloor \frac{2n}{5} \rfloor) \cdot \frac{k}{k+1} \geq 0$, the simplified expression stated in the theorem is also correct. □

4.3 Lower Bounds on Depth-Reduction

The depth-reduction algorithm of Section 4.1 is better in the sense it removes only a sublinear number of nodes, while those of Section 4.2 are better in the sense they drastically reduce the depth of the circuit. One may wonder if it is possible to improve on this result, and reduce the depth of any circuit polynomially (*i.e.* from d to d^ϵ for some constant $0 \leq \epsilon < 1$) while still only removing a sublinear number of vertices. Unfortunately, Schnitger [Sch83, Theorem A] showed that the sub-polynomial limitation on depth-reduction was inherent by producing family of *constant fan-in* circuits whose depth cannot be reduced polynomially without removing a linear fraction of vertices. Alwen, Blocki, and Pietrzak [ABP17] later introduced a technique to reduce a circuit family's fan-in from δ to 2 while preserving its depth-robustness up to a factor δ. We state the combined result in Theorem 4.

Theorem 4 (Polynomial depth-reduction requires removing a linear number of vertices in some graphs, Combination of [Sch83, Theorem A] and [ABP17, Lemma 1]). *For each ϵ ($0 \leq \epsilon < 1$), there is a family of in-degree-2 DAGs $(G_{n,\epsilon})_{n \in \mathbb{N}^\star}$ such that $G_{n,\epsilon}$ has $\mathcal{O}(n)$ vertices, but $\Omega(n)$ vertices have to be removed to reduce its depth to $(n/\log n)^\epsilon$.*

Proof. Let $\epsilon \in [0, 1)$. By [Sch83, Theorem A], there is a family of *constant* in-degree DAGs $(\tilde{G}_{n,\epsilon})_{n \in \mathbb{N}^\star}$ such that $\tilde{G}_{n,\epsilon}$ has n vertices, but $\Omega(n)$ vertices have to be removed to reduce its depth to $(n/\log n)^\epsilon$. Let δ be the in-degree of $(\tilde{G}_{n,\epsilon})_{n \in \mathbb{N}^\star}$. Applying [ABP17, Lemma 1] (with $\gamma = 1$) yields the desired result. □

4.4 Applications to Cryptography

We conclude by combining the results of Section 3.3, which re-casts cryptographic results in the lens of our pebbling game, and the depth-reduction algorithms of Sections 4.1 and 4.2.
By combining Lemma 2 and theorem 2 we obtain Corollary 4.

Corollary 4 (Fractionally linear-communication MPC in the correlated randomness model, Concrete Result). *Let C be an n-input, m-output, depth-d, $\mathbb{F}$-arithmetic circuit with s non-output computation gates. There exists a passive, perfectly secure N-party protocol for securely computing C in the correlated randomness model, using the following resources (in bits):*

- Correlated randomness per party: $((11 + 2/3) \cdot s + n + 16 \cdot m) \cdot N \cdot \log |\mathbb{F}|$
- Total communication: $(\frac{2}{3}s + n + m) \cdot N \cdot \log |\mathbb{F}|$
- Local computation per party: $\mathcal{O}((\frac{2s}{3} + m) \cdot (N + \log |\mathbb{F}|))$

If instead we combine Lemma 2 and theorem 3 we obtain Corollary 5.

Corollary 5 (Fractionally linear-communication MPC in the correlated randomness model, Asymptotic Result). *Let C be an n-input, m-output, depth-d, $\mathbb{F}$-arithmetic circuit with s non-output computation gates. For every $\epsilon \geq \frac{3}{2 \log \log s}$, there exists a passive, perfectly secure N-party protocol for securely computing C in the correlated randomness model, using the following resources (in bits):*

- Correlated randomness per party: $n + 2^{\sqrt{8}^{1/\epsilon}}(s + m)N \cdot \log |\mathbb{F}|$
- Total communication: $((1 + \epsilon)\frac{2}{5}s + n + m) \cdot N \cdot \log |\mathbb{F}|$
- Local computation per party: $\mathcal{O}(2^{\sqrt{8}^{1/\epsilon}}((1 + \epsilon)\frac{2}{5}s + m) \cdot (N + \log^2 |\mathbb{F}|))$

