Private Circuits with Quasilinear Randomness

Vipul Goyal[1,2], Yuval Ishai[3], and Yifan Song[1]

[1] Carnegie Mellon University, USA.
vipul@cmu.edu, yifans2@andrew.cmu.edu
[2] NTT Research, USA.
[3] Technion, Israel.
yuvali@cs.technion.ac.il

Abstract. A *t-private* circuit for a function f is a randomized Boolean circuit C that maps a randomized encoding of an input x to an encoding of the output $f(x)$, such that probing t wires anywhere in C reveals nothing about x. Private circuits can be used to protect embedded devices against side-channel attacks. Motivated by the high cost of generating fresh randomness in such devices, several works have studied the question of minimizing the randomness complexity of private circuits.
The best known upper bound, due to Coron et al. (Eurocrypt 2020), is $O(t^2 \cdot \log ts)$ random bits, where s is the circuit size of f. We improve this to $O(t \cdot \log ts)$, including the randomness used by the input encoder, and extend this bound to the stateful variant of private circuits. Our constructions are semi-explicit in the sense that there is an efficient randomized algorithm that generates the private circuit C from a circuit for f with negligible failure probability.

1 Introduction

The notion of *private circuits*, due to Ishai, Sahai, and Wagner (ISW) [9], is a simple abstraction of leakage-resilience computation. It is simple both in terms of the underlying computational model, namely Boolean circuits, and in terms of the class of leakage attacks it should protect against.

ISW defined two flavors of private circuits: stateless and stateful. The former captures a one-time computation that maps an encoded (or "secret-shared") input to an encoded output, whereas the latter captures an evolving computation that may update a secret internal state. We will start by considering the simpler stateless model and later discuss an extension to the stateful case.

A (stateless) t-private circuit for a function f is defined by a triple (I, C, O), where I is a trusted, randomized input encoder, mapping an input x to an encoded input $\hat{x}$, C is a randomized Boolean circuit mapping $\hat{x}$ to an encoded output $\hat{y}$, and O is a trusted, deterministic output decoder mapping the encoding $\hat{y}$ to an output y. The natural correctness requirement is that for every input x, we have $O(C(I(x))) = f(x)$ (with probability 1). The security requirement asserts that an adversary who can probe any set of t wires of C learns nothing about x. Moreover, it is required that I and O be *universal* in the sense that they

are independent of (and are typically much smaller than) C and may depend only on the input length and t. This rules out trivial solutions in which I or O compute f. The default encoder I, referred to as the *canonical* encoder, independently splits each input bit x_i into $t+1$ random bits whose parity is x_i.

The simplicity of the private circuits model makes it attractive as an object of theoretical study, enabling a clean and rigorous analysis of the achievable tradeoffs between security and efficiency. On the downside, the same simplicity that makes the model theoretically appealing also makes it a very crude approximation of reality. In particular, the bounded probing attacks that private circuits are designed to protect against are much too restrictive to capture real-life side-channel attacks.

Somewhat unexpectedly, private circuits have gained popularity as a practical method for "higher-order masking" countermeasures that protect embedded devices against realistic side-channel attacks. A partial theoretical explanation was given by Duc et al. [6], who showed that the security of private circuits against probing attacks is enough to also guarantee security against a certain level of *noisy leakage*, which independently leaks a small amount information about each wire.

Minimizing randomness complexity. A natural complexity measure for private circuits is their *randomness complexity*, measured by default as the total number of random bits used by the randomized circuit C. (We will later also address the goal of minimizing the randomness used by the input encoder.) The question of minimizing the randomness complexity of private circuits is not only a natural theoretical question, but is also motivated by the high costs of generating fresh randomness on embedded devices.[4] This question has also found unexpected applications to efficiently mitigating selective failure attacks in secure computation protocols based on garbled circuits [8].

The original ISW construction [9] used $O(t^2)$ fresh random bits to compute each AND gate in a circuit computing f. This gives an upper bound of $O(t^2 \cdot s)$ where s is the circuit size of f. Several subsequent works obtained improvements to this bound, with the canonical encoder described above.

Ishai et al. [8], in the broader context of studying robust pseudorandomness generators, show how to reduce the randomness complexity to $O(t^{3+\epsilon} \cdot \log ts)$, for any $\epsilon > 0$, thus making it almost independent of the circuit size s. Belaïd et al. [2] focus on the randomness complexity of implementing a single AND gate, for which they present a probabilistic construction of a gadget that requires only $O(t \log t)$ random bits, compared to $O(t^2)$ in ISW. However, their technique does not efficiently extend to multiple gates, and their construction is not composable in the sense of [1,3] (see Section 2 for further discussion). Faust et al. [7] present a practically-oriented approach for reusing randomness across multiple AND gadgets, reducing the total amount of randomness for small values of t

[4] Even if one settles for computational security, alternative approaches based on cryptographic pseudorandom generators (PRGs) are also quite expensive, especially since the PRG computation itself is subject to leakage [9].

($t \leq 7$) by a constant factor. The current state of the art was obtained by the recent construction of Coron et al. [5], which uses $O(t^2 \cdot \log ts)$ random bits. This construction not only improves the bound of [8] by more than a factor of t, but also eliminates the use of low-degree expander graphs that hurts concrete efficiency.

1.1 Our Contribution

In this work, we improve the best previous asymptotic bound of Coron et al. [5] by an additional factor of t. Concretely, we show that any function f computed by a Boolean circuit of size s admits a t-private circuit which uses only $O(t \cdot \log ts)$ random bits.

Our construction is only semi-explicit in the sense that there is an efficient randomized algorithm that generates C from a Boolean circuit for f with negligible failure probability. However, similarly to the construction from [5] (and unlike the earlier construction from [8]), our construction does not require the circuit C to use a good low-degree expander, and we do not see inherent barriers to good concrete efficiency (which we did not attempt to optimize).

We also present the following additional extensions of our main result:

- **Randomness-efficient input encoder.** By using a randomness-efficient encoder I instead of the canonical one, we can obtain the same $O(t \cdot \log ts)$ bound *even when counting the internal randomness of the input encoder*. This is optimal up to the logarithmic factor.
- **Leakage-tolerant circuits.** We can get the same bound in the (stronger) model of *leakage-tolerant* private circuits [8]. In this model there are no trusted input encoder or output decoder, and an adversary probing t wires of C is allowed to learn a similar number of input and output bits.
- **Stateful private circuits.** Finally, our $O(t \cdot \log ts)$ bound applies also to the standard *stateful* model of private circuits from [9] (see the full version), counting the number of fresh random bits in each invocation.

Open questions. We conjecture that our $O(t \cdot \log ts)$ upper bound is asymptotically optimal in all of the above settings, and leave open the question of proving (or disproving) this conjecture. We also leave open the existence of a fully explicit variant of our constructions.

2 Technical Overview

In this section, we give a detailed technical overview of our main results, starting with some necessary background.

Background: Leakage-resilient and Leakage-tolerant Private Circuits. In [9], Ishai et al. introduced the fundamental notion of private circuits. This notion comes in two flavors, a stateful variant and a simpler stateless variant. Here we will

start by focusing on the latter for simplicity, but will later show how to extend our results to the stateful model.

Informally, for a function f, a private (stateless) circuit consists of an input encoder I, a compiled circuit C, and an output decoder O. The compiled circuit C takes an encoded input $\hat{x}$ generated by $I(x)$ and computes an encoded output $\hat{y}$ such that $O(y) = f(x)$. The security requires that any t wires in C should be independent of the function input. This notion is also referred to as leakage-resilient private circuits. We focus by default on the canonical encoder: I encodes each input bit x_i by a vector of $t+1$ random bits with parity x_i. (Later we will consider a different encoder, in the context of minimizing the randomness complexity of the encoder.) As to the decoder, we consider by default a relaxation of the canonical decoder: To decode each output bit, O takes the parity of a block of bits (which are not restricted to be $t+1$ bits).

- As noted in [8], without any requirement on I and O, a trivial solution is having I compute a secret sharing of $f(x)$ which is passed by C to the decoder.
- In this work, we focus on the additional randomness used in the compiled circuit C. We want to avoid encoders which output a large amount of randomness.

We may also define the notion of leakage-tolerant private circuits. In this setting, the input and output are not encoded. I.e., the encoder and the decoder are the identity function. The security requires that any set of at most t wires in C can be simulated by the *same number* of input and output wires.

In this work, we show that for any function f with circuit size s, there is a (leakage-resilient) private circuit which uses $O(t \cdot \log ts)$ random bits.

Background: (Strong) t-wise Independent Pseudo-random Generator. Our construction makes use of the notion of (strong) t-wise independent pseudo-random generators (PRG). Informally, a function $G : \{0,1\}^n \to \{0,1\}^m$ is a t-wise independent PRG if any t bits in $G(x)$ are independent and uniformly random when the input x is uniformly distributed in $\{0,1\}^n$. Moreover, if any t bits in $(x, G(x))$ are independent and uniformly random, then we say G is a *strong* t-wise independent PRG.

We say that G is linear if any output bit of G is equal to the XOR of a subset of its input bits. See Section 3.2 for an explicit construction of a *linear and strong* t-wise independent PRG with input size $n = O(t \cdot \log m)$.

Limitations of Previous Approaches. We first recall the definition of robust t-wise independent PRGs introduced in [8]. Intuitively, any probing attack towards a robust t-wise independent PRG is equivalent to a probing attack towards the output bits. The following definition corresponds to the strong robust t-wise independent PRGs in [8].

Definition 1 (Robust t-wise Independent PRGs [8]). *A circuit implementation C of a function $G : \{0,1\}^m \to \{0,1\}^n$ is a (t, k, q)-robust pseudo-random*

generator if the following holds. Let $\boldsymbol{u}$ *be a vector of* m *uniformly random bits, and* $\boldsymbol{r} = G(\boldsymbol{u})$*. For any set* S *of at most* k *wires in* C*, there is a set* T *of at most* $q|S|$ *output bits such that conditioned on any fixing of the values* C_S *of the wires in* S *and* $\boldsymbol{r}_I$*, where* $I = \{i : r_i \in T\}$*, the values* $\boldsymbol{r}_{\bar{I}}$ *of the output bits not in* T *are* t*-wise independent.*

A generic approach of derandomizing a private circuit introduced in [8] works as follows:

1. First, derandomize a private circuit by assuming an access to t-wise independent random source. This is achieved by considering the notion of randomness locality of a private circuit. The randomness locality of a private circuit is the number of random bits that are used to compute each wire. If each wire uses at most ℓ random bits (i.e., the randomness locality is ℓ), then we may replace the uniform random source by a $(\ell \cdot t)$-wise independent random source to protect against t-probing attacks. It is because any t wires depend on at most $\ell \cdot t$ random bits, and therefore, the distribution of these t wires when using uniform random source is identical to that when using $(\ell \cdot t)$-wise independent random source.
2. Then, replace the $(\ell \cdot t)$-wise independent random source by a robust $(\ell \cdot t)$-wise independent PRG.

In [8], Ishai et al. constructed a private circuit with randomness locality $O(t^2)$ based on the private circuit constructed in [9]. The recent work of Coron et al. [5] further improves the randomness locality to $O(t)$. Combined with a robust $O(t^2)$-wise independent PRG from [8], this gives a private circuit which uses only $\tilde{O}(t^{2+\epsilon})$[5] random bits, for any constant $\epsilon > 0$. Moreover, they also show that in their construction the robust $O(t^2)$-wise independent PRG can be replaced by $O(t)$ independent $O(t)$-wise independent PRGs. This reduces the randomness complexity to $\tilde{O}(t^2)$ and results in better concrete efficiency.

However, the approach of using randomness locality inherently requires $\Omega(t^2)$ random bits. Intuitively, the randomness locality of a private circuit cannot be smaller than t, or otherwise, an adversary may learn extra information about the input by probing a wire and all random bits that are used to compute this wire. It means that the random source should be at least t^2-wise independent, which requires $\Omega(t^2)$ random bits.

To overcome this bottleneck:

1. We first switch the view from computing secret shared wire values, the mainstream method in the literature of private circuits [9,8,5], to computing *masked* wire values. We note that computing masked wire values is commonly used in the literature on secure multi-party computation. However, to the best of our knowledge, this method was not used in the literature on private circuits. We use this to bypass the limitation of using randomness locality and reduce the problem of constructing a private circuit to

[5] In this paper, we use $\tilde{O}$ notation to hide logarithmic factors in either a PRG output size or a circuit size.

that of constructing a leakage-tolerant private circuit for the XOR function, which we refer to as a leakage-tolerant XOR gadget. Specifically, assuming the existence of a leakage-tolerant XOR gadget, we construct a private circuit which uses $\tilde{O}(t)$ random bits (excluding the randomness used in the leakage-tolerant XOR gadgets).

