Randomness in Private Sequential Stateless Protocols

Hari Krishnan P. Anilkumar[1] *, Varun Narayanan[2] **, Manoj Prabhakaran[3] ***, and Vinod M. Prabhakaran[1] *

[1] TIFR, Mumbai, India {hari.a,vinodmp}@tifr.res.in
[2] UCLA, Los Angeles, USA varunnkv@gmail.com
[3] IIT Bombay, Mumbai, India mp@cse.iitb.ac.in

Abstract. A significant body of work in information-theoretic cryptography has been devoted to the fundamental problem of understanding the power of randomness in private computation. This has included both in-depth study of the randomness complexity of specific functions (e.g., Couteau and Rosén, ASIACRYPT 2022, gives an upper bound of 6 for n-party AND), and results for broad classes of functions (e.g., Kushilevitz et al. STOC 1996, gives an $O(1)$ upper bound for all functions with linear-sized circuits). In this work, we make further progress on both fronts by studying randomness complexity in a new simple model of secure computation called Private Sequential Stateless (PSS) model.

We show that functions with $O(1)$ randomness complexity in the PSS model are *exactly* those with constant-width branching programs, restricting to "speak-constant-times" protocols and to "read-constant-times" branching programs.

Towards this our main construction is a novel PSS protocol for "strongly regular branching programs" (SRBP). As we show, any constant-width branching program can be converted to a constant-width SRBP, yielding one side of our characterization. The converse direction uses ideas from Kushilevitz et al. to translate randomness to communication.

Our protocols are concretely efficient, has a simple structure, covers the broad class of functions with small-width, read-once (or read-a-few-times) branching programs, and hence may be of practical interest when 1-privacy is considered adequate. Also, as a consequence of our general result for SRBPs, we obtain an improvement over the protocol of Couteau and Rosén for ANDin certain cases — not in terms of the number of bits of randomness, but in terms of a simpler protocol structure (sequential, stateless).

Keywords - Randomness complexity; Private computation; Private Sequential Stateless model; lower bound; branching programs.

---

* Hari Krishnan P. Anilkumar and Vinod M. Prabhakaran were supported by Department of Atomic Energy, Government of India, under project no. RTI4001.

** Varun Narayanan was supported by NSF Grants CNS-2246355, CCF-2220450, and CNS-2001096.

*** Manoj Prabhakaran was supported by IIT Bombay Trust Lab and Algorand Centre of Excellence by Algorand Foundation.

1 Introduction

In information-theoretic cryptography, randomness is the key resource available for creating secrecy (unlike in computational cryptography, where computational hardness plays an equally important role). As such, understanding the amount of randomness needed for various tasks is a problem of great theoretical and practical importance in this area.

In this paper we study randomness complexity in the context of private multi-party computation (i.e., multi-party computation secure against honest-but-curious adversary). Not surprisingly, this fundamental question has received a great amount of attention in the literature [KR94, KOR96, KM96, KOR98, BSPV99, BGP99, GR03, JLR03, BGP07, DPP16, RU19, KOP+19, CR22, GIS22].

We shall focus on the most basic setting of 1-privacy, wherein the adversary can corrupt only a single party, which itself has been studied in depth. In one of the early works in this line, it was shown that for any function which has a linear sized circuit, the the randomness complexity of 1-private computation is constant, irrespective of the number of parties holding the inputs. Several subsequent works improved on the exact constant when considering a specific function, namely the n-party AND function [KOP+19, CR22].

We introduce a simplified model of multi-party computation called the **Private Sequential Stateless** (PSS) model that is intrinsically connected to randomness complexity. Various simplified models of MPC, like *Private Simultaneous Message* (PSM) model [FKN94, IK97], *Non-Interactive Secure Computation* (NISC) [IKO+11], *Conditional Disclosure of Secrets* (CDS) [GIKM98], etc., have proven influential in shaping our understanding of private computation and information-theoretic cryptography. We propose PSS in a similar spirit, but it deals with a different aspect of complexity. While PSM, NISC, CDS all impose a "star" topology of communication, PSS considers a *chain* model of communication. In PSS, given correlated randomness, the parties communicate in a (pre-determined) sequence from one party to the next; the parties do not retain any state between rounds, except for their own input and their share of the correlated randomness.

The randomness cost of a PSS protocol – namely, the number of random bits used to prepare the correlated randomness for the parties – is a crucial factor that determines whether a function has such a protocol or not. Indeed, without any restriction on the amount of randomness used, the restrictions in the PSS model can be subverted.[4] Our focus will be on PSS protocols which have a constant randomness cost for a family of functions (with variable input length), independent of the number of parties (each with a bit of the input). Also, unless

[4] Statelessness can be subverted by each party sending out an encryption (using one-time pads) of its state as part of the communication, which will be forwarded as it is until the party is again invoked (at which point it can update the contents of the state and re-encrypt using a new one-time pad). Once state is allowed, the sequentiality requirement can be subverted by letting a sender communicate to many parties one-by-one over several rounds.

otherwise specified, we shall also restrict to *speak-constant-times* PSS protocols, in which the number of times each party speaks is at most a constant.

Our main result is an *exact characterization* of boolean functions with *constant-randomness speak-constant-times PSS protocols* in terms of Branching Programs: they are exactly those functions which have *constant-width read-constant-times branching programs* (read-constant-times refers to the condition that each input is used at most a constant number of times in evaluating the branching program). Towards proving this result we present a PSS protocol for branching programs, and also, conversely, show how to convert a PSS protocol into a branching program, respecting the cost constraints.

Our contribution can alternately be viewed as developing concretely efficient PSS protocols for a large class of interesting functions. This class includes functions like AND, inner product, and any boolean function which can be computed by a streaming algorithm with constant memory and a constant number of passes over the input sequence.[5] The converse demonstrates an optimality of this result – that one could not have constructed such protocols for a wider class of functions.

Along the way to constructing our protocol, we identify a class of branching programs that we term *strongly regular branching programs* (SRBP). Strong regularity captures the technical conditions necessary for our protocol construction to be private. While some functions naturally have branching programs that are strongly regular, we observe that any branching program can be converted into an SRBP with a polynomial blow-up in the width (keeping the other size parameters intact). SRBPs may be of independent interest as they are a restriction of (a natural generalization of) "regular branching programs" as studied in [LPV23].[6]

For the converse, we rely on a lemma from [KOR96], who proved a similar characterization of functions with constant randomness complexity for 1-private computation in terms of circuit complexity, and adapt it to the setting of PSS protocols.

Finally, as a related result of interest, we also obtain a lower bound for the 3-party AND function. While the best known lower bound result shows that 1 bit of randomness is insufficient [KOP$^+$19], we show that at least 3 bits are necessary for computing this function (even without the sequentiality and memorylessness restrictions). Our result uses a reduction from a recently introduced problem called 3-secret sharing [ARN$^+$23] instantiated for a suitable set of secrets, for which the exact randomness complexity was determined to be 3 bits.

1.1 Our Results

We briefly list our contributions, and expand on them below.

[5] Understanding the exact computational power of constant-width read-m-times branching programs is an interesting problem in its own right.

[6] The regular branching programs defined in [LPV23] can be termed 2-regular; the generalization referred to here allows d-regularity for any $d \geq 2$. Strong regularity imposes additional requirements on top of d-regularity. See Section 5 for more details.

– We introduce a simple model of private protocols with pre-processed correlated randomness, called *Private Sequential Stateless* (PSS) model.
– Our main result is to show a tight connection between functions computable using PSS and Branching Programs, in both directions.
– As an intermediate step, we identify a new model of branching programs called *Strongly Regular Branching Programs* (SRBP) which may be of independent interest.
– We show how our PSS protocols can be adapted to an "unassisted" setting without correlated randomness. While this result applies to a broad class of functions, applying it to the function AND (which has received a significant amount of attention in the literature) yields results matching or closely matching the state-of-the-art results [CR22], but with a simpler protocol structure.
– Continuing to focus on AND, we present a new lower bound on the randomness complexity of 3-party AND for 1-private computation (even with correlated randomness).

In a PSS protocol, the parties are deterministic and stateless, except for the correlated randomness and the input that they receive at the beginning. At every round a single pre-determined party receives a message and then sends a message in the next round to a single other party, without updating its state. The last message in the protocol is sent to a special output party who produces the final output (using its share of the correlated randomness along with the message it received). A PSS protocol is defined to be a 1-private protocol – i.e., with information theoretic security against semi-honest corruption of one party. Two costs of interest to us in a PSS protocol are the number of times any party speaks in the protocol and the amount of randomness used in the protocol, that is, the number of random bits used to prepare the correlated randomness for the parties.

We show a close connection between functions computable using Private Sequential Stateless and Branching Programs.

Theorem 1 (Main Result (informal)). *An n-input boolean function has a speak-constant-times constant-randomness PSS protocol iff it has a read-constant-times constant-width branching program.*

To construct a PSS protocol from a branching program, first we convert it, if necessary, into a *strongly regular branching program* (SRBP), with a polynomial blow-up in the width (keeping the other parameters intact). SRBP is a new definition we introduce, which may be of independent interest (see Section 2.3 and Section 5 for more details). The upper bound on randomness complexity that we obtain depends on the width of the strongly regular branching program and the number of times each input is read while evaluating the branching program – *but not on the number of inputs* or the length of the branching program.

To build a PSS protocol from SRBP, we start by considering a read-once strongly regular branching program (abbreviated as 1-SRBP), in which each input is used for only one transition in the branching program. We obtain the following result:

Theorem 2. *The randomness complexity of PSS computation of n-input boolean functions which have 1-SRBPs of width w, is $O(w \log w)$. This is achieved by a speak-once PSS protocol.*

To prove the theorem, we present a concretely efficient protocol, which requires sampling just 4 uniformly random permutations over $[w]$ and 2 uniform bits in addition. This is comparable to the state-of-the-art concrete parameters obtained in prior work on randomness complexity of 1-private computation for a specific function, namely AND (which has a width-2 1-SRBP). The prior work considered a model without the restrictions of PSS model, but also without external parties to supply correlated randomness and produce the final output. To fairly compare to those results (since additional parties without input *can* help save on randomness), we show how our PSS protocol can be modified so that the pre-processing computation and the final output computation can be carried out by two of the parties with inputs. As shown in Section 6, the modified PSS protocol we obtain for read-once AND uses 6 or 9 bits of randomness (depending on odd or even number of inputs).

We generalize Theorem 2 to functions with an k-SRBP, in which each input is used for at most k transitions in the branching program.

Theorem 3. *The randomness complexity of PSS computation of n-input boolean functions which have k-SRBPs of width w, is $O(kw \log w)$. This is achieved by a speak-$(2k-1)$-times PSS protocol.*

We remark that the $O(k)$ factor in the result above is in fact an upper bound for the chromatic number of a "conflict graph" associated with the branching program; for specific branching programs, this factor could be lower.

Next, we show that one cannot hope to improve this result significantly in terms of the functions covered. That is, we show that constant-randomness PSS protocols exist only for functions which are computable using constant-width branching programs.

Theorem 4. *Boolean functions with constant-randomness speak-k-times PSS protocols have constant-width read-k-times branching programs.*

Note that a sequential protocol with constant communication can be naturally translated to a constant-width branching program. When the protocol is 1-private, randomness cost can be translated to communication cost, as was first shown in [KOR96]. We adapt this result to the setting of PSS protocols to establish our result above.

Finally, on the lower bound front, we obtain the following result for the 3-party AND functionality.

Theorem 5. *Randomness complexity of 1-private computation of $\mathcal{F}^*_{\mathsf{AND}}$ for 3 parties is at least 3 bits.*

1.2 Related Work

As mentioned above, the fundamental question of randomness complexity of secure computation has received much attention. Among these, Kushilevitz, Ostrovsky and Rosén [KOR96] obtained a result analogous to our main result, but in the context of circuits and unrestricted protocols (as opposed to branching programs and PSS protocols).

A line of works have focused on upper bounding the randomness complexity of AND [KOR96, KOP+19, CR22]. Some of these protocols are in the PSS model (or rather, the unassisted PSS model, as defined in Section 6.1), and forms the basic motivation for studying this model. However, the state-of-the-art results in [CR22] are not (which, we show, can be attained in the unassisted PSS model, when the number of parties is odd).

Lower bound results for randomness complexity of 1-private computation have been fewer, with the notable exception of [KOP+19]. Our lower bound result relies on information-theoretic techniques from [DPP16, ARN+23].

Branching programs are a well-studied model of computation [INW94, Bar86]. In particular, a notion of regular branching programs studied in [LPV23] is closely related to the notion of strongly regular branching programs we introduce.

While we have focused on 1-privacy, the question of t-privacy has also been studied in the literature [KM96, CKOR00, GIS22]. We leave it for future work to study the power of t-Private Sequential Stateless model.

2 Technical Overview

We build 1-private sequential stateless protocols for n-party functions that are computable using a family of branching programs called strongly regular branching programs (SRBPs). We first construct a PSS protocol for read-once SRBP (1-SRBP). We will then elaborate on the notion of strong regularity, and present a conversion from a general branching program to strongly regular branching program with a polynomial blow-up in width. A more complex protocol is then presented that realizes PSS for general (read-m) SRBP. As an application of our main result, we construct a 1-private protocol with a limited communication pattern for standard n-party AND functionality by modifying the PSS protocol for AND. Our construction matches the best known randomness cost for AND for odd values of n.

2.1 Branching Programs and Private Sequential Stateless Protocols.

A width-w and length-ℓ branching program for an n input (alternatively n-party) function $f : \{0,1\}^n \to \{0,1\}$ is described by a layer assignment function $\sigma : [\ell] \to [n]$ mapping each layer $i \in [\ell]$ to a party $\mathsf{P}_{\sigma(i)}$; for each layer $i \in [\ell]$, a pair of transition functions $g_0^{(i)}, g_1^{(i)} : [w] \to [w]$ one of which will be chosen according to the input of $\mathsf{P}_{\sigma(i)}$ to map the incoming state to layer i to its outgoing state; and an output function ϕ that maps the final state outgoing from layer ℓ

to an output bit. The branching program computes an n-party function f if, for all $(x_1, \ldots, x_n) \in \{0,1\}^n$, $f(x_1, \ldots, x_n) = \phi(u_\ell)$, where, for $i \in [\ell]$, the outgoing state u_i from layer i (which is the incoming state for layer $i+1$) is defined as follows: $u_i = g^{(i)}_{x_{\sigma(i)}}(u_{i-1})$ and $u_0 = 1$.

If every party is assigned at most m layers, i.e., $|\sigma^{-1}(i)| \leq m$ for each i, the branching program is said to be read-m. A branching program is said to be read-once if $\ell = n$ and σ is the identity function (without loss of generality).

In this paper, we draw a connection between branching programs and a simplified model of secure multi-party computation called the private sequential stateless (PSS) protocols. A PSS protocol $\pi = (\mathsf{Prep}_\pi, \varsigma_\pi, \mathsf{Next}_\pi, \mathsf{Out}_\pi)$ is a semi-honest 1-private protocol in the correlated randomness setting with sequential communication and stateless computation. A set of parties $\mathsf{P}_i, i \in [n]$, each P_i holding an input bit x_i want to deliver a boolean $f(x_1, \ldots, x_n)$ to an external party P_{n+1} who has no input. π starts off with a preprocessing step Prep_π in which a trusted party samples correlated randomness $(r_1, \ldots, r_{n+1})$ (independent of the inputs), and delivers r_i to each $\mathsf{P}_i, i \in [n]$ and r_{n+1} to P_{n+1}. The protocol then proceeds sequentially for T rounds, with $\mathsf{P}_{\varsigma_\pi(t)}$ sending a message m_t to $\mathsf{P}_{\varsigma_\pi(t+1)}$ in round $t \in [T]$. The party $\mathsf{P}_{\varsigma_\pi(t)}$ computes m_t using a stateless next message function (corresponding to round t) that takes the message m_{t-1} the party received in round-$t-1$, the party's own input, and the randomness it received during preprocessing. The receiver of the message in round T is necessarily the output party P_{n+1} who computes the output as $\mathsf{Out}_\pi(m_T, r_{n+1})$.

Our first result shows that if f is privately computed by a speak-m-times PSS protocol (where each party speaks in at most m rounds) and using constant randomness, then f is computable using a read-m branching program of constant width independent of the number of parties and rounds. Using an argument along the lines of [KOR96], we first show that for any fixing of the randomness chosen during preprocessing, the number of possible messages received in any round in the now deterministic protocol while ranging over all possible inputs is at most $2^{\rho+2}$ irrespective of the number of rounds, when ρ is the randomness cost of π. Thus, when the randomness is fixed arbitrarily, there is a map η_{t-1} that maps the set of all possible messages m_{t-1} received by the speaker of any round t to the set $[2^{\rho+2}]$. For any round t, with $\mathsf{P}_{\varsigma_\pi(t)}$ as speaker, we can set the transition function $g^{(t)}_0 : [2^{\rho+2}] \to [2^{\rho+2}]$ as a translation of the next message function invoked with 0 as user's input (and randomness fixed) induced when m_{t-1} and m_t are mapped to $[2^{\rho+2}]$ by η_{t-1} and η_t, respectively; $g^{(t)}_1$ is defined similarly. Finally, the output function for the branching program is a translation of the output function of that in π induced by the mapping of m_T to $[2^{\rho+2}]$. The resulting branching program has width $2^{\rho+2}$ and length T (number of rounds in π), and the layer assignment function is the same as the speaker assignment function of π.