By combining Lemma 3 and theorem 1 we get Corollary 6. To clarify, the quantification in Corollary 6 is as follows: for every infinite family of circuits $(C_\lambda)_{\lambda \in \mathbb{N}}$ of size $s = s(\lambda)$ such that there exists a vanishing function $\alpha(\cdot) \in o(1)$ such that the depth of C_λ is at most $\log^{1+\alpha(s(\lambda))}(s(\lambda))$ (resp. $[\log \log(s(\lambda))]^{1+\alpha(s(\lambda))}$), there exists a protocol for securely computing C_λ assuming HSS support logarithmic (resp. doubly logarithmic) depth homomorphic evaluations.

Corollary 6 (Sub-polynomially deeper HSS-based sublinear-communication secure computation). *Assuming the existence of N-party homomorphic secret-sharing supporting logarithmic depth (respectively doubly logarithmic depth circuits) $\mathbb{F}$-arithmetic fan-in two circuits, there exists sublinear-communication secure N-party computation for* all $\log^{1+o(1)}$*-depth (resp.* $(\log \log)^{1+o(1)}$*-depth) circuits.*

By combining Lemma 5 and theorem 1 we obtain Corollary 7, establishing an upper bound on the number of bootstraps required in FHE, given a maximum noise level.

Corollary 7 (FHE-based secure computation with reduced bootstrapping, informal). *Given an FHE scheme tolerating a maximum noise level of L, only $\frac{2}{5}s \cdot (1 + \frac{1.5}{L+1})$ bootstraps are required to homomorphically evaluate a size-s circuit (over an arbitrary basis of binary gates).*

Corollary 8 (Upper bounds on the 1-out-of-M OT-complexity of secure multiparty computation). *Let $f \colon \{0,1\}^n \to \{0,1\}$ be a boolean function. For every $M \geq 2$, there is a two-party protocol for passively securely computing f (we assume the parties initially hold shares of the inputs, but this captures the case where each input is held by one of the parties) with perfect security in the $\binom{M}{1}$-OT hybrid model while making*

$$1 + \frac{2}{5}|f| \cdot \left(1 + \frac{3/2}{1 + \lceil \log \log M \rceil}\right)$$

calls to the OT functionality, where $|f|$ is the computational complexity of f with respect to the basis of all binary boolean gates.

Acknowledgments

We thank Mikaël Rabie and Elette Boyle for helpful discussions and pointers to respectively degenerate graphs and memory-hard functions.
Geoffroy Couteau was supported by the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE39-0001 (project SCENE), and by the France 2030 ANR Project ANR22-PECY-003 SecureCompute. Pierre Meyer was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreements numbers 852952 (HSS) and 803096 (SPEC).

References

AB16. Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 241–271. Springer, Heidelberg, August 2016.

ABH17. Joël Alwen, Jeremiah Blocki, and Ben Harsha. Practical graphs for optimal side-channel resistant memory-hard functions. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1001–1017. ACM Press, October / November 2017.

ABP17. Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 3–32. Springer, Heidelberg, April / May 2017.

ADOS22. Damiano Abram, Ivan Damgård, Claudio Orlandi, and Peter Scholl. An algebraic framework for silent preprocessing with trustless setup and active security. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 421–452. Springer, Heidelberg, August 2022.

AJL+12. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

ARS24. Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. To appear at EUROCRYPT 2024, 2024.

BCM22. Elette Boyle, Geoffroy Couteau, and Pierre Meyer. Sublinear secure computation from new assumptions. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 121–150. Springer, Heidelberg, November 2022.

BCM23. Elette Boyle, Geoffroy Couteau, and Pierre Meyer. Sublinear-communication secure multiparty computation does not require FHE. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 159–189. Springer, Heidelberg, April 2023.

BDBS14. Mieczysław Borowiecki, Ewa Drgas-Burchardt, and Elżbieta Sidorowicz. A feedback vertex set of 2-degenerate graphs. *Theoretical Computer Science*, 557:50–58, 2014.