2. Then, we focus on the leakage-tolerant XOR gadget. We start with a straightforward construction of a leakage-tolerant XOR gadget assuming the access to correlated random bits. Then we use a special kind of robust PRG, which we refer to as a *robust parity sharing generator*, to generate the correlated random bits. Compared with [8], we extend the notion of robust PRGs in the following two directions: (1) the output bits of the robust PRG should be *correlated* as required by the basic construction of the XOR gadget, and (2) unlike the robust t-wise independent PRG [8], which only focuses on the *number* of random bits on which the probed wires depend, we also take into account the concrete dependence on the random bits induced by the construction of the XOR gadget. This more fine-grained approach allows us to bypass the limitation of using randomness locality when constructing leakage-tolerant XOR gadgets. By using probabilistic arguments, we obtain a semi-explicit construction of a robust parity sharing generator which uses $\tilde{O}(t)$ random bits. This yields a leakage-tolerant XOR gadget with randomness complexity $\tilde{O}(t)$.

Combining the above two steps, we obtain a private circuit in the plain model which uses $\tilde{O}(t)$ random bits. We note that a similar approach is used in [2] to reduce the randomness complexity of a single multiplication gate. Concretely, Belaïd et al. [2] use probabilistic arguments and show the existence of a t-private multiplication circuit, which takes *shared inputs* and produces *shared output*, with randomness complexity $O(t \log t)$. However there are three key differences which make their technique difficult to work for a general circuit (i.e., not restricting to a single multiplication gate):

- The solution in [2] only reduces the randomness complexity for a single multiplication gate. However, to obtain our result, we have to reuse the randomness across all gadgets or otherwise the randomness complexity will be proportional to the circuit size. It is unclear how their result can be extended to reducing the randomness complexity for multiple multiplication gates. Our solution, on the other hand, reuses the randomness across all leakage-tolerant XOR gadgets. See more discussion in Remark 2.
- As discussed in Section 7.2 in [2], their private multiplication circuit is not composable. As a result, Belaïd et al. [2] cannot obtain a private circuit for a general function from their private multiplication circuit without affecting the achieved randomness complexity per gate.
- The construction in [2] takes *shared inputs* and computes *shared output*. As discussed above, derandomizing shared wire values may require at least $\Omega(t^2)$ random bits.

Extensions. Beyond the basic construction, we consider the following two extensions:

1. The first extension is replacing the canonical encoder by a randomness-efficient encoder. Note that the canonical encoder requires $t \cdot n_i$ random bits to encode n_i input bits. We construct a randomness-efficient encoder which reduces the randomness complexity to $O(t \cdot \log n_i)$. As a result, for any function with circuit size s, we obtain a private circuit which uses $O(t \cdot \log ts)$ random bits *including the randomness used in the encoder*.
2. The second extension is to construct a private *stateful* circuit. A stateful circuit models the scenario where the circuit has an internal state. In each invocation, the circuit computes the function which takes as input its internal state and the external input, outputs the result of the function, and stores its new internal state. The security of a private stateful circuit requires that an adversary, which has control over the external input, cannot learn any information about the internal state of the circuit over multiple executions of the circuit with the power of adaptively choosing a set of t internal wires before each execution depending on the output values and the wire values observed in previous executions. We show how to extend our basic construction to a private stateful circuit which uses $O(t \cdot \log t|C|)$ random bits for any stateful circuit C in each invocation.

More details can be found in Section 2.3.

2.1 Outer Construction: Private Circuits via Leakage-tolerant XOR Gadgets

For a boolean function $f : \{0,1\}^{n_i} \to \{0,1\}^{n_o}$, let $\tilde{C}$ denote a circuit that computes the function f. To protect the wire values in $\tilde{C}$, our idea is to mask each wire value in $\tilde{C}$ by an output bit of a t-wise independent PRG. Since any t output bits of a t-wise independent PRG are independent and uniformly distributed, any t masked wire values in $\tilde{C}$ leak no information about the input.

In our construction, we choose to use a linear and strong $2t$-wise independent PRG $G : \{0,1\}^m \to \{0,1\}^{|\tilde{C}|}$. Informally, this is because

- A probing attack can not only probe the masked wire values in $\tilde{C}$, but also the input random bits of G. We need to protect the input random bits of G by using a *strong* $2t$-wise independent PRG.
- A linear PRG allows us to compute each output bit of G by simply computing the XOR of a subset of its input bits. As we will show later, our construction directly uses a leakage-tolerant XOR gadget to compute each masked wire value. In this way, the implementation of G is hidden in the leakage-tolerant XOR gadgets and we do not need to worry about the intermediate wires in the implementation of G.
- As we will see later, our construction may leak at most $2t$ masked wire values within t probes. It requires us to use a $2t$-wise independent PRG.

We use $\boldsymbol{u} = (u_1, u_2, \ldots, u_m)$ to denote the input of G and $\boldsymbol{r} = (r_1, r_2, \ldots, r_{|\tilde{C}|})$ to denote the output of G. Since G is linear, each output bit r_i is the XOR of a subset of $\{u_1, u_2, \ldots, u_m\}$. We refer to this set as the support of r_i, denoted by $\mathtt{supp}(r_i)$. For the circuit $\tilde{C}$, all wires are denoted by $g_1, g_2, \ldots, g_{|\tilde{C}|}$ (including input wires and output wires). The goal is to compute $g_i \oplus r_i$ for all $i \in \{1, 2, \ldots, |\tilde{C}|\}$. Note that we can view $\{g_i \oplus r_i\} \bigcup \mathtt{supp}(r_i)$ as an additive sharing of g_i since the XOR of the bits in $\mathtt{supp}(r_i)$ is equal to r_i. We use $[g_i]$ to represent the set $\{g_i \oplus r_i\} \bigcup \mathtt{supp}(r_i)$. At first glance, one may think that our approach is very similar to the generic approach of constructing private circuits in [9,8,5] since the latter also computes an additive sharing for each wire value. However, we would like to point out that there are two key differences between our approach and the generic approach:

- First, for each additive sharing $[g_i]$ in our approach, the number of shares depends on the size of $\mathtt{supp}(r_i)$, which may vary for different i. On the other hand, the additive sharings in the generic approach all have the same number of shares.
- Second, in our approach, the shares are reused among different additive sharings, which is not the case in the generic approach.

As a result, our approach does not need to derandomize the shares of additive sharings since they reuse the input random bits of the PRG G, which is of size $O(t \log |\tilde{C}|)$. It allows us to bypass the limitation of using randomness locality.

In the outer construction, we assume the existence of a leakage-tolerant XOR gadget (or equivalently, a leakage-tolerant private circuit for the XOR function). Recall that the input encoder is the canonical encoder, which encodes each input bit x_i by a vector of $t+1$ random bits with parity x_i. We may view the encoding of x_i as an additive sharing of x_i, denoted by $[x_i]$. The outer construction works as follows:

1. We first transform the input encoding $[x_i]$ to the masked input bit. Let g_i be the input wire of $\tilde{C}$ which is initialized to x_i. Then we want to compute $g_i \oplus r_i = x_i \oplus r_i$. This is done by using a leakage-tolerant XOR gadget to XOR the bits in $[x_i] \bigcup \mathtt{supp}(r_i)$.
2. For each addition gate with input wires g_a, g_b and output wire g_c in $\tilde{C}$, the masked output wire $g_c \oplus r_c$ is computed by using a leakage-tolerant XOR gadget to XOR the bits in $[g_a] \bigcup [g_b] \bigcup \mathtt{supp}(r_c)$.
3. For each multiplication gate with input wires g_a, g_b and output wire g_c in $\tilde{C}$, we first define $[g_a] \otimes [g_b] = \{u \cdot v : u \in [g_a], v \in [g_b]\}$. Note that $g_a = \oplus_{u \in [g_a]} u$ and $g_b = \oplus_{v \in [g_b]} v$. Therefore

$$g_c = g_a \cdot g_b = (\oplus_{u \in [g_a]} u) \cdot (\oplus_{v \in [g_b]} v) = \oplus_{u \in [g_a], v \in [g_b]} u \cdot v,$$

   which means that g_c is equal to the XOR of the bits in $[g_a] \otimes [g_b]$. The circuit first computes $u \cdot v$ for all $u \in [g_a]$ and $v \in [g_b]$. The masked output wire $g_c \oplus r_c$ is then computed by using a leakage-tolerant XOR gadget to XOR the bits in $([g_a] \otimes [g_b]) \bigcup \mathtt{supp}(r_c)$.

4. For each output gate with input wire g_a in $\tilde{C}$, we have computed $g_a \oplus r_a$. Recall that $[g_a] = \{g_a \oplus r_a\} \bigcup \mathtt{supp}(r_a)$ and the XOR of bits in $[g_a]$ is equal to g_a. The circuit simply outputs bits in $[g_a]$.

Let C denote the circuit constructed above. The correctness of C is straightforward from the description. The security follows from the following three facts:

- By the definition of leakage-tolerant private circuits, any probing attack towards a leakage-tolerant XOR gadget is equivalent to a probing attack (that probes the same number of wires) towards the input wires and the output wire of this gadget. Therefore, we only need to focus on two kinds of wires: (1) the input wires of C, and (2) the rest of wires which excludes the internal wires of leakage-tolerant XOR gadgets. The second kind contains the input bits of the PRG G and the masked wire values, i.e., $\{u_i\}_{i=1}^{m} \bigcup \{g_i \oplus r_i\}_{i=1}^{|\tilde{C}|}$, and the values in $[g_a] \otimes [g_b]$ for all multiplication gate with input wires g_a and g_b in $\tilde{C}$.
- For input wires of C, since each input bit x is encoded by $t+1$ random bits with parity x, any t bits are uniformly random. Therefore, the input wires are leakage resilient.
- For the second kind of wires, note that each value $u \cdot v \in [g_a] \otimes [g_b]$ can be computed by $u \in [g_a]$ and $v \in [g_b]$. Thus, any t wires can be determined by at most $2t$ values in $\{u_i\}_{i=1}^{m} \bigcup \{g_i \oplus r_i\}_{i=1}^{|\tilde{C}|}$. The security follows from that G is a strong $2t$-wise independent PRG.

We refer the readers to Section 4 for more details.

2.2 Inner Construction and Robust Parity Sharing Generator

Given the outer construction in Section 2.1, it is sufficient to move our focus to the construction of a leakage-tolerant XOR gadget. Our main technical contribution is a new notion of robust PRGs which we refer to as robust parity sharing generators.

Basic Construction of Leakage-Tolerant XOR Gadgets. We first introduce a straightforward construction of leakage-tolerant XOR gadgets assuming access to ideal correlated random bits, which is adapted from the basic MPC protocol for XOR of Kushilevitz and Mansour [10]. Let $x_1, x_2, \ldots, x_n$ denote the input bits of the gadget $\mathcal{G}$. The goal is to compute the output $\oplus_{i=1}^{n} x_i$. The circuit is given access to n random bits $r_1, r_2, \ldots, r_n$ with parity 0. (Note that here x_i's and r_i's are different variables from those in Section 2.1.)

The basic construction works as follows:

1. For all $i \in \{1, 2, \ldots, n\}$, the circuit computes $g_i := x_i \oplus r_i$ in parallel. (Here too, g_i is a different variable from that in Section 2.1.)
2. Then the circuit computes the XOR of $g_1, g_2, \ldots, g_n$ by using a $\lceil \log n \rceil$-depth addition circuit: In each round, the addition circuit partitions the input bits into groups of size 2. For each group, the addition circuit computes the XOR

of the bits in this group. The output bits are provided as input bits to the next round.

In fact, our construction works for any addition circuit with the same randomness complexity. Here we choose to use the addition circuit with the smallest depth so as to minimize the depth of the private circuit. In [10], the basic protocol computes $\oplus_{i=1}^{n} g_i$ by adding up $\{g_i\}_{i=1}^{n}$ from g_1 to g_n.

We show that this simple construction is leakage-tolerant. First note that the input wires and output wires of the gadget can always be simulated by having access to the corresponding wires. In the following, we only focus on the intermediate wires in $\mathcal{G}$.

We may divide the intermediate wires in $\mathcal{G}$ into two sets: (1) the first set contains the correlated random bits $r_1, r_2, \ldots, r_n$, and (2) the second set contains all wires in the addition circuit, which is either g_i for some $i \in \{1, 2, \ldots, n\}$ or a linear combination of $\{g_1, g_2, \ldots, g_n\}$. If all probed wires are in the first set, the simulator can simply sample the random bits $r_1, r_2 \ldots, r_n$ with parity 0 and output the probed wires. This requires no information about the input and output bits. Suppose at least one wire in the second set is probed. The main observation is that, after mask x_i by r_i, $g_1, g_2, \ldots, g_n$ are uniformly random bits with parity $\oplus_{i=1}^{n} x_i$. Thus, the simulator works as follows:

1. The simulator first queries the output bit $\oplus_{i=1}^{n} x_i$, and then randomly samples $g_1, g_2, \ldots, g_n$ with parity $\oplus_{i=1}^{n} x_i$. In this way, any probed wire in the second set can be simulated.
2. For each wire r_i in the first set, the simulator queries x_i and computes $r_i = g_i \oplus x_i$.