2.2 A Protocol Idea for Branching Programs

Let $f : \{0,1\}^n \to \{0,1\}$ be an n-party function computable using a *read-once* branching program of width w with transition functions $g_0^{(i)}, g_1^{(i)} : [w] \to [w]$ for $i \in [n]$, and sets $S_0, S_1 \subset [w]$ such that, for any $x_1, \ldots, x_n \in \{0,1\}$,

$$g_{x_n}^{(n)} \circ g_{x_{n-1}}^{(n-1)} \circ \ldots \circ g_{x_1}^{(1)}(1) \in \begin{cases} S_0 \text{ if } f(x_1, \ldots, x_n) = 0 \\ S_1 \text{ if } f(x_1, \ldots, x_n) = 1 \end{cases}.$$

A *non-private* sequential protocol for f can be built as follows: P_1 sends the message $u_1 = g_{x_1}^{(1)}(1)$ to P_2, P_2 sends $u_2 = g_{x_2}^{(2)}(u_1) = g_{x_2}^{(2)} \circ g_{x_1}^{(1)}(1)$ to P_3 and so on with P_i sending $u_i = g_{x_i}^{(i)}(u_{i-1}) = g_{x_i}^{(i)} \circ g_{x_{i-1}}^{(i-1)} \circ \ldots \circ g_{x_1}^{(1)}(1)$ to P_{i+1}. Party P_n, who receives u_{n-1}, sends $u_n = g_{x_n}^{(n)}(u_{n-1}) = g_{x_n}^{(n)} \circ g_{x_{n-1}}^{(n-1)} \circ \ldots \circ g_{x_1}^{(1)}(1)$ to P_{n+1}, who outputs $b \in \{0,1\}$ if $u_n \in S_b$.

The above non-private protocol admits a straightforward conversion to a PSS protocol: we will arrange each P_i to compute and forward a random permutation of u_i, say $\alpha_i(u_i)$, where α_i is a random permutation, in the place of u_i. For this, the (descriptions of) functions $(\hat{g}_0^{(1)}, \hat{g}_1^{(1)}) = (\alpha_1 \circ g_0^{(1)}, \alpha_1 \circ g_1^{(1)})$ are sent to P_1 during the preprocessing phase, and $(\hat{g}_0^{(i)}, \hat{g}_1^{(i)}) = (\alpha_i \circ g_0^{(i)} \circ \alpha_{i-1}^{-1}, \alpha_i \circ g_1^{(i)} \circ \alpha_{i-1}^{-1})$ are sent to P_i for each $2 \leq i \leq n$. When $(\hat{g}^{(i)}, \hat{g}^{(i)})$ is replaced with $(g^{(i)}, g^{(i)})$ in the aforementioned non-private protocol, each $\mathsf{P}_i, i > 1$ receives $\alpha_{i-1}(u_{i-1})$ instead of u_i, which hides u_{i-1} as long as α_{i-1} is sampled independent of α_i. Finally, P_{n+1} who receives $\alpha_n(u_n)$ from P_n, is also sent $\{\alpha_n(j) : j \in S_0\}$ and $\{\alpha_n(j) : j \in S_1\}$ during preprocessing. Now, P_{n+1} can check whether u_n belongs to S_0 or S_1, allowing it to decode $f(x_1, \ldots, x_n)$. In the process, P_{n+1} only learns whether u_n belongs to S_0 or S_1 since α_n is a random permutation unknown to P_{n+1}. Since 1-privacy is ensured as long as each α_i is uniformly random and independent of α_{i-1}, we can set $\alpha_i = \alpha_{\mathsf{odd}}$ for all odd i, and $\alpha_i = \alpha_{\mathsf{even}}$ for all even i, where $\alpha_{\mathsf{odd}}, \alpha_{\mathsf{even}}$ are randomly sampled from $\mathrm{Sym}(w)$: the set of all permutations of $[w]$. Thus, preprocessing uses a randomness domain of size $2 \log w!$.

The approach sketched above can be shown to be 1-private if f is computed using a read-once *permutation* branching program where $g_b^{(i)}$ is a one-to-one function for each $i \in [n]$ and $b \in \{0,1\}$. However, it fails to be 1-private when $\{g_b^{(i)}\}$ are not all one-to-one. For instance, consider the n-party AND function, computable using a read-once branching program of width 2 such that, for each $i \in [n]$, $g_1^{(i)} : b \mapsto b$ and $g_0^{(i)} : b \mapsto 0$. P_i receives $\alpha_{i-1}(u_{i-1})$ from P_{i-1}, and $(\alpha_i \circ g_0^{(i)} \circ \alpha_{i-1}^{-1}, \alpha_i \circ g_1^{(i)} \circ \alpha_{i-1}^{-1})$ during preprocessing. Since $g_0^{(i)}$ maps all inputs to 0, P_i can learn $\alpha_i(0)$ from $\alpha_i \circ g_0^{(i)} \circ \alpha_{i-1}^{-1}$ which further reveals α_i. But then, α_i and $\alpha_i \circ g_1^{(i)} \circ \alpha_{i-1}^{-1}$ can be used to learn α_{i-1} since $g_1^{(i)}$ is the identity function. Thus P_i can recover $u_{i-1} = x_1 \cdot \ldots \cdot x_{i-1}$ from $\alpha_{i-1}(u_{i-1})$, breaking security of the protocol.

To get around this, the preprocessing step samples a random bit r_i and permutations $\alpha_{i,0}, \alpha_{i,1}$ for each $i \in [n]$, and sets $\alpha_{0,0}$ and $\alpha_{0,1}$ to be the identity

function. To each P_i, it sends r_i, and the following functions:

$$\hat{g}^{(i)}_{0,0} = \alpha_{i,0} \circ g^{(i)}_0 \circ \alpha^{-1}_{i-1,r_{i-1}} \qquad \hat{g}^{(i)}_{0,1} = \alpha_{i,0} \circ g^{(i)}_0 \circ \alpha^{-1}_{i-1,r_{i-1}\oplus 1} \tag{1}$$

$$\hat{g}^{(i)}_{1,0} = \alpha_{i,1} \circ g^{(i)}_1 \circ \alpha^{-1}_{i-1,r_{i-1}} \qquad \hat{g}^{(i)}_{1,1} = \alpha_{i,1} \circ g^{(i)}_1 \circ \alpha^{-1}_{i-1,r_{i-1}\oplus 1} \tag{2}$$

In the protocol, we will ensure the invariant $\alpha_{i,x_i}(u_i) = \alpha_{i,x_i} \circ g^{(i)}_{x_i}(u_{i-1})$. For this, P_1 sends $x_1 \oplus r_1$ and $\alpha_{1,x_1} \circ g^{(1)}_{x_1}(0) = \alpha_{1,x_1}(u_1)$ to P_2. To propagate the invariant, it suffices to show that P_i can compute $x_i \oplus r_i$ and $\alpha_{i,x_i}(u_i) = \alpha_{i,x_i} \circ g^{(i)}_{x_i}(u_{i-1})$, assuming P_{i-1} sends $x_{i-1} \oplus r_{i-1}$ and $\alpha_{i-1,x_{i-1}}(u_{i-1})$ to P_i. Observe that

$$\hat{g}^{(i)}_{x_i,x_{i-1}\oplus r_{i-1}} = \alpha_{i,x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{i-1,x_{i-1}}.$$

Hence, using $\hat{g}^{(i)}_{x_i,x_{i-1}\oplus r_{i-1}}$, r_i, $x_{i-1} \oplus r_{i-1}$ and $\alpha_{i-1,x_{i-1}}(u_{i-1})$, P_i can compute $x_i \oplus r_i$ and

$$\begin{aligned}\hat{g}^{(i)}_{x_i,x_{i-1}\oplus r_{i-1}}(\alpha_{i-1,x_{i-1}}(u_{i-1})) &= \alpha_{i,x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{i-1,x_{i-1}} \circ \alpha_{i-1,x_{i-1}}(u_{i-1}) \\ &= \alpha_{i,x_i} \circ g^{(i)}_{x_i}(u_{i-1}) = \alpha_{i,x_i}(u_i)\end{aligned}$$

preserving the invariant. To allow computing the output, during preprocessing, P_{n+1} is sent

$$\hat{S}_{c,0} = \{\alpha_{n,r_n\oplus c}(j) : j \in S_0\}, \qquad \hat{S}_{c,1} = \{\alpha_{n,r_n\oplus c}(j) : j \in S_1\}, \quad c \in \{0,1\}.$$

P_{n+1} outputs b satisfying $u_n \in \hat{S}_{x_n\oplus r_n,b}$, on receiving $\alpha_{n,x_n}(u_n)$ and $x_n \oplus r_n$ from P_n. It is easy to verify that P_{n+1} outputs b such that $u_n \in S_b$.

Security against P_1 and P_{n+1} are straight-forward to argue: the former does not receive any message, and the view of the latter can be easily simulated using the output of the function. For $1 < i \leq n$, P_i receives $x_{i-1} \oplus r_{i-1}$ and $\alpha_{i-1,x_{i-1}}(u_{i-1})$ from P_{i-1}, and r_i and $\{\hat{g}^{(i)}\}_{c,c'\in\{0,1\}}$. Although $x_{i-1} \oplus r_{i-1}, \alpha_{i-1,x_{i-1}}(u_{i-1})$, and r_i do not reveal any information to P_i, taken together with $\{\hat{g}^{(i)}_{c,c'}\}_{c,c'\in\{0,1\}}$, these random variables can indeed break security for certain branching programs even when $\alpha_{i,b}$ is chosen uniformly and independently for all $i \in [n]$ and $b \in \{0,1\}$. Using a careful analysis, we characterize the family of branching programs for which the protocol remain perfectly private. We refer to this family as strongly regular branching programs (discussed in the next section). We further show that 1-privacy is maintained even when randomness is reused as follows: set $(\alpha_{i,0}, \alpha_{i,1}, r_i) = (\alpha_{\mathsf{odd},0}, \alpha_{\mathsf{odd},1}, r_{\mathsf{odd}})$ for all odd i, and $(\alpha_{i,0}, \alpha_{i,1}, r_i) = (\alpha_{\mathsf{even},0}, \alpha_{\mathsf{even},1}, r_{\mathsf{even}})$ for all even i, where $\alpha_{\mathsf{odd},0}, \alpha_{\mathsf{odd},1}, \alpha_{\mathsf{even},0}, \alpha_{\mathsf{even},1}$ are randomly sampled from $\mathrm{Sym}(w)$ and $r_{\mathsf{odd}}, r_{\mathsf{even}}$ are random bits. Thus, the protocol used a randomness domain of $4 \log w! + 2$ to carry out the private computation.

2.3 Strongly Regular Branching Programs

For the PSS protocol outlined in the previous section (and its generalization to read-m branching programs) to achieve 1-security requires the branching program to satisfy a technical condition we call strong regularity. In this section, we

will briefly describe this condition and sketch the intuition behind a construction that converts *any* branching program to a strongly regular branching program while incurring a quadratic blow-up in the width, but preserving the length of the branching program.

A branching program is said to be strongly regular if the pair of transition functions $g_0^{(i)}, g_1^{(i)} : [w] \to [w]$ are strongly regular for every layer i in the program. Strong regularity of $g_0^{(i)}, g_1^{(i)}$ requires that the preimages of $g_0^{(i)}$ form a partition of $[w]$ into sets of equal size (ignoring empty pre-images); similarly for $g_1^{(i)}$. Further, the intersection of these partitions created by $g_0^{(i)}$ and $g_1^{(i)}$ is also a partition into sets of equal size, say d (again ignoring empty intersections). Strong regularity requires an additional technical condition: For this define a bipartite graph H with the same set $[w]$ as both left and right vertices. There is an edge between a left vertex u and a right vertex v if the preimage of u under $g_0^{(i)}$ intersects the preimage of v under $g_0^{(i)}$ (this intersection is of size d by previous conditions). The final condition for strong regularity demands that a random automorphism of H maps every edge to a uniformly random edge in H. Here, by an automorphism of of H, we mean any permutation of the left and right vertices of H under which every edge is mapped to some edge of H. The security of our protocols depend crucially on the strong regularity of transition functions, and hence that of the branching program.

In Theorem 7, we show how to transform an arbitrary branching program into a strongly regular one while scaling the width from w to w^2. Leaving the layer assignment function σ unchanged, we define new transition functions $\{h_b^{(i)}\}_{i\in[\ell],b\in\{0,1\}}$ and output function ϕ' as follows. Given a pair of transition functions $g_0^{(i)}, g_1^{(i)} : [w] \to [w]$, we construct functions $h_0^{(i)}, h_1^{(i)} : [w]\times[w] \to [w]\times[w]$ as follows: assign $h_b^{(i)}(u, v) = (u, g_b^{(i)}(u))$ if i is odd and $h_b^{(i)}(u, v) = (g_b^{(i)}(v), v)$ if i is even for all $(u, v) \in [w]^2$, $b \in \{0, 1\}$. Thus, for odd i (the case of even i is similar), every node u in $[w]$ is replaced by a set of w nodes $(u, 1), \ldots, (u, w)$ and $h_b^{(i)}$ maps all of them to $(u, g_b^{(i)}(u))$. This implies strong regularity as

1. for each $u \in [w]$, the nodes $(u, 1), \ldots, (u, w)$ in the domain of $h_b^{(i)}$ are all mapped to $(u, g_b^{(i)}(u))$, while all the other nodes of the form $(u, v), v \neq g_b^{(i)}(u))$ in the co-domain have an empty pre-image; therefore $|(h_b^{(i)})^{-1}(u, v)|$ is either w or 0,
2. for $u', v', u'', v'' \in [w]$, the pre-images of (u', v') under $h_0^{(i)}$ and (u'', v'') under $h_1^{(i)}$ have a non-empty intersection if and only if $u' = u''$, and $v' = v'' = g_0^{(i)}(u') = g_1^{(i)}(u'')$, and in such a case, their pre-images are both $\{(u', 1), \ldots, (u', w)\}$; hence, the size of the intersection of pre-images of (u', v') and (u'', v'') is either w or 0, and
3. the H graph consists of an edge between $(u, g_0^{(i)}(u))$ and $(u, g_1^{(i)}(u))$ for each $u \in [w]$; therefore, the edge set E consists of w edges where no two of them have any common vertices, which implies that every edge is mapped to every edge with the same probability under a uniformly chosen $\mathrm{Aut}(H)$.

Furthermore, note that if (u_0, v_0) is the initial state of the program, then, for odd i,

$$(u_i, v_i) = (u_{i-1}, g^{(i)}_{x_{\sigma(i)}} \circ g^{(i-1)}_{x_{\sigma(i-1)}} \circ \cdots \circ g^{(1)}_{x_{\sigma(1)}}(u_0)),$$

and for even i,

$$(u_i, v_i) = (g^{(i)}_{x_{\sigma(i)}} \circ g^{(i-1)}_{x_{\sigma(i-1)}} \circ \cdots \circ g^{(1)}_{x_{\sigma(1)}}(u_0), v_{i-1}).$$

Hence, defining the new output function as $\phi'(u, v) = \phi(v)$ for odd ℓ and $\phi'(u, v) = \phi(u)$ for even ℓ ensures that the output is identical to that of the original branching program. Thus, for a function $f : \{0,1\}^n \to \{0,1\}$, given a length-$\ell$ and width-w branching program, we have a length-ℓ and width-w^2 strongly regular branching program.

2.4 Beyond Read-Once Branching Programs

We generalize our construction for read-once strongly regular branching programs to accommodate strongly regular branching programs in which a party may have inputs at multiple layers. Suppose $f : \{0,1\}^n \to \{0,1\}$ computable using a strongly regular branching program of width w and length ℓ. That is, there exist $\sigma : [\ell] \to [n]$; for each $t \in [\ell]$, a pair of functions $g^{(t)}_0, g^{(t)}_1 : [w] \to [w]$; and sets $S_0, S_1 \subset [w]$ such that for $(x_1, \ldots, x_n) \in \{0,1\}^n$,

$$g^{(\ell)}_{x_{\sigma(\ell)}} \circ \ldots \circ g^{(2)}_{x_{\sigma(2)}} \circ g^{(1)}_{x_{\sigma(1)}}(1) \in \begin{cases} S_0 \text{ if } f(x_1, \ldots, x_n) = 0, \\ S_1 \text{ if } f(x_1, \ldots, x_n) = 1. \end{cases}$$

The straightforward extension of the protocol sketched in Section 2.2 (without randomness reuse) continues to private for read-m strongly regular branching programs. That is, during preprocessing, $(\alpha_{t,0}, \alpha_{t,1})$ and $(r_{t,0}, r_{t,1})$ are sampled independently and uniformly at random for each $t \in [\ell]$, and the corresponding party $\mathsf{P}_{\sigma(t)}$ receives 'masked functions' $\{\hat{g}^{(i)}_{b,b'}\}_{b,b'} \in \{0,1\}$, as defined in eq. (1). To aid in computing the output, P_{n+1} is sent

$$\hat{S}_{c,0} = \{\alpha_{\ell, r_\ell \oplus c}(j) : j \in S_0\}, \qquad \hat{S}_{c,1} = \{\alpha_{\ell, r_\ell \oplus c}(j) : j \in S_1\}, \quad c \in \{0,1\}.$$

This results in a protocol where the randomness cost grows with ℓ, the length of the branching program. In the read once case, we could bring the randomness cost down by using the same permutations for all odd parties, and a independently sampled set of permutations for even parties. Reusing randomness in this manner will break security if a party P_i appears more than once in the branching program.

We next describe how to reuse randomness for the read-m case. Let $z_t = x_{\sigma(t)}$ be the input of $\mathsf{P}_{\sigma(t)}$ at layer $t \in [\ell]$. We already observed that the view of $\mathsf{P}_{\sigma(t)}$ consisting of $z_{t-1} \oplus r_{t-1}, \alpha_{t-1, z_{t-1}}(u_{t-1})$, and $\{\hat{g}^{(t)}_{b,b'}\}$ is private. Hence, to ensure privacy against P_i in the protocol, it suffices to ensure that,

$\{\alpha_{t-1,0}, \alpha_{t-1,1}, \alpha_{t,0}, \alpha_{t,0} : t \in [\ell], \sigma(t) = i\}$ are sampled uniformly and independently from $\mathrm{Sym}(w)$, and $\{r_{t-1}, r_t : t \in [\ell], \sigma(t) = i\}$ are sampled independently from $\{0,1\}$.