Bea92. Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Heidelberg, August 1992.

BFKR91. Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 62–76. Springer, Heidelberg, August 1991.

BGI16a. Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016.

BGI16b. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

BIKK14. Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 317–342. Springer, Heidelberg, February 2014.

BKKS11. Boštjan Brešar, František Kardoš, Ján Katrenič, and Gabriel Semanišin. Minimum k-path vertex cover. *Discrete Applied Mathematics*, 159(12):1189–1195, 2011.

BKS19. Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.

BLMZ17. Fabrice Benhamouda, Tancrède Lepoint, Claire Mathieu, and Hang Zhou. Optimization of bootstrapping in circuits. In Philip N. Klein, editor, *28th SODA*, pages 2423–2433. ACM-SIAM, January 2017.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

CCD88. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

CE91. Marek Chrobak and David Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theoretical Computer Science*, 86(2):243–266, 1991.

CM21. Geoffroy Couteau and Pierre Meyer. Breaking the circuit size barrier for secure computation under quasi-polynomial LPN. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 842–870. Springer, Heidelberg, October 2021.

COS+22. Ilaria Chillotti, Emmanuela Orsini, Peter Scholl, Nigel Paul Smart, and Barry Van Leeuwen. Scooby: Improved multi-party homomorphic secret sharing based on FHE. *SCN 2022*, 2022. `https://eprint.iacr.org/2022/862`.

Cou19. Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 473–503. Springer, Heidelberg, May 2019.

DFH12. Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 54–74. Springer, Heidelberg, March 2012.

DIJL23. Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 315–348. Springer, Heidelberg, August 2023.

DNNR17. Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 167–187. Springer, Heidelberg, August 2017.

EH66. P. Erdös and A. Hajnal. On chromatic number of graphs and set-systems. *Acta Mathematica Academiae Scientiarum Hungarica*, 17:61–99, 1966.

FGJS17. Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from paillier encryption. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *ProvSec 2017*, volume 10592 of *LNCS*, pages 381–399. Springer, Heidelberg, October 2017.

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

GJ11. Anna Gál and Jing-Tang Jang. The size and depth of layered boolean circuits. *Information Processing Letters*, 111(5):213–217, 2011.

GMW86. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pages 174–187. IEEE Computer Society Press, October 1986.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

IKM+13. Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 600–620. Springer, Heidelberg, March 2013.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

LP13. Tancrède Lepoint and Pascal Paillier. On the minimal number of bootstrappings in homomorphic circuits. In Andrew A. Adams, Michael Brenner, and Matthew Smith, editors, *Financial Cryptography and Data Security*, pages 189–200, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

LW70. Don R. Lick and Arthur T. White. k-degenerate graphs. *Canadian Journal of Mathematics*, 22(5):1082–1096, 1970.

Mat68. David W. Matula. A min-max theorem for graphs with application to graph coloring. *SIAM 1968 National Meeting, SIAM Review*, 10(4):481–482, 1968.

Nau09. Uwe Naumann. Dag reversal is np-complete. *Journal of Discrete Algorithms*, 7(4):402–410, 2009.

OSY21. Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 678–708. Springer, Heidelberg, October 2021.

PV16. Marie Paindavoine and Bastien Vialla. Minimizing the number of bootstrappings in fully homomorphic encryption. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 25–43. Springer, Heidelberg, August 2016.

RS21. Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 687–717, Virtual Event, August 2021. Springer, Heidelberg.

Sch83. Georg Schnitger. On depth-reduction and grates. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 323–328, 1983.

Tu22. Jianhua Tu. A survey on the k-path vertex cover problem. *Axioms*, 11(5), 2022.

Val77. Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In Jozef Gruska, editor, *Mathematical Foundations of Computer Science 1977*, pages 162–176, Berlin, Heidelberg, 1977. Springer Berlin Heidelberg.

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

YZW+19. Yu Yu, Jiang Zhang, Jian Weng, Chun Guo, and Xiangxue Li. Collision resistant hashing from sub-exponential learning parity with noise. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 3–24. Springer, Heidelberg, December 2019.