Efficient Distributed Randomness Generation from Minimal Assumptions where PArties Speak Sequentially Once

Chen-Da Liu-Zhang* Elisaweta Masserova† João Ribeiro‡ Pratik Soni§
Sri AravindaKrishnan Thyagarajan¶

February 25, 2025

Abstract

We study efficient public randomness generation protocols in the PASSO (PArties Speak Sequentially Once) model for multi-party computation (MPC). PASSO is a variation of traditional MPC where n parties are executed in sequence and each party "speaks" only once, broadcasting and sending secret messages only to parties further down the line. Prior results in this setting include information-theoretic protocols in which the computational complexity scales exponentially with the number of corruptions t (CRYPTO 2022), as well as more efficient computationally-secure protocols either assuming a trusted setup phase or DDH (FC 2024). Moreover, these works only consider security against static adversaries.

In this work, we focus on computational security against *adaptive adversaries* and *from minimal assumptions*, and improve on the works mentioned above in several ways:

- Assuming the existence of non-interactive perfectly binding commitments, we design protocols with $n = 3t + 1$ or $n = 4t$ parties that are efficient and secure *whenever t is small compared to the security parameter λ* (e.g., t is constant). This improves the resiliency of all previous protocols, even those requiring a trusted setup. It also shows that $n = 4$ parties are necessary and sufficient for $t = 1$ corruptions in the computational setting, while $n = 5$ parties are required for information-theoretic security.
- Under the same assumption, we design protocols with $n = 4t + 2$ or $n = 5t + 2$ parties (depending on the adversarial network model) which are efficient whenever $t = \mathsf{poly}(\lambda)$. This improves on the existing DDH-based protocol both in terms of resiliency and the underlying assumptions.
- We design efficient protocols with $n = 5t + 3$ or $n = 6t + 3$ parties (depending on the adversarial network model) assuming the existence of one-way functions.

We complement these results by studying lower bounds for randomness generation protocols in the computational setting.

*Lucerne University of Applied Sciences and Arts and Web3 Foundation. chen-da.liuzhang@hslu.ch.

†Carnegie Mellon University. elisawem@andrew.cmu.edu.

‡Instituto de Telecomunicações and Instituto Superior Técnico, Universidade de Lisboa. jribeiro@tecnico.ulisboa.pt. Work mainly done while at NOVA LINCS and NOVA School of Science and Technology.

§University of Utah. psoni@cs.utah.edu.

¶University of Sydney. t.srikrishnan@gmail.com.

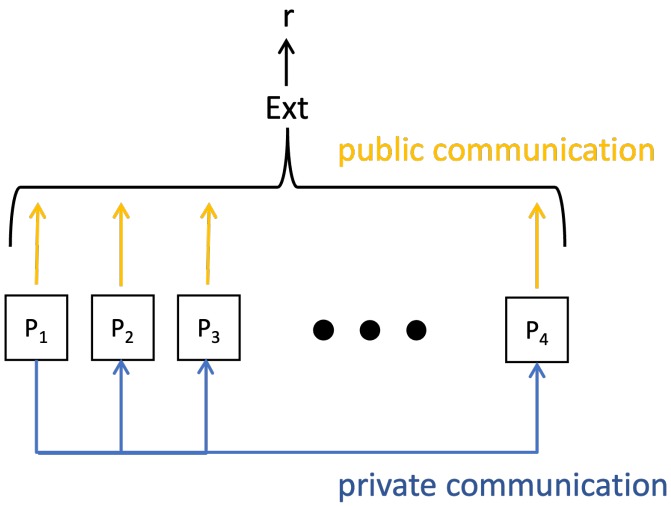

Figure 1: Communication model from [36, Figure 1]. Parties P_i speak one after the other, send secrets to future parties P_j for $j > i$, and publish public values, which are available to all parties.

1 Introduction

A reliable source of unpredictable public randomness which cannot be manipulated by any party is the bedrock of numerous cryptographic tasks and applications. Centralized *randomness beacons* – an ideal service which emits unpredictable and unbiased random bits at regular intervals – have been extensively studied starting from Rabin's work in 1983 [39] to the NIST beacon [30]. With the advent of modern decentralized applications like blockchains, the centralized approach is undesirable due to its single-point-of-failure problem. An alternative approach is to design a *distributed* algorithm where mutually distrusting parties together provide a randomness beacon service where the random value generated is both unpredictable and unbiased.

Recently, Nielsen, Ribeiro, and Obremski [36] proposed a simple model for studying such protocols, inspired by the YOSO (You Only Speak Once) model studied earlier by Gentry, Halevi, Krawczyk, Magri, Nielsen, Rabin, and Yakoubov [22]. In [36], the authors consider n parties, each with its own local source of randomness, to execute sequentially. One after the other, each party "speaks", i.e., broadcasts and/or sends private messages to future parties (see Figure 1). We refer to this model as PASSO (PArties Speak Sequentially Once). [1]

Both the PASSO and YOSO models involve parties communicating once. YOSO encompasses additional aspects of the "theoretical model to practical system" pipeline with specific blockchain applications in mind. In more detail, YOSO differentiates between physical machines (which can retain state long-term) and ephemeral *roles*, which are deployed on demand to perform a certain task, and a role-assignment functionality selects which physical machine is performing which role. Assuming that role-assignment is secure, i.e., the adversary cannot predict which machine will execute which role, YOSO schemes can withstand adaptive adversaries with corruption budgets

[1] Nielsen, Ribeiro, and Obremski did not name their model, and Liu-Zhang, Masserova, Ribeiro, Soni, and Thyagarajan [34] refer to it as YOSO with worst-case corruptions. We believe that PArties Speak Sequentially Once better highlights the model's key features.

that exceed the size of the committee. YOSO protocols typically crucially rely on a *uniformly random* role-assignment, and allow parties to send private messages to future roles for which the party-role correspondence may not even be known yet.

We believe that the PASSO MPC model, isolated from potential existing applications, is theoretically interesting on its own and beyond applications to YOSO. The usage of "PASSO" is meant to emphasize this separation. For example, protocol design in PASSO, which features worst-case corruptions, brings new theoretical challenges, as we need to depart from standard (committee-based) MPC techniques. With YOSO applications in mind, security against worst-case corruptions also means that PASSO protocols are more resilient to failures in the role-assignment mechanism. Furthermore, in part due to its simplicity, we believe that PASSO will prove relevant beyond YOSO, in particular in settings where the parties participating in the protocol are known beforehand, in which case components such as the role-assignment and private messages to the future (which were challenging to implement in the YOSO setting) are simply not needed or easy to realize.

The original work [36] on PASSO studied the *feasibility* of randomness generation protocols with information-theoretic security (i.e., where the attacker is computationally unbounded). The corresponding protocols are secure against t worst-case corruptions with $n = 5t$ or $n = 6t + 1$ parties, depending on the underlying adversarial network model. Unfortunately, these protocols have a significant drawback: their computational and communication complexities scale exponentially with the corruption threshold t. Motivated by this, a recent work by Liu-Zhang, Masserova, Ribeiro, Soni, and Thyagarajan [34] focused on the design of *efficient* PASSO randomness generation protocols secure against *computationally-bounded adversaries*, in which computational and communication complexities scale polynomially with the corruption threshold t. More specifically, they present two protocols that offer different tradeoffs between the tolerated threshold corruption and the required cryptographic and setup assumptions. The first protocol is proven secure for $n = 3t + 2$ parties assuming a trusted setup phase given the use of a publicly-verifiable secret sharing (PVSS) scheme (irrespective of the adversarial network model under consideration). The second protocol requires no setup and is proven secure under the DDH assumption for $n = 4t+4$ or $n = 5t + 4$ parties, depending on the strength of the adversarial network model. In summary, the only known efficient randomness generation protocols in the model above either require a trusted setup or rely on a concrete algebraic assumption, limiting possible instantiations. For example, the $n = 4t + 4$ protocol of [34] is not post-quantum secure. Given the significance of randomness for secure computing, it is imperative to understand the weakest cryptographic assumptions necessary to generate useful randomness in the distributed setting. In particular, we ask

> *What are the minimal assumptions under which poly-time computationally secure randomness generation is feasible in the PASSO model?*

1.1 Our Results

In this work, we make significant progress towards addressing the above question. More concretely, we improve upon the plain model result of [34] in terms of the supported adversarial threshold, provide protocols which offer further interesting trade-offs in terms of assumptions and/or resiliency, and supplement our positive results by providing lower bounds on the required number of parties.
Setting. We consider two adversarial models – *sending-leaks* and *execution-leaks* – as done in [36, 34]. Intuitively, in the execution-leaks model the adversary only obtains messages addressed to corrupted parties upon their execution. In the stronger sending-leaks model, the adversary obtains

Table 1: PASSO randomness generation in the plain model, sending-leaks.