To satisfy these conditions, define a conflict graph $G = ([\ell], E)$ where $\{t, t'\} \in E$ if $t \neq t' \in [\ell]$ and $\{\sigma(t), \sigma(t+1)\} \cap \{\sigma(t'), \sigma(t'+1)\} \neq \emptyset$. Let $\mathsf{col} : [\ell] \to [\chi]$ be an optimal vertex coloring of G. Sample $\alpha_{j,0}$ and $\alpha_{j,1}$ uniformly from $\mathrm{Sym}(w)$, and r_j uniformly from $\{0,1\}$ for each $j \in [\chi]$ and set

$$\alpha_{t,0} = \alpha_{\mathsf{col}(t),0} \quad \alpha_{t,1} = \alpha_{\mathsf{col}(t),1} \quad r_t = r_{\mathsf{col}(t)}, \quad t \in [\ell].$$

Such an assignment satisfies the constraints given above by the construction of the conflict graph, ensuring 1-privacy. The randomness cost of such a protocol is $O(\chi) \cdot w!$.

2.5 Private Computation of AND

As an application of our positive result, we modify our PSS protocol construction to realize with 1-privacy the AND functionality which takes a bit from each of the n parties and delivers their product to all parties. The resulting protocol achieves AND computation with 6 bits of randomness when n is odd, matching the best randomness upper bound in the literature [CR22]. When n is even we get a randomness cost of 9 bits. In this section, we outline the modified construction. To complement this upper bound, we also show that 1-private computation of AND functionality among 3 parties requires at least 3 bits of randomness even while employing non-sequential protocols. The previously best known lower bound [KOP+19] result is that 1 bit of randomness is insufficient (for any number of parties).

Our protocol for n-party AND in the PSS model consumes 6 bits of randomness. We convert this into the standard model by getting rid of the preprocessing step, and internalizing the output party P_{n+1}. For odd values of n, we effect this transformation without requiring any extra randomness; whereas, for even n, our transformation consumes 3 more bits, resulting in 9 bits of randomness. The preprocessing step can be removed from the PSS protocol for AND (or any read-once branching program in general) by letting P_1 sample and deliver the randomness supplied in the preprocessing step. Since P_1 does not receive messages during in the online step, this change does not affect security against P_1. To remove the output party P_{n+1}, we transfer the role of P_{n+1} to P_2, by redirecting the P_n's messages $\alpha_{n,x_n \oplus r_n}(u_n)$ and $x_n \oplus r_n$ to P_2 instead of P_{n+1}, and also redirecting the randomness used for computing the output to P_2. To ensure privacy against P_2 despite these extra messages, $\alpha_{n,0}, \alpha_{n,1}$ and r_n are sampled using 3 bits of fresh randomness, and the randomness needed for computing the output is sampled appropriately. When n is odd, we observe that some randomness can be recycled avoiding the need of fresh randomness for sampling $\alpha_{n,0}, \alpha_{n,1}$ and r_n, and maintaining randomness cost of 6 bits[7]. Finally, P_2 distributes the decoded output to all the parties.

[7] An involved construction can bring down the randomness cost for even values of n from 9 bits to $6 + \log 3$ bits. We do not present this construction in this work.

The protocol obtained by this transformation preserves the sequentiality except in the initial step where P_1 sends correlated randomness to all parties, and in the last step where P_2 delivers output to all parties; we refer to such a protocol as an unassisted PSS protocol or simply a uPSS protocol. We note that this results in a much sparser communication pattern compared to some of the protocols in the literature with low randomness cost [KOP+19, CR22].

Our lower bound of 3 bits for 3-party AND is obtained using a lower bound for the so-called 3-Secret Sharing (3SS) problem, recently presented in [ARN+23]. In a 3SS for secret domain M, the dealer, with input $(m_1, m_2, m_3) \in M$ wants to compute shares $s_{\{1,2\}}, s_{\{2,3\}}, s_{\{1,3\}}$ such that for any distinct $i, j, k \in [3]$, $s_{\{i,j\}}$ and $s_{\{i,k\}}$ form a secret sharing of m_i. We show that, in any 3-party 1-private AND protocol, the transcripts $T_{\{i,j\}}$ and $T_{\{i,k\}}$ between P_i and P_j, and between P_i and P_k, respectively, form a secret sharing of x_i, the input of P_i, for any distinct $i, j, k \in [3]$. Consequently, for the secret domain $D = (x_1, x_2, x_3) \in \{0,1\}^3 \setminus \{(1,1,1)\}$, the transcripts $\{T_{\{i,j\}}\}$ of the AND protocol with input (m_1, m_2, m_3) forms a 3SS of $(m_1, m_2, m_3) \in D$. At this point, we invoke the fact that randomness complexity of 3SS for D is 3 bits to obtain the desired lower bound for AND computation.

3 Preliminaries

We use the standard notion of 1-private computation and randomness complexity associated with it. By default, we shall use a model with correlated randomness generated during a pre-processing phase, and no other randomness, as this is the setting we shall use in Section 4 and Section 5; however, local (uncorrelated) randomness can be modeled as a special case of this.

For the sake of being self-contained, we summarize the standard protocol model below, with notation that will be convenient for us. For our purposes, a T-round protocol π (over private channels) with n input-parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ and an output party P_{n+1}, is specified by a correlated-randomness generation function Prep_π, a deterministic next message function Next_π, and output function Out_π which behave as follows in an execution of the protocol. A random element $R \leftarrow \mathcal{R}$ is sampled first, where $\mathcal{R}$ is a finite set representing the randomness space of the protocol. Prep_π, on input (i, R) where $i \in [n+1]$, outputs a string R_i (corresponding to the share of correlated-randomness for party P_i). Next_π takes as input $(i, \mathsf{View}^{(i)}_{\pi,t})$, where $i \in [n]$ is an index, and $\mathsf{View}^{(i)}_{\pi,t}$ (for $0 \le t < T$) is the *view* of P_i in t rounds – consisting of its input, the string R_i (obtained from Prep_π), and all the messages received from all the other parties till then – and outputs a set of messages for P_i to send to all the other parties in round $t+1$. Out_π takes as input $(i, \mathsf{View}^{(i)}_{\pi,T})$ and produces an output for party P_i. We define the random variables $\mathsf{View}^{(i)}_\pi(x_1, \ldots, x_n)$ and $\pi(x_1, \ldots, x_n)$ to be, respectively, the view of P_i in a complete execution of π with parties using inputs $(x_1, \ldots, x_n)$, and the outputs produced by the parties at the end of such an execution.

An n-party functionality $\mathcal{F}$ is simply a function that takes n inputs, one from each party, and deterministically produces n outputs, one for each party.

We say that $\pi = (\mathsf{Prep}_\pi, \mathsf{Next}_\pi)$ is a 1-private realization of the functionality $\mathcal{F}$ (or simply, π is a protocol for $\mathcal{F}$) if the following conditions hold:

- **Correctness.** For any set of inputs $(x_1, \ldots, x_n)$, $\Pr[\pi(x_1, \ldots, x_n) = \mathcal{F}(x_1, \ldots, x_n)] = 1$ (where the probability is over the random input $R \in \mathcal{R}$ given to Prep_π).
- **1-Privacy.** For any $i \in [n]$ and any two sets of inputs $(x_1, \ldots, x_n)$ and $(x'_1, \ldots, x'_n)$ such that $x_i = x'_i$ and the i^{th} output of $\mathcal{F}$ on both are equal, $\mathsf{View}_\pi^{(i)}(x_1, \ldots, x_n)$ and $\mathsf{View}_\pi^{(i)}(x'_1, \ldots, x'_n)$ are identically distributed.

For a function $f : \{0,1\}^n \to \{0,1\}$, we define an $(n+1)$-party functionality $\mathcal{F}_f$, which takes an input bit x_i from party P_i for $i \in [n]$ (and empty input from P_{n+1}) and outputs $f(x_1, \ldots, x_n)$ to party P_{n+1} (and empty output to the other parties).

Branching Programs

Definition 1. *A width w and length ℓ branching program for a function $f : \{0,1\}^n \to \{0,1\}$ is a collection of functions $(\sigma, \{g_b^{(t)}\}_{t \in [\ell], b \in \{0,1\}}, \phi)$ where $\sigma : [\ell] \to [n]$ encodes the order in which inputs are accessed, $g_b^{(t)} : [w] \to [w]$ denotes the transition function for each choice bit b and $t \in [\ell]$, and $\phi : [w] \to \{0,1\}$ denotes the output function, such that for all $(x_1, \ldots, x_n) \in \mathcal{X}_1, \times \ldots, \times \mathcal{X}_n$, $f(x_1, \ldots, x_n) = \phi(u_\ell)$, where u_i is defined as follows: $u_0 = 1$ and for $t \geq 1$, $u_t = g^{(t)}_{x_{\sigma(t)}}(u_{t-1})$.*

We shall refer to a length ℓ branching program as having ℓ *layers*. σ in the above definition is said to be the *input label function*, which maps each layer to an input index. Also, given a branching program as above and an input $(x_1, \ldots, x_n)$ for it, we shall refer to $x_{\sigma(t)}$ as the *choice bit* at layer t. We shall also be interested in a natural complexity measure of a branching program (apart from width and length), namely the number of layers at which the same input is used: we say that a branching program is a read-k-times branching program if for all $i \in [n]$, $|\{t : \sigma(t) = i\}| \leq k$. By default, all the branching programs we consider, unless otherwise specified, are read-constant-times branching programs; note that in this case the length $\ell = O(n)$.

4 Private Sequential Stateless Protocols

In this section we define the PSS model and further show that constant - randomness speak-constant-times PSS protocols imply constant-width read-constant-times branching programs.

A Private Sequential Stateless protocol is a 1-private protocol with certain restrictions on its communication pattern (sequential) and computation (stateless). Below we define a PSS protocol π in terms of functions $(\mathsf{Prep}_\pi, \varsigma_\pi, \mathsf{Next}_\pi, \mathsf{Out}_\pi)$, where ς_π determines which party speaks at each round, Prep_π computes the correlated randomness given to the parties in the pre-processing phase, Next_π

is the next-message function used by a party (who is speaking at a round) to generate the message for the next round based solely on the message sent to it in the current round and its share of correlated randomness and input (since it does not update its state during the online phase), and Out_π is the function used by the output party (who does not participate in the protocol otherwise) to generate the final output.

Definition 2 (Private Sequential Stateless Protocol). *A T-round* Private Sequential Stateless protocol π for $f : \{0,1\}^n \to \{0,1\}$ *is a tuple* $(\mathsf{Prep}_\pi, \varsigma_\pi, \mathsf{Next}_\pi, \mathsf{Out}_\pi)$*, with* $\mathsf{Prep}_\pi : [n+1] \times \mathcal{R} \to \{0,1\}^*$*,* $\varsigma_\pi : [T+1] \to [n+1]$*,* $\mathsf{Next}_\pi : [T] \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\} \to \{0,1\}^*$*, and* $\mathsf{Out}_\pi : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$*, such that and the following is a 1-private protocol for the $n+1$-party functionality $\mathcal{F}_f$ in the pre-processing model:*

- *First, in the pre-processing phase $R \leftarrow \mathcal{R}$ is sampled and for each $i \in [n+1]$, P_i receives is $r_i := \mathsf{Prep}_\pi(i; R)$ as its share of correlated randomness;*
- *then for each $i \in [n]$, party P_i receives an input bit x_i;*
- *then, at each round $t \in [T]$, party $\mathsf{P}_{\varsigma_\pi(t)}$ receives a message m_{t-1} (m_0 is defined as the empty string) and sends the message $m_t := \mathsf{Next}_\pi(t, m_{t-1}, r_{\varsigma_\pi(t)}, x_{\varsigma_\pi(t)})$ to party $\mathsf{P}_{\varsigma_\pi(t+1)}$; it is required that $\varsigma_\pi(t) = n+1$ iff $t = T+1$.*
- *Finally, party P_{n+1} outputs the bit $\mathsf{Out}_\pi(m_{T+1}, r_{n+1})$.*

We call the PSS protocol π a speak-k-times protocol if $|\varsigma_\pi^{-1}(i)| \leq k$ for all $i \in [n]$. The randomness cost of π is defined as $\log_2 |\mathcal{R}|$ bits.

By default in the PSS model, unless otherwise specified, we always consider speak-constant-times protocols (i.e., speak-k-times protocols where k does not grow with the input size n).

It is worth emphasising that statelessness is a structural feature of a PSS protocol, and it does not alter the security model of 1-privacy: the adversary can corrupt a party at the beginning of the protocol and it can see all the messages ever sent to that party.

4.1 PSS Protocols to Branching Programs

In this section we prove the following result.

Theorem 6. *For any constant k, boolean functions over $\{0,1\}^n$ which have speak-k-times, constant-randomness-cost PSS protocols also have read-k-times, constant-width branching programs.*

Proof. The proof uses the ideas from the transformation of protocols into circuits in [KOR96]. We make the required transformation in two steps, firstly, converting the given randomized protocol into a deterministic protocol by freezing the randomnness in the system, and then, converting this deterministic protocol into a branching program by defining appropriate functions on the set of these messages. The width of the branching program will be determined by the number of different messages a party can receive at any round.

We start by bounding the number of different views a party can have under a fixed randomness $R \in \mathcal{R}$ (analogous to Lemma 4 from [KOR96]).

Lemma 1. *If π is a PSS protocol with randomness cost ρ, then for any fixed choice of randomness, over all the choices of inputs, the total number of communication transcripts seen by any party P_i is at most $2^{\rho+2}$.*

Proof. Let π be a PSS protocol for a function $f : \{0,1\}^n \to \{0,1\}$. For an input $\boldsymbol{x} \in \{0,1\}^n$, let $C_i(\boldsymbol{x})$ denote the set of communication transcripts a party P_i can see in executions of π, using all choices of the randomness $R \in \mathcal{R}$. Note that $|C_i(\boldsymbol{x})| \leq |\mathcal{R}| = 2^\rho$. Secondly, for all $\boldsymbol{x}$ in an equivalence class such that x_i and $f(\boldsymbol{x})$ are equal (there are four such equivalence classes), the distribution, and hence support, of the views of P_i are identical; this follows from the 1-privacy guarantee of a PSS protocol. Since the view contains the communication (as well as the party's share of correlated randomness), for all $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the same equivalence class, $C_i(\boldsymbol{x}) = C_i(\boldsymbol{x}')$. Hence, taking the union over all $\boldsymbol{x}$ in the same equivalence class, $|\bigcup_{\boldsymbol{x}} C_i(\boldsymbol{x})| \leq 2^\rho$. Since there are four such equivalence classes, we have

$$\left| \bigcup_{\boldsymbol{x} \in \{0,1\}^n} C_i(\boldsymbol{x}) \right| \leq 2^{\rho+2}.$$

In particular, for any fixed choice of randomness, the transcripts seen by P_i comes from this set of size $2^{\rho+2}$, as claimed. □

We now proceed to transform the given protocol into a branching program. Let π' be a deterministic protocol obtained by fixing the randomness of π to $R^* \in \mathcal{R}$. To convert π' to a branching program we shall interpret the message sent in round t from party P_i to P_j (where $i = \varsigma_\pi(t)$ and $j = \varsigma_\pi(t+1)$) as a state in the $t+1^{\text{st}}$ layer of the branching program. Since the number of different messages that P_j can receive in a round (over all inputs $\boldsymbol{x} \in \{0,1\}^n$) is upper bounded by the total number of communication transcripts, which is in turn bounded by $2^{\rho+2}$ by Lemma 1, the width of the branching program can be set to $w = 2^{\rho+2}$. The length of the branching program $\ell = T$, the number of rounds in π, and the input-reading function σ is the same as ς_π, but restricted to $[T]$ (rather than $[T+1]$).

The transition functions g_b^t from layer $t-1$ to layer t will implement $\mathsf{Next}_\pi(t, \cdot, r^*, b)$, for $b \in \{0,1\}$ and where $r^* = \mathsf{Prep}_\pi(\varsigma_\pi(t), R^*)$, under a mapping of messages to states. In more detail, for $t \in [T]$, let M^t denote the set of messages that can be sent in round t, over all possible inputs $\boldsymbol{x} \in \{0,1\}^n$ (with the randomness fixed to R^*); also let $M^0 = \{\epsilon\}$. We noted above that $|M^t| \leq w$. Let $\eta^t : M^t \to [w]$ be an arbitrary injective function for each $t \in [T]$; also let $\eta^0(\epsilon) = 1$ to set the start state (in layer 1) to be 1. Then we define $g_b^t : [w] \to [w]$, for $b \in \{0,1\}$, such that if $u = \eta^{t-1}(m)$ for $m \in M^{t-1}$, let $g_b^t(u) = \eta^t(\mathsf{Next}_\pi(t, m, r^*, b))$ where $r^* = \mathsf{Prep}_\pi(\varsigma_\pi(t), R^*)$; if u is not in the image of η^t, we set $g_b^t(u)$ arbitrarily. Finally, The output function $\phi : [w] \to \{0,1\}$ is defined as follows: if $u = \eta^T(m)$, then $\phi(u) = \mathsf{Out}_\pi(m, \mathsf{Prep}_\pi(n+1, R^*))$; if u is not in the image of η^T, we set $\phi(u)$ arbitrarily.

From the perfect correctness of π, it follows that this branching program computes f. Also, since $\sigma = \varsigma_\pi$ (restricted to $[T]$), if π is a speak-k-times protocol, then the branching program constructed above will be a read-k-times branching program. Finally, as required, if π's randomness cost ρ is a constant, so is the width $w = 2^{\rho+2}$. □

5 PSS Protocols From Branching Programs

We begin by formally defining strong regularity and strongly regular branching programs. We then present Private Sequential Stateless protocols for strongly regular branching programs.

Definition 3 (Strong Regularity). *A pair of functions $g_0, g_1 : [w] \to [w]$ is* strongly regular *if the following conditions are met:*

1. *There exists c_0, c_1 such that $|g_0^{-1}(u)| \in \{c_0, 0\}$, and $|g_1^{-1}(u)| \in \{c_1, 0\}$ for all $u \in [w]$, when $g_0^{-1}(u)$ and $g_1^{-1}(u)$ denote the preimages of u under g_0 and g_1, respectively.*
2. *There exists c such that $|g_0^{-1}(u) \cap g_1^{-1}(v)| \in \{c, 0\}$ for all $u, v \in [w]$.*
3. *Define a bipartite graph $H = (L \cup R, E)$ where $L = [w]$ and $R = [w]$ (disjoint copies) are the left and right set of vertices respectively, and $E = \{(u, v) \in L \times R : g_0^{-1}(u) \cap g_1^{-1}(v) \neq \emptyset\}$ is the edge set. Let $\mathrm{Aut}(H)$ be the set of all automorphisms of H that respect the left and right parts; i.e., $\mathrm{Aut}(H) = \{(\mu, \nu) \in \mathrm{Sym}(w) \times \mathrm{Sym}(w) : (\mu(u), \nu(v)) \in E \Leftrightarrow (u, v) \in E\}$. Then,*

$$\mathrm{Prob}\left[(\mu(u), \nu(v)) = (u', v') | (\mu, \nu) \leftarrow \mathrm{Aut}(H)\right] = 1/|E|, \forall (u, v), (u', v') \in E. \tag{3}$$

We shall be interested in branching programs where, at all layers, the pairs of transition functions are strongly regular. We capture this in the following definition.

Definition 4 (SRBP and k-SRBP). *A branching program with input labeling function $\sigma : [\ell] \to [n]$ and transition functions $\{(g_0^{(t)}, g_1^{(t)})\}_{t \in [\ell]}$ is a strongly regular branching program (SRBP) if for every $t \in [\ell]$, the pair $(g_0^{(t)}, g_1^{(t)})$ is strongly regular. It is said to be a k-SRBP if for all $i \in [n]$, $|\{t : \sigma(t) = i\}| \leq k$.*

A special case of interest is a 1-SRBP: in this case, we may w.l.o.g. assume that $\ell = n$ (adding layers with identity functions as transition functions, if necessary), and σ is the identity function (by permuting the order of the arguments to the function evaluated by the SRBP, if necessary).

While SRBPs may appear restrictive, they are in fact quite expressive, and any branching program can be converted to one with only a polynomial blow-up in the width, and no change to the length or the input label function. We give an overview this conversion in Section 2.3. We state this formally here and present the proof in Appendix C.

Theorem 7. *For any branching program of width w and length ℓ there is an SRBP computing the same function of the same length and input label function, and width w^2.*

Examples. Apart from the fact that any constant width branching program can be converted to a constant width SRBP, natural branching programs to compute some interesting functions are already constant width SRBPs. We mention three such examples of 1-SRBP below:

- $\mathsf{AND}(x_1, \ldots, x_n) = x_1 \wedge \ldots \wedge x_n$ has a width-2 1-SRBP.
- $\mathsf{IP}(x_1, y_1, \ldots, x_n, y_n) = \bigoplus_i (x_i \wedge y_i)$ has a width-4 1-SRBP.
- Every permutation branching program (in which all transition functions are permutations) is an SRBP (with $c_0 = c_1 = c = 1$ and H being a perfect matching, in Definition 3).

Strong Regularity and Regularity. It is instructive to compare SRBP with the notion of a regular branching program from [LPV23]. Let us call a pair of functions $g_0, g_1 : [w] \to [w]$ (c_1, c_2)-regular if for all $u \in [w]$, $|g_0^{-1}(u)| + |g_1^{-1}(u)| \in \{0, c_1, c_2, c_1 + c_2\}$. Note that a strongly regular pair (as a consequence of the first condition in Definition 3) is regular according to this definition. For the special case of $(1,1)$-regularity, we require $|g_0^{-1}(u)| + |g_1^{-1}(u)| \leq 2$; but since the average value of $|g_0^{-1}(u)| + |g_1^{-1}(u)|$ is 2, it must be the case that for each u, $|g_0^{-1}(u)| + |g_1^{-1}(u)| = 2$. This is the definition of regularity used in [LPV23].

Restricting to $(1,1)$-regular branching programs results in somewhat crippled computational power: even a simple function like n-input AND requires a $(1,1)$-regular branching program to have width that grows (exponentially) with n. On the other hand, AND has a width 2 branching program that is $(2,1)$-regular. As such, regular branching programs as generalized above (or possibly with the restriction that all layers use the same (c_1, c_2) – since the transformation in Theorem 7 yields a (w, w)-regular branching program) is an interesting class on its own right.

Strong regularity imposes additional constraints beyond (c_1, c_2)-regularity. One may in fact add even more constraints, and yet retain the result in Theorem 7 (e.g., require the bipartite graph H in Definition 3 to be a complete bipartite graph after pruning 0-degree nodes), but this will rule out some of the examples above (e.g., permutation branching programs).

5.1 PSS Protocols From 1-SRBP

Proof of Theorem 2. Let $\left(\{g_b^{(i)}\}_{i \in [n], b \in \{0,1\}}, \phi\right)$ be a width w 1-SRBP computing the function $f : \{0,1\}^n \to \{0,1\}$. The protocol given in Figure 1 is a speak-once PSS protocol which computes f. All the variables used in the sequel are defined in Figure 1. We will separately prove the correctness and privacy of the protocol.

Correctness. We claim,

$$v_i = \alpha_{\mathsf{col}(i), x_i} \circ g_{x_i}^{(i)} \circ g_{x_{i-1}}^{(i-1)} \circ \ldots \circ g_{x_1}^{(1)}(1), \qquad i \in [n]. \tag{4}$$

A PSS protocol to compute f having a 1-SRBP

Let $\left(\{g_b^{(t)}\}_{t\in[n],b\in\{0,1\}}, \phi\right)$ be a width-w 1-SRBP computing the function $f : \{0,1\}^n \to \{0,1\}$. For each $i \in [n]$, let $x_i \in \{0,1\}$ be the input to party P_i. and let $\mathsf{col}(i)$ be the parity of i, *i.e.*, $\mathsf{col}(i) = 0$ if i is even, and 1 otherwise. Set $\varsigma_\pi : [n+1] \to [n+1]$ to be the identity map.

Preprocessing

1. $\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, r_0, r_1 \leftarrow \mathcal{R}$ is sampled where $\mathcal{R} = \mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \{0,1\} \times \{0,1\}$.
2. P_1 receives $\mathsf{Prep}_\pi(1; \alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, r_0, r_1) := (r_1, \alpha_{1,0}, \alpha_{1,1})$.
3. For each $2 \le i \le n$: P_i receives $\mathsf{Prep}_\pi(i; \alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, r_0, r_1) := (r_{\mathsf{col}(i)}, \hat{g}^{(i)}_{0,0}, \hat{g}^{(i)}_{0,1}, \hat{g}^{(i)}_{1,0}, \hat{g}^{(i)}_{1,1})$ where
$$\begin{aligned}
\hat{g}^{(i)}_{0,0} &= \alpha_{\mathsf{col}(i),0} \circ g^{(i)}_0 \circ \alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}} \\
\hat{g}^{(i)}_{0,1} &= \alpha_{\mathsf{col}(i),0} \circ g^{(i)}_0 \circ \alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}\oplus 1} \\
\hat{g}^{(i)}_{1,0} &= \alpha_{\mathsf{col}(i),1} \circ g^{(i)}_1 \circ \alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}} \\
\hat{g}^{(i)}_{1,1} &= \alpha_{\mathsf{col}(i),1} \circ g^{(i)}_1 \circ \alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}\oplus 1}.
\end{aligned}$$
4. P_{n+1} receives $\mathsf{Prep}_\pi(n+1; \alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, r_0, r_1) := (S_{b,y})_{(b,y)\in\{0,1\}^2}$, where $S_{b,y} = \{j : \alpha^{-1}_{\mathsf{col}(n),r_{\mathsf{col}(n)}\oplus b}(j) \in \phi^{-1}(y)\}$.

Computation.

1. P_1 sends $\mathsf{Next}_\pi(1, m_0, (r_1, \alpha_{1,0}, \alpha_{1,1}), x_1) := (s_1, v_1)$ to P_2, where $s_1 = r_{\mathsf{col}(1)} \oplus x_1$ and $v_1 = \alpha_{\mathsf{col}(1),x_1} \circ g^{(1)}_{x_1}(1)$.
2. For each $2 \le i \le n$:
   P_i sends $\mathsf{Next}_\pi(i, (s_{i-1}, v_{i-1}), (r_{\mathsf{col}(i)}, \hat{g}^{(i)}_{0,0}, \hat{g}^{(i)}_{0,1}, \hat{g}^{(i)}_{1,0}, \hat{g}^{(i)}_{1,1}), x_i) := (s_i, v_i)$ to P_{i+1}, where $s_i = r_{\mathsf{col}(i)} \oplus x_i$ and $v_i = \hat{g}^{(i)}_{x_i,s_{i-1}}(v_{i-1})$.
3. P_{n+1} outputs $\mathsf{Out}_\pi((s_n, v_n), \{S_{b,y}\}_{b,y\in\{0,1\}}) = y$, where y is s.t. $v_n \in S_{s_n,y}$.

Fig. 1. A PSS protocol to compute f having a 1-SRBP.

Before proving this, we show that it implies correctness. We have, $s_n = r_{\mathsf{col}(n)} \oplus x_n$, and

$$S_{s_n,y} = \{j \in [w] \text{ s.t. } \alpha^{-1}_{\mathsf{col}(n),r_{\mathsf{col}(n)}\oplus s_n}(j) = \alpha^{-1}_{\mathsf{col}(n),x_n}(j) \in \phi^{-1}(y)\}, \forall y \in \{0,1\}.$$

Hence, by eq. (4) (for $i = n$), P_{n+1} outputs y such that

$$\alpha^{-1}_{\mathsf{col}(n),r_{\mathsf{col}(n)}\oplus s_n}(v_n) = \alpha^{-1}_{\mathsf{col}(n),x_n}\left(\alpha_{\mathsf{col}(n),x_n} \circ g^{(n)}_{x_n} \circ g^{(n-1)}_{x_{n-1}} \circ \ldots g^{(1)}_{x_1}(1)\right) \in \phi^{-1}(y).$$

Therefore P_{n+1} outputs y such that $\phi(g^{(n)}_{x_n} \circ g^{(n-1)}_{x_{n-1}} \circ \ldots g^{(1)}_{x_1}(1)) = y$, ensuring correctness.

To conclude the proof of correctness, we prove (4) by induction. Clearly, (4) holds for $i = 1$. Assume that (4) holds for $i-1$. Since $x_{i-1} = s_{i-1} \oplus r_{\mathsf{col}(i-1)}$,

$$\begin{aligned} v_i = \hat{g}^{(i)}_{x_i,s_{i-1}}(v_{i-1}) &= \alpha_{\mathsf{col}(i),x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}\oplus s_{i-1}}(v_{i-1}) \\ &= \alpha_{\mathsf{col}(i),x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{\mathsf{col}(i-1),x_{i-1}}(v_{i-1}) \\ &= \alpha_{\mathsf{col}(i),x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{\mathsf{col}(i-1),x_{i-1}} \circ \alpha^{-1}_{\mathsf{col}(i-1),x_{i-1}} \\ &\quad \circ g^{(i-1)}_{x_{i-1}} \circ \ldots \circ g^{(1)}_{x_1}(1) \\ &= \alpha_{\mathsf{col}(i),x_i} \circ g^{(i)}_{x_i} \circ g^{(i-1)}_{x_{i-1}} \circ \ldots \circ g^{(1)}_{x_1}(1). \end{aligned}$$

Security. The view of P_1 consists of its input and the correlated randomness received during preprocessing. Hence, privacy against P_1 follows trivially.

We next show privacy against P_i for $i \in \{2, \ldots, n\}$. The view of P_i consists of its input and the messages received from P_{i-1} and the correlated randomness received during preprocessing, *viz.*, x_i, s_{i-1}, v_{i-1} and $\{\hat{g}^{(i)}_{b,b'}\}_{b,b' \in \{0,1\}}$. We first simplify the above expression. By eq. (4), $v_{i-1} = \alpha_{\mathsf{col}(i-1),x_{i-1}}(u_{i-1})$, where $u_{i-1} = g^{(i-1)}_{x_{i-1}} \circ \ldots \circ g^{(1)}_{x_1}(1)$. Further, $\{\hat{g}^{(i)}_{b,b'}\}_{b,b' \in \{0,1\}}$ is a function of $\hat{g}^{(i)}_{0,0}, \hat{g}^{(i)}_{1,0}$ and $\alpha_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}} \circ \alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}\oplus 1}$. For brevity, we will denote u_{i-1}, s_{i-1} and x_{i-1} by u, s and x, $r_{\mathsf{col}(i-1)}$ and $r_{\mathsf{col}(i)}$ by r and r'; $g^{(i)}_0$ and $g^{(i)}_1$ by g_0 and g_1; $\alpha_{\mathsf{col}(i),0}$ and $\alpha_{\mathsf{col}(i),0}$ by α_0 and α_1; and $\alpha_{\mathsf{col}(i-1),0}$ and $\alpha_{\mathsf{col}(i-1),1}$ by β_0 and β_1. Recalling the definitions of $\hat{g}^{(i)}_{0,0}, \hat{g}^{(i)}_{1,0}$, the view is determined by

$$\left(x, s, \beta_x(u), r', \alpha_0 \circ g_0 \circ \beta_r^{-1}, \alpha_1 \circ g_1 \circ \beta_r^{-1}, \beta_r \circ \beta_{r\oplus 1}^{-1}\right).$$

Hence, the protocol is private against P_i if the following lemma holds:

Lemma 2. *For all permutations $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\beta}_0, \hat{\beta}_1 \in \mathrm{Sym}(w)$, $b, b' \in \{0,1\}$, and $v \in [w]$, there exists a constant μ such that, for all $x \in \{0,1\}$, and $u \in [w]$,*

$$\mathrm{Prob}\left[\begin{array}{c|c} \alpha_0 \circ g_0 \circ (\beta_r)^{-1} = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0 & \alpha_0, \alpha_1, \beta_0, \beta_1 \leftarrow \mathrm{Sym}(w) \\ \alpha_1 \circ g_1 \circ (\beta_r)^{-1} = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0 & r \leftarrow \{0,1\} \\ \beta_r \circ \beta_{r\oplus 1}^{-1} = \hat{\beta}_0^{-1} \circ \hat{\beta}_1 & s = x \oplus r \\ s = b, \quad r' = b', \quad \beta_x(u) = v & r' \leftarrow \{0,1\} \end{array}\right] = \mu. \tag{5}$$

This is proved in Appendix A. We provide an intuition of the proof. Fix $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\beta}_0, \hat{\beta}_1 \in \mathrm{Sym}(w)$, $b, b' \in \{0,1\}$, and $v \in [w]$. For $x \in \{0,1\}$ and $u \in [w]$, let the LHS of eq. (5) be defined as $\mu(x, w)$. We observe that, there is a well structured set

$$\Lambda = \{\beta_0 \in \mathrm{Sym}(w) : \exists \alpha, \alpha' \in \mathrm{Sym}(w) \text{ s.t. } (g_0 \circ \beta_0 = \alpha \circ g_0) \wedge (g_1 \circ \beta_0 = \alpha' \circ g_0)\},$$

such that

$$\Big\{(\beta_r)^{-1} \in \mathrm{Sym}(w) : \alpha_0 \circ g_0 \circ (\beta_r)^{-1} = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0, \\ \alpha_1 \circ g_1 \circ (\beta_r)^{-1} = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0\Big\} = \{\beta_0 \circ \hat{\beta}_0 : \beta_0 \in \Lambda\}.$$

Further, for all $\beta_0 \circ \hat{\beta}_0$ such that $\beta_0 \in \Lambda$, over the randomness of α_0 and α_1 choosen uniformly and independently from $\mathrm{Sym}(w)$, the events $\alpha_0 \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0$ and $\alpha_1 \circ g_1 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0$ simultaneously occur with the same probability. Using the above two observations, and standard renaming of variables, we simply the expression for $\mu(x, w)$ considerably to get the following: there exists a constant c such that,

$$\mu(x,w)/c = \mathrm{Prob}\left[\begin{matrix} s = b \\ \Gamma_s(u) = v \end{matrix} \middle| \begin{matrix} \beta_0 \leftarrow \Lambda^0 \cap \Lambda^1, \qquad \beta_1 = \beta_0 \circ \hat{\beta}_1 \\ s \leftarrow \{0,1\} \\ \Gamma_0 = (\beta_0 \circ \hat{\beta}_0)^{-1} \quad \Gamma_1 = \beta_1^{-1} \end{matrix}\right].$$

At this point, the following suffices to prove the lemma:

$$\begin{aligned}\mathrm{Prob}\left[(\beta_0 \circ \hat{\beta}_0)^{-1}(u) = v \middle| \beta_0 \leftarrow \Lambda^0 \cap \Lambda^1\right] \\ = \mathrm{Prob}\left[(\beta_0 \circ \hat{\beta}_1)^{-1}(u) = v \middle| \beta_0 \leftarrow \Lambda^0 \cap \Lambda^1\right], \quad \forall u \in [w].\end{aligned}$$

When (g_0, g_1) is strongly regular, we argue that this is indeed the case (see Lemma 8 in Appendix A).

Finally, we prove privacy against P_{n+1}. The view of P_{n+1} is $\{S_{b,y}\}_{(b,y)\in\{0,1\}^2}$, v_n and s_n. We once again, simplify the notation by denoting $x_n, u_n, r_{\mathsf{col}(n)}$, $\alpha_{\mathsf{col}(n),0}$ and $\alpha_{\mathsf{col}(n),1}$ by x, u, r, α_0 and α_1. Since $S_{b,0} = [w] \setminus S_{b,1}$ for $b \in \{0,1\}$, $\{S_{b,y}\}_{b\in\{0,1\},y\in\{0,1\}}$ is a function of $(S_{0,0}, S_{1,0})$ We have $v_n = \alpha_{x_n}(u_n)$, $s_n = x_n \oplus r$, $S_{0,0} = \alpha_r(\phi^{-1}(0))$ and $S_{1,0} = \alpha_{r\oplus 1}(\phi^{-1}(0))$. Here, $\alpha_r(\phi^{-1}(0)) = \{\alpha_r(j) : j \in \phi^{-1}(0)\}$. To prove privacy against P_{n+1}, we will show that, when $x, x' \in \{0,1\}$ and $u, u' \in [w]$ such that $\phi(u) = \phi(u')$,

$$\begin{aligned}(\alpha_r(\phi^{-1}(0)), \alpha_{r\oplus 1}(\phi^{-1}(0)), \alpha_x(u), x \oplus r) \\ \equiv (\alpha_r(\phi^{-1}(0)), \alpha_{r\oplus 1}(\phi^{-1}(0)), \alpha_{x'}(u'), x' \oplus r). \quad (6)\end{aligned}$$

We show this using a sequence of equivalences:

$$\begin{aligned}&(\alpha_r(\phi^{-1}(0)), \alpha_{r\oplus 1}(\phi^{-1}(0)), \alpha_x(u), x \oplus r) \\ &\quad \equiv (\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x\oplus r}(u), x \oplus r) \\ &\quad \equiv (\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x'\oplus r}(u), x' \oplus r). \quad (7)\end{aligned}$$

The first equivalence is obtained by replacing $(\alpha_r, \alpha_{r\oplus 1})$ with the identically distributed pair (α_0, α_1); and the second equivalence is obtained by replacing r with identically distributed $r \oplus x \oplus x'$. Since $\phi(u) = \phi(u')$, there exists $\hat{\alpha} \in \mathrm{Sym}(w)$ such that, $\hat{\alpha}(u) = u'$ and, for all $u'' \in [w]$, $\phi \circ \hat{\alpha}(u'') = \phi(u'')$. We replace (α_0, α_1) with the identically distributed pair $(\alpha_0 \circ \hat{\alpha}, \alpha_1 \circ \hat{\alpha})$ to obtain the following equivalence:

$$\begin{aligned}&(\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x'\oplus r}(u), x' \oplus r) \\ &\quad \equiv (\alpha_0 \circ \hat{\alpha}(\phi^{-1}(0)), \alpha_1 \circ \hat{\alpha}(\phi^{-1}(0)), \alpha_{x'\oplus r} \circ \hat{\alpha}(u), x' \oplus r) \\ &\quad \equiv (\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x'\oplus r}(u'), x' \oplus r). \quad (8)\end{aligned}$$

The second equivalence used the following facts: $\alpha_b \circ \hat{\alpha}(\phi^{-1}(0))$ is identically distributed as $\alpha_b(\phi^{-1}(0))$ for $b \in \{0,1\}$; and $\hat{\alpha}(u) = u'$. Using the reasoning in eq. (7) in the reverse direction, we can show that

$$(\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x' \oplus r}(u'), x' \oplus r) \\ \equiv (\alpha_r(\phi^{-1}(0)), \alpha_{r \oplus 1}(\phi^{-1}(0)), \alpha_{x'}(u'), x' \oplus r). \quad (9)$$

Equations (7) to (9) prove eq. (6) concluding the proof of privacy.

Randomness complexity: Since we need four independent samples from the set of permutations of $[w]$ and two random bits for this protocol, the randomness cost is $\log(2 + 4w!)$ bits, which is $O(w \log w)$. This completes the proof. □

5.2 PSS Protocols From k-SRBP

Normal Form SRBP. In our constructions, for a cleaner presentation, we will consider *normal form* (strongly regular) branching programs. This especially makes the presentation of the conflict graph easier. A length ℓ SRBP is said to be in normal form if it satisfies that for all $t \in [\ell - 1]$, $\sigma(t) \neq \sigma(t + 1)$. That is, the same party doesn't feed inputs to two consecutive layers of the branching program.

It is easy to modify an k-SRBP for a function (with at least 3 inputs) into a normal form $(2k - 1)$-SRBP, with the same width and computing the same function. We show this is in the following lemma which is proved in Appendix D.

Lemma 3. *Any function $f : \{0,1\}^n \to \{0,1\}$, where $n \geq 3$, computable using a k-SRBP is also computable using a $(2k - 1)$-SRBP in the normal form.*

We now present the protocol for computation of functions having k-SRBP. Our construction follows the blueprint of our construction for 1-SRBP. Since a party feeds their input to the branching program only once in an 1-SRBP, we could get away with using the same permutations for masking the state of the branching program in alternating layers, resulting in a protocol that uses only 4 permutations and two bit masks to realize 1-privacy. In a general SRBP, each party can feed their inputs in several layers of the BP. Hence, the main challenge in the protocol is to come up with a strategy for recycling randomness while ensuring that a reappearing party does not learn any intermediate state of the branching program due to this reuse. We define our strategy for randomness reuse using a conflict graph associated with the branching program we want to compute. We present the protocol in the following proof.

Proof of Theorem 3. Suppose f is computable using an k-SRBP

$$\Pi = \left(\sigma, \{g_b^{(t)}\}_{i \in [\ell], b \in \{0,1\}}, \phi\right).$$

We will show that the protocol in Figure 2 computes f with 1-privacy.

A PSS protocol computing f computable using a k-SRBP

Let $\Pi = \left(\sigma, \{g_b^{(t)}\}_{t\in[\ell],b\in\{0,1\}}, \phi\right)$ be a width-w normal form k-SRBP computing the function $f : \{0,1\}^n \to \{0,1\}$. We define a conflict graph $G_\Pi = ([\ell], E)$ for Π, where $\{t, t'\} \in E$ if $t \neq t' \in [\ell]$ and $\{\sigma(t), \sigma(t+1)\} \cap \{\sigma(t'), \sigma(t'+1)\} \neq \emptyset$. Let $\mathsf{col} : [\ell] \to [\chi]$ be a vertex coloring of G_Π. Define $z_t = x_{\sigma(t)}$ for $t \in [\ell]$. Assign $\varsigma_\pi(t) = \sigma(t)$, $t \in [\ell]$ and $\varsigma_\pi(\ell+1) = \mathsf{P}_{n+1}$.

Preprocessing Phase

1. $(\alpha_{c,0}, \alpha_{c,1}, r_c)_{c\in[\chi]} \leftarrow \mathcal{R}$ is sampled where $\mathcal{R} = (\mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \{0,1\})^{|\chi|}$.
2. For $i \in [n]$ such that $i \neq \sigma(1)$, P_i receives

$$\mathsf{Prep}_\pi(i; (\alpha_{c,0}, \alpha_{c,1}, r_c)_{c\in[\chi]}) := \left(r_{\mathsf{col}(t)}, \hat{g}_{0,0}^{(t)}, \hat{g}_{0,1}^{(t)}, \hat{g}_{1,0}^{(t)}, \hat{g}_{1,1}^{(t)}\right)_{t\in[\ell]:\sigma(t)=i},$$

   where

$$\begin{aligned}
\hat{g}_{0,0}^{(t)} &= \alpha_{\mathsf{col}(t),0} \circ g_0^{(t)} \circ \alpha_{\mathsf{col}(t-1),r_{\mathsf{col}(t-1)}} \\
\hat{g}_{0,1}^{(t)} &= \alpha_{\mathsf{col}(t),0} \circ g_0^{(t)} \circ \alpha_{\mathsf{col}(t-1),r_{\mathsf{col}(t-1)}\oplus 1} \\
\hat{g}_{1,0}^{(t)} &= \alpha_{\mathsf{col}(t),1} \circ g_1^{(t)} \circ \alpha_{\mathsf{col}(t-1),r_{\mathsf{col}(t-1)}} \\
\hat{g}_{1,1}^{(t)} &= \alpha_{\mathsf{col}(t),1} \circ g_1^{(t)} \circ \alpha_{\mathsf{col}(t-1),r_{\mathsf{col}(t-1)}\oplus 1}.
\end{aligned}$$

3. $\mathsf{P}_{\sigma(1)}$ receives $\mathsf{Prep}_\pi(\sigma(1); (\alpha_{c,0}, \alpha_{c,1}, r_c)_{c\in[\chi]})$ which is defined as

$$\left((r_{\mathsf{col}(1)}, \alpha_{\mathsf{col}(1),0}, \alpha_{\mathsf{col}(1),1}), \left(r_{\mathsf{col}(t)}, \hat{g}_{0,0}^{(t)}, \hat{g}_{0,1}^{(t)}, \hat{g}_{1,0}^{(t)}, \hat{g}_{1,1}^{(t)}\right)_{t\in[\ell]\setminus\{1\}:\sigma(t)=\sigma(1)}\right),$$

   where the $\hat{g}$ functions are as in Step 2 above.
4. P_{n+1} receives $\mathsf{Prep}_\pi(n+1; (\alpha_{c,0}, \alpha_{c,1}, r_c)_{c\in[\chi]}) := (S_{b,y})_{(b,y)\in\{0,1\}^2}$ where

$$S_{b,y} = \left\{j : \alpha^{-1}_{\mathsf{col}(\ell),r_{\mathsf{col}(\ell)}\oplus b}(j) \in \phi^{-1}(y)\right\}, \quad b, y \in \{0,1\}.$$

Computation

1. $\mathsf{P}_{\sigma(1)}$ sends $\mathsf{Next}_\pi(1, m_0, (r_{\mathsf{col}(1)}, \alpha_{\mathsf{col}(1),0}, \alpha_{\mathsf{col}(1),1}), z_1) := (s_1, v_1)$ to $\mathsf{P}_{\sigma(2)}$, where $s_1 = r_{\mathsf{col}(1)} \oplus z_1$ and $v_1 = \alpha_{\mathsf{col}(1),z_1} \circ \hat{g}_{z_1}^{(1)}(1)$. Note that we defined $z_t = x_{\sigma(t)}$ for $t \in [\ell]$.
2. For $2 \leq t \leq \ell - 1$:
   $\mathsf{P}_{\sigma(t)}$ sends $\mathsf{Next}_\pi(t, (s_{t-1}, v_{t-1}), (r_{\mathsf{col}(t)}, \hat{g}_{0,0}^{(t)}, \hat{g}_{0,1}^{(t)}, \hat{g}_{1,0}^{(t)}, \hat{g}_{1,1}^{(t)}), z_t) := (s_t, v_t)$ to $\mathsf{P}_{\sigma(t+1)}$, where $s_t = r_{\mathsf{col}(t)} \oplus z_t$ and $v_t = \hat{g}_{z_t,s_{t-1}}^{(t)}(v_{t-1})$.
3. $\mathsf{P}_{\sigma(\ell)}$ sends $\mathsf{Next}_\pi(\ell, (s_{\ell-1}, v_{\ell-1}), (r_{\mathsf{col}(\ell)}, \hat{g}_{0,0}^{(\ell)}, \hat{g}_{0,1}^{(\ell)}, \hat{g}_{1,0}^{(\ell)}, \hat{g}_{1,1}^{(\ell)}), z_\ell) := (s_\ell, v_\ell)$ to P_{n+1}, where $s_\ell = r_{\mathsf{col}(\ell)} \oplus z_\ell$ and $v_\ell = \hat{g}_{z_\ell,s_{\ell-1}}^{(\ell)}(v_{\ell-1})$.
4. P_{n+1} outputs $\mathsf{Out}_\pi((s_\ell, v_\ell), \{S_{b,y}\}_{b,y\in\{0,1\}}) = y$, where y is s.t. $v_\ell \in S_{s_\ell,y}$.

Fig. 2. A PSS protocol computing f having an k-SRBP.

Correctness. The computation of s_t for each $t \in [\ell]$ is carried out exactly as in the protocol in Figure 1, except using a different *coloring* function col. Hence, using the same line of argument used to show eq. (4) in the proof of Theorem 2,

$$\begin{aligned} v_t &= \alpha_{\mathsf{col}(t),z_t} \circ g^{(t)}_{z_t} \circ g^{(t-1)}_{z_{t-1}} \circ \ldots \circ g^{(1)}_{z_1}(1), \quad t \in [\ell], \\ v_\ell &= \alpha_{\mathsf{col}(\ell),z_\ell} \circ g^{(\ell)}_{z_\ell} \circ g^{(\ell-1)}_{z_{\ell-1}} \circ \ldots \circ g^{(1)}_{z_1}(1), \\ S_{s_\ell,y} &= \{j \in [w] \text{ s.t. } \alpha^{-1}_{\mathsf{col}(\ell),r_{\mathsf{col}(\ell)} \oplus s_\ell}(j) \in \phi^{-1}(y)\}, \end{aligned} \tag{10}$$

where we have used the notation $z_t = x_{\sigma(t)}$ for $t \in [\ell]$ from Figure 2. Thus, P_{n+1} outputs y such that $y = g^{(\ell)}_{z_\ell} \circ \ldots \circ g^{(1)}_{z_1}(1) = f(x_1, \ldots, x_n)$.

Security. We first show that the protocol is private against P_i for each i such that $\sigma(1) \neq i$, *i.e.*, all the parties except the party who implements the first layer. Fix such a i. Let $(\tilde{x}_1, \ldots, \tilde{x}_n)$ and $(\hat{x}_1, \ldots, \hat{x}_n)$ be any pair of inputs such that $f(\tilde{x}_1, \ldots, \tilde{x}_n) = f(\hat{x}_1, \ldots, \hat{x}_n)$ and $\tilde{x}_i = \hat{x}_i$. We will prove that the view of P_i in an execution of the protocol with $(\tilde{x}_1, \ldots, \tilde{x}_n)$ as inputs is identically distributed as in an execution with $(\hat{x}_1, \ldots, \hat{x}_n)$ as inputs.

Let $\tilde{x}_i = \hat{x}_i = x_i$ and $f(\tilde{x}_1, \ldots, \tilde{x}_n) = f(\hat{x}_1, \ldots, \hat{x}_n) = y$. Let $\tilde{b}_t = \tilde{x}_{\sigma(t)}$ and $\hat{b}_t = \hat{x}_{\sigma(t)}$ for all $t \in [\ell]$. For each $t \in [\ell]$, let $\tilde{u}_t = g^{(t)}_{\tilde{b}_t} \circ \ldots \circ g^{(1)}_{\tilde{b}_1}(1)$ and $\hat{u}_t = g^{(t)}_{\hat{b}_t} \circ \ldots \circ g^{(1)}_{\hat{b}_1}(1)$. The view of P_i in an execution of the protocol with $(\tilde{x}_1, \ldots, \tilde{x}_n)$ as input is

$$\widetilde{\text{View}} = \Big(x_i, y, \Big\{\tilde{s}_{t-1} = r_{\mathsf{col}(t-1)} \oplus \tilde{b}_{t-1}, \tilde{v}_{t-1} = \alpha_{\mathsf{col}(t-1),\tilde{b}_{t-1}}(\tilde{u}_{t-1}), \\ r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\Big\}_{t:\sigma(t)=i}\Big).$$

The view of P_i in an execution of the protocol with $(\hat{x}_1, \ldots, \hat{x}_n)$ as input is

$$\widehat{\text{View}} = \Big(x_i, y, \Big\{\hat{s}_{t-1} = r_{\mathsf{col}(t-1)} \oplus \hat{b}_{t-1}, \hat{v}_{t-1} = \alpha_{\mathsf{col}(t-1),\hat{b}_{t-1}}(\hat{u}_{t-1}), \\ r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\Big\}_{t:\sigma(t)=i}\Big).$$

We will show $\widetilde{\text{View}} \equiv \widehat{\text{View}}$ using a hybrid argument. Let $t_1, \ldots, t_\zeta$ be an arbitrary ordering of the set $\mathcal{L}_i = \{t : \sigma(t) = i\}$ where $\zeta = |\mathcal{L}_i|$. For each $0 \leq h \leq \zeta$ we define a hybrid view

$$\text{Hyb}_h = \left(\begin{array}{l} \\ x_i, y, \\ \\ \end{array} \begin{array}{l} \Big\{r_{\mathsf{col}(t-1)} \oplus \tilde{b}_{t-1}, \alpha_{\mathsf{col}(t-1),\tilde{b}_{t-1}}(\tilde{u}_{t-1}), \\ \qquad r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\Big\}_{t \in \{t_1, \ldots, t_h\}} \\ \Big\{r_{\mathsf{col}(t-1)} \oplus \hat{b}_{t-1}, \alpha_{\mathsf{col}(t-1),\hat{b}_{t-1}}(\hat{u}_{t-1}), \\ \qquad r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\Big\}_{t \in \{t_{h+1}, \ldots, t_\zeta\}} \end{array}\right).$$

Then, $\text{Hyb}_0 = \widetilde{\text{View}}$ and $\text{Hyb}_\zeta = \widehat{\text{View}}$. Hence, $\widetilde{\text{View}} \equiv \widehat{\text{View}}$ if $\text{Hyb}_{h-1} \equiv \text{Hyb}_h$ for all $h \in [\zeta]$, which we prove below: Since (a) col is a coloring of G_Π, and $\mathcal{L}_i$

forms a clique in G_{Π}, and (b) $\sigma(t) \neq \sigma(t+1)$ for any $t \in [\ell-1]$ due to normal form of Π, for any t such that $\sigma(t) = i$,

$$\begin{aligned}&\left(r_{\mathsf{col}(t-1)}, r_{\mathsf{col}(t)}, \{\alpha_{\mathsf{col}(t-1),b}, \alpha_{\mathsf{col}(t),b}\}_{b\in\{0,1\}}\right)\\ &\qquad\qquad \perp\!\!\!\perp \left(r_{\mathsf{col}(t'-1)}, r_{\mathsf{col}(t')}, \{\alpha_{\mathsf{col}(t'-1),c}, \alpha_{\mathsf{col}(t'),c}\}_{c\in\{0,1\}}\right)_{t'\in\mathcal{L}_i\setminus\{t\}}.\end{aligned}$$

Hence, for any $h \in [\zeta]$, to prove that $\mathrm{Hyb}_{h-1} \equiv \mathrm{Hyb}_h$, it suffices to show that for $t = t_h$,

$$\begin{aligned}&\left(r_{\mathsf{col}(t-1)} \oplus \hat{b}_{t-1}, \alpha_{\mathsf{col}(t-1),\hat{b}_{t-1}}(\hat{u}_{t-1}), r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\right)\\ &\qquad \equiv \left(r_{\mathsf{col}(t-1)} \oplus \tilde{b}_{t-1}, \alpha_{\mathsf{col}(t-1),\tilde{b}_{t-1}}(\tilde{u}_{t-1}), r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\right).\end{aligned}$$

But, this follows from Lemma 2; see the proof of privacy against P_i for $2 \leq i \leq n$ in Theorem 2.