Note that the number of input and output wires queried by the simulator is bounded by the number of probed wires. The main issue of the basic construction is that it uses too many random bits: it requires $n - 1$ uniformly random bits to compute the XOR of n input bits. To reduce the randomness complexity, we first analyse what kind of random source is sufficient for the basic construction. We note that the analysis in [10] is specific to their order of computing $\{g_i\}_{i=1}^{n}$ and appears difficult to generalize to other orders. Our analysis uses a different argument, described below.

Sufficient Conditions for the Random Source in the Basic Construction. First of all, the correctness of our construction requires that the random variables $(r_1, r_2, \ldots, r_n)$ have parity 0. Now we want to relax the requirement that $r_1, r_2, \ldots, r_n$ are uniformly random bits with parity 0. Going forward, let $r_1, r_2, \ldots, r_n$ be arbitrary possibly correlated bits with parity 0. Let $\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n$ be uniformly random bits with parity 0.

We observe that a direct sufficient condition is that the distribution of the probed wires when using $r_1, r_2, \ldots, r_n$ is identical to the distribution of the probed wires when using $\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n$. This is because we have shown that the basic protocol is leakage-tolerant when using uniformly random bits $\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n$ with parity 0. With the above sufficient condition, we can simulate the probed

wires in the same way as that for the basic construction when replacing $\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n$ with $r_1, r_2, \ldots, r_n$.

More concretely, let $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, and $\tilde{\boldsymbol{r}} = (\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n)$. For a set W of at most t intermediate wires in $\mathcal{G}$, the above sufficient condition requires that

$$\texttt{Dist}((\boldsymbol{x}, \boldsymbol{r}), W) = \texttt{Dist}((\boldsymbol{x}, \tilde{\boldsymbol{r}}), W),$$

where $\texttt{Dist}((\boldsymbol{x}, \boldsymbol{r}), W)$ refers to the distribution of W when instantiated by $(\boldsymbol{x}, \boldsymbol{r})$. We will reduce this to a requirement on the random source $\boldsymbol{r}$.

Recall that there are two sets of intermediate variables in $\mathcal{G}$: (1) the first set contains the correlated random bits $r_1, r_2, \ldots, r_n$, and (2) the second set contains all wires in the addition circuit, which is either g_i for some $i \in \{1, 2, \ldots, n\}$ or a linear combination of $\{g_1, g_2, \ldots, g_n\}$. Note that each bit in the second set can be written as $\oplus_{i \in L} g_i = (\oplus_{i \in L} x_i) \oplus (\oplus_{i \in L} r_i)$ for some set $L \subset \{1, 2, \ldots, n\}$. Since $\boldsymbol{x}$ are fixed input bits, it is sufficient to only consider the distribution of $\oplus_{i \in L} r_i$. Therefore, consider the set $\mathcal{A}$ defined as:

$$\mathcal{A} = \{r_1, r_2, \ldots, r_n\} \bigcup \{\oplus_{i \in L} r_i : \oplus_{i \in L} g_i \text{ is an intermediate wire in } \mathcal{G}\}.$$

Each variable in $\mathcal{A}$ is a linear combination of $\{r_1, r_2, \ldots, r_n\}$. The sufficient condition above can be transformed to that, for any set W of at most t variables in $\mathcal{A}$, $\texttt{Dist}(\boldsymbol{r}, W) = \texttt{Dist}(\tilde{\boldsymbol{r}}, W)$ holds.

We refer to $\mathcal{A}$ as the access structure of the random variables $\{r_1, r_2, \ldots, r_n\}$. Formally, an access structure $\mathcal{A}$ of a set of variables $\{r_1, r_2, \ldots, r_n\}$ is a set which satisfies that (1) for all $i \in \{1, 2, \ldots, n\}$, $r_i \in \mathcal{A}$, and (2) every variable in $\mathcal{A}$ is a linear combination of $(r_1, r_2, \ldots, r_n)$. One may think that a probing attack can only probe variables in $\mathcal{A}$.

Therefore, we can summarize the sufficient conditions for the distribution of the random source $\boldsymbol{r}$ in the basic construction as follows:

1. The parity of $(r_1, r_2, \ldots, r_n)$ is 0.
2. Let $\mathcal{A}$ be the access structure of $(r_1, r_2, \ldots, r_n)$ as defined above. Let $\tilde{\boldsymbol{r}} = (\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n)$ be uniformly random bits with parity 0. For any set W of at most t variables in $\mathcal{A}$, $\texttt{Dist}(\boldsymbol{r}, W) = \texttt{Dist}(\tilde{\boldsymbol{r}}, W)$.

Robust Parity Sharing Generator. Now we consider to use a robust PRG to generate the above correlated random bits. We follow the notion of robust t-wise independent PRGs in [8] and define what we refer to as robust parity sharing generators as follows.

Definition 2 (Robust Parity Sharing Generators). *Let $G : \{0,1\}^m \to \{0,1\}^n$ be a function and $\mathcal{A}$ be an access structure of the output bits of G. Let $\boldsymbol{u}$ be a vector of m uniformly random bits, and $\boldsymbol{r} = G(\boldsymbol{u})$. A circuit implementation C of the function G is a (t, k, q)-robust parity sharing generator with respect to $\mathcal{A}$ if the following holds:*

- *The parity of the output bits $\boldsymbol{r}$ is 0.*

– *Let* $\tilde{\boldsymbol{r}} \in \{0,1\}^n$ *be uniformly random bits with parity* 0*. For any set* S *of at most* k *wires in* C*, there is a set* T *of at most* $q|S|$ *output bits such that for any set* W *of* t *variables in* $\mathcal{A}$ *and for any fixing of the values* C_S *of the wires in* S *and* $\boldsymbol{r}_I$*, where* $I = \{i : r_i \in T\}$*,*

$$\texttt{Dist}(\boldsymbol{r}|_{C_S,\boldsymbol{r}_I}, W) = \texttt{Dist}(\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I=\boldsymbol{r}_I}, W),$$

where $\texttt{Dist}(\boldsymbol{r}, W)$ *is the distribution of the variables in* W *when they are instantiated by* $\boldsymbol{r}$*,* $\boldsymbol{r}|_{C_S,\boldsymbol{r}_I}$ *is the random variable* $\boldsymbol{r}$ *conditioned on* C_S *and* $\boldsymbol{r}_I$*, and* $\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I=\boldsymbol{r}_I}$ *is the random variable* $\tilde{\boldsymbol{r}}$ *conditioned on* $\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I$*.*

Informally, a robust parity sharing generator outputs n random bits with parity 0. The output of a robust parity sharing generator satisfies that any t variables in the access structure $\mathcal{A}$ have the same distribution when these t variables are instantiated by n uniformly random bits with parity 0. As a robust t-wise independent PRG, any probing attack towards a robust parity sharing generator is equivalent to a probing attack towards the output bits. In Lemma 1, we formally prove that by replacing the randomness source by a robust parity sharing generator in the basic construction, we obtain a leakage-tolerant XOR gadget in the plain model.

Remark 1. The notion of robust parity sharing generators extends the notion of robust t-wise independent PRGs in the following two directions: (1) the parity of the output bits should be 0, and (2) an adversary may access to the output bits by learning not only a single output bit, but also a linear combination specified in the access structure $\mathcal{A}$.

If $\mathcal{A}$ only contains all the output bits, we may obtain a robust t-wise independent PRG from a robust parity sharing generator by removing the last output bit.

Remark 2. Recall that the outer construction uses the leakage-tolerant XOR gadget to compute each masked wire value. We note that if we use fresh randomness for each leakage-tolerant XOR gadget, then the total number of random bits will depend on the circuit size. To solve it, our construction uses a *single* leakage-tolerant private circuit for all XOR functions, which we refer to as a multi-phase leakage-tolerant XOR gadget, to replace the leakage-tolerant XOR gadgets invoked in the outer construction. Note that it is sufficient for our purpose since the number of probed wires is bounded by t in the whole circuit. Correspondingly, we also extend the notion of robust parity sharing generators to what we refer to as multi-phase robust parity sharing generators. We refer the readers to Section 5.2 for more details.

In the following, however, we still focus on robust parity sharing generators for simplicity. The idea can be easily extended to the multi-phase version.

Construction of a Robust Parity Sharing Generator. Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_m)$ denote the input random bits of the generator G. Our idea is to use a matrix $\boldsymbol{M}$ of size $n \times m$ to compute the output correlated random bits $\boldsymbol{r} = (r_1, r_2, \ldots, r_n) = \boldsymbol{M} \cdot \boldsymbol{u}$. To compute r_i,

1. For all $w \in \{1, 2, \ldots, m\}$, G computes $M_{i,w} \cdot u_w$ in parallel. Here $M_{i,w}$ is the entry at i-th row and w-th column in $\boldsymbol{M}$.
2. G computes the $r_i = \oplus_{w=1}^{m} M_{i,w} \cdot u_w$ by using a $\lceil \log m \rceil$-depth addition circuit (See Section 2.2 for more details about the addition circuit).

Requiring that the parity of $r_1, r_2, \ldots, r_n$ is 0 is equivalent to requiring that $\sum_{i=1}^{n} \boldsymbol{M}_i = \mathbf{0}$.

Our idea is to use a random matrix $\boldsymbol{M}$ and show that the construction of G is a robust parity sharing generator with high probability when $m = O(t \cdot \log tn)$. Note that the structure of G is independent of the matrix $\boldsymbol{M}$. Although a probing attack can depending on $\boldsymbol{M}$, the set of all possible probing attacks is independent of $\boldsymbol{M}$. This allows us to first analyse the probability that G is secure against a fixed probing attack and then apply the union bound on all possible probing attacks.

Therefore, the problem becomes that for a fixed set S of at most k wires in G and a fixed set W of at most t variables in $\mathcal{A}$, there is a set T of at most $q|S|$ output bits (where T only depends on S) such that for any fixing of the values G_S of the wires in S and $\boldsymbol{r}_I$, where $I = \{i : r_i \in T\}$,

$$\texttt{Dist}(\boldsymbol{r}|_{C_S, \boldsymbol{r}_I}, W) = \texttt{Dist}(\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I}, W).$$

At a high-level, the proof works as follows:

1. We first determine the set T. For each wire in S, if it is r_i or an intermediate wire when computing r_i, we insert r_i in T. Then T contains at most $|S|$ output bits. Intuitively, T corresponds to the set of output bits whose distributions are affected by the wires in S.
2. We note that if a variable w in W is a linear combination of other variables in $W \bigcup T$, then we can safely remove w and only consider $W \backslash \{w\}$. Note that if the argument holds for $W \backslash \{w\}$, then it also holds for W since w is fully determined by the variables in $W \bigcup T \backslash \{w\}$. Therefore, our second step is to find $\tilde{W} \subset W$ such that no variable in $\tilde{W}$ is a linear combination of other variables in $\tilde{W} \bigcup T$.
3. Recall that $\tilde{W} \subset W \subset \mathcal{A}$ and $T \subset \mathcal{A}$. Thus, all variables in $\tilde{W} \bigcup T$ are linear combinations of $r_1, r_2, \ldots, r_n$. We show that the distribution $\texttt{Dist}(\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I}, \tilde{W})$ is the same as the distribution of $|\tilde{W}|$ uniform bits. Therefore, the problem is reduced to analyse the probability that the distribution $\texttt{Dist}(\boldsymbol{r}|_{C_S, \boldsymbol{r}_I}, \tilde{W})$ is the distribution of $|\tilde{W}|$ uniform bits.
4. We prove that this is equivalent to showing that the probability that for all *non-empty* subset $X \subset \tilde{W}$ and for all subset $Y \subset S \bigcup T$, the XOR of all bits in $X \bigcup Y$ is uniformly distributed. To this end, we first analyse the probability for fixed sets X, Y and then apply the union bound on all possible X, Y.
5. Finally, we note that all variables in $S \bigcup T \bigcup \tilde{W}$ are linear combinations of the input random bits $u_1, u_2, \ldots, u_m$. Therefore, there exists a vector $\boldsymbol{v}(w)$ for all $w \in S \bigcup T \bigcup \tilde{W}$ such that $w = \boldsymbol{v}(w) \cdot \boldsymbol{u}$. The XOR of all bits in $X \bigcup Y$ is uniformly distributed if and only if the summation of the vectors $\boldsymbol{v}(w)$ for

all $w \in X \bigcup Y$ is a non-zero vector. Since $\boldsymbol{M}$ is a uniformly random matrix, we show that this holds with overwhelming probability.