Assumptions	# of Parties	Poly-time	Source
Unconditional	$6t+1$	For $t = O(\log \lambda)$	[36]
DDH	$5t+4$	Any t	[34]
One-way functions	$6t+3$	Any t	**Theorem 3**
Non-interactive commitments	$5t+2$	Any t	**Theorem 2**
Non-interactive commitments	$4t$	For $t = O(\log \lambda)$	**Theorem 1**
Impossible	$< 3t+1$	–	**Theorem 9**

Table 2: PASSO randomness generation in the plain model, execution-leaks.

Assumptions	# of Parties	Poly-time	Source
Unconditional	$5t$	For $t = O(\log \lambda)$	[36]
DDH	$4t+4$	Any t	[34]
One-way functions	$5t+3$	Any t	**Theorem 3**
Non-interactive commitments	$4t+2$	Any t	**Theorem 2**
Non-interactive commitments	$3t+1$	For $t = O(\log \lambda)$	**Theorem 1**
Impossible	$t = 1, n \leq 3$	–	**Theorem 8**
Impossible	$t = 2, n \leq 6$	–	**Theorem 10**

the messages addressed to corrupted parties immediately upon the sender sending the message.

In this context, we say that a PASSO protocol is (t, n)-*computationally secure* in the execution-leaks model (resp. sending-leaks model) if it outputs a bit that is negligibly close to uniform in statistical distance even when a PPT adversary is allowed to adaptively corrupt and control any t parties. We say that a protocol is *efficient* if its computational and communication complexities scale polynomially with the number of parties n and the security parameter λ. Our lower bounds hold even against non-adaptive adversaries. See Tables 1 and 2 for a summary of our results and comparison to prior work in the plain model.

Feasibility. As our first contribution, in Section 4 we design protocols with $n = 3t+1$ parties in the execution-leaks model (and $n = 4t$ in the sending-leaks model) which incur exponential computational and communication complexities in the corruption threshold t, assuming the existence of non-interactive commitments.[2] Therefore, the resulting protocol is only efficient and secure when $t = O(\log \lambda)$, where λ is the security parameter. This is a relevant setting, as it also makes sense to consider settings where the adversary's running time grows much faster than the number of parties. In particular, the protocol is efficient when the number of parties is constant.

Theorem 1. *Let $t = O(\log \lambda)$, where λ is the security parameter. Then, assuming the existence of non-interactive perfectly binding commitments, there are (t, n)-computationally secure PASSO randomness generation protocols with $n = 3t+1$ and $n = 4t$ parties in the execution-leaks and sending-leaks models, respectively, and polynomial (in λ) computational and communication.*

[2]Non-interactive commitments can be instantiated from a variety of concrete assumptions including factoring [8, 42, 24], more recently from LWE and LPN [25], and even from the general assumption of injective one-way functions. While black-box separations between general one-way functions and non-interactive commitments are known [35], non-interactive commitments are fundamental and one of the weakest complexity-theoretic cryptographic assumptions.

We highlight the following takeaways from this result. Coupled with our lower bounds below, this result tells us that $n = 4$ parties are necessary and sufficient for computationally secure PASSO randomness generation against $t = 1$ corruption. It also tells us that $n = 7$ parties are necessary and sufficient against $t = 2$ corruptions in the execution-leaks model, while $n = 7$ parties are necessary and $n = 8$ parties are sufficient in the sending-leaks model. In contrast, information-theoretic PASSO randomness generation secure against $t = 1$ corruptions requires $n = 5$ parties [36].

Next, we trade off the number of parties for protocol efficiency under the same assumption. Namely, in Section 5 we obtain protocols based on non-interactive commitments which remain efficient and secure when $t = \mathsf{poly}(\lambda)$ and require $n = 4t + 2$ parties in the execution-leaks model, or $n = 5t + 2$ parties in the sending-leaks model. This protocol improves upon the DDH-based protocol of [34] in terms of the required number of parties and underlying assumptions.

Theorem 2. *Assuming the existence of non-interactive perfectly binding commitments, there are efficient (t, n)-computationally secure PASSO randomness generation protocols with $n = 4t+2$ and $n = 5t + 2$ parties in the execution-leaks and sending-leaks models, respectively.*

Finally, in a quest for designing protocols under as minimal assumptions as possible, we obtain an efficient protocol for $n = 5t + 3$ parties in the execution-leaks model based only on one-way functions in Section 6. The protocol can be adapted to be secure in the stronger sending-leaks model while requiring $n = 6t + 3$ parties.

Theorem 3. *Assuming the existence of one-way functions, there are efficient (t, n)-computationally secure PASSO randomness generation protocols in the execution-leaks and sending-leaks models with $n = 5t + 3$ and $n = 6t + 3$ parties, respectively.*

Lower bounds. We complement the positive results above by studying lower bounds for computationally secure PASSO protocols without setup in Section 7. First, we prove impossibility for computationally secure PASSO randomness generation with $n = 3$ parties and $t = 1$ corruptions. As we show, this result is tight and extends to $t > 1$ corruptions and $n = 3t$ parties in the sending-leaks model. Next, we investigate whether it is also possible to extend the $(t = 1, n = 3)$ impossibility in the computational setting to $t > 1$ and $n = 3t$ *in the execution-leaks model*. Here, we take the first step with a novel approach and show the impossibility of $(t = 2, n = 6)$ PASSO randomness generation. We leave extending this result to $t > 2$ as an interesting open problem. Our results complement those of [36] that proved impossibility of *information-theoretic* PASSO randomness generation with $n = 4$ parties and $t = 1$ corruptions, which is tight and extends to $t > 1$ corruptions and $n = 4t$ parties in the sending-leaks model.[3]

1.2 Other Related Work

Other works have considered models with parties communicating sequentially. In particular, several prior works have considered the setting of randomly-selected committees speaking in sequence (e.g., YOSO MPC [22, 10, 31], Fluid MPC [15, 6, 18], player-replaceability [13, 7, 14], Layered MPC [17, 19], SCALES [1, 2]). PASSO protocols are not based on committees. In particular, these committee-based protocols typically rely on having an honest majority in each committee and do

[3]The work [36] claims that the information-theoretic $(t = 1, n = 4)$ impossibility result also extends directly to $t > 1$ and $n = 4t$ in the weaker execution-leaks model, but it is not clear whether this holds (this is acknowledged in the updated ePrint version of [36]). We discuss this in more detail in Section 7.