Next, we argue privacy against $\mathsf{P}_{\sigma(1)}$. The view of $\mathsf{P}_{\sigma(1)}$ differ from other parties as it receives $\alpha_{\mathsf{col}(1),b} \circ g^{(1)}_b$ for $b \in \{0,1\}$ and $r_{\mathsf{col}(1)}$. Appealing to the properties of graph coloring of G_{Π}, $\alpha_{\mathsf{col}(1),b} \circ g^{(1)}_b$ for $b \in \{0,1\}$ is independent of the remaining part of the view of P_1. Hence, we can prove privacy against P_1 exactly as we proved the privacy against other parties after excluding these parts of the view.

The view of P_{n+1} is exactly the same as that which is given in the protocol in Theorem 2, *i.e.*, (s_ℓ, v_ℓ) and the set $S_{b,y}$ for $b, y \in \{0,1\}$ and therefore the proof of privacy follows from the proof given for Theorem 2.

Randomness complexity. For every color $c \in [\chi]$, the protocol samples fresh random variables α_c (from a permutation of size w) and coin r_c. Therefore the randomness complexity is bounded by $O(2\chi w \log w)$. From Brooks' theorem [Bro41], the number of colors needed to color a graph greedily is Δ_G+1 where Δ_G is its maximum degree. In this case, since an input is read at most $2m-1$ times and from the definition of $G_{\Pi} = (\ell, E)$, $\{t, t'\} \in E$ if $\sigma(t) = \sigma(t')$, $\sigma(t) = \sigma(t'+1)$, $\sigma(t) = \sigma(t'-1)$ or $\sigma(t-1) = \sigma(t'-1)$, there are at most $8m-4$ vertices having an edge with t. Therefore $\Delta_G = 4$. This gives that the randomness complexity is $O(mw \log w)$. □

6 Private Computation of AND

In this section we focus on private computation of the n-party AND function, which has received significant attention in the literature. We present new results regarding upper and lower bounds on the randomness complexity of AND. Thanks to AND having a 1-SRBP (see Section 5), we have a PSS protocol for it, from Figure 1. However, before we can compare our results fairly to prior results, we need to cast our PSS protocol into a setting without an external source supplying correlated randomness and without a separate output party (with all input-parties getting the output, instead). Towards this, we define an

unassisted PSS (uPSS) protocol, which is a private protocol (with uncorrelated randomness) in the sense in Section 3, but is also sequential and stateless, and further has only one party being randomized.

Unassisted Private Sequential Stateless (uPSS) Protocol. A uPSS protocol for a function $f : \{0,1\}^n \to \{0,1\}$ is specified by a tuple $(\mathsf{Prep}_\pi, \varsigma_\pi, \mathsf{Next}_\pi, \mathsf{Out}_\pi)$, similar to a PSS protocol, but with the following differences:

- There are only n parties, $\mathsf{P}_1, \ldots, \mathsf{P}_n$ (no separate output party).
- In the pre-processing phase P_1 samples $R \leftarrow \mathcal{R}$, computes $r_i = \mathsf{Prep}_\pi(i, R)$ and sends it to P_i for each $i \in [n]$.
- After this each party P_i receives its input x_i, and they all carry out the protocol using ς_π and Next_π exactly as in the PSS model.
- In addition, each of them produces an output over the last $[n]$ rounds. For this we require that in a $T+n$-round protocol, $\varsigma_\pi : [T] \to [n]$, when restricted to the domain $\{T-n+1, \ldots, T\}$, is a bijection with $[n]$; also in round $t > T-n$, party P_i, where $i = \varsigma_\pi(t)$, produces the output $\mathsf{Out}_\pi(t, m_{t-1}, x_i, r_i)$.

We require this protocol to be a 1-private protocol (without correlated randomness) for the n-party functionality $\mathcal{F}_f^*$, which is similar to $\mathcal{F}_f$ but delivers the output to all n parties.

6.1 uPSS Protocol for 1-SRBP

Below, we describe the necessary modifications to be made to the protocol in Figure 1 to turn it into a uPSS protocol for a function f with a 1-SRBP.

1. **Preprocessing phase.**
   (a) P_1 samples $(\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, r_0, r_1) \leftarrow \mathcal{R}$ is sampled where $\mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \{0,1\} \times \{0,1\}$ and sends the appropriate correlated randomness r_i to P_i for each $1 \le i \le n$ (including itself) as in the description of the protocol in Figure 1.
   (b) Additionally, P_1 samples γ_0 and γ_1 uniformly from $\mathrm{Sym}(w)$ and a random bit r'; it sends r' and (γ_0, γ_1) to P_n, and the sets $S_{c,y} = \{j : \alpha^{-1}_{\mathsf{col}(n), r' \oplus b} \circ \gamma^{-1}_{r' \oplus b}(j) \in \phi^{-1}(y)\}$ for $b \in \{0,1\}$ and $y \in \{0,1\}$ to P_2.
2. **Computation Phase.**
   (a) For $i = 1, \ldots, n-1$, each P_i (including P_1) follows the instruction in the protocol in Figure 1.
   (b) P_n computes $v_n = \hat{g}^{(n)}_{x_n, s_{n-1}}(v_{n-1})$ as in the previous protocol, but sends (s'_n, v'_n) to P_2, where $s'_n = r' \oplus x_n$ and $v'_n = \gamma_{x_n}(v_n)$.
   (c) P_2 computes y such that $v'_n \in S_{s'_n, y}$.
   (d) Over the next n rounds, each party (starting with P_2) outputs y and sends it to the next party to output.

Theorem 8. *Suppose $f : \{0,1\}^n \to \{0,1\}$ is computable using a 1-SRBP of width w. There exists a uPSS protocol that realizes $\mathcal{F}_f^*$ with 1-privacy using $O(w \log w)$ bits of randomness, all sampled by a single party.*

Proof. The proof of security and correctness follow closely to that of the protocol in Figure 1 which we established in Theorem 2. Throughout the proof, we refer to this as the 'previous protocol'.

Correctness. The computation of (s_i, v_i) for $1 \leq i \leq n$, proceeds exactly as in the previous protocol. Hence, $v_n = \alpha_{\mathsf{col}(n),x_n} \circ g_{x_n}^{(n)} \circ g_{x_{n-1}}^{(n-1)} \circ \ldots \circ g_{x_1}^{(1)}(1)$, and $v'_n = \gamma_{x_n} \circ \alpha_{\mathsf{col}(n),x_n} \circ g_{x_n}^{(n)} \circ \ldots \circ g_{x_1}^{(1)}(1)$. Since $s'_n = r' \oplus x_n$,

$$S_{s'_n,y} = \{j \in [w] \text{ s.t. } \alpha^{-1}_{\mathsf{col}(n),r' \oplus s'_n} \circ \gamma^{-1}_{r' \oplus s'_n}(j) = \alpha^{-1}_{\mathsf{col}(n),x_n} \circ \gamma^{-1}_{x_n}(j) \in \phi^{-1}(y)\}.$$

Hence, P_2 outputs y such that

$$\begin{aligned}\alpha^{-1}_{\mathsf{col}(n),x_n} \circ \gamma^{-1}_{x_n}(v'_n) &= \alpha^{-1}_{\mathsf{col}(n),x_n} \circ \gamma^{-1}_{x_n} \circ \gamma_{x_n} \circ \alpha_{\mathsf{col}(n),x_n} \circ g_{x_n}^{(n)} \circ g_{x_{n-1}}^{(n-1)} \circ \ldots g_{x_1}^{(1)}(1) \\ &= g_{x_n}^{(n)} \circ g_{x_{n-1}}^{(n-1)} \circ \ldots g_{x_1}^{(1)}(1) \in \phi^{-1}(y).\end{aligned}$$

Thus, $y = f(x_1, \ldots, x_n)$.

Security. The only message received by P_1 throughout the protocl is y from P_n. We have established $y = f(x_1, \ldots, x_n)$, hence the protocol is secure against P_1. For $3 \leq i < n$, the view of P_i is identical to that in the previous protocol (with the exception of y that they receive at the end of the new protocol). Hence, security against them follow from our argument in Theorem 2. The view of P_n additionally consists of r' and functions (γ_0, γ_{r_1}). But, these random variables are sampled independent of all the other messages received by P_n. Hence, the security against P_n in the new protocol follows from that in the old protocol.

Finally, we argue security against P_2. Compared to its view in the previous protocol, the view of P_2 in the new protocol additionally contains $(S_{0,y}, S_{1,y})$ for $y \in \{0,1\}$ that it received in the preprocessing phase, and (s'_n, v'_n) that it receives from P_n. Since (γ_0, γ_1) are sampled independent of $(\alpha_{\mathsf{col}(i),0}, \alpha_{\mathsf{col}(i),1})$ for all $i \in [n]$, the additional values in the view of P_2 are independent of all the other messages it received. Further, the view of P_2 in the previous protocol is established to be secure. Hence, to argue security against P_2, it suffices to show that the additional messages received by P_2 in the new protocol do not break security. But, it can be seen by inspection that $(S_{0,y}, S_{1,y})$ for $y \in \{0,1\}$ and s'_n, v'_n are distributed exactly as the view of P_{n+1} in the previous protocol. But, the previous protocol is secure against P_{n+1} who does not have any input to the protocol. The security against P_2 in the new protocol now follows from the security of the previous protocol against P_2 and P_{n+1}.

Randomness Complexity: Since we need 6 independent samples from the set of permutations of $[w]$ $(\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, \gamma_0, \gamma_1)$ and three binary coins (r_0, r_1, r'), the randomness cost of the given protocol is $3+6\log(w!)$ bits, which is $O(w \log w)$ bits. □

Since AND has a width-2 branching program, the following corollary follows immediately.

Corollary 1. *There exists a uPSS protocol for $\mathcal{F}^*_{\mathsf{AND}}$ with randomness cost of 9 bits.*

However, when the number of inputs n is odd, we obtain an optimization to 6 bits, which matches the state-of-the-art result for (non uPSS) 1-private computation of $\mathcal{F}^*_{\mathsf{AND}}$.[8] The optimized protocol is described in Figure 3. Instead of presenting the full protocol, we describe how the protocol differs from Figure 1. We remark that this optimization is possible only because of the properties of AND function. We crucially use the fact that in AND, if a party has 0 as input, it can essentially ignore the information it received so far. Furthermore, we rely on the total number of parties being odd to make the message sent by last party independent of the rest of the view for one of the parties. With this optimization, we bring down the randomness cost from 9 bits in the general uPSS protocol to 6 bits. The following theorem states this fact and it is proved in Appendix B.

Theorem 9. *There exists a uPSS protocol for $\mathcal{F}^*_{\mathsf{AND}}$ for an odd number of parties, with randomness cost of 6 bits.*

Preprocessing – P_1 samples $(\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, r_0, r_1) \leftarrow \mathcal{R}$ where $\mathcal{R} = (\mathrm{Sym}(2) \times \mathrm{Sym}(2) \times \mathrm{Sym}(2) \times \mathrm{Sym}(2) \times \{0,1\} \times \{0,1\})$.
- For $3 \leq i \leq n$, P_1 sends the appropriate correlated random variables r_i to P_i as given in the description of the protocol in Figure 1.
- P_1 sends $(\alpha_{0,0}, r_0, \tilde{\alpha}_0, \tilde{\alpha}_1)$ to P_2 where $\tilde{\alpha}_0 = \alpha_{0,r_1}$ and $\tilde{\alpha}_1 = \alpha_{0,r_1 \oplus 1}$.

Computation – P_1 sends $v_1 = \alpha_{0,1} \circ g^{(1)}_{x_1}(1)$ to P_2.
- P_2 sends (s_2, v_2) to P_3 where $v_2 = \begin{cases} v_1 & x_2 = 1 \\ \alpha_{0,0} & x_2 = 0 \end{cases}$ and $s_2 = x_2 \oplus r_0$.
- For each $i \in \{3, \ldots, n-1\}$, P_i computes and sends (s_i, v_i) to P_{i+1} where (s_i, v_i) is defined as per the description of Figure 1.
- P_n sends (s_n, v_n) to P_2 where (s_n, v_n) is defined as per the description of Figure 1.
- P_2 computes and outputs $y = 1$ if $\tilde{\alpha}^{-1}_{s_n}(v_n) = 1$ $y = 0$ if $\tilde{\alpha}^{-1}_{s_n}(v_n) = 2$ and sends to P_3.
- For $i \in \{3, \ldots, n-1\}$, P_i sends y to P_{i+1}.
- P_n sends y to P_1.

Fig. 3. Optimized uPSS protocol for n-party $\mathcal{F}^*_{\mathsf{AND}}$ for odd n

6.2 Lower bound on randomness complexity of AND for 3 parties

In this section, we prove Theorem 5, namely that the randomness complexity of AND for 3 parties is at least 3 bits.

[8] For even n too, the randomness cost can be improved from 9 to $6 + \log_2 3$ bits. We omit this optimization as it still falls short of the non-uPSS state-of-the-art.

We show this lower bound through a reduction from a secret sharing problem. Recently, [ARN+23] characterized the randomness complexity of the following problem, referred to as 3-Secret Sharing (3SS): Let $\mathcal{S} \subseteq \{0,1\}^3$. Given $(x_1, x_2, x_3) \in \mathcal{S}$, a dealer produces three different shares $W_{1,2}, W_{2,3}$ and $W_{3,1}$ such that the pair of shares $(W_{1,2}, W_{2,3})$ together reveals x_2 and nothing more about (x_1, x_3) – i.e., nothing other than what can be inferred from learning x_2 and the fact that $(x_1, x_2, x_3) \in \mathcal{S}$. Similarly x_1 (and nothing more about (x_2, x_3)) can be obtained from shares $(W_{1,2}, W_{3,1})$, while $(W_{2,3}, W_{3,1})$ reveals x_3 and nothing more.

We shall argue that a 1-private 3-party MPC protocol for the AND function yields a 3SS scheme for the set $\mathcal{S} = \{0,1\}^3 \setminus \{(1,1,1)\}$ as follows: To share $(x_1, x_2, x_3) \in \mathcal{S}$, let $W_{i,j}$ denote the transcript between parties P_i and P_j in the MPC protocol for AND, in which the input of each party P_i is x_i. Note that $(W_{1,2}, W_{2,3})$ together is part of the view of party P_2 in the MPC protocol, and reveals nothing more about x_1, x_3 beyond what x_2 and $x_1 \wedge x_2 \wedge x_3$ reveals. But the latter only reveals that $(x_1, x_2, x_3) \in \mathcal{S}$. The analogous conditions hold for $(W_{2,3}, W_{3,1})$ (for party P_3) and $(W_{3,1}, W_{1,2})$ (for party P_1). Hence this scheme satisfies the privacy condition of 3SS. It remains to check that this scheme also meets the correctness conditions of 3SS, namely that $(W_{1,2}, W_{2,3})$ do determine x_2, and so on. We verify this in the following lemma, which is proved in Appendix E using elementary arguments based on the properties of private protocols.

Lemma 4. *In any 1-private 3-party MPC protocol for the 3-party AND function where all parties learn the output, the pair of transcripts in the view of each party uniquely determines its input.*

Theorem 5 now follows from the fact that for $\mathcal{S} = \{0,1\}^3 \setminus \{(1,1,1)\}$ 3SS has a lower bound of 3 bits on randomness complexity [ARN+23].

In contrast, the best known upper bound is 6 bits [CR22], and we leave it as an open problem to bridge this gap. Incidentally, our approach cannot yield a higher lower bound than 3, since the randomness complexity of 3SS for $\{0,1\}^3 \setminus \{(1,1,1)\}$ is exactly 3 bits.

References

ARN+23. Hari Krishnan P. Anilkumar, Aayush Rajesh, Varun Narayanan, Manoj M. Prabhakaran, and Vinod M. Prabhakaran. Randomness requirements for three-secret sharing. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 252–257. IEEE, 2023.

Bar86. David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 1–5, 1986.

BGP99. Carlo Blundo, Clemente Galdi, and Pino Persiano. Randomness recycling in constant-round private computations (extended abstract). In

Prasad Jayanti, editor, *Distributed Computing, 13th International Symposium, Bratislava, Slovak Republic, September 27-29, 1999, Proceedings*, volume 1693 of *Lecture Notes in Computer Science*, pages 138–150. Springer, 1999.

BGP07. Carlo Blundo, Clemente Galdi, and Giuseppe Persiano. Low-randomness constant-round private XOR computations. *Int. J. Inf. Sec.*, 6(1):15–26, 2007.

Bro41. Rowland Leonard Brooks. On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 37, pages 194–197. Cambridge University Press, 1941.

BSPV99. Carlo Blundo, Alfredo De Santis, Giuseppe Persiano, and Ugo Vaccaro. Randomness complexity of private computation. *Comput. Complex.*, 8(2):145–168, 1999.

CKOR00. Ran Canetti, Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Randomness versus fault-tolerance. *Journal of cryptology*, 13:107–142, 2000.

CR22. Geoffroy Couteau and Adi Rosén. Random sources in private computation. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 443–473. Springer, Heidelberg, December 2022.

DPP16. Deepesh Data, Vinod M. Prabhakaran, and Manoj M. Prabhakaran. Communication and randomness lower bounds for secure computation. *IEEE Trans. Inf. Theory*, 62(7):3901–3929, 2016.

FKN94. Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 554–563, 1994.

GIKM98. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 151–160, 1998.

GIS22. Vipul Goyal, Yuval Ishai, and Yifan Song. Tight bounds on the randomness complexity of secure multiparty computation. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 483–513. Springer, Heidelberg, August 2022.

GR03. Anna Gál and Adi Rosén. Lower bounds on the amount of randomness in private computation. In *35th ACM STOC*, pages 659–666. ACM Press, June 2003.

IK97. Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pages 174–183. IEEE, 1997.

IKO+11. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*, pages 406–425. Springer, 2011.

INW94. Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 356–364, 1994.

JLR03. Andreas Jakoby, Maciej Liskiewicz, and Rüdiger Reischuk. Private computations in networks: Topology versus randomness. In Helmut Alt and Michel Habib, editors, *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March*

1, 2003, Proceedings, volume 2607 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2003.

KM96. Eyal Kushilevitz and Yishay Mansour. Randomness in private computations. In James E. Burns and Yoram Moses, editors, *15th ACM PODC*, pages 181–190. ACM, August 1996.

KOP+19. Eyal Kushilevitz, Rafail Ostrovsky, Emmanuel Prouff, Adi Rosén, Adrian Thillard, and Damien Vergnaud. Lower and upper bounds on the randomness complexity of private computations of AND. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 386–406. Springer, Heidelberg, December 2019.

KOR96. Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Characterizing linear size circuits in terms of privacy. In *28th ACM STOC*, pages 541–550. ACM Press, May 1996.

KOR98. Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Amortizing randomness in private multiparty computations. In Brian A. Coan and Yehuda Afek, editors, *17th ACM PODC*, pages 81–90. ACM, June / July 1998.

KR94. Eyal Kushilevitz and Adi Rosén. A randomnesss-rounds tradeoff in private computation. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 397–410. Springer, Heidelberg, August 1994.

LPV23. Chin Ho Lee, Edward Pyne, and Salil Vadhan. On the power of regular and permutation branching programs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

RU19. Adi Rosén and Florent Urrutia. A new approach to multi-party peer-to-peer communication complexity. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 64:1–64:19. LIPIcs, January 2019.

Appendices

A Proofs of Lemma 2

Lemma 5 (Restatement of Lemma 2). *If (g_0, g_1) is strongly regular, for all permutations $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\beta}_0, \hat{\beta}_1 \in \mathrm{Sym}(w)$, $b \in \{0,1\}$, and $v \in [w]$, there exists a constant μ such that, for all $x \in \{0,1\}$, and $u \in [w]$,*

$$\mathrm{Prob}\left[\begin{array}{l|c} \alpha_0 \circ g_0 \circ (\beta_r)^{-1} = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0 & \alpha_0, \alpha_1, \beta_0, \beta_1 \leftarrow \mathrm{Sym}(w) \\ \alpha_1 \circ g_1 \circ (\beta_r)^{-1} = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0 & r \leftarrow \{0,1\} \\ \beta_r \circ \beta_{r\oplus 1}^{-1} = \hat{\beta}_0^{-1} \circ \hat{\beta}_1 & s = x \oplus r \\ s = b, \quad r' = b', \quad \beta_x(u) = v & r' \leftarrow \{0,1\} \end{array}\right] = \mu \qquad (11)$$

We first prove the following lemmas that will be used in proving Lemma 2.

Lemma 6. *For each $b \in \{0,1\}$, define $\Lambda^b = \{\lambda \in \mathrm{Sym}(w) : \exists \rho \in \mathrm{Sym}(w)$ s.t. $g_b \circ \lambda = \rho \circ g_b\}$. For any $\alpha, \beta \in \mathrm{Sym}(w)$, define $\Lambda^0_{\alpha,\beta} = \{\lambda \in \mathrm{Sym}(w) : \exists \rho \in \mathrm{Sym}(w)$ s.t. $\alpha \circ g_0 \circ \beta = \rho \circ g_0 \circ \lambda \circ \beta\}$. Then, $\Lambda^0_{\alpha,\beta} = \Lambda^0$. Similarly, $\Lambda^1_{\alpha,\beta} = \{\lambda \in \mathrm{Sym}(w) : \exists \rho \in \mathrm{Sym}(w)$ s.t. $\alpha \circ g_1 \circ \beta = \rho \circ g_1 \circ \lambda \circ \beta\} = \Lambda^1$*

Proof. We will show that $\Lambda^0_{\alpha,\beta} = \Lambda^0$; the other statement can be shown analogously. We first prove that $\Lambda^0 \subseteq \Lambda^0_{\alpha,\beta}$. Let $\lambda \in \Lambda^0$, and let ρ be such that $g_0 \circ \lambda = \rho \circ g_0$. Then,

$$(\alpha \circ \rho^{-1}) \circ g_0 \circ \lambda \circ \beta = \alpha \circ \rho^{-1} \circ \rho \circ g_0 \circ \beta = \alpha \circ g_0 \circ \beta \implies \lambda \in \Lambda^0_{\alpha,\beta}.$$

Next, we prove $\Lambda^0_{\alpha,\beta} \subseteq \Lambda^0$: Let $\lambda \in \Lambda^0_{\alpha,\beta}$, and let ρ be such that $\alpha \circ g_0 \circ \beta = \rho \circ g_0 \circ \lambda \circ \beta$. Then,

$$\rho^{-1} \circ \alpha \circ g_0 = \rho^{-1} \circ (\alpha \circ g_0 \circ \beta) \circ \beta^{-1} = \rho^{-1} \circ (\rho \circ g_0 \circ \lambda \circ \beta) \circ \beta^{-1} = g_0 \circ \lambda \implies \lambda \in \Lambda^0.$$

This concludes the proof. □

Lemma 7. *For any pair of permutations $\alpha, \beta \in \mathrm{Sym}(w)$,*

$$|\{\rho \in \mathrm{Sym}(w) : \rho \circ g_0 \circ \beta = \alpha \circ g_0 \circ \beta\}| = |\{\rho \in \mathrm{Sym}(w) : \rho \circ g_0 = g_0\}|.$$

Proof. $\alpha \circ g_0 \circ \beta = \rho \circ g_0 \circ \beta$ implies that $\alpha^{-1} \circ \rho \circ g_0 = g_0$. The claim now follows from the fact that $\alpha^{-1} \circ \rho = \alpha^{-1} \circ \rho'$ if and only if $\rho = \rho'$. □

Lemma 8. *For all x, y and y',*

$$\mathrm{Prob}\left[\lambda(x) = y \middle| \lambda \leftarrow \Lambda^0 \cap \Lambda^1\right] = \mathrm{Prob}\left[\lambda(x) = y' \middle| \lambda \leftarrow \Lambda^0 \cap \Lambda^1\right] \qquad (12)$$

if and only if the pair (g_0, g_1) is strongly regular.

Proof. For each $(u,v) \in E$, define $S_{u,v} = g_0^{-1}(u) \cap g_1^{-1}(v)$. Enforce an arbitrary ordering over each $S_{u,v}$. For any $x \in [w]$, there exists unique (u,v) and $i \in S_{u,v}$ such that x is 'the ith element of $S_{u,v}$'. Hence, each $x \in [w]$ can be equivalently represented as (i,u,v) for some $(u,v) \in E$ and $i \in [|S_{u,v}|]$. In the sequel, we will often refer to a member of $[w]$ as (i,u,v) where $i \in [|S_{u,v}|]$ and $(u,v) \in E$.

Define

$$\Lambda = \left\{ \lambda \in \mathrm{Sym}(w) : \begin{array}{r} \exists (\mu,\nu) \in \mathrm{Aut}(H), \text{ and } \forall (u,v) \in E, \exists \lambda_{u,v} \in \mathrm{Sym}(|S_{u,v}|), \\ \text{s.t. } \forall (u,v) \in E, i \in [|S_{u,v}|], \\ \lambda((i,u,v)) = (\lambda_{u,v}(i), \mu(u), \nu(v)) \end{array} \right\}. \tag{13}$$

We will first show that $\Lambda^0 \cap \Lambda^1 = \Lambda$.

Proof of $\Lambda \subseteq \Lambda^0 \cap \Lambda^1$. Suppose $\lambda \in \Lambda$. Let $(\mu,\nu) \in \mathrm{Aut}(H)$ and $\lambda_{u,v} \in \mathrm{Sym}(|S_{u,v}|)$ for each $(u,v) \in E$ be such that

$$\lambda((i,u,v)) = (\lambda_{u,v}(i), \mu(u), \nu(v)), \forall (u,v) \in E, i \in [|S_{u,v}|].$$

Recall, $x \in [w]$ is identified with (i,u,v) such that x is the ith element in $S_{u,v}$, where $g_0(x) = u$ and $g_1(x) = v$. Hence, $g_0((i,u,v)) = u$ and $g_1((i,u,v)) = v$ for all $x \in [w]$. Thus, for all $(i,u,v) \in [w]$,

$$g_0 \circ \lambda((i,u,v)) = g_0((\lambda_{u,v}(i), \mu(u), \nu(v))) = \mu(u) = \mu \circ g_0((i,u,v)).$$

Since μ is a permutation of $[w]$, $\lambda \in \Lambda^0$. Similarly, $\lambda \in \Lambda^1$ since

$$g_1 \circ \lambda((i,u,v)) = \nu \circ g_1((i,u,v)) \text{ for all } (i,u,v).$$

Proof of $\Lambda^0 \cap \Lambda^1 \subseteq \Lambda$. We will show that $\lambda \notin \Lambda^0 \cap \Lambda^1$ if $\lambda \notin \Lambda$. Let $\lambda \in \mathrm{Sym}(w)$. If $\lambda \notin \Lambda$, we claim there exist $x, x' \in [w]$ such that (i.) $g_0(x) = g_0(x')$ but $g_0 \circ \lambda(x) \neq g_0 \circ \lambda(x')$ or (ii). $g_1(x) = g_1(x')$ but $g_1 \circ \lambda(x) \neq g_1 \circ \lambda(x')$. In case (i), $\lambda \notin \Lambda^0$ by definition of Λ^0, and in case (i), $\lambda \notin \Lambda^1$. Thus, $\Lambda^0 \cap \Lambda^1 \subseteq \Lambda$.

We will prove the contrapositive of the above claim. Suppose, for any x, x' such that $g_0(x) = g_0(x')$, we have $g_0 \circ \lambda(x) = g_0 \circ \lambda(x')$. Then, for every u, there exists $\hat{u}$ such that, for any v such that $(u,v) \in E$ and $i \in [|S_{u,v}|]$, $\lambda((i,u,v)) = (\hat{i},\hat{u},\hat{v})$ for some $\hat{v}$ such that $(\hat{u},\hat{v}) \in E$ and $\hat{i} \in [|S_{\hat{u},\hat{v}}|]$. This is an immediate consequence of the fact that, for any $(i,u,v) \in [w]$, $g_0(i,u,v) = u$. Similarly, if, for any x, x' such that $g_1(x) = g_1(x')$, we have $g_1 \circ \lambda(x) \neq g_1 \circ \lambda(x')$. Then, for any v, there exists $\hat{v}$ such that for any v such that $(u,v) \in E$ and $i \in [|S_{u,v}|]$, $\lambda((i,u,v)) = (\hat{i},\hat{u},\hat{v})$ for some $\hat{v}$ such that $(\hat{u},\hat{v}) \in E$ and $\hat{i} \in [|S_{\hat{u},\hat{v}}|]$. Finally, since λ is a permutation of $[w]$, $\lambda((i,u,v)) \neq \lambda((i',u',v'))$ if $(i,u,v) \neq (i',u',v')$. Using these three observations, we conclude that, if

$$\begin{aligned} g_0(x) = g_0(x') \implies g_0 \circ \lambda(x) = g_0 \circ \lambda(x') \text{and} \\ g_1(x) = g_1(x') \implies g_1 \circ \lambda(x) = g_1 \circ \lambda(x'), \forall x, x' \in [w], \end{aligned} \tag{14}$$

then, there exist $(\mu, \nu) \in \mathrm{Aut}(H)$ and $\{\lambda_{u,v} \in \mathrm{Sym}(|S_{u,v}|)\}_{(u,v)\in E}$ such that, for all $(i, u, v) \in [w]$, $\lambda((i, u, v)) = (\lambda_{u,v}(i), \mu(u), \nu(v))$; in other words, $\lambda \in \Lambda$.

We have showed that $\Lambda^0 \cap \Lambda^1 = \Lambda$. Therefore, $\forall\, (i, u, v), (i', u', v'), (i'', u'', v'') \in [w]$,

$$\begin{aligned}\mathrm{Prob}\left[\lambda((i,u,v)) = (i',u',v') \middle| \lambda \leftarrow \Lambda^0 \cap \Lambda^1\right] \\ = \mathrm{Prob}\left[\lambda((i,u,v)) = (i'',u'',v'') \middle| \lambda \leftarrow \Lambda^0 \cap \Lambda^1\right]\end{aligned}$$

if and only if

$$\mathrm{Prob}\left[\lambda((i,u,v)) = (i',u',v') | \lambda \leftarrow \Lambda\right] = \mathrm{Prob}\left[\lambda((i,u,v)) = (i'',u'',v'') | \lambda \leftarrow \Lambda\right] \tag{15}$$

Suppose (g_0, g_1) is strongly regular. Note, by the second property of strongly regular functions (Definition 3), there exists c such that $|S_{u,v}| = c$ for all $(u, v) \in E$ when (g_0, g_1) is strongly regular. Then, for any $(\mu, \nu) \in \mathrm{Aut}(H)$ and $\lambda_{u,v} \in \mathrm{Sym}(c)$, the map $\lambda : ((i, u, v)) \mapsto (\lambda_{u,v}, \mu(u), \nu(v))$ for all $(i, u, v) \in [w]$ belongs to Λ. Then, for any $(i, u, v), (i', u', v') \in [w]$,

$$\begin{aligned}&\mathrm{Prob}\left[\lambda((i,u,v)) = (i',u',v') | \lambda \leftarrow \Lambda\right] \\ &\quad = \mathrm{Prob}\left[\lambda_{u,v}(i) = i', (\mu(u), \nu(v)) = (u', v') | \lambda_{u,v} \leftarrow \mathrm{Sym}(c), (\mu, \nu) \leftarrow \mathrm{Aut}(H)\right] \\ &\quad = \mathrm{Prob}\left[\lambda_{u,v}(i) = i' | \lambda_{u,v} \leftarrow \mathrm{Sym}(c)\right] \\ &\qquad \times \mathrm{Prob}\left[(\mu(u), \nu(v)) = (u', v') | (\mu, \nu) \leftarrow \mathrm{Aut}(H)\right] \\ &\quad = 1/(w \cdot |E|).\end{aligned}$$

Here, the final equality used the third property eq. (3) of strongly regular functions.

Suppose (g_0, g_1) is not strongly regular. We will first establish that, if the first or second condition in Definition 3 is not met, it is easy to see that there exist $(u, v), (u', v') \in E$ such that, there exists no $\lambda \in \Lambda$ for which $\lambda((i, u, v)) = (\hat{i}, \hat{u}, \hat{v})$ for some $i \in [|S_{u,v}|]$ and $i' \in [|S_{u',v'}|]$. This will contradict eq. (15) when $i, i', i'' = 1$, $(u'', v'') = (u, v)$.

Suppose $(u, v), (u', v') \in E$ but $|S_{u,v}| \neq |S_{u',v'}|$; this occurs if (g_0, g_1) contradicts the second condition in Definition 3. In this case, there exists no $\lambda \in \Lambda$ such that $\lambda((i, u, v)) = (i', u', v')$ for some $i' \in [|S_{u',v'}|]$. This is a consequence of λ being one-to-one and $|S_{u,v}| \neq |S_{u',v'}|$. Next, suppose there exists $u, u' \in [w]$ such that $|\{v : (u, v) \in E\}| \neq |\{v : (u', v) \in E\}|$; i.e., degree of u and u' in H are distinct and non-zero. Note, this is true if the second condition in Definition 3 is met but the first condition is not met. In this case, there exists no $(\mu, \nu) \in \mathrm{Aut}(H)$ such that $(\mu(u), \nu(v)) = (u', v')$ for any v, v' such that $(u, v), (u', v') \in E$. We are left with (g_0, g_1) that satisfy the first two conditions but fails to satisfy the third condition. Since $|S_{u,v}| = c$ for all $(u, v) \in E$ for some c, as we previously observed, for any $(\mu, \nu) \in \mathrm{Aut}(H)$ and $\lambda_{u,v} \in \mathrm{Sym}(c)$, $\lambda : ((i, u, v)) \mapsto (\lambda_{u,v}, \mu(u), \nu(v))$ for all $(i, u, v) \in [w]$ belongs to Λ. Hence, eq. (15) holds for all $(i, u, v), (i', u', v'), (i'', u'', v'') \in [w]$ only if the third condition in Definition 3 is met. This concludes the proof of the lemma. □

Proof of Lemma 2. Fix $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\beta}_0, \hat{\beta}_1 \in \mathrm{Sym}(w)$, $b \in \{0,1\}$, and $v \in [w]$. In the sequel, we will denote $r \oplus 1$ by $\bar{r}$. For any x, u, let the LHS of eq. (11) be defined as $\mu(x,u)$. We can rewrite $\mu(x,u)$ as follows by replacing $(\beta_r^{-1}, \beta_{\bar{r}}^{-1})$ by (β_0, β_1), and noting that r' is independent of all the other random variables.

$$\begin{aligned}\mu(x,u) = \mathrm{Prob}\left[r' = b' | r' \leftarrow \{0,1\}\right] \\ \times \mathrm{Prob}\left[\begin{matrix}\alpha_0 \circ g_0 \circ \beta_0 = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0 \\ \alpha_1 \circ g_1 \circ \beta_0 = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0 \\ \beta_0^{-1} \circ \beta_1 = \hat{\beta}_0^{-1} \circ \hat{\beta}_1 \\ s = b \qquad \beta_{x \oplus r}^{-1}(u) = v\end{matrix} \middle| \begin{matrix}\alpha_0, \alpha_1, \beta_0, \beta_1 \leftarrow \mathrm{Sym}(w) \\ r \leftarrow \{0,1\} \\ s = x \oplus r\end{matrix}\right]\end{aligned} \tag{16}$$

This can be further rewritten as follows by replacing (β_0, β_1) by $(\beta_0 \circ \hat{\beta}_0, \beta_1)$, and defining $\Gamma_0 = (\beta_0 \circ \hat{\beta}_0)^{-1}$ and $\Gamma_1 = \beta_1^{-1}$.

$$\mu(x,u) = \frac{1}{2} \cdot \mathrm{Prob}\left[\begin{matrix}\alpha_0 \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0 \\ \alpha_1 \circ g_1 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0 \\ (\beta_0 \circ \hat{\beta}_0)^{-1} \circ \beta_1 = \hat{\beta}_0^{-1} \circ \hat{\beta}_1 \\ s = b \qquad \Gamma_s(u) = v\end{matrix} \middle| \begin{matrix}\alpha_0, \alpha_1, \beta_0, \beta_1 \leftarrow \mathrm{Sym}(w) \\ s \leftarrow \{0,1\}\end{matrix}\right] \tag{17}$$

For any $\beta_0, \hat{\beta}_0 \in \mathrm{Sym}(w)$, denoting the indicator function by I,

$$\begin{aligned}&|\{\alpha : \alpha \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha} \circ g_0 \circ \hat{\beta}_0\}| \\ &\quad = I(\exists \tilde{\alpha} : \tilde{\alpha} \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha} \circ g_0 \circ \hat{\beta}_0) \\ &\qquad\qquad \times |\{\alpha : \alpha \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \tilde{\alpha} \circ g_0 \circ \beta_0 \circ \hat{\beta}_0\}| \\ &\quad = I(\exists \tilde{\alpha} : \tilde{\alpha} \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha} \circ g_0 \circ \hat{\beta}_0) \\ &\qquad\qquad \times |\{\alpha' : \alpha' \circ g_0 = g_0\}|.\end{aligned}$$

Here, the second equality follows from Lemma 7. A similar condition holds for g_1 as well. Hence, for any $c \in \{0,1\}$,

$$\begin{aligned}&\mathrm{Prob}\left[\alpha_c \circ g_c \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_c \circ g_c \circ \hat{\beta}_0 \middle| \alpha_c, \beta_0 \leftarrow \mathrm{Sym}(w)\right] \\ &\quad = \mathrm{Prob}\left[g_c = \alpha_c \circ g_c | \alpha_c \leftarrow \mathrm{Sym}(w)\right] \\ &\qquad \times \mathrm{Prob}\left[\exists \alpha_c : \alpha_c \circ g_c \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_c \circ g_c \circ \hat{\beta}_0 \middle| \beta_0 \leftarrow \mathrm{Sym}(w)\right].\end{aligned}$$

Since α_0 and α_1 are uniformly sampled independent of each other and other random variables, by the above observation,

$$\mu(x,u) = \frac{1}{2} \cdot \text{Prob}\left[g_0 = \alpha_0 \circ g_0 | \alpha_0 \leftarrow \text{Sym}(w)\right] \tag{18}$$

$$\times \text{Prob}\left[g_1 = \alpha_1 \circ g_1 | \alpha_1 \leftarrow \text{Sym}(w)\right] \tag{19}$$

$$\times \text{Prob}\left[\begin{matrix}\exists \alpha_0 : \alpha_0 \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0 \\ \exists \alpha_1 : \alpha_1 \circ g_1 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0 \\ (\beta_0 \circ \hat{\beta}_0)^{-1} \circ \beta_1 = \hat{\beta}_0^{-1} \circ \hat{\beta}_1 \\ s = b \qquad \Gamma_s(u) = v\end{matrix} \middle| \begin{matrix}\beta_0, \beta_1 \leftarrow \text{Sym}(w) \\ s \leftarrow \{0,1\}\end{matrix}\right] \tag{20}$$

Using the definition of $\Lambda^0_{\hat{\alpha_0},\hat{\beta}_0}$ and $\Lambda^1_{\hat{\alpha_1},\hat{\beta}_0}$, and invoking Lemma 6,

$$2\mu(x,u)/\text{Prob}\left[g_0 = \alpha_0 \circ g_0 | \alpha_0 \leftarrow \text{Sym}(w)\right] \cdot \text{Prob}\left[g_1 = \alpha_1 \circ g_1 | \alpha_1 \leftarrow \text{Sym}(w)\right]$$

$$= \text{Prob}\left[\begin{matrix}\exists \alpha_0 : \alpha_0 \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0 \\ \exists \alpha_1 : \alpha_1 \circ g_1 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0 \\ (\beta_0 \circ \hat{\beta}_0)^{-1} \circ \beta_1 = \hat{\beta}_0^{-1} \circ \hat{\beta}_1 \\ s = b \qquad \Gamma_s(u) = v\end{matrix} \middle| \begin{matrix}\beta_0, \beta_1 \leftarrow \text{Sym}(w) \\ s \leftarrow \{0,1\}\end{matrix}\right]$$

$$= \text{Prob}\left[\beta_0 \in \Lambda^0_{\hat{\alpha_0},\hat{\beta}_0} \cap \Lambda^1_{\hat{\alpha_1},\hat{\beta}_0} \middle| \beta_0 \leftarrow \text{Perm}\right]$$

$$\times \text{Prob}\left[\begin{matrix}\beta_0^{-1} \circ \beta_1 = \hat{\beta}_1 \\ s = b \\ \Gamma_s(u) = v\end{matrix} \middle| \begin{matrix}\beta_0 \leftarrow \Lambda^0_{\hat{\alpha_0},\hat{\beta}_0} \cap \Lambda^1_{\hat{\alpha_1},\hat{\beta}_0} \\ \beta_1 \leftarrow \text{Sym}(w), \\ s \leftarrow \{0,1\}\end{matrix}\right]$$

$$= \text{Prob}\left[\beta_0 \in \Lambda^0 \cap \Lambda^1 \middle| \beta_0 \leftarrow \text{Perm}\right] \cdot \text{Prob}\left[\begin{matrix}\beta_0^{-1} \circ \beta_1 = \hat{\beta}_1 \\ s = b \\ \Gamma_s(u) = v\end{matrix} \middle| \begin{matrix}\beta_0 \leftarrow \Lambda^0 \cap \Lambda^1 \\ \beta_1 \leftarrow \text{Sym}(w) \\ s \leftarrow \{0,1\}\end{matrix}\right]$$

$$= \text{Prob}\left[\beta_0 \in \Lambda^0 \cap \Lambda^1 \middle| \beta_0 \leftarrow \text{Perm}\right] \times \text{Prob}\left[\beta_1 = \beta_0 \circ \hat{\beta}_1 \middle| \beta_1 \leftarrow \text{Perm}\right]$$

$$\times \text{Prob}\left[\begin{matrix}s = b \\ \Gamma_s(u) = v\end{matrix} \middle| \begin{matrix}\beta_0 \leftarrow \Lambda^0 \cap \Lambda^1 \\ \beta_1 = \beta_0 \circ \hat{\beta}_1 \\ s \leftarrow \{0,1\}\end{matrix}\right].$$

Thus, there exists a constant c that does not depend on x and u such that,

$$\mu(x,u) = c \cdot \text{Prob}\left[\begin{matrix}s \\ \Gamma_s(u) = v\end{matrix} \middle| \begin{matrix}\beta_0 \leftarrow \Lambda^0 \cap \Lambda^1 \\ \beta_1 = \beta_0 \circ \hat{\beta}_1 \\ s \leftarrow \{0,1\}\end{matrix}\right].$$

Given the above observations, recalling the definition of Γ, the following suffices to prove the lemma:

$$\text{Prob}\left[(\beta_0 \circ \hat{\beta}_0)^{-1}(u) = v \middle| \beta_0 \leftarrow \Lambda^0 \cap \Lambda^1\right]$$

$$= \text{Prob}\left[(\beta_0 \circ \hat{\beta}_1)^{-1}(u) = v \middle| \beta_0 \leftarrow \Lambda^0 \cap \Lambda^1\right] \forall u. \tag{21}$$

When (g_0, g_1) is strongly regular, this is implied by Lemma 8. This concludes the proof. □

B Proof of Theorem 9

Proof. We describe a UPSS protocol in Figure 3 which is another variation of the protocol presented in Figure 1. Note that $\mathcal{R}$ is of size 2^6, therefore, the randomness cost of the protocol is 6 bits. We now show the correctness and privacy of this protocol. Since the superscript for layer index is obvious for a 1-SRBP, we omit the same.

Correctness: We claim that $v_2 = \alpha_{\mathsf{col}(2),x_2} \circ g_{x_2} \circ g_{x_1}(1)$. To see this, note that when $x_2 = 1$, then, g_{x_2} is identity, so, $v_2 = v_1 = \alpha_{0,1} \circ g_{x_1}(1) = \alpha_{0,x_2} \circ g_{x_2} \circ g_{x_1}(1)$. When $x_2 = 0$, $g_{x_2} = 1$, so, $v_2 = \alpha_{0,0} = \alpha_{0,x_2} \circ g_{x_2} \circ g_{x_1}(1)$. So we have that the output of P_2 is (s_2, v_2) where v_2 respects the invariant given in Equation (4). Furthermore, since the computation of (s_i, v_i) for $3 \leq i \leq n$ is the same as that given in Figure 1, therefore, the invariant given in Equation (4) holds further for $i \in [n]$. From this we have that,

$$v_n = \alpha_{0,x_n} \circ g_{x_n} \circ \cdots \circ g_{x_2} \circ g_{x_1}(1) = \alpha_{0,x_n}(u_n). \tag{22}$$

$y = \tilde{\alpha}_{s_n}^{-1}(v_n) = \alpha_{0,r_1 \oplus s_n}^{-1} \circ \alpha_{0,x_n}(u_n) = \alpha_{0,x_n}^{-1} \circ \alpha_{0,x_n}(u_n) = u_n$. So, if $u_n = 1, y = 0$ and if $u_n = 2, y = 1$, therefore establishing the correctness.

Privacy: Party P_1 doesn't receive anything from other parties except y, so the privacy against P_1 holds trivially. For $3 \leq i \leq n$, the view of party P_i is the same as the view of P_i while running the protocol given in Figure 1 and therefore the provacy follows from the proof of Theorem 2. Note that P_2's view is $(x_2, y, \alpha_{0,0}, r_0, \tilde{\alpha}_0, \tilde{\alpha}_1, v_1, s_n, v_n)$. Since $(\alpha_{0,0}, r_0)$ can be sampled independent of $(\tilde{\alpha}_0, \tilde{\alpha}_1, s_n, v_n)$ and $(\alpha_{0,0}, r_0)$ are uniformly random variables, it remains to show that, for privacy, $(\tilde{\alpha}_0, \tilde{\alpha}_1, s_n, v_n)$ do not break security, but note that this is the exact same view of P_{n+1} in Figure 1 when the protcol is computing a branching program of width 2. By Theorem 2, we know that the view of P_{n+1} in Figure 1 doesn't break privacy. Privacy against P_2 therefore follows from privacy against P_2. □

C Proof of Theorem 7

Proof. Given a branching program $\Pi = (\sigma, \{g_0^{(t)}, g_1^{(t)}\}_{t \in [\ell]}, \phi)$ of width w and length ℓ, we build an equivalent SRBP $\Pi' = (\sigma, \{h_0^{(t)}, h_1^{(t)}\}_{t \in [\ell]}, \phi')$ of width w^2. Note that Π' has the same input labelling function σ as Π. We further construct $h_0^{(t)}$ and $h_1^{(t)}$ for every $t \in [\ell]$ and ϕ' and show that $(h_0^{(t)}, h_1^{(t)})$ is a strongly regular pair of functions. For odd t, define $h_b^{(t)} : [w]^2 \to [w]^2$ for $b \in \{0,1\}$ as $h_b^{(t)}(u, v) = (u, g_b^{(t)}(u))$ for every $(u, v) \in [w]^2$. For even t, define $h_b^{(t)} : [w]^2 \to [w]^2$ for $b \in \{0,1\}$ as $h_b^{(t)}(u, v) = (g_b^{(t)}(v), v)$ for every $(u, v) \in [w]^2$. It remains to show that $(h_0^{(t)}, h_1^{(t)})$ is strongly regular for all t. First, fix t to be odd and denote the functions $g_b^{(t)}$ and $h_b^{(t)}$ with g_b and h_b for $b \in \{0,1\}$ for simplicity.

By construction, for $b \in \{0,1\}$, for every $(u',v') \in [w]^2$, $|h_b^{-1}(u',v')| = w$ if $v' = g_b(u')$ and 0 otherwise therefore satisfying the first condition with $c_0 = c_1 = w$. For any two (u',v') and $(u'',v'') \in [w]^2$, $|h_0^{-1}(u',v') \cap h_1^{-1}(u'',v'')| = w$ only when $u' = u''$, $v' = g_0(u') = v'' = g_1(u'')$ and 0 otherwise, therefore satisfying the second condition of strong regularity with $c = w$. Observe that the edge set E consists of w edges, where each edge is between $(u, g_0(u))$ and $(u, g_1(u)), u \in [w]$ and therefore no two edges share the same vertex. The third condition for strong regularity is also satisfied since each edge maps to every other edge with the same probability under a uniformly random $\mathrm{Aut}(H)$. Strong regularity can be shown for $(h_0^{(t)}, h_1^{(t)})$ for an even t in a similar way.

It remains to define ϕ' and show the equivalence of Π' and Π. To this end, assuming that (u_0, v_0) is the initial state of the modified branching program, we show that the following invariant holds true using induction on t: for odd t,

$$(u_t, v_t) = (u_{t-1}, g^{(t)}_{x_{\sigma(t)}} \circ g^{(t-1)}_{x_{\sigma(t-1)}} \circ \cdots \circ g^{(1)}_{x_{\sigma(1)}}(u_0)), \tag{23}$$

and for even t,

$$(u_t, v_t) = (g^{(t)}_{x_{\sigma(t)}} \circ g^{(t-1)}_{x_{\sigma(t-1)}} \circ \cdots \circ g^{(1)}_{x_{\sigma(1)}}(u_0), v_{t-1}). \tag{24}$$

Note that for the base case, when $t = 1$, $(u_1, v_1) = h^{(1)}_{\sigma(x_1)}(u_0, v_0) = (u_0, g^{(1)}_{\sigma(x_1)}(u_0))$ and for $t = 2$, $(u_2, v_2) = h^{(2)}_{\sigma(x_2)}(u_1, v_1) = (g^{(2)}_{\sigma(x_2)}(v_1), v_1) = (g^{(2)}_{\sigma(x_2)} \circ g^{(1)}_{\sigma(x_1)}(u_0), v_1)$. Assume Equation (23) to be true (for an odd t). Then, for an even $t+1$, we have that,

$$h^{(t+1)}_{\sigma(x_{t+1})}(u_t, v_t) = (g^{(t+1)}_{\sigma(x_{t+1})}(v_t), v_t) \tag{25}$$

$$= (g^{(t+1)}_{\sigma(x_{t+1})} \circ g^{(t)}_{\sigma(x_t)} \circ \cdots \circ g^{(1)}_{\sigma(x_1)}(u_0), v_t). \tag{26}$$

which proves the hypothesis for the even case. The hypothesis for the odd case can be proven similarly. This invariant naturally gives the following output function ϕ'. If ℓ is odd, output $\phi'(u_\ell, v_\ell) = \phi(v_\ell)$ and if ℓ is even, output $\phi'(u_\ell, v_\ell) = \phi(u_\ell)$. This completes the proof. □

D Proof of Lemma 3

Proof. Note that we can insert a "dummy layer" at any point in a given SRBP, with any input label, with both transition functions being the identity function (which indeed satisfies strongly regularity) and an arbitrary choice-bit function, without changing the function being computed.

Let $\eta : [n] \to [n]$ be a permutation with no fixed point (i.e., $\forall j \in [n], \eta(j) \neq j$). Then between any two layers t and $t+1$ with $\sigma(t) = \sigma(t+1)$ in the given branching program, we insert a dummy layer with input label $\eta(\sigma(t))$. Since an input label j can occur at most k times originally, there can be at most $k-1$ positions t with $\sigma(t) = \sigma(t+1) = j$; hence for the dummy layers inserted $\eta(j)$ can be assigned at most $k-1$ times. This results in a $(2k-1)$-SRBP. □

E Proof of Lemma 4

Proof. Consider an execution of an MPC protocol for AND by 3 parties $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3$ with inputs x_1, x_2, x_3 respectively. Consider any party, taken to be P_1 w.l.o.g. We seek to prove that the transcripts in the view of P_1 determine x_1.

Towards this, consider a 2-party protocol derived from the given 3-party protocol, between P_1 and a party P'_1 which internally carries out the roles of both P_2 and P_3. It is enough to show that the transcript of this protocol, which corresponds to the transcripts that are in the view of P_1 in the original protocol, determine x_1.

Let T_{ab} denote the set of all transcripts that occur with positive probability in the new 2-party protocol above, for inputs $x_1 = a$ and some inputs (x_2, x_3) such that $x_2 \wedge x_3 = b$. Note that in this protocol we still have the privacy requirement that when $x_1 = 0$, P_1 does not learn $x_2 \wedge x_3$. Hence

$$T_{00} = T_{01}.$$

We also use the protocol structure of the transcripts: the probability of a transcript w occurring in a 2-party protocol given inputs x, y to the two parties can be factorized as $\Pr[w|x, y] = \alpha(w, x)\beta(w, y)$ for some functions α and β that capture the local actions of the two parties. Hence $T_{01} \cap T_{10} \subseteq T_{11}$, since every w in the LHS has $\alpha(w, 1) > 0$ (since $w \in T_{10}$) and $\beta(w, (1, 1)) > 0$ (since $w \in T_{01}$), making it part of RHS as well. By intersecting both sides of this inclusion with T_{01} we can write

$$T_{01} \cap T_{10} \subseteq T_{01} \cap T_{11}.$$

We seek to prove that the transcript determines x_1, or in other words there is no transcript which occurs for $x_1 = 0$ and for $x_1 = 1$ (with either value of $x_2 \wedge x_3$ in each case): i.e., $(T_{00} \cup T_{01}) \cap (T_{10} \cup T_{11}) = \emptyset$. Now

$$\begin{aligned}
(T_{00} \cup T_{01}) \cap (T_{10} \cup T_{11}) &= T_{01} \cap (T_{10} \cup T_{11}) && \text{since } T_{00} = T_{01} \\
&= (T_{01} \cap T_{10}) \cup (T_{01} \cap T_{11}) \\
&= T_{01} \cap T_{11} && \text{since } T_{01} \cap T_{10} \subseteq T_{01} \cap T_{11}
\end{aligned}$$

Finally, the correctness requirement for P'_1, when $x_2 = x_3 = 1$ implies that $T_{01} \cap T_{11} = \emptyset$. Hence we have $(T_{00} \cup T_{01}) \cap (T_{10} \cup T_{11}) = \emptyset$, as required. □