We refer the readers to Section 6 for more details. Combining all the components we construct, we have the following theorem.

Theorem 1. *Any function f with circuit size s and output size n_o admits a t-private implementation (I, C, O) with the canonical encoder I and the decoder O which, for each output bit, takes the parity of a block of bits (which are not restricted to $t+1$ bits), where C uses $O(t \cdot \log ts)$ random bits. Moreover, there exists a PPT algorithm which takes $(\tilde{C}_f, 1^t, 1^\lambda)$ as input, where $\tilde{C}_f$ is a circuit of size s that computes f, and outputs, except with $\leq 2^{-\lambda}$ probability, a t-private implementation (I, C, O) such that C uses $O(t \cdot \log ts + \lambda)$ random bits.*

In the full version, we show how to construct a t-leakate-resilient private circuit with the canonical decoder at the cost of $O(t \cdot \log \lambda)$ extra random bits.

Randomness Complexity of t-Leakage-Tolerant Private Circuits. In Remark 3, we show that a t-leakage-tolerant private circuit can be obtained from our outer construction with small modifications. When the leakage-tolerant XOR gadgets are instantiated by a multi-phase leakage-tolerant XOR gadget, we have the following theorem.

Theorem 2. *Any function f with circuit size s admits a t-leakage-tolerant private implementation C, where C uses $O(t \cdot \log ts)$ random bits. Moreover, there exists a PPT algorithm which takes $(\tilde{C}_f, 1^t, 1^\lambda)$ as input, where $\tilde{C}_f$ is a circuit of size s that computes f, and outputs, except with $\leq 2^{-\lambda}$ probability, a t-leakage-tolerant implementation C such that C uses $O(t \cdot \log ts + \lambda)$ random bits.*

2.3 Extensions

Replacing the Canonical Encoder with a Randomness-Efficient Encoder. We note that the canonical encoder has already required $t \cdot n_i$ random bits for the input in I. When $n_i \geq t$, it means that the encoding of the input has already contained $O(t^2)$ random bits. It may lead to the following objection: a potential solution may reuse the randomness output by the encoder and may even be deterministic due to the large amount of randomness output by the encoder. We show that we can replace the canonical encoder by a randomness-efficient encoder, and achieve randomness complexity of $O(t \cdot \log ts)$ including the input encoder.

We first construct an encoder which only requires $O(t \cdot \log n_i)$ random bits to encode n_i bits by using a linear and strong t-wise independent PRG $G : \{0,1\}^m \to \{0,1\}^{n_i}$. The construction works as follows:

1. The encoder $\mathtt{Enc}$ takes $\boldsymbol{x} \in \{0,1\}^{n_i}$ as input and $\boldsymbol{\rho} \in \{0,1\}^m$ as randomness.
2. The encoder $\mathtt{Enc}$ first computes $\boldsymbol{r} = G(\boldsymbol{\rho})$. Then it computes $\boldsymbol{x} \oplus \boldsymbol{r}$.
3. The output of $\mathtt{Enc}$ is $(\boldsymbol{\rho}, \boldsymbol{x} \oplus \boldsymbol{r})$.

Note that it follows the same idea as the outer construction. We then show that we can directly replace the canonical encoder with this construction. Informally, this is because each input bit x_i is equal to the XOR of a subset of output bits of the encoder. To see this, since G is linear, each output bit r_i of G is equal to the XOR of a subset $\mathtt{supp}(r_i)$ of bits of the input $\boldsymbol{\rho}$. Therefore, each output bit x_i is equal to the XOR of the bits in $\mathtt{supp}(r_i) \bigcup \{x_i \oplus r_i\}$ which are all in the output of the encoder. Thus, we may define $[x_i] = \mathtt{supp}(r_i) \bigcup \{x_i \oplus r_i\}$ and we can use the same outer construction to transform the input encoding to the masked input bits.

As a result, we have the following theorem. We refer the readers to the full version for more details.

Theorem 3. *Any function f with circuit size s and input size n_i admits a t-private implementation (I, C, O), where I uses $O(t \cdot \log n_i)$ random bits and C uses $O(t \cdot \log ts)$ random bits. Moreover, there exists a PPT algorithm which takes $(\tilde{C}_f, 1^t, 1^\lambda)$ as input, where $\tilde{C}_f$ is a circuit of size s that computes f, and outputs, except with $\leq 2^{-\lambda}$ probability, a t-private implementation (I, C, O) such that C uses $O(t \cdot \log ts + \lambda)$ random bits.*

Private Stateful Circuit. We follow the same argument as [9] to transform our basic construction to a private stateful circuit. Concretely, in the first step, we extend our construction to support unprotected input bits and output bits. The unprotected input bits and output bits are not encoded and can be observed by the public. Note that for an unprotected input x_i, we may set $[x_i] = \{x_i\}$ so that we can continue to use our basic construction.

In the second step, we use the randomness-efficient encoder $\mathtt{Enc}$ constructed above to encode the initial state. In each invocation, we use the private circuit which takes as input the encoded state and the unprotected external input, and outputs the encoded updated state and the external output. Note that since $\mathtt{Enc}$ follows from the same idea as the outer construction, the encoded updated state produced by our private circuit has the same form as that computed by $\mathtt{Enc}$. Therefore, the encoded updated state produced by our private circuit is stored and will be used in the next invocation. The security directly follows from the private circuit that supporting unprotected input bits and output bits.

As a result, we have the following theorem. We refer the readers to the full version for more details.

Theorem 4. *Any stateful circuit C with initial state s_0 admits a t-private implementation $C'[s'_0]$ which uses $O(t \cdot \log t|C|)$ random bits. Moreover, there exists a PPT algorithm which takes $(C, s_0, 1^t, 1^\lambda)$ as input and outputs, except with $\leq 2^{-\lambda}$ probability, a t-private implementation $C'[s'_0]$ such that C' uses $O(t \cdot \log t|C| + \lambda)$ random bits.*

3 Preliminaries

3.1 Private Circuits

We start by defining the simple "stateless" variant of private circuits.

Definition 3 (Private Circuit [9]). *A private (stateless) circuit for* $f : \{0,1\}^{n_i} \to \{0,1\}^{n_o}$ *is a triple* (I, C, O) *where* $I : \{0,1\}^{n_i} \to \{0,1\}^{\hat{n}_i}$ *is a randomized input encoder,* C *is a randomized boolean circuit with input* $\hat{\boldsymbol{x}} \in \{0,1\}^{\hat{n}_i}$*, output* $\hat{\boldsymbol{y}} \in \{0,1\}^{\hat{n}_o}$*, and randomness* $\boldsymbol{\rho} \in \{0,1\}^m$*, and* $O : \{0,1\}^{\hat{n}_o} \to \{0,1\}^{n_o}$ *is an output decoder, such that for any input* $x \in \{0,1\}^{n_i}$*, we have*

$$\Pr[O(C(I(\boldsymbol{x}), \boldsymbol{\rho})) = f(\boldsymbol{x})] = 1,$$

where the probability is over the randomness of I *and* $\boldsymbol{\rho}$*.*

In this work, the term "private circuit" will refer to the above stateless notion by default. In the full version, we also discuss the stateful variant and show how to extend our main result to this stronger model.

We will be interested in two different notions of security for private circuits: the standard notion of *leakage-resilience* (against probing attacks) and a more refined notion of *leakage-tolerance*. We define both notions below.

Leakage-resilient Private Circuit. In the setting of leakage-resilient private circuits, we consider the canonical encoder: I encodes each input bit x_i by a vector of $t+1$ random bits with parity x_i. This is mainly to avoid encoders which are function-dependent or provide large amount of randomness. For each input bit x, we use $[x_i]$ to denote the set of bits in the encoding of x_i. As to the decoder, we consider a relaxation of the canonical decoder: To decode each output bit, O takes the parity of a block of bits (which are not restricted to be $t+1$ bits).

We note that the canonical encoder consumes $O(t \cdot n_i)$ random bits to encode an input $\boldsymbol{x} \in \{0,1\}^{n_i}$. Later on, we will also consider a randomness-efficient encoder which only uses $O(t \log n_i)$ random bits.

Definition 4 (t-leakage-resilient Privacy [9]). *We say that* C *is a* t*-leakage-resilient private implementation of* f *with encoder* I *and decoder* O *if for any* $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{n_i}$ *and any set* P *of* t *wires in* C*, the distributions* $C_P(I(\boldsymbol{x}), \boldsymbol{\rho})$ *and* $C_P(I(\boldsymbol{x}'), \boldsymbol{\rho})$ *are identical, where* C_P *denotes the set of* t *bits on the wires from* P*.*

In the following, whenever we say t-private circuit, we refer to a t-leakage-resilient private circuit.

Leakage-tolerant Private Circuit. In the setting of leakage-tolerant private circuits, we restrict the encoder and the decoder to be the identity function. The security requires that any set of at most t wires in C should leak at most the *same number* of input and output bits. Formally,

Definition 5 (t-leakage-tolerant privacy). *We say that* C *is a* t*-leakage-tolerant private implementation of* f *if there exists a simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that for all* $\boldsymbol{x}$ *and any set* P *of at most* t *wires in* C*,* $\mathcal{S}_1(C, P)$ *outputs a set* P' *of* $|P|$ *input and output wires in* C *such that*

$$C_P(\boldsymbol{x}, \boldsymbol{\rho}) = \mathcal{S}_2(C, C_{P'}(\boldsymbol{x}, \boldsymbol{\rho})),$$

where C_P *(w.r.t.* $C_{P'}$*) denotes the set of bits on the wires from* P *(w.r.t.* P'*).*

We note that the input wires and output wires can be naively simulated by having access to those wires. Therefore, it is sufficient to only consider the intermediate wires in C. We have the following equivalent definition.

Definition 6. *We say that C is a t-leakage-tolerant private implementation of f if there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all $\boldsymbol{x}$ and any set P of at most t* intermediate *wires in C, $\mathcal{S}_1(C, P)$ outputs a set P' of $|P|$ input and output wires in C such that*

$$C_P(\boldsymbol{x}, \boldsymbol{\rho}) = \mathcal{S}_2(C, C_{P'}(\boldsymbol{x}, \boldsymbol{\rho})),$$

where C_P (w.r.t. $C_{P'}$) denotes the set of bits on the wires from P (w.r.t. P').

3.2 Strong t-wise Independent Pseudo-random Generator

Our work will make use of the following notion of (strong) t-wise independent pseudo-random generator.

Definition 7 ((Strong) t-wise Independent PRG). *A function $G : \{0,1\}^n \to \{0,1\}^m$ is a t-wise independent pseudo-random generator (or PRG for short) if any subset of t bits of $G(x)$ is uniformly random and independently distributed when x is uniformly sampled from $\{0,1\}^n$.*

If any subset of t bits of $(x, G(x))$ is uniformly random and independently distributed when x is uniformly sampled from $\{0,1\}^n$, then we say G is a strong *t-wise independent PRG.*

We say a (strong) t-wise independent PRG G is linear if any output bit of $G(x)$ is equal to the XOR of a subset of bits in x.

Generic Construction of Linear and Strong t-wise Independent PRGs. In [4], Chor et al. introduce the notion of t-resilient functions. A t-resilient function $\texttt{Ext} : \{0,1\}^n \to \{0,1\}^{n'}$ satisfies that the output of $\texttt{Ext}$ is uniformly random given any t bits from the input.

A generic approach of constructing a linear and strong t-wise independent PRG is to combine a *linear* t-resilient function $\texttt{Ext} : \{0,1\}^n \to \{0,1\}^{n'}$ and a *linear* t-wise independent PRG $G' : \{0,1\}^{n'} \to \{0,1\}^m$. Consider the function $G : \{0,1\}^n \to \{0,1\}^m$ which is defined to be $G(x) = G'(\texttt{Ext}(x))$. Since $\texttt{Ext}$ and G' are linear, G is also linear. Note that after fixing at most t bits in x, the output of $\texttt{Ext}(x)$ is uniformly distributed, which means that the output of $G(x) = G'(\texttt{Ext}(x))$ is t-wise independent. Therefore, any subset of t bits of $(x, G(x))$ is uniformly random and independently distributed, which means that G is a linear and strong t-wise independent PRG.

A Concrete Instance. For our purposes, it will suffice to use the following simple construction using polynomial evaluation over a finite field $\mathbb{F}_{2^k}$. We use $\alpha_0, \alpha_1, \ldots, \alpha_{2^k-1}$ to represent field elements in $\mathbb{F}_{2^k}$. Let $\boldsymbol{r} = (r_0, r_1, \ldots, r_{t-1}) \in$

$F_{2^k}^t$. Consider the degree-$(t-1)$ polynomial $h_{\boldsymbol{r}}(\cdot)$ which satisfies that $h_{\boldsymbol{r}}(\alpha_i) = r_i$ for all $i \in \{0, 1, \ldots, t-1\}$. That is

$$\boldsymbol{r} = (h_{\boldsymbol{r}}(\alpha_0), h_{\boldsymbol{r}}(\alpha_1), \ldots, h_{\boldsymbol{r}}(\alpha_{t-1})).$$

If $\boldsymbol{r}$ is a uniform vector in $\mathbb{F}_{2^k}^t$, then $h_{\boldsymbol{r}}(\cdot)$ is a random degree-$(t-1)$ polynomial, which means that any t distinct evaluation points of $h_{\boldsymbol{r}}(\cdot)$ are uniformly random and independently distributed. Thus, for all $\ell \leq 2^k - t$ we may define $G : \mathbb{F}_{2^k}^t \to \mathbb{F}_{2^k}^\ell$ by:

$$G(\boldsymbol{r}) = (h_{\boldsymbol{r}}(\alpha_t), h_{\boldsymbol{r}}(\alpha_{t+1}), \ldots, h_{\boldsymbol{r}}(\alpha_{\ell+t-1})).$$

To see why G is a strong t-wise independent PRG, note that $(\boldsymbol{r}, G(\boldsymbol{r})) = (h_{\boldsymbol{r}}(\alpha_0), h_{\boldsymbol{r}}(\alpha_1), \ldots, h_{\boldsymbol{r}}(\alpha_{\ell+t-1}))$, and any t distinct evaluation points of $h_{\boldsymbol{r}}(\cdot)$ are uniformly random and independently distributed when $\boldsymbol{r}$ is uniformly sampled from $\mathbb{F}_{2^k}^t$.

Since every element in $\mathbb{F}_{2^k}$ can be represented by k bits, it gives us a strong t-wise independent PRG which takes as input $t \cdot k$ bits and outputs $\ell \cdot k$ bits. Note that ℓ can be as large as $2^k - t + 1$. Therefore, to construct a strong t-wise independent PRG which outputs m bits, we only need to use a field of size $O(m)$, which means that the input size can be as small as $O(t \cdot \log m)$.

We note that in the above construction, each output element $h_{\boldsymbol{r}}(\alpha_i)$ can be written as a linear combination of $\boldsymbol{r}$. Since we are in an extension field of the binary field where addition is equivalent to coordinate-wise XOR, it implies that every output bit of G is the XOR of a subset of its input. Therefore, for all m, we obtain a linear and strong t-wise independent PRG which takes as input $O(t \cdot \log m)$ bits.

4 Outer Construction: *t*-private Circuit via Leakage-tolerant XOR Gadgets

For a boolean function $f : \{0,1\}^{n_i} \to \{0,1\}^{n_o}$, let $\tilde{C}$ denote a circuit that computes the function f. Let G be a linear and strong $2t$-wise independent PRG $G : \{0,1\}^m \to \{0,1\}^{|\tilde{C}|}$. We use $\boldsymbol{u} = (u_1, u_2, \ldots, u_m)$ to denote the input of G and $\boldsymbol{r} = (r_1, r_2, \ldots, r_{|\tilde{C}|})$ to denote the output of G. Since G is linear, each output bit r_i is the XOR of a subset of the input bits in $u_1, u_2, \ldots, u_m$. We refer to this set as the support of r_i, denoted by $\mathtt{supp}(r_i)$.

Our idea is to compute a masked bit for each wire value in $\tilde{C}$ using the output bit of G. Concretely, suppose all the wire values in $\tilde{C}$ are denoted by $g_1, g_2, \ldots, g_{|\tilde{C}|}$ (including input wires and output wires). For all $i \in 1, 2, \ldots, |\tilde{C}|$, we want to compute $g_i \oplus r_i$, where r_i is the i-th output bit of G. Intuitively, any set of t bits in $\{u_1, u_2, \ldots, u_m, g_1 \oplus r_1, g_2 \oplus r_2, \ldots, g_{|\tilde{C}|} \oplus r_{|\tilde{C}|}\}$ are uniformly random and independently distributed and therefore can be simulated by simply choosing t uniform bits. Note that for each wire value g_i, we can view $\{g_i \oplus r_i\} \bigcup \mathtt{supp}(r_i)$ as an additive sharing of g_i since the XOR of the bits in $\mathtt{supp}(r_i)$ is equal to r_i. We use $[g_i]$ to represent the set $\{g_i \oplus r_i\} \bigcup \mathtt{supp}(r_i)$.

We will first construct a t-private circuit by using a t-leakage-tolerant private circuit for the XOR function, which we referred to as a t-leakage-tolerant XOR gadget.

Intuitively, a leakage-tolerant XOR gadget satisfies that any probing attack towards the gadget is equivalent to a probing attack (that probes the same number of wires) towards the input wires and the output wire of this gadget. The construction of the circuit C works as follows.

1. The circuit C takes as input the input encoding $[x_1], [x_2], \ldots, [x_{n_i}]$ and the randomness $\rho \in \{0,1\}^m$. We use $\boldsymbol{\rho}$ as the input of the PRG G.
2. Transforming Input Encoding: We first transform the input encoding to the masked input bits using the corresponding output bits from G. For each input x_i, let g_i denote the input wire in $\tilde{C}$ that takes x_i as input. Then we want to compute the bit $g_i \oplus r_i$ in the circuit C. This is done by using a t-leakage-tolerant XOR gadget $\mathcal{G}$ with input bits in $[x_i]$ and $\mathtt{supp}(r_i)$.
3. Evaluating Addition Gates in $\tilde{C}$: For each addition gate in $\tilde{C}$ with input wires g_a, g_b and output wire g_c. Suppose we have constructed the circuit to compute $g_a \oplus r_a$ and $g_b \oplus r_b$ in C. To compute $g_c \oplus r_c$, we insert a t-leakage-tolerant XOR gadget $\mathcal{G}$ with input bits in $[g_a], [g_b]$ and $\mathtt{supp}(r_c)$.
4. Evaluating Multiplication Gates in $\tilde{C}$: For each multiplication gate in $\tilde{C}$ with input wires g_a, g_b and output wire g_c. Suppose we have constructed the circuit to compute $g_a \oplus r_a$ and $g_b \oplus r_b$ in C. Recall that $[g_a] = \{g_a \oplus r_a\} \bigcup \mathtt{supp}(r_a)$ and $[g_b] = \{g_b \oplus r_b\} \bigcup \mathtt{supp}(r_b)$. Let $[g_a] \otimes [g_b] := \{u \cdot v : u \in [g_a], v \in [g_b]\}$. Then the XOR of all bits in $[g_a] \otimes [g_b]$ is equal to $g_a \cdot g_b = g_c$. Thus, to compute $g_c \oplus r_c$, we first compute $u \cdot v$ for all $u \in [g_a], v \in [g_b]$ and then insert a t-leakage-tolerant XOR gadget $\mathcal{G}$ with input bits in $[g_a] \otimes [g_b]$ and $\mathtt{supp}(r_c)$.
5. Transforming to Output Encoding: The last step is to transform each masked output bit to the encoding of this output bit. For each output wire g_a in $\tilde{C}$, suppose we have constructed the circuit to compute $g_a \oplus r_a$ in C. Recall that $[g_a] = \{g_a \oplus r_a\} \bigcup \mathtt{supp}(r_a)$. Therefore, the XOR of the bits in $[g_a]$ is equal to g_a. The circuit simply outputs the wires in $[g_a]$ as the encoding of the output bit g_a.

Theorem 5. *Assume $\mathcal{G}$ is a t-leakage-tolerant XOR gadget. The circuit C constructed above is a t-private implementation of the function f.*

Proof. Following the definition of the t-privacy in Definition 4, it is sufficient to show that, for any $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{n_i}$ and any set P of t wires in C, the distributions $C_P(I(\boldsymbol{x}), \boldsymbol{\rho})$ and $C_P(I(\boldsymbol{x}'), \boldsymbol{\rho})$ are identical, where C_P denotes the set of t bits on the wires from P. Since $\mathcal{G}$ is t-leakage-tolerant, any set of at most t intermediate variables within $\mathcal{G}$ can be perfectly simulated by probing the same number of the input wires and the output wire of $\mathcal{G}$. Therefore, it is sufficient to only focus on the set of t wires in C that does not include intermediate wires in the gadgets.

We first divide the wires in C (excluding the intermediate wires in the gadgets) into two disjoint sets:

1. The set of input wires of C: $\{[x_1], [x_2], \ldots, [x_{n_i}]\}$.

2. The set of the rest of wires in C. It is consist of the random bits $\boldsymbol{\rho} \in \{0,1\}^m$, the masked bits of the wire values $g_1 \oplus r_1, g_2 \oplus r_2, \ldots, g_{|\tilde{C}|} \oplus r_{|\tilde{C}|}$, and the bits in the set $[g_a] \otimes [g_b]$ for each multiplication gate in $\tilde{C}$ with input wires g_a and g_b.

Note that the output wires are included in the second set. We strengthen the argument by allowing to choose t wires in each of these two sets and show that these $2t$ wires can be simulated without knowing the input and the output of f.

For the first set, note that each sharing of $x_1, \ldots, x_{n_i}$ is a random additive sharing with $t+1$ shares. Therefore, any t shares in the first set are uniformly random. We can simulate the bits on the t wires chosen in the first set by uniform values.

For the second set, let $\boldsymbol{\rho} = (u_1, u_2, \ldots, u_m) \in \{0,1\}^m$. Note that any wire $u \cdot v \in [g_a] \otimes [g_b]$, where $u \in [g_a]$ and $v \in [g_b]$, is determined by u and v. Therefore, any t wires in the third set are determined by at most $2t$ wires in the set $T = \{u_1, u_2, \ldots, u_m, g_1 \oplus r_1, g_2 \oplus r_2, \ldots, g_{|\tilde{C}|} \oplus r_{|\tilde{C}|}\}$. Recall that $(r_1, r_2, \ldots, r_{|\tilde{C}|})$ is the output of G on input $\boldsymbol{\rho} = (u_1, u_2, \ldots, u_m) \in \{0,1\}^m$. Since G is a strong $2t$-wise independent PRG, when $\boldsymbol{\rho} = (u_1, u_2, \ldots, u_m) \in \{0,1\}^m$ is chosen uniformly, $(u_1, u_2, \ldots, u_m, r_1, r_2, \ldots, r_{|\tilde{C}|})$ are $2t$-wise independent. Thus, we can simulate the bits on the t wires chosen in the third set by first sampling $2t$ random bits for the wires in T that determine these t wires and then compute the bits on these t wires.

Remark 3. Recall that in the setting of leakage-tolerant private circuits, function input and output are not encoded. We note that the above construction with small modifications gives a *t-leakage-tolerant* private implementation of f: (1) for each input bit x_i, we use $[x_i] = \{x_i\}$ in the above construction; and (2) for each output bit y_i, we use an additional leakage-tolerant XOR gadget with bits in $[y_i]$ to compute y_i.

Recall that in the setting of leakage-tolerant private circuits, the security requires that any set of t wires in C can be simulated by the same number of input and output bits. To show security, we divide the wires in C (after the modifications) into two disjoint sets: (1) the set of all input wires and output wires, and (2) the set of the rest of wires. Note that the second set is consist of the random bits $\boldsymbol{\rho} \in \{0,1\}^m$, the masked bits of the wire values $g_1 \oplus r_1, g_2 \oplus r_2, \ldots, g_{|\tilde{C}|} \oplus r_{|\tilde{C}|}$, and the bits in the set $[g_a] \otimes [g_b]$ for each multiplication gate in $\tilde{C}$ with input wires g_a and g_b. The simulator works by querying the corresponding input and output wires in the first set and simulating the wires in the second set in the same way as described in the proof of Theorem 5.

5 Inner Construction: Leakage-tolerant XOR Gadget

Following from Theorem 5, it is sufficient to construct a leakage-tolerant XOR Gadget. We first start with a basic construction which is given access to correlated randomness.

5.1 Basic Construction via Correlated Randomness

Let $x_1, x_2, \ldots, x_n$ denote the input bits of the gadget $\mathcal{G}$. The goal is to compute the output $\oplus_{i=1}^{n} x_i$. The construction of the circuit works as follows:

1. The circuit takes as input n bits $x_1, x_2, \ldots, x_n$. The circuit is given an access to n correlated random variables $r_1, r_2, \ldots, r_n$ which are uniformly random bits with parity 0.
2. The circuit first computes $g_i = x_i \oplus r_i$ for all $i \in \{1, 2, \ldots, n\}$ in parallel.
3. The circuit computes $\oplus_{i=1}^{n} g_i$ using a $\lceil \log n \rceil$-depth addition circuit. Concretely, in each iteration, all input bits are divided into groups of size 2 and then, for each group, we compute the XOR of the two bits in this group. The results are provided as input bits for the next iteration. Note that in each iteration, we reduce the number of input bits by a factor of 2. The whole process will end after $\lceil \log n \rceil$ iterations.

We show that this simple construction is t-leakage-tolerant given the correlated random variables $r_1, r_2, \ldots, r_n$.

First note that there are two different kinds of intermediate variables: (1) the first kind contains the correlated random variables $\{r_1, r_2, \ldots, r_n\}$, (2) the second kind contains the variables $\{g_1, g_2, \ldots, g_n\}$ computed in Step 2, and all intermediate variables when computing the addition circuit. Note that they are all in the form of $\oplus_{i \in L} g_i$ where $L \subset \{1, 2, \ldots, n\}$.

For any set W of $t_1 (\leq t)$ intermediate variables, we first determine the set $T \subset \{x_1, x_2, \ldots, x_n, \oplus_{i=1}^{n} x_i\}$ of size at most t_1 that will be used to simulate the intermediate variables in W. We first define I to be a subset of the indices $\{1, 2, \ldots, n\}$ such that for all $r_i \in W$, $i \in I$; or equivalently $I := \{i : r_i \in W\}$. Initially, we set T to be an empty set.

- For all $i \in I$, we insert the i-th input bit x_i in T.
- If W contains any intermediate variable of the second kind (i.e., an intermediate bit in the form of $\oplus_{i \in L} g_i$ where $L \subset \{1, 2, \ldots, n\}$), we insert the output bit $\oplus_{i=1}^{n} x_i$ in T.

Note that the size of T is at most t_1.

Now we show how to simulate the intermediate variables in W using the input bits and the output bit in T. For all $r_i \in W$, we sample a random bit as r_i. Since $x_i \in T$, we also compute $g_i = x_i \oplus r_i$. If W does not contain any intermediate variable of the second kind, then we are done. Otherwise, the rest of variables in W are all in the form of $\oplus_{i \in L} g_i$ where $L \subset \{1, 2, \ldots, n\}$. If we can generate g_i for all $i \in \{1, 2, \ldots, n\}$, then we can simulate the rest of variables in W. Since we have computed g_i for all $i \in I$, it is sufficient to focus on g_i where $i \notin I$.

Recall that for all $i \in I$, we have $x_i \in T$. Also recall that if W contains any intermediate variable of the second kind, then $\oplus_{i=1}^{n} x_i \in T$. Therefore, we can compute the parity of $\{x_i : i \notin I\}$ by $\oplus_{i \notin I} x_i = (\oplus_{i \in I} x_i) \oplus (\oplus_{i=1}^{n} x_i)$. Note that $\{r_i : i \notin I\}$ are random bits with parity $\oplus_{i \notin I} r_i = \oplus_{i \in I} r_i$. For all $i \notin I$, since we

use r_i to mask the bit x_i, $\{g_i : i \notin I\}$ are random bits with parity $\oplus_{i \notin I} g_i$. Thus, we first compute

$$\oplus_{i \notin I} g_i = (\oplus_{i \notin I} x_i) \oplus (\oplus_{i \notin I} r_i) = (\oplus_{i \in I} x_i) \oplus (\oplus_{i=1}^{n} x_i) \oplus (\oplus_{i \in I} r_i).$$

Then, we sample $n - |I|$ random bits with parity $\oplus_{i \notin I} g_i$ as $\{g_i : i \notin I\}$. Finally, we compute the intermediate variables of the second kind in W using $\{g_1, g_2, \ldots, g_n\}$.

5.2 Robust Parity Sharing Generator

Now, we consider to use a generation circuit G for correlated random variables $r_1, r_2, \ldots, r_n$ in the basic construction. In this case, an adversary can also probe wires in G. We first review the definition of robust t-wise independent PRGs introduced in [8]. Then we will extend the notion of robust t-wise independent PRGs to what we refer to as robust parity sharing generators. The following definition corresponds to the strong robust t-wise independent PRGs in [8].

Definition 1 (Robust t-wise Independent PRGs [8]). *A circuit implementation C of a function $G : \{0,1\}^m \to \{0,1\}^n$ is a (t, k, q)-robust pseudo-random generator if the following holds. Let $\boldsymbol{u}$ be a vector of m uniformly random bits, and $\boldsymbol{r} = G(\boldsymbol{u})$. For any set S of at most k wires in C, there is a set T of at most $q|S|$ output bits such that conditioned on any fixing of the values C_S of the wires in S and $\boldsymbol{r}_I$, where $I = \{i : r_i \in T\}$, the values $\boldsymbol{r}_{\bar{I}}$ of the output bits not in T are t-wise independent.*

Intuitively, any probing attack towards a robust t-wise independent PRG is equivalent to a probing attack towards the output bits. In [8], Ishai, et al. show that a private circuit can be derandomized by the following two steps:

1. First, derandomzie a private circuit by assuming an access to t-wise independent random source. This is achieved by considering the notion of randomness locality of a private circuit. The randomness locality of a private circuit is the number of random bits that are used to compute each wire. If each wire uses at most ℓ random bits (i.e., the randomness locality is ℓ), then we may replace the uniform random source by a $(\ell \cdot t)$-wise independent random source to protect against t-probing attacks. It is because any t wires depend on at most $\ell \cdot t$ random bits, and therefore, the distribution of these t wires when using uniform random source is identical to that when using $(\ell \cdot t)$-wise independent random source.
2. Then, replace the $(\ell \cdot t)$-wise independent random source by a robust $(\ell \cdot t)$-wise independent PRG.

However, this approach may inherently requires $\Omega(t^2)$ random bits. Intuitively, the randomness locality of a private circuit cannot be smaller than t, or otherwise, an adversary may learn extra information about the input by probing a wire and all random bits that are used to compute this wire. It means that

the random source should be at least t^2-wise independent, which requires $\Omega(t^2)$ uniform random bits.

To overcome this bottleneck, our idea is to use a different approach to derandomize our basic construction of the leakage-tolerant XOR gadget. We first analyse what random source we need in our construction.

Sufficient Conditions of the Random Source in the Basic Construction. First of all, the correctness of our construction requires that the random variables $(r_1, r_2, \ldots, r_n)$ should have parity 0. Now we want to relax the requirement that $(r_1, r_2, \ldots, r_n)$ are uniformly random bits with parity 0.

We use $\boldsymbol{r}$ to denote the random variables $(r_1, r_2, \ldots, r_n)$ and $\boldsymbol{x}$ to denote the input bits $(x_1, x_2, \ldots, x_n)$. For a set W of intermediate variables in $\mathcal{G}$, we use $\mathtt{Dist}((\boldsymbol{x}, \boldsymbol{r}), W)$ to denote the distribution of the variables in W when they are instantiated by $(\boldsymbol{x}, \boldsymbol{r})$. Let $\tilde{\boldsymbol{r}} = (\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n)$ be uniformly random bits with parity 0. Since we have shown that $\mathcal{G}$ is a leakage-tolerant XOR gadget when using $\tilde{\boldsymbol{r}}$ as random source, a straightforward sufficient condition of the distribution of $\boldsymbol{r}$ is that, for any set W of $t_1(\leq t)$ intermediate variables, $\mathtt{Dist}((\boldsymbol{x}, \boldsymbol{r}), W) = \mathtt{Dist}((\boldsymbol{x}, \tilde{\boldsymbol{r}}), W)$ holds for all $\boldsymbol{x}$. With this condition, when using $\boldsymbol{r}$ as the random source in $\mathcal{G}$, we can simulate the intermediate variables in W by using the same way as that when the random source is $\tilde{\boldsymbol{r}}$.

Recall that in our construction, there are two kinds of intermediate variables: (1) the first kind contains the random variables $\{r_1, r_2, \ldots, r_n\}$, (2) the second kind contains the variables $\{g_1, g_2, \ldots, g_n\}$ computed in Step 2, and all intermediate bits when computing the addition circuit, which are in the form of $\oplus_{i \in L} g_i$ where $L \subset \{1, 2, \ldots, n\}$. Note that $\oplus_{i \in L} g_i = (\oplus_{i \in L} x_i) \oplus (\oplus_{i \in L} r_i)$. Since $\boldsymbol{x}$ are fixed input bits, it is sufficient to only consider the distribution of $\oplus_{i \in L} r_i$. Therefore, consider the set $\mathcal{A}$ defined as:

$$\mathcal{A} = \{r_1, r_2, \ldots, r_n\} \bigcup \{\oplus_{i \in L} r_i : \oplus_{i \in L} g_i \text{ is an intermediate wire in } \mathcal{G}\}.$$

Each variable in $\mathcal{A}$ is a linear combination of $\{r_1, r_2, \ldots, r_n\}$. The sufficient condition above can be transformed to that, for any set W of $t_1(\leq t)$ variables in $\mathcal{A}$, $\mathtt{Dist}(\boldsymbol{r}, W) = \mathtt{Dist}(\tilde{\boldsymbol{r}}, W)$ holds, where $\mathtt{Dist}(\boldsymbol{r}, W)$ refers to the distribution of the variables in W when they are instantiated by $\boldsymbol{r}$. We refer to $\mathcal{A}$ as the access structure of the random variables $\{r_1, r_2, \ldots, r_n\}$. Formally, an access structure $\mathcal{A}$ of a set of variables $\{r_1, r_2, \ldots, r_n\}$ is a set which satisfies that (1) for all $i \in \{1, 2, \ldots, n\}$, $r_i \in \mathcal{A}$, and (2) every variable in $\mathcal{A}$ is a linear combination of $(r_1, r_2, \ldots, r_n)$. One may think that an probing attack can only probe variables in $\mathcal{A}$.

Therefore, we can summarize the sufficient conditions of the distribution of the random source $\boldsymbol{r}$ in the basic construction as follows:

1. The parity of $(r_1, r_2, \ldots, r_n)$ is 0.
2. Let $\mathcal{A}$ be the access structure of $(r_1, r_2, \ldots, r_n)$ as defined above. Let $\tilde{\boldsymbol{r}} = (\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_n)$ be uniformly random bits with parity 0. For any set W of $t_1(\leq t)$ variables in $\mathcal{A}$, $\mathtt{Dist}(\boldsymbol{r}, W) = \mathtt{Dist}(\tilde{\boldsymbol{r}}, W)$.

Robust Parity Sharing Generator. Now we are ready to define the notion of robust parity sharing generators.

Definition 2 (Robust Parity Sharing Generators). *Let $G : \{0,1\}^m \to \{0,1\}^n$ be a function and $\mathcal{A}$ be an access structure of the output bits of G. Let $\boldsymbol{u}$ be a vector of m uniformly random bits, and $\boldsymbol{r} = G(\boldsymbol{u})$. A circuit implementation C of the function G is a (t, k, q)-robust parity sharing generator with respect to $\mathcal{A}$ if the following holds:*

- *The parity of the output bits $\boldsymbol{r}$ is 0.*
- *Let $\tilde{\boldsymbol{r}} \in \{0,1\}^n$ be uniformly random bits with parity 0. For any set S of at most k wires in C, there is a set T of at most $q|S|$ output bits such that for any set W of t variables in $\mathcal{A}$ and for any fixing of the values C_S of the wires in S and $\boldsymbol{r}_I$, where $I = \{i : r_i \in T\}$,*

$$\texttt{Dist}(\boldsymbol{r}|_{C_S,\boldsymbol{r}_I}, W) = \texttt{Dist}(\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I=\boldsymbol{r}_I}, W),$$

where $\texttt{Dist}(\boldsymbol{r}, W)$ is the distribution of the variables in W when they are instantiated by $\boldsymbol{r}$, $\boldsymbol{r}|_{C_S,\boldsymbol{r}_I}$ is the random variable $\boldsymbol{r}$ conditioned on C_S and $\boldsymbol{r}_I$, and $\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I=\boldsymbol{r}_I}$ is the random variable $\tilde{\boldsymbol{r}}$ conditioned on $\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I$.

Let $\mathcal{G}$ be the t-leakage-tolerant XOR gadget we constructed in Section 5.1 that uses correlated randomness. Recall that $(x_1, x_2, \ldots, x_n)$ are input bits, $(r_1, r_2, \ldots, r_n)$ are random bits with parity 0, and $g_i = x_i \oplus r_i$ for all $i \in \{1, 2, \ldots, n\}$. Also recall that the access structure $\mathcal{A}$ of $\{r_1, r_2, \ldots, r_n\}$ is defined by

$$\mathcal{A} = \{r_1, r_2, \ldots, r_n\} \bigcup \{\oplus_{i\in L} r_i : \oplus_{i\in L} g_i \text{ is an intermediate wire in } \mathcal{G}\}.$$

We show how to construct a t-leakage-tolerant XOR gadget $\mathcal{G}'$ in the plain model (i.e., without access to correlated randomness) by using a $(t, t, 1)$-robust parity sharing generator with respect to $\mathcal{A}$. The construction simply uses a $(t, t, 1)$-robust parity sharing generator with respect to $\mathcal{A}$ to generate correlated randomness $(r_1, r_2, \ldots, r_n)$ for $\mathcal{G}$ and then uses $\mathcal{G}$ to compute the output.