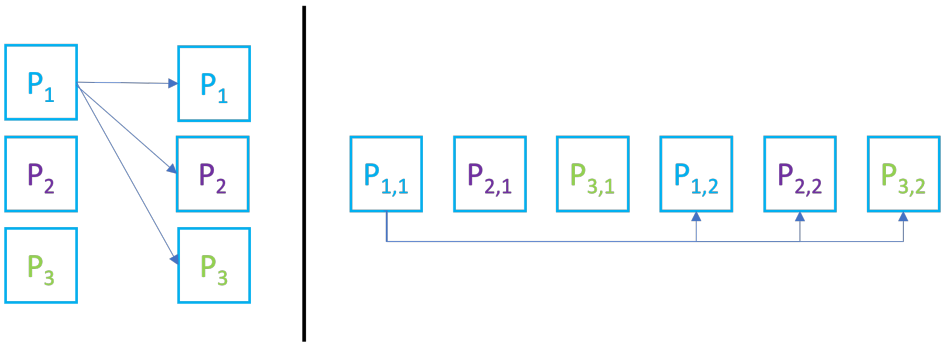

Figure 2: Traditional protocol vs. its naively linearized version.

not directly work in PASSO, since the adversary can concentrate corruptions in a single committee so that there is a dishonest majority.

Goyal et al. [27] designed MPC with GOD where parties speak once under a strong assumption of conditional storage and retrieval systems (protocols of Benhamouda et al. [5] and Goyal et al. [26] can serve as instantiations of such systems). ALBATROSS [12] (and the earlier protocol SCRAPE [11]) are based on PVSS. Randomness generation protocols based on PVSS can be easily translated to the PASSO model. Indeed, [34] introduced a PVSS-based protocol in PASSO already. Our focus here is on protocols with minimal assumptions and no setup or random oracle.

2 Technical Overview

We now provide an overview of our feasibility results. We begin by briefly outlining some natural strategies for building randomness generation protocols in the PASSO model and the problems we face when following them.

2.1 First Attempts at PASSO Protocols

As a first attempt, one can naively build a PASSO protocol from any stateful r-round multiparty computation protocol that tolerates t-out-of-n corruptions.

Linearizing Round-Based Protocols. We can implement the behavior of each party P_i in the r rounds using $r+1$ different parties $P_{i,k}$ for $k \in [r+1]$. Party $P_{i,k}$ plays the role of P_i in round k, and sends its private state to party $P_{i,k+1}$ (see Figure 2). Then, each $P_{i,r+1}$ receives the messages from the last round r and publishes their output. This approach, however, leads to very low resiliency. Naively, it requires $n \cdot (r+1)$ parties, while still only t corruptions can be tolerated. For example, consider the standard approach of letting everyone verifiably secret-share a random value and reconstruct the sum. If we consider the most round-efficient VSS protocol for $n = 3t+1$, which takes 2 rounds for sharing and 1 to reconstruct [37][4], we end up with $12t+4$ parties. Similarly, the most round-efficient poly-time VSS for $n = 2t+1$ takes 4 rounds to share and 1 round to reconstruct [32] in our setting, and we would end up with $12t+6$ parties.[5]

[4]The protocol in the standard setting requires 1 round for reconstruction when the adversary is non-rushing. However, since in our model one party speaks in each round, there are no rushing attacks.

[5]The works [32, 3] also show an inefficient VSS for $n = 2t+1$ that takes 3 rounds to share, leading to $10t+5$ parties.

A Strawman Protocol. As a second attempt, say we are in the execution-leaks model, and we want to generate a single unbiased bit. Given a corruption threshold t, consider the following simple commit-reveal protocol for $n = 3t + 2$ parties, split into a group of $t + 1$ *dealers* followed by a group of $2t + 1$ *receivers*. Each dealer P_i generates a bit x_i uniformly at random and publishes a commitment to x_i. Dealer P_i also sends the opening information to each receiver and never speaks again. Once every dealer has spoken, each receiver P'_i publishes every valid opening they received. The final coin is set to $\bigoplus_{i \in I} x_i$, where I is a set of secrets for which at least $t + 1$ correct openings exist. One could hope that the scheme is secure, as at least one of the x_i's in the computation of the coin output is uniformly random and independent of x_j for $j \neq i$. However, the adversary can bias the final coin value through a "conditional abort" attack. To see this, let the adversary corrupt the parties P_{t+1} and P'_{2t+1}, and let P_{t+1} commit to $x_{t+1} = 1$. Party P_{t+1} sends valid openings to t honest parties. After the first receiver has spoken, the adversary knows all honest values. As t honest parties will publish P_{t+1}'s opening, P'_{2t+1} can choose whether the outcome should be 0 or 1 by choosing whether to publish the opening for P_{t+1} or not. This "conditional abort" attack is inherent to these types of schemes, and simply changing the threshold t does not help.

2.2 A PASSO Protocol for $n = 3t + 1$ Parties

We now discuss our first construction, an execution-leaks protocol for $n = 3t + 1$ parties, which yields Theorem 1. To circumvent the "conditional abort" issue above, we ensure that the coin output is fixed *prior* to the reveal phase. At a very high level, we do so by ensuring that the coin output is shared among *sets* of parties, where any party in each set can reconstruct the set's share, and simultaneously at least one party in each set is honest.

As a first step, we let a single party (dealer) distribute a random bit. For now, we only want to ensure that assuming all parties in a set $\mathcal{S}$ are honest, the distributed bit is hidden before reconstruction starts. Additionally, if there is at least one honest party among $\mathcal{S}$, the sharing is either discarded before reconstruction starts, or the value shared by the dealer is fixed and guaranteed to be reconstructed. We use a non-interactive commitment scheme COM that is perfectly binding and computationally hiding to achieve this. The sharing proceeds as follows:

1. The dealer D samples a random bit $r \leftarrow\!\!\$\ \{0, 1\}$ and broadcasts a commitment com to r. D also sends the opening of com to everyone in $\mathcal{S}$ and to all receivers.

2. Each party $P \in \mathcal{S}$ forwards the opening it received to all receivers. If P did not receive a valid opening, it broadcasts $(\texttt{Complain}, D)$.

If any party complained, the sharing is discarded. During the reconstruction, receivers can broadcast any valid opening. Clearly, as long as everyone in $\mathcal{S}$ is honest, bit r is hidden before the reconstruction, and as long as at least one party in $\mathcal{S}$ is honest, it will either complain, or forward a valid opening to the receivers. If at least one receiver is honest, it will then broadcast this valid opening.

We now use this sharing scheme to get secure randomness. Intuitively, we need to generate sufficiently many bits so that at least one of them was generated by an honest dealer, is hidden before reconstruction, and malicious parties cannot "deny" it. At the same time we need to ensure that sets $\mathcal{S}$ contain at least one honest party (to prevent a conditional abort attack if a dealer is malicious).

Let ℓ be the number of dealers (to be set later). Each sharing is represented by a nonzero vector $v \in \{0,1\}^\ell$, and our set $\mathcal{S}$ above will correspond to $\mathsf{Supp}(v)$, where $\mathsf{Supp}(v)$ denotes the support of v. The dealer corresponds to $\min \mathsf{Supp}(v)$, i.e., the position of the leftmost 1 in v. Overall, the m total sharings are represented by rows of a binary "t-sharing matrix" $M \in \{0,1\}^{m \times \ell}$. Given this matrix, the full protocol works as follows:

1. The ℓ dealers $P_1, \ldots, P_\ell$ perform sharings according to the m rows of M.
2. Party $P_{\ell+1}$ outputs a random bit $r^\star \leftarrow\!\!\$\ \{0,1\}$.
3. The $t+1$ receivers $P_{\ell+2}, \ldots, P_{\ell+t+1}$ broadcast everything they receive from the dealers $P_1, \ldots, P_\ell$.

To reconstruct the coin from the public broadcasts, we use the information published by the receivers to open the commitments published by the dealers, and then XOR the random bits r_v for all rows v of M with the special random bit $r^\star$. If a complaint was broadcast during the sharing of some bit r_v, then we ignore that sharing (i.e., replace it by 0 in the XOR).

We identify two properties of the t-sharing matrix M that are sufficient to yield a secure protocol against t corruptions:

1. Every row of M has Hamming weight t (i.e., $|\mathsf{Supp}(v)| = t$ for all rows v);
2. For any $t-1$ columns $M_{\cdot j_1}, M_{\cdot j_2}, \ldots, M_{\cdot j_{t-1}}$ of M, there exists an index i such that $M_{ij_1} = M_{ij_2} = \cdots = M_{ij_{t-1}} = 0$. In other words, the coordinate-wise union of any $t-1$ columns is not the all-1s vector.

The second property is reminiscent of the notion of *t-disjunct matrices*, which are useful in the design of non-adaptive group testing schemes (see [28, Chapter 22] for a discussion of this topic).

Intuition behind the security proof. We give some intuition on why the properties above are sufficient to establish security. First, note that if the adversary corrupts t dealers, then they cannot bias the coin because they do not know the random bit published by $P_{\ell+1}$ (who is honest), and, furthermore, all the receivers are also honest. Therefore, we may assume that the adversary corrupts at most $t-1$ dealers, and possibly some other non-dealer parties.

By Property 1 above, the assumption that only at most $t-1$ dealers are dishonest means that every sharing contains at least one honest party. This forces the dishonest parties in that particular sharing to commit to *some* (possibly non-random) value during the sharing procedure, as otherwise the honest party would identify an inconsistency and complain. If $P_{\ell+1}$ is honest, then the output of the protocol will be unbiased, because this party is only executed after the sharing phase has concluded and so its published bit $r^\star$ is uniformly random and independent of the bits generated in the sharing phase. On the other hand, if $P_{\ell+1}$ is dishonest then we invoke Property 2, which ensures that there is a sharing in which all participating parties are honest. Intuitively, the random bit produced in this fully honest sharing is hidden (by the security of the commitment scheme) until the receivers are executed, at which point other shared values have already been committed to and party $P_{\ell+1}$ has already been executed.

Constructing the t-sharing matrix. It remains to give a construction of a t-sharing matrix $M \in \{0,1\}^{m \times \ell}$ satisfying the properties laid out above. A simple option that works, and which we

use, is to consider $\ell = 2t-1$ columns and $m = \binom{2t-1}{t}$ rows, one per weight-t vector of length $2t-1$. Instantiating the framework above with this matrix yields a protocol with

$$n = \ell + 1 + (t+1) = (2t-1) + 1 + (t+1) = 3t+1$$

parties. To complement this, we use a simple argument to show that any t-sharing matrix $M \in \{0,1\}^{m \times \ell}$ must satisfy $m \geq \binom{\ell}{t-1}/\binom{\ell-t}{t-1}$. In particular, when $\ell = 2t-1$, which is the minimum number of columns in a t-sharing matrix, then it is not possible to do better than our simple construction. We can hope to obtain some complexity *versus* number of parties trade-off by increasing the number of columns and decreasing the number of rows, but this lower bound also shows that the number of sharings remains exponential in t unless we significantly increase the number of dealers.

Finally, we can extend our execution-leaks protocol to work in the sending-leaks model by adding $t-1$ additional parties (see Section 4).

2.3 Lower Bounds Without Setup

As we will now see, the result from the previous section is tight in the sense that $n = 4$ parties are both necessary and sufficient for $t = 1$ (both in the sending-leaks and execution-leaks models). This argument can be easily extended to show that $n = 3t+1$ parties are required against any $t \geq 1$ corruptions in the sending-leaks model. For the execution-leaks model the situation is more subtle, and such an extension is not clear. Using a novel approach, we show that $n = 7$ parties are required against $t = 2$ corruptions in the execution-leaks model, which is also tight by the result from the previous section.

We find it enlightening to begin by discussing the impossibility result for $t = 1$ corruption and $n = 3$ parties, where there is no distinction between the execution-leaks and sending-leaks models. Consider a protocol with parties P_1, P_2, and P_3. First, we consider corrupting P_3. There are two cases – either two independent honest executions of P_3 lead to two different final coins with non-negligible probability, or the final coin is determined (except with negligible probability) by the public values of P_1 and P_2. If the first case holds, then the protocol is not secure. If the second case holds, then we consider corrupting P_2. Now, either two independent honest executions of P_2 lead to different coin values (and P_2 can predict the final coin with high probability in each execution because we have assumed that P_3's behavior does not matter, regardless of the content of private messages sent to it), in which case the protocol is not secure, or P_1's public value determines the final coin with high probability, in which case the protocol is also clearly not secure. Briefly, the reason why this argument does not extend to $t = 1$ corruption and $n = 4$ parties $P_1, \ldots, P_4$, unlike in [36], is that in order for a corrupt P_2 to predict the final coin in a given honest execution (which it must do to be able to choose the execution that biases the coin to its preferred value) it would have to efficiently simulate the private message from P_1 to P_3 *conditioned on P_1's already broadcast public value*. It is not clear how this can be done.

The argument above extends to give an impossibility for any $t \geq 1$ corruptions and $n = 3t$ parties *in the sending-leaks model*. Concretely, for $t = 2$ corruptions and a protocol Π with $n = 3 \cdot 2 = 6$ parties $P_1, \ldots, P_6$, we simply apply the argument above to the 3-party protocol Π' with parties P'_1, P'_2, P'_3 where each P'_i emulates a block of 2 parties in Π (i.e., P'_1 emulates (P_1, P_2), P'_2 emulates (P_3, P_4), and P'_3 emulates (P_5, P_6)). Remarkably, this emulation argument does not work in the execution-leaks setting.

Intuitively, to extend this argument to the execution-leaks setting it would be sufficient to show that every efficient execution-leaks adversary corrupting 1 party that breaks Π' can be transformed into an efficient execution-leaks adversary corrupting t parties that breaks protocol Π (recall that for 1 corruption the execution-leaks and sending-leaks settings are identical). The key issue precluding this is that in the resulting 3-party protocol Π' all private messages from P'_1 (who emulates P_1, P_2 of the original protocol Π) to P'_2 (who emulates parties P_3, P_4 of Π) are revealed to P'_2 *as soon as* P'_2 *starts its execution*. Therefore, an adversary for Π' that corrupts P'_2 can adversarially choose the behavior of P_3 based on messages that would only be sent to later parties in Π. On the other hand, an execution-leaks adversary for Π that corrupts the corresponding parties P_3, P_4 does not know the messages sent to P_4 when executing P_3, and so it is not clear how the execution-leaks adversary for Π can efficiently emulate the adversary for Π'. This subtlety has also been acknowledged in the updated ePrint version of [36]. While there may be a way to get around this issue, we do not know how to generally overcome it in the computational setting.

As our main contribution on lower bounds, we give a novel argument that overcomes the barriers above and yields an impossibility result for $t = 2$ corruptions and $n = 6$ parties *in the execution-leaks model*. Our argument proceeds by analyzing corruption patterns that go beyond consecutive blocks of 2 parties. First, similarly to the previous argument, we consider corrupting P_5 and P_6. This lets us conclude that the public values of $P_1, \ldots, P_4$ in an honest execution of the protocol determine the final coin with high probability (otherwise the protocol would not be secure). If we followed the previous approach, we would proceed by considering the corruption of P_3 and P_4. But we run into trouble, since when executing P_3 we do not know the private messages sent by P_1 and P_2 to P_4, which may affect P_4's behavior and hence the final coin. Therefore, crucially departing from the previous approach, we consider corrupting P_1 and P_4. This lets us show that the private message from P_1 to P_4 can be fixed to a special symbol $\perp$ without loss of generality, provided that P_1, P_2, P_3 behave honestly, which unlocks the remainder of the argument. Indeed, consider corrupting P_2 and P_3. Using that P_1's private message to P_4 is fixed to $\perp$ and that we control P_2, we show that either P_3 can locally predict and bias the final coin, in which case the protocol is not secure, or it is already determined by P_1 and P_2, in which case the protocol is also not secure since we can simultaneously corrupt P_1 and P_2.