Lemma 1. *The gadget $\mathcal{G}'$ constructed above is a t-leakage-tolerant XOR gadget.*

Proof. It is sufficient to show that for any $t_1 \leq t$ and any set W of t_1 intermediate variables, there exists a subset $T \subset \{x_1, x_2, \ldots, x_n, \oplus_{i=1}^n x_i\}$ of size at most t_1 such that the t_1 intermediate variables in W can be perfectly simulated from the bits in T. In the following, we use G to denote the $(t, t, 1)$-robust parity sharing generator.

We first divide the intermediate variables in $\mathcal{G}'$ into two categories:

- The first category contains all the wires in G, including the input random source $(u_1, u_2, \ldots, u_m)$ and the output correlated random variables $(r_1, r_2, \ldots, r_n)$.

– The second category contains the rest of intermediate wires in $\mathcal{G}'$. In other words, the second category contains all intermediate wires in $\mathcal{G}$ except the correlated randomness $(r_1, r_2, \ldots, r_n)$. For all $i \in \{1, 2, \ldots, n\}$, let $g_i = x_i \oplus r_i$. By the construction of $\mathcal{G}$, each variable in the second category is a linear combination of $(g_1, g_2, \ldots, g_n)$.

Let S be the set of intermediate variables in W that belong to the first category, and W' be the set of intermediate variables in W that belong to the second category. Then $|W| = |S| + |W'|$, and $|S|, |W'| \leq t$.

We first determine the set $T \subset \{x_1, x_2, \ldots, x_n, \oplus_{i=1}^n x_i\}$ that is used to simulate the intermediate variables in W. Let T' be the set of output bits of G in Definition 2. Then $|T'| = |S|$. There are two cases.

– If $W' = \emptyset$, then $T = \{x_i : r_i \in T'\}$. In this case $|T| = |T'| = |S| = |W| = t_1$.
– If $W' \neq \emptyset$, then $T = \{x_i : r_i \in T'\} \bigcup \{\oplus_{i=1}^n x_i\}$. In this case, $|T| = |T'| + 1 \leq |S| + |W'| = |W| = t_1$.

Now we describe the simulation of intermediate wires in W.

– For all intermediate wires in S, we sample uniformly random bits as $u_1, u_2, \ldots, u_m$ and compute G by taking $u_1, u_2, \ldots, u_m$ as input. Then we output the values associated with the intermediate wires in S.
– The following step is only done if $W' \neq \emptyset$. In this case, $T = \{x_i : r_i \in T'\} \bigcup \{\oplus_{i=1}^n x_i\}$. Let $I = \{i : r_i \in T'\}$. Then T is also equal to $\{x_i : i \in I\} \bigcup \{\oplus_{i=1}^n x_i\}$
For all intermediate wires in W', let $\tilde{W} = \{\oplus_{i \in L} r_i : \oplus_{i \in L} g_i \text{ is in } W'\}$. Then $\tilde{W} \subset \mathcal{A}$ and $|\tilde{W}| = |W'| \leq t$. Since G is a $(t, t, 1)$-robust parity sharing generator, by Definition 2,

$$\texttt{Dist}(\boldsymbol{r}|_{G_S, \boldsymbol{r}_I}, \tilde{W}) = \texttt{Dist}(\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I}, \tilde{W}).$$

Since $(x_1, x_2, \ldots, x_n)$ are fixed input bits, we have $\texttt{Dist}(\boldsymbol{r}|_{C_S, \boldsymbol{r}_I}, W') = \texttt{Dist}(\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I}, W')$. Therefore, it is sufficient to simulate the intermediate wires in W' by using correlated randomness $\tilde{\boldsymbol{r}}$ subject to $\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I$.
Recall that each variable in W' is a linear combination of $(g_1, g_2, \ldots, g_n)$. Therefore, we first generate $g_1, g_2, \ldots, g_n$ and then compute the variables in W'. For all $i \in I$, we compute $g_i = x_i \oplus \tilde{r}_i$. Note that $\{\tilde{r}_i : i \notin I\}$ are uniformly random bits with parity $\oplus_{i \notin I} \tilde{r}_i = \oplus_{i \in I} \tilde{r}_i$. Also note that we can compute the parity of $\{x_i : i \notin I\}$ by $(\oplus_{i=1}^n x_i) \oplus (\oplus_{i \in I} x_i)$. Therefore, $\{g_i = x_i \oplus \tilde{r}_i : i \notin I\}$ are uniformly random bits with parity $(\oplus_{i \notin I} \tilde{r}_i) \oplus (\oplus_{i \notin I} x_i)$. We generate uniformly random bits with parity $(\oplus_{i \notin I} \tilde{r}_i) \oplus (\oplus_{i \notin I} x_i)$ as $\{g_i : i \notin I\}$. Now we can compute variables in W' by using $(g_1, g_2, \ldots, g_n)$.

Multi-Phase Leakage-Tolerant XOR Gadget and Multi-Phase Robust Parity Sharing Generator. We note that if we use a robust parity sharing generator for each gadget $\mathcal{G}$, then the total number of random bits will depend on the circuit size. To solve it, we first consider what we call multi-phase leakage-tolerant XOR gadgets, which can compute a bounded number of XOR functions. Formally,

Definition 8 (Multi-Phase Leakage-Tolerant XOR Gadget). *Let p and $n_1, n_2, \ldots, n_p$ be positive integers. For all $j \in \{1, 2, \ldots, p\}$, the function f takes as input a sequence of n_j bits $x_1^{(j)}, x_2^{(j)}, \ldots, x_{n_j}^{(j)}$ and outputs $\oplus_{i=1}^{n_j} x_i^{(j)}$. We say $\mathcal{G}$ is a multi-phase t-leakage-tolerant XOR gadget if it is a t-leakage-tolerant private implementation of f.*

Note that it is strictly weaker than p compositions of t-leakage-tolerant XOR gadgets since a multi-phase t-leakage-tolerant XOR gadget can only tolerate t probes across all phases while p compositions of t-leakage-tolerant XOR gadgets can tolerate t probes for each gadget, i.e., $p \cdot t$ probes in total. However, a multi-phase t-leakage-tolerant XOR gadget is sufficient to replace the t-leakage-tolerant XOR gadgets used in the outer protocol (see Section 4) since the number of probed wires is bounded by t in the whole circuit.

To construct a multi-phase t-leakage-tolerant XOR gadget, we extend the t-robust parity sharing generator to the multi-phase version as follows.

Definition 9 (Multi-Phase Robust Parity Sharing Generators). *Let p and $n_1, n_2, \ldots, n_p$ be positive integers, $G : \{0,1\}^m \to \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \ldots \times \{0,1\}^{n_p}$ be a function, $\boldsymbol{u}$ be a vector of m uniformly random bits, and $\boldsymbol{r} = (\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(2)}, \ldots, \boldsymbol{r}^{(p)}) = G(\boldsymbol{u})$, where $\boldsymbol{r}^{(j)} \in \{0,1\}^{n_j}$ for all $j \in \{1, 2, \ldots, p\}$. For each $\boldsymbol{r}^{(j)}$, let $\mathcal{A}_j$ be an access structure of the output bits $\{r_1^{(j)}, r_2^{(j)}, \ldots, r_{n_j}^{(j)}\}$. Let $\mathcal{A} = \bigcup_{j=1}^{p} \mathcal{A}_j$. A circuit implementation C of the function G is a multi-phase (t, k, q)-robust parity sharing generator with respect to $\mathcal{A}$ if the following holds:*

- *For all $j \in \{1, 2, \ldots, p\}$, the parity of the output bits $\boldsymbol{r}^{(j)}$ is 0.*
- *Let $\tilde{\boldsymbol{r}} = (\tilde{\boldsymbol{r}}^{(1)}, \tilde{\boldsymbol{r}}^{(2)}, \ldots, \tilde{\boldsymbol{r}}^{(p)}) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \ldots \times \{0,1\}^{n_p}$ be uniformly random bits such that for all $j \in \{1, 2, \ldots, p\}$, the parity of $\{\tilde{r}_1^{(j)}, \tilde{r}_2^{(j)}, \ldots, \tilde{r}_{n_j}^{(j)}\}$ is 0. For any set S of at most k wires in C, there is a set T of at most $q|S|$ output bits such that for any set W of t variables in $\mathcal{A}$ and for any fixing of the values C_S of the wires in S and $\boldsymbol{r}_I$, where $I = \{(j,i) : r_i^{(j)} \in T\}$ and $\boldsymbol{r}_I$ is the vector that contains all bits in $\{r_i^{(j)} : (j,i) \in I\}$,*

$$\texttt{Dist}(\boldsymbol{r}|_{C_S, \boldsymbol{r}_I}, W) = \texttt{Dist}(\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I}, W),$$

where $\texttt{Dist}(\boldsymbol{r}, W)$ is the distribution of the variables in W when they are instantiated by $\boldsymbol{r}$, $\boldsymbol{r}|_{C_S, \boldsymbol{r}_I}$ is the random variable $\boldsymbol{r}$ conditioned on C_S and $\boldsymbol{r}_I$, and $\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I}$ is the random variable $\tilde{\boldsymbol{r}}$ conditioned on $\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I$.

Let $\mathcal{G}$ be the t-leakage-tolerant XOR gadget we constructed in Section 5.1 that uses correlated randomness. We construct a multi-phase t-leakage-tolerant XOR gadget $\mathcal{G}'$ as follows:

1. First, we use a multi-phase t-robust parity sharing generator G with respect to a proper access structure $\mathcal{A}$ to prepare the correlated randomness for all phases.
2. Then, we use $\mathcal{G}$ to compute the XOR function in each phase.

The access structure $\mathcal{A}$ is defined as follows. For all $j \in \{1, 2, \ldots, p\}$, let $\boldsymbol{x}^{(j)} = (x_1^{(j)}, x_2^{(j)}, \ldots, x_{n_j}^{(j)})$ be the input bits and $\boldsymbol{r}^{(j)} = (r_1^{(j)}, r_2^{(j)}, \ldots, r_{n_j}^{(j)})$ be the random bits with parity 0, and $g_i^{(j)} = x_i^{(j)} \oplus r_i^{(j)}$ for all $i \in \{1, 2, \ldots, n\}$. We define the access structure $\mathcal{A}_j$ of $\{r_1^{(j)}, r_2^{(j)}, \ldots, r_{n_j}^{(j)}\}$ to be

$$\mathcal{A}_j = \{r_1^{(j)}, r_2^{(j)}, \ldots, r_{n_j}^{(j)}\} \bigcup \{\oplus_{i \in L} r_i^{(j)} : \oplus_{i \in L} g_i^{(j)} \text{ is an intermediate wire of } \mathcal{G}(\boldsymbol{x}^{(j)}; \boldsymbol{r}^{(j)})\}.$$

Then, the access structure $\mathcal{A} = \bigcup_{j=1}^{p} \mathcal{A}_j$. We show that this simple construction $\mathcal{G}'$ is a multi-phase t-leakage-tolerant XOR gadget.

Lemma 2. *The gadget $\mathcal{G}'$ constructed above is a multi-phase t-leakage-tolerant XOR gadget.*

The proof can be found in the full version.

Randomness Complexity of Our t-Private Circuit. In Section 6, we will show the following theorem.

Theorem 6. *For all positive integers p and $n_1, n_2, \ldots, n_p$, let $N = \sum_{i=1}^{p} n_i$ and $\mathcal{A}$ be the access structure defined above. There exists a PPT algorithm which takes $(1^t, 1^\lambda, \mathcal{A})$ as input, and outputs a multi-phase $(t, t, 1)$-robust parity sharing generator with respect to $\mathcal{A}$ with probability $1 - 2^{-\lambda}$ such that the input size $m = O(t \cdot \log tN + \lambda)$.*

The proof follows from Lemma 3 in Section 6.

We analyse the randomness complexity of our t-private circuit. Recall that f is the function we want to compute and n_i is the input size of f.

- Recall that in the outer construction, the random bits are used as input of a linear and strong $2t$-wise independent PRG G with output size $|\tilde{C}|$, where $\tilde{C}$ is a circuit that computes the function f. By using the construction in Section 3.2, the number of random bits that are used in G is bounded by $O(t \cdot \log |\tilde{C}|)$.
- For the inner construction, we only need to use random bits to instantiate the multi-phase $(t, t, 1)$-robust parity sharing generator in Theorem 6. To this end, we analyse the number of phases and the input size of each phase. In the outer construction, we invoke the leakage-tolerant XOR gadget for each wire in $\tilde{C}$. Therefore, the number of phases $p = |\tilde{C}|$.
  Recall that for each wire value g_i in the outer construction, we use $[g_i]$ to represent the set $\{g_i \oplus r_i\} \bigcup \mathtt{supp}(r_i)$, where r_i is the output bit of the $2t$-wise independent PRG G that is associated with g_i. Since the input size of G is bounded by $O(t \cdot \log |\tilde{C}|)$, the size of $[g_i]$ is also bounded by $O(t \cdot \log |\tilde{C}|)$. Note that:
  - For an input wire which carries the value x_i, the input size of the leakage-tolerant XOR gadget is $|[x_i]| + |\mathtt{supp}(r_i)| = O(t \cdot \log |\tilde{C}|)$.
  - For an addition gate with input wires g_a, g_b and output wire g_c, the input size of the leakage-tolerant XOR gadget is $|[g_a]| + |[g_b]| + |\mathtt{supp}(r_c)| = O(t \cdot \log |\tilde{C}|)$.

- For a multiplication gate with input wires g_a, g_b and output wire g_c, the input size of the leakage-tolerant XOR gadget is $|[g_a] \otimes [g_b]| + |\mathtt{supp}(r_c)| = O(t^2 \log^2 |\tilde{C}|)$.

Therefore, $N \leq p \cdot \max\{n_1, n_2, \ldots, n_p\} = O(|\tilde{C}| \cdot t^2 \log^2 |\tilde{C}|)$. Thus, inner construction requires $O(t \cdot \log(t|\tilde{C}| \cdot t^2 \log^2 |\tilde{C}|) + \lambda) = O(t \cdot \log t|\tilde{C}| + \lambda)$ random bits.

In summary, the randomness complexity of our t-private circuit is $O(t \cdot \log t|\tilde{C}| + \lambda)$.

Theorem 1. *Any function f with circuit size s and output size n_o admits a t-private implementation (I, C, O) with the canonical encoder I and the decoder O which, for each output bit, takes the parity of a block of bits (which are not restricted to $t+1$ bits), where C uses $O(t \cdot \log ts)$ random bits. Moreover, there exists a PPT algorithm which takes $(\tilde{C}_f, 1^t, 1^\lambda)$ as input, where $\tilde{C}_f$ is a circuit of size s that computes f, and outputs, except with $\leq 2^{-\lambda}$ probability, a t-private implementation (I, C, O) such that C uses $O(t \cdot \log ts + \lambda)$ random bits.*

Randomness Complexity of t-leakage-tolerant Private Circuits. As we discussed in Remark 2, we can obtain a t-leakage-tolerant private circuit from our outer construction with small modifications. As for the randomness complexity, we need to invoke one more time of the leakage-tolerant XOR gadget for each output bit and the input size of the XOR gadget is bounded by $O(t \cdot \log |\tilde{C}|)$. When the leakage-tolerant XOR gadgets are instantiated by a multi-phase leakage-tolerant XOR gadget, the number of phases $p = |\tilde{C}| + n_o = O(|\tilde{C}|)$, where n_o is the number of output bits of f, and $N \leq p \cdot \max\{n_1, n_2, \ldots, n_p\} = O(|\tilde{C}| \cdot t^2 \log^2 |\tilde{C}|)$. We have the following theorem.

Theorem 2. *Any function f with circuit size s admits a t-leakage-tolerant private implementation C, where C uses $O(t \cdot \log ts)$ random bits. Moreover, there exists a PPT algorithm which takes $(\tilde{C}_f, 1^t, 1^\lambda)$ as input, where $\tilde{C}_f$ is a circuit of size s that computes f, and outputs, except with $\leq 2^{-\lambda}$ probability, a t-leakage-tolerant implementation C such that C uses $O(t \cdot \log ts + \lambda)$ random bits.*

Circuit Size. Our construction can be viewed as a composition of two parts: (1) the computation of a multi-phase $(t, t, 1)$-robust parity sharing generator in the inner construction, and (2) the computation for input wires, addition gates, multiplication gates, and the leakage-tolerant XOR gadgets (and for output wires in the leakage-tolerant variant).

For the first part, our construction of the multi-phase robust parity sharing generator has size $O(m \cdot N)$, where m is the input size of the generator, and N is the output size (see Section 6). Let s denote the circuit size of $\tilde{C}$ that computes f. Thus, the first part has size $O((t \cdot \log ts + \lambda) \cdot t^2 s \cdot \log^2 s) = O(t^3 s \cdot \log^2 s \cdot \log ts + \lambda \cdot t^2 s \cdot \log^2 s)$.

For the second part, for each input wire and addition gate (and output wire in the leakage-tolerant variant), we use the leakage-tolerant XOR gadget with

input size $O(t \cdot \log s)$. For each multiplication gate, we first compute $O(t^2 \cdot \log^2 s)$ multiplications and then use the leakage-tolerant XOR gadget with input size $O(t^2 \cdot \log^2 s)$. Thus, the second part has size $O(t^2 s \cdot \log^2 s)$.

Thus, the overall circuit size is $O(t^3 s \cdot \log^2 s \cdot \log ts + \lambda \cdot t^2 s \cdot \log^2 s)$. Thus, assuming $\lambda \leq \tilde{O}(t)$, the circuit size is $\tilde{O}(t^3 s)$.

6 Construction of Multi-Phase Robust Parity Sharing Generator

In this section, we show that there exists a multi-phase $(t, t, 1)$-robust parity sharing generator with randomness complexity $O(t \cdot \log tN)$, where $N = \sum_{j=1}^{p} n_j$. In the following, addition and multiplication operations are in the binary field $\mathbb{Z}_2$.

Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_m)$ denote the input of G. For all $j \in \{1, 2, \ldots, p\}$, our idea is to use a matrix $\boldsymbol{M}^{(j)} \in \{0,1\}^{n_j \times m}$ to compute $\boldsymbol{r}^{(j)} = (r_1^{(j)}, r_2^{(j)}, \ldots, r_{n_j}^{(j)}) = \boldsymbol{M}^{(j)} \cdot \boldsymbol{u}$. Specifically, to compute $r_i^{(j)}$,

1. The circuit G first computes the coordinate-wise multiplication $\boldsymbol{M}_i^{(j)} * \boldsymbol{u}$, where $\boldsymbol{M}_i^{(j)}$ is the i-th row of $\boldsymbol{M}^{(j)}$. That is, the circuit G computes $M_{i,w}^{(j)} \cdot u_w$ for all $w \in \{1, 2, \ldots, m\}$.
2. Then, the circuit G computes $r_i^{(j)} = \sum_{w=1}^{m} M_{i,w}^{(j)} \cdot u_w$ by using a $\lceil \log m \rceil$-depth addition circuit.

The requirement that the parity of $\boldsymbol{r}^{(j)}$ is 0 is equivalent to $\sum_{i=1}^{n_j} \boldsymbol{M}_i^{(j)} = \boldsymbol{0}$.

Let $\mathcal{A}$ be the access structure defined in the construction of the multi-phase t-leakage-tolerant gadget in Section 5.2. We will show that when using random matrices for $\{\boldsymbol{M}^{(j)}\}_{j=1}^{p}$ (with $m = O(t \log tN + \log(1/\epsilon))$) which are subject to $\sum_{i=1}^{n_j} \boldsymbol{M}_i^{(j)} = \boldsymbol{0}$ for all $j \in \{1, 2, \ldots, p\}$, with probability $1 - \epsilon$, the above construction is a multi-phase $(t, t, 1)$-robust parity sharing generator with respect to the access structure $\mathcal{A}$.

To this end, for any set S of at most t wires in G, we first determine the set T of at most $|S|$ output bits. For each wire in S, if it is $r_i^{(j)}$ or an intermediate wire when computing $r_i^{(j)}$, we insert $r_i^{(j)}$ in T. Then, it is clear that the size of T is bounded by $|S|$. We will prove the following argument in the full version.

Lemma 3. *Let $\{\boldsymbol{M}^{(j)}\}_{j=1}^{p}$ be random matrices subject to $\sum_{i=1}^{n_j} \boldsymbol{M}_i^{(j)} = \boldsymbol{0}$ for all $j \in \{1, 2, \ldots, p\}$, and G be the circuit constructed above. Let $\mathcal{A}$ be the access structure defined in the construction of the multi-phase t-leakage-tolerant gadget in Section 5.2. For any set S of at most t wires in G, let T be the set of at most $|S|$ output bits defined above.*

Then, when $m = O(t \log tN + \log(1/\epsilon))$, where $N = \sum_{j=1}^{p} n_j$, with probability $1 - \epsilon$, for any set S of at most t wires in G, any set W of t variables in $\mathcal{A}$ and for any fixing of the values G_S of the wires in S and $\boldsymbol{r}_I$, where $I = \{(j, i) : r_i^{(j)} \in T\}$ and $\boldsymbol{r}_I$ is the vector that contains all bits in $\{r_i^{(j)} : (j, i) \in I\}$,

$$\mathtt{Dist}(\boldsymbol{r}|_{G_S, \boldsymbol{r}_I}, W) = \mathtt{Dist}(\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I}, W),$$

where $\texttt{Dist}(\boldsymbol{r}, W)$ *is the distribution of the variables in* W *when they are instantiated by* $\boldsymbol{r}$*,* $\boldsymbol{r}|_{G_S,\boldsymbol{r}_I}$ *is the random variable* $\boldsymbol{r}$ *conditioned on* G_S *and* $\boldsymbol{r}_I$*, and* $\tilde{\boldsymbol{r}}|_{\tilde{\boldsymbol{r}}_I=\boldsymbol{r}_I}$ *is the random variable* $\tilde{\boldsymbol{r}}$ *conditioned on* $\tilde{\boldsymbol{r}}_I = \boldsymbol{r}_I$*.*

Acknowledgements. Y. Ishai supported by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20. V. Goyal and Y. Song were supported by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award. Y. Song was also supported by a Cylab Presidential Fellowship.

References

1. G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, P.-Y. Strub, and R. Zucchini, *Strong Non-Interference and Type-Directed Higher-Order Masking*, in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, New York, NY, USA, 2016, Association for Computing Machinery, p. 116129.
2. S. Belaïd, F. Benhamouda, A. Passelègue, E. Prouff, A. Thillard, and D. Vergnaud, *Randomness Complexity of Private Circuits for Multiplication*, in Advances in Cryptology – EUROCRYPT 2016, M. Fischlin and J.-S. Coron, eds., Berlin, Heidelberg, 2016, Springer Berlin Heidelberg, pp. 616–648.
3. G. Cassiers and F.-X. Standaert, *Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference*, IEEE Transactions on Information Forensics and Security, 15 (2020), pp. 2542–2555.
4. B. Chor, O. Goldreich, J. Hasted, J. Freidmann, S. Rudich, and R. Smolensky, *The bit extraction problem or t-resilient functions*, in 26th Annual Symposium on Foundations of Computer Science (sfcs 1985), 1985, pp. 396–407.
5. J.-S. Coron, A. Greuet, and R. Zeitoun, *Side-Channel Masking with Pseudo-Random Generator*, in Advances in Cryptology – EUROCRYPT 2020, A. Canteaut and Y. Ishai, eds., Cham, 2020, Springer International Publishing, pp. 342–375.
6. A. Duc, S. Dziembowski, and S. Faust, *Unifying Leakage Models: From Probing Attacks to Noisy Leakage.*, in Advances in Cryptology – EUROCRYPT 2014, P. Q. Nguyen and E. Oswald, eds., Berlin, Heidelberg, 2014, Springer Berlin Heidelberg, pp. 423–440.
7. S. Faust, C. Paglialonga, and T. Schneider, *Amortizing Randomness Complexity in Private Circuits*, in ASIACRYPT 2017, Part I, 2017, pp. 781–810.
8. Y. Ishai, E. Kushilevitz, X. Li, R. Ostrovsky, M. Prabhakaran, A. Sahai, and D. Zuckerman, *Robust Pseudorandom Generators*, in Automata, Languages, and Programming, F. V. Fomin, R. Freivalds, M. Kwiatkowska, and D. Peleg, eds., Berlin, Heidelberg, 2013, Springer Berlin Heidelberg, pp. 576–588.
9. Y. Ishai, A. Sahai, and D. Wagner, *Private Circuits: Securing Hardware against Probing Attacks*, in Advances in Cryptology - CRYPTO 2003, D. Boneh, ed., Berlin, Heidelberg, 2003, Springer Berlin Heidelberg, pp. 463–481.
10. E. Kushilevitz and Y. Mansour, *Randomness in Private Computations*, SIAM Journal on Discrete Mathematics, 10 (1997), pp. 647–661. Earlier version in PODC 1996.