2.4 An Efficient PASSO Protocol from Non-Interactive Commitments

Continuing on our quest towards obtaining efficient PASSO protocols, we note that our previous scheme worked only for $t = O(\log \lambda)$. We now show how to remove this assumption at the cost of slightly larger number of parties, which will result in Theorem 2. Our scheme relies on *verifiable secret sharing* (VSS). A (t, n)-VSS allows threshold secret sharing of a secret to n parties such that (1) a secret shared by an honest dealer is always reconstructed correctly by any set of $t + 1$ parties, (2) prior to reconstruction phase, no information is leaked about the secret to any set of t parties, and (3) receivers can verify that the dealer behaved correctly, i.e., there exists a unique secret corresponding to the sharing phase, and it can be correctly reconstructed. Our scheme can be seen as a variant of the protocol from [34], which is a custom version of a sequence of $t + 1$ instantiations of Pedersen's VSS protocol [38], where each dealer shares a random value. More precisely, each instantiation distinguishes the following parties:

1. Party D, who acts as the dealer distributing the secrets (publishing commitments to the coefficients of the degree-t polynomial and bilaterally sending to each receiver a share evaluation),

and sends its state to its counterpart D'.

2. $2t+1$ receivers R_i, who receive and verify the secret shares, complain about the shares if applicable, and otherwise send these to each party R'_i.

3. Party D' who obtains a state from D and uses it to publish the shares of the receivers that complained. If D' cannot resolve a complaint, this instance is aborted.

4. $t+1$ receivers R'_i who receive all the shares from each party R_i, as well as set their shares to the ones broadcast by D' (if the counterpart R_i complained), and publicly reveal all these shares.

The idea is that after the first three steps, the dealer has committed to a random value, which will be reconstructed in Step 4. Note that before Step 4, if both D and D' are honest, no information about the committed random value is revealed. Therefore, to generate a random coin, one can use a standard linearization of $t+1$ instances of the above protocol (where the first three steps of each instance are executed, and subsequently all committed random values are reconstructed). The final coin is then the sum of the $t+1$ random values. This works because since there are $t+1$ dealers, at least one of them is honest; moreover, before the reconstruction phase starts (Step 4 of each instance), the adversary submits secrets without knowing the honest secrets, and every instance that succeeded, is guaranteed to be reconstructed.

While [34] uses ElGamal commitments, our goal is to generalize to allow for any non-interactive commitments (even non-homomorphic ones). The above construction requires the commitment to be homomorphic, since the receivers need to compute commitments to the point evaluations from the commitments to the polynomial coefficients. We observe that one can instead let the dealer D publish $2t+1$ commitments to the points themselves, rather than the $t+1$ coefficients, and send its state to D'. However, the difference is that now a cheating dealer could commit to a polynomial that is not of degree t.

To solve this, one can let the dealer commit to all the points of a bivariate degree-t polynomial $F(x,y)$, and send to each receiver R_i the openings corresponding to the i-th projection (horizontal and vertical), i.e. openings to the commitments of the points $\{F(i,j)\}_{j\in[2t+1]}$ and $\{F(j,i)\}_{j\in[2t+1]}$. Each receiver can now directly check the openings against the published commitments, and also check that the two projections are of degree t. The party D' will publish the openings to the points of any R_i that complained.

After resolving the complaints, observe that the projections corresponding to any two honest receivers R_i and R_j are consistent among each other and have degree t, and therefore the committed bivariate polynomial F has degree t. Moreover, the state of each R_i is sent to all receivers R'_i, and therefore any honest receiver R'_i can provide enough information to reveal the whole polynomial.

At a high level, the protocol described above uses a total of $n = 5t+4$ parties: a group $\mathcal{D}$ of size $t+1$, group $\mathcal{R}$ of size $2t+1$, group $\mathcal{D}'$ of size $t+1$, and group $\mathcal{R}'$ of size $t+1$. The parties execute the following:

- Each $D_i \in \mathcal{D}$ executes the protocol of the dealer D in the i-th linearization.
- Each $R_i \in \mathcal{R}$ executes the protocol of the i-th receiver R_i in *each* of the $t+1$ linearizations.
- Each $D'_i \in \mathcal{D}'$ executes the protocol of the dealer D' in the i-th linearization.
- Each $R'_i \in \mathcal{R}'$ executes the protocol of the i-th receiver R'_i in *each* of the $t+1$ linearizations.

However, one can slightly improve the number of parties to $4t+4$ with the following modification. Instead of letting $t+1$ dealers, each of whom shares secrets among the *same set* $\mathcal{R}$ of $2t+1$ parties, we let each dealer share secrets among the *next* $2t+1$ parties. More details can be found in Section 5.

2.5 An Efficient PASSO Protocol based on One-Way Functions

Finally, we discuss the approach behind our Theorem 3. We again turn to VSS to ensure that the secret is fixed at the end of the commit (*sharing*) phase, and the adversary corrupting at most t parties learns nothing about the secret. As before, if we ensure that the coin reconstruction starts only after the sharing phase of *all* secrets is complete, the adversary can no longer bias the outcome.

Our first goal is to design an efficient PASSO VSS protocol with a good resiliency and as minimal assumptions as possible. We will then see how to use this VSS protocol to build a full-fledged PASSO randomness generation protocol. In fact, we show that our randomness generation protocol can be based on a weaker version of VSS, which we call *split-dealer VSS*.

Unoptimized Stateful VSS. Our starting point is the elegant *stateful* VSS protocol by Hirt and Zikas [29] that is based on the BGW VSS protocol [4] and the work of Cramer, Damgård, Dziembowski, Hirt, and Rabin [16]. The protocol requires $n = 2t+1$ parties who hold secret shares and rely on a standard signature scheme (which follows from one-way functions), private communication channels, and access to a broadcast channel.[6] This protocol consists of two phases each with several rounds as described below.

Sharing Phase:

(1) *Share round:* Dealer D with secret s selects a uniform bi-variate polynomial $f(x,y)$ of degree at most t in each variable, such that $f(0,0) = s$. Let $s_{i,j} = f(i,j)$. D privately sends shares $\{s_{k,i}\}_{k\in[n]}$ and $\{s_{i,k}\}_{k\in[n]}$, along with signatures on these values to each party P_i. P_i denotes these values as $\{s^{(i)}_{k,i}\}_{k\in[n]}$ and $\{s^{(i)}_{i,k}\}_{k\in[n]}$.

(2) *Share check round:* Each party P_i checks whether the values they received are *t-consistent*, i.e., fit onto a polynomial of degree at most t, and contain valid signatures from D. If not, P_i broadcasts a complaint.

(3) *Dealer response round:* Dealer D addresses the complaint of each party P_i by broadcasting the correct values for P_i, along it's signatures. If these values are not t-consistent or the signatures are invalid, D is deemed corrupt and the execution halts. Otherwise, P_i adopts the new values as the messages it received in the sharing round.

(4) *Subshare exchange round:* Party P_i sends the value $s^{(i)}_{i,j}$ and both *its own, and the dealer's* signature on it privately to P_j.

(5) *First subshare check round:* Party P_i checks if they received a message along with valid signatures from every other party. If a message from P_j is missing or does not contain valid signatures, P_i broadcasts a complaint.

(6) *Resolve complaints round:* Party P_i checks if there is a complaint by any P_j about P_i. If yes, P_i broadcasts $s^{(i)}_{i,j}$ along with D's and its own signature. If P_i is silent, or any of the signatures are invalid, P_i is deemed corrupt, and everyone sets signatures of P_i to $\perp$. Otherwise, P_j adopts the message broadcast by P_i as the message it received during the subshare exchange.

[6] In our protocol, each party signs an a-priori bounded number of messages, and hence we can use Lamport's signatures [33] which are known from one-way functions.

(7) *Second subshare check round:* Party P_i checks if it received any value $s_{j,i}^{(j)}$ during the subshare exchange or resolve complaints round which is inconsistent with its view. If yes, P_i broadcasts $s_{j,i}^{(j)}$, $s_{j,i}^{(i)}$, along with D's signature on both values. If the two values are different, and have valid signatures, D is deemed corrupt and the execution halts.

Reconstruction Phase: In this phase each party P_i broadcasts $\{s_{k,i}^{(i)}\}_{k\in[n]}$, along with the signature for each $s_{j,i}^{(i)}$ that P_i received from P_j. Each party checks if the values broadcast by every P_i are t-consistent, and all signatures are valid. If not, P_i is disqualified. The values of all non-disqualified parties are interpolated to compute $f(0,0)$.

The protocol's correctness relies on honest parties only sending or broadcasting consistent, correctly signed values, with their shares being sufficient to compute the secret. Privacy is maintained as any t shares reveal no information about the secret, and the adversary learns nothing additional from honest parties during the sharing phase. For verifiability, if the dealer is not disqualified, a sufficient number of honest parties ($n - t = t + 1$) possess consistent shares that define a unique secret. Even a malicious dealer cannot prevent the secret's reconstruction, as no honest party would sign inconsistent shares.

Reducing Round Complexity. The scheme above is not well-suited for a direct transformation into the PASSO setting. In fact, a naive transformation of the above protocol to the PASSO model (see Section 2.1) would require $n = 6(2t + 1) + 2 = 12t + 8$ parties. The round complexity of the above VSS is the first clear bottleneck in our approach. Therefore, we aim to reduce the number of rounds, and thus reduce the number of parties needed in the PASSO model.

Merging Subshare Checks. Observe that if round 5 and round 7 subshare checks could be merged, followed by a one-shot "resolve complaints" (round 6), this would reduce the round complexity by one round, resulting in a reduction of $2t + 1$ parties in the PASSO setting.

The intuition behind having the "resolve complaint" phase between the two subshare checks is the following: If party P_i complains in (5), party P_j resolves the complaint *publicly* in (6) while including its own and the dealer's signatures. Everyone can verify whether P_j's response is valid. Even if so, P_i might still be unhappy, as the dealer could have given inconsistent shares to P_i and P_j. In this case, P_i can complain again, this time including its own and the dealer's signature on the corresponding value. Since all the complaints and the resolving messages are public, anyone can conclude if the dealer is to be blamed or not. We make the following crucial observation: P_i *never changes* its own share based on the resolving message from P_j in round (6). Additionally, P_i can verify whether the value it received from P_j in the subshare exchange (4) is consistent with its own shares received from the dealer *directly after (4).*

These observations allow us to change the protocol as follows, while retaining its security. If the share that P_i was supposed to receive from P_j during the subshare exchange (4) is missing, contains invalid signatures, *or is inconsistent with its own share*, P_i complains *and includes its own and the dealer's signatures.* Then, P_j is forced to publicly respond. As before, if P_j's response is missing or contains invalid signatures, P_j is discarded. If P_j's response is inconsistent with P_i's, but the signatures are valid, everyone can conclude that the dealer misbehaved. This has the same effect as before – either the parties agree on their shares, or either a malicious party P_i or the malicious dealer is disqualified.

Merging Share Check and Subshare Exchange. Next, we seek to merge the share check (2) and the subshare exchange (4) rounds, which would result in another reduction of $2t + 1$ parties.

Currently, the dealer ensures in (3) that either all parties are *happy* with their shares, or the

dealer can be deemed corrupt for *publicly* providing inconsistent shares. Thus after (3), all parties *must* have complete sharings signed by the dealer, and if there are complaints in later rounds, we can definitively assign blame to either the dealer or a party. Observe that if a party complains in (2), *everyone* knows that the party is unhappy, and other happy parties can still crosscheck their values. However, we must ensure that if the dealer resolves complaints *after* the subshare checks, the new shares are *still* consistent with the shares of the happy ones. Subtle modifications allow merging (2) and (4) and delaying the dealer response until the sharing phase's end.

Happy parties proceed with the subshare exchange, while unhappy party P_i skips this phase and broadcasts a complaint. P_i also skips the subshare checks and the resolve complaints round. If happy party P_j complains about a missing message from P_i, P_j includes $s_{j,i}^{(j)}$ and $s_{i,j}^{(j)}$, along with its own and the dealer's signature. After the resolve complaint phase, the dealer addresses P_i's complaint. Everyone verifies that the values posted by the dealer are consistent, in particular with the *non-complaining parties*, like P_j. As P_j broadcasts $s_{j,i}^{(j)}, s_{i,j}^{(j)}$ along with valid signatures from the dealer, anyone can see if the new share distributed by the dealer is consistent, and the dealer can be discarded if this is not the case.

Removing "Resolve Complaints". Currently, the "resolve complaints" round is followed by the dealer response. We might hope that the dealer can resolve the complaints *on behalf* of every party P_j, eliminating the need for the resolve complaints round. However, this modification requires some care: A malicious dealer and a malicious party P_i can provide a share inconsistent with an honest happy party P_j. Imagine a malicious P_i complaining about an honest party P_j, while including a valid signature on $s_{j,i}^{(j)}, s_{i,j}^{(j)}$ from a malicious dealer. Previously, P_j would have responded publicly by providing the dealer's signature on its own share, thus unmasking the malicious dealer. However, now the dealer can simply confirm P_i's share, and thus the share held by P_j becomes inconsistent.

We rectify this issue by using $3t+1$ share receivers (instead of $2t+1$), and skipping the "resolve complaints" round, *without having the dealer resolve parties' complaints about each other*. Every point on which P_i did not get a valid signature from P_j is now essentially lost for reconstruction. P_i still checks whether the points from P_j contain valid signatures of the dealer on an inconsistent share, and blames the dealer if so. P_i also broadcasts a complaint for a missing message from P_j, along with the dealer's signature on its corresponding share to ensure that any share that the dealer will broadcast for an unhappy P_j remains consistent with P_i's share. In the reconstruction phase, we consider any polynomial of P_i to be valid if it has $2t+1$ points $s_{j,i}^{(i)}$, such that (1) each point is either correctly signed by P_j, or broadcast by the dealer if P_j complained during the share check, and (2) all these points lie on a polynomial of degree at most t. Intuitively, requiring $2t+1$ valid points ensures that any party's share is consistent with at least $2t+1-t = t+1$ honest parties. Since any $t+1$ honest shares fix the secret, any valid share is thus consistent with the secret. Simultaneously, given $3t+1$ share receivers, any party can provide at least $2t+1$ such points, as at most t of the points can be unavailable due to malicious parties. While this adds t receivers, we can cut the resolve complaints round and save $t+1$ parties with further optimisations discussed below.

Obtaining PASSO VSS. In the scheme we arrived at, the sharing phase consists of the following rounds: *sharing, share check with subshare exchange, subshare checks*, and *dealer response*. Consider a "linearized" version of this scheme in the PASSO setting, where parties are executed one after the other: It requires one party D_1 to be the dealer in share round, and another party D_2 to execute the role of the dealer in dealer response (this will be the resolver). It further requires a set of $3t+1$

parties $\mathcal{P}^1$ to execute share check with subshare exchange, another set $\mathcal{P}^2$ of size $3t+1$ to execute the subshare check, and a set $\mathcal{P}'$ of $3t+1$ parties to perform the reconstruction. The scheme works as follows: First, the dealer D_1 distributes shares of secrets to parties in $\mathcal{P}^1$, and additionally sends its entire state to the future dealer D_2. Then, one after the other, each party $P_i^1 \in \mathcal{P}^1$ performs the share check, and either sends its subshares to the parties in $\mathcal{P}^2$, or complains about the dealer. Additionally, P_i^1 sends its state to its future counterpart $P_i^2 \in \mathcal{P}_2$. Each $P_i^2 \in \mathcal{P}^2$ performs the subshare checks, complains if necessary, and sends its state to the future counterpart P_i'. Finally, parties in P_i' output the data from which s can be computed according to the reconstruction phase.

We observe the following: during the sharing check with subshare exchange, when party P_j^1 is executed, it could have already obtained the shares of all parties P_i^1, for $i < j$, and perform the subshare checks, as those parties have *already* executed the subshare exchange. If we were to require the checks to be performed only by such *ordered* pairs of parties, in the PASSO model we could remove the $3t+1$ parties $\mathcal{P}^2$ which are currently used to execute the subshare check round of the stateful construction. Note that in the stateful version, we already do not require parties P_i to resolve the complaints about them, as we were able to *remove the resolve complaints round* by requiring $3t+1$ share receivers and $2t+1$ valid points in the reconstruction phase instead. Thus, we simply must ensure that if honest parties notice an inconsistency in the prior version of the protocol, this inconsistency still becomes public, even if the check is done only for pairs (P_i, P_j), where $i < j$. Note that given two honest parties P_i, P_j, it is sufficient if only one of them checks their shares for consistency and complains on behalf of the pair if necessary. For this, we let the complaint include signatures on the respective values from *both* P_i and P_j, as well as the dealer's signatures on these values. This way, in the PASSO version of the construction, we can slash additional $3t+1$ parties $\mathcal{P}^2$ by merging the sharing check with subshare exchange, and subshare checks phases. In the following, we have only the set $\mathcal{P}$, instead of two separate sets $\mathcal{P}^1$ and $\mathcal{P}^2$.

Optimizing Reconstruction in Sending-Leaks. Having a one-to-one correspondence between the $3t+1$ parties P and parties P' seems wasteful, as intuitively $2t+1$ reconstructors ought to be enough. Consider the following optimization: Instead of having each $P_i \in \mathcal{P}$ send the state only to its counterpart $P_i' \in \mathcal{P}'$, party P_i *secret shares* it to parties $\mathcal{P}'$, $|\mathcal{P}'| = 2t+1$, using a standard $(t, 2t+1)$-Shamir secret sharing, while including its own signature on the share. The privacy of the honest shares is still preserved for the duration of the sharing phase by the privacy of the secret sharing scheme. The $2t+1$ reconstructors now broadcast the signed shares of each P_i's state. As only the shares correctly signed by P_i will be used for reconstruction, no adversary can modify the reconstructed state of an honest P_i. Further, $t+1$ correctly signed shares are available for any honest P_i as there are at least $t+1$ honest reconstructors. Now, we only require $2t+1$ parties in the reconstruction phase. This finalizes our scheme in the sending-leaks model. We give a full formal description in Protocol 7.

Optimizing Reconstruction in Execution-Leaks. In the execution-leaks model we can further reduce the number of reconstructors compared to the sending-leaks model. Consider the following modification: Instead of having each $P_i \in \mathcal{P}$ send the state only to its counterpart $P_i' \in \mathcal{P}'$, P_i sends it (signed) to *every* party in $\mathcal{P}'$. As in the execution-leaks model the adversary obtains the values only when an adversarial party is being executed, the privacy of the honest shares is preserved for the duration of the sharing phase. Now, we only require $t+1$ parties in the reconstruction phase to ensure there is at least 1 honest reconstructor in $\mathcal{P}'$. These parties gather the information sent to them by the parties P_i, and reveal all shares that are verified, along with all available signatures. This finalizes our scheme in the execution-leaks model. We give the full construction in Protocol 3,

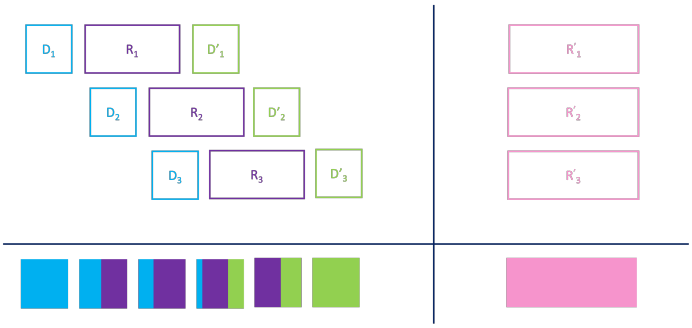

Figure 3: Stacking technique, where D_2 is part of R_1, D'_1 is part of R_2, and so on.

and provide a formal security proof in Section 6.1.

Putting It Together: Efficient PASSO Randomness Generation. We now compile several instances of our PASSO SD-VSS protocol introduced above into the randomness generation protocol we aimed for. Concretely, we take $t+1$ instances of our PASSO SD-VSS so that we have $t+1$ dealers and the final coin is computed through some deterministic function of the non-misbehaving dealers' secrets. This compilation step has to be done carefully to minimize the overall required number of parties without compromising security.

As a first step, consider the following construction: $t+1$ dealers in $\mathcal{D}$ share their secrets via PASSO SD-VSS to the *same* set of share receivers $\mathcal{P}$. Each $P_i \in \mathcal{P}$ verifies its shares, distributes subshares, submits complaints and sends its state to the future as specified by PASSO SD-VSS above. Then, for each dealer $P_i \in \mathcal{D}$ its counterpart $P'_i \in \mathcal{D}'$, dubbed *resolver*, addresses the complaints of the receivers. The reconstructors publish data according to the PASSO SD-VSS. Given this information, anyone can compute the value shared by each dealer, and output for instance the XOR of each valid value that was reconstructed. This requires $6t+4$ parties in total in the execution-leaks model, and $7t+4$ parties in the sending-leaks model.

Stacking. We use a stacking technique to reduce the number of parties needed. Notice that when dealer $P_i \in \mathcal{D}$ is active, each dealer P_j for $j < i$ has already distributed their share. Thus, P_i can act as a dealer for its secret, a *share receiver* of all secrets from prior dealers P_j for $j < i$, and a receiver for the sharing of its own secret. This allows us to "stack" multiple instances of SD-VSS, so each party performs multiple "roles" during execution. Using stacking we merge (1) dealers and share recipients, and (2) share recipients and resolvers. See Figure 3 for a pictorial representation. Security still holds, as at most t dealers can be corrupt, ensuring at least one honest value is used in the output. For each SD-VSS, at most t recipients are corrupt, preserving the security properties of the stacked construction. However, care is required when using stacking in general: For example, security immediately breaks down if we let a party perform roles both in sharing and reconstruction phase. Using stacking in combination with PASSO SD-VSS, we obtain a PASSO randomness generation with $n = 5t+3$ in the execution-leaks model ($n = 6t+3$ in sending-leaks) under the assumption of the existence of one-way functions.

3 Our Model and Notation

We now outline our notation and model. We discuss standard basic cryptographic building blocks in Appendix A.

The sampling of a value x according to a distribution X is denoted by $x \leftarrow\!\!\$\ X$. If S is a set, we also write $x \leftarrow\!\!\$\ S$ when x is sampled uniformly at random from S. The support of the distribution X is denoted by $\mathsf{Supp}(X)$. We denote by $\lambda \in \mathbb{N}$ the security parameter and by $x \leftarrow \mathcal{A}(\mathsf{in}; r)$ the output of the algorithm $\mathcal{A}$ on input in using $r \leftarrow\!\!\$\ \{0,1\}^*$ as its randomness. We often omit this randomness and only mention it explicitly when required. We consider *probabilistic polynomial time* (PPT) machines as efficient algorithms. For integers $m, n \in \mathbb{N}$ we write $[n] = \{1, 2, \ldots, n\}$ and $[m, n] = \{m, m+1, \ldots, n\}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if it is asymptotically smaller than any inverse-polynomial function, namely, for every constant $c > 0$ there exists an integer N_c such that $\mathsf{negl}(\lambda) \leq \lambda^{-c}$ for all $\lambda > N_c$.

3.1 The PASSO Model and Security Definitions

We formally define our network and threat models, as well as security notions.

3.1.1 Network Model

We consider n parties $P_1, \ldots, P_n$. Party P_1 outputs a public value x_1 and sends secret values $s_{1,2}, \ldots, s_{1,n}$ to be received by parties speaking in rounds $i = 2, \ldots, n$, respectively. In the i-th round for $2 \leq i \leq n$, party P_i outputs a public value x_i which depends on the previously broadcast public values $(x_1, \ldots, x_{i-1})$, along with the secret values sent to the party speaking in the i-th round, $(s_{1,i}, \ldots, s_{i-1,i})$. Party P_i then sends secret values $(s_{i,i+1}, \ldots, s_{i,n})$ to be received by parties P_{i+1} through P_n, respectively.

The end goal of a randomness generation protocol in this network model is to generate a random coin based on the public values $x_1, \ldots, x_n$ broadcast by the parties. We now discuss two different adversarial models and our security definition, which mirror those considered in [36].

3.1.2 Adversarial Models

We consider a computationally-bounded adversary who is allowed to corrupt a subset of parties of size at most t. We call t the *corruption threshold*. We study *adaptive* adversaries in the fully malicious setting. An adaptive adversary has a total budget of t corruptions and behaves as follows. After the i-th party P_i speaks, the adversary decides whether to corrupt the $(i+1)$-st party P_{i+1} based on its view of the protocol so far, provided the adversary has not spent all its budget already. Parties that have been corrupted in the past cannot be "uncorrupted", and the adversary *cannot* corrupt previous parties that have already spoken. This is reasonable since, as parties are only executed once, honest parties can (and should) erase their private states immediately after speaking. Corrupted parties can deviate arbitrarily from the protocol.

We consider two network models which differ based on the information made available to the adversary when it executes each dishonest party P_i:

- In the *sending-leaks* model, the adversary immediately learns secret values once they are sent to a dishonest party. In other words, when the adversary is executing the corrupted party P_i, then it is allowed to see the previously broadcast public values $x_1, \ldots, x_{i-1}$ along with all

secret values sent by parties P_1 through P_{i-1} to *all* already corrupted parties P_j, even when $j > i$. We call an adversary in this model a *sending-leaks adversary*.

- We also consider a weaker network model – *execution-leaks* – where secret values sent to some corrupted party P_i are only revealed to the adversary once P_i is executed. Thus, for an *execution-leaks adversary* the behavior of an adversarial party P_i depends on the public values $x_1, \ldots, x_{i-1}$ and only the secret values $s_{j,j'}$ for $1 \leq j < j' \leq i$ and corrupted $P_{j'}$. We call an adversary in this model an *execution-leaks adversary*.

3.1.3 Security Definition

We now define the randomness generation security guarantee. Let λ denote a security parameter. Consider an interaction of an adversary A with the randomness generation protocol and let $\mathsf{OUT}(A)$ denote the coin output of this protocol with adversary A. Let $L(\lambda)$ denote the length of this output. Let D be a distinguisher. Consider the following experiment:

1. $b \xleftarrow{\$} \{0,1\}$.
2. $r \xleftarrow{\$} \{0,1\}^{L(\lambda)}$.
3. If $b = 0$, set $\mathsf{coin} \leftarrow \mathsf{OUT}(A)$. Otherwise, set $\mathsf{coin} \leftarrow r$.
4. $b' \leftarrow D(\mathsf{coin})$.

Definition 1 (Computationally secure PASSO randomness generation)**.** *A PASSO randomness generation protocol is* (t, n)-computationally secure in the sending-leaks (resp. execution-leaks) model *if for all PPT adaptive sending-leaks (resp. execution-leaks) adversaries A corrupting t out of n parties and PPT distinguishers D we have* $\left|\Pr[b = b'] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$ *for some negligible function* $\mathsf{negl}(\lambda)$ *in the experiment above. We call* $\left|\Pr[b = b'] - \frac{1}{2}\right|$ *the* bias of the protocol with respect to A and D.

4 Protocols for Small Number of Parties from Non-Interactive Commitments

We now present our randomness generation protocols for $n = 3t + 1$ parties using non-interactive commitments in the execution-leaks model. These protocols are efficient whenever the corruption threshold t is small. We make use of a combinatorial object that we formally define below.

4.1 t-Sharing Matrices

Definition 2 (t-sharing matrix)**.** *We say that a matrix $M \in \{0,1\}^{m \times \ell}$ is a* t-sharing matrix *if every row of M has Hamming weight t and for any set $S \subseteq [\ell]$ of size $t-1$ there exists $i \in [m]$ such that $M_{ij} = 0$ for all $j \in S$.*

Our next lemma exhibits a lower bound on the number of rows of any t-sharing matrix M, which we prove in Appendix B.1.

Lemma 1. *If $M \in \{0,1\}^{m \times \ell}$ is a t-sharing matrix, then $m \geq \frac{\binom{\ell}{t-1}}{\binom{\ell-t}{t-1}}$.*

We use the following simple construction of a t-sharing matrix in our general protocol for $n = 3t + 1$, which we prove in Appendix B.2.

Lemma 2. *There exists a t-sharing matrix $M \in \{0,1\}^{m \times \ell}$ with $\ell = 2t-1$ columns and $m = \binom{2t-1}{t}$ rows. Moreover, this matrix can be constructed in time polynomial in ℓ and m.*

By Lemma 1, we get that the number of rows in the construction of Lemma 2 cannot be improved if the number of rows is kept as is.

4.2 Our Protocol

With the aid of a t-sharing matrix M as defined above, in Protocol 1 we present our randomness generation protocol in the PASSO execution-leaks model.

Protocol 1 Randomness Generation with $n = 3t + 1$ in Exec.-Leaks Model.

We have a t-sharing matrix $M \in \{0,1\}^{m \times \ell}$ according to Lemma 2, where $\ell = 2t - 1$ and $m = \binom{2t-1}{t}$.

1. For $i \in [2t-1]$:
   (a) For $j \in [m]$, party P_i does the following:
      i. If $\forall k \in [i-1], M_{jk} = 0$ and $M_{ji} = 1$,
         A. Choose value $s_j \leftarrow\!\!\$\ \{0,1\}^{\ell_m(\lambda)}$ and generate $\mathsf{com}_j \leftarrow \mathsf{Commit}(s_j; r_j)$ for random coins $r_j \in \{0,1\}^{\ell_r(\lambda)}$.
         B. Broadcast com_j and send the opening (s_j, r_j) to all P_k where $k \in [i+1, \ell]$ and $M_{jk} = 1$, and to all P_k where $k \in [2t+1, 3t+1]$.
      ii. Else if $M_{ji} = 1$,
         A. Receive (s_j, r_j) from party P_k for some $k \in [i-1]$ and $M_{jk} = 1$.
         B. Broadcast $(\texttt{Complain}, j)$ if nothing was received or if $\mathsf{Commit}(s_j; r_j) \neq \mathsf{com}_j$. Else, send (s_j, r_j) to all P_k where $k \in [2t+1, 3t+1]$.
2. P_{2t} samples $s^* \leftarrow\!\!\$\ \{0,1\}^{\ell_m(\lambda)}$ and broadcasts s^*.
3. For $i \in [2t+1, 3t+1]$, P_i does the following:
   (a) For any $j \in [m]$, if no message $(\texttt{Complain}, j)$ was seen, then receive all messages (s_j, r_j) such that $\mathsf{Commit}(s_j; r_j) = \mathsf{com}_j$, and output (s_j, r_j).

Let $C \subseteq [m]$ be the set of j's such that no message $(\texttt{Complain}, j)$ was seen and (s_j, r_j) was broadcast satisfying $\mathsf{Commit}(s_j; r_j) = \mathsf{com}_j$. The final random string is $s = \bigoplus_{j \in C} s_j \oplus s^*$.

Theorem 4. *If* COM *is a perfectly binding and computationally hiding non-interactive commitment scheme, then Protocol 1 is a $(t, n = 3t+1)$-comp. secure PASSO randomness generation in the execution-leaks model, provided that $n \leq \mathsf{poly}(\lambda)$ and $2^t \leq \mathsf{poly}(\lambda)$ with λ the security parameter. Its computation and communication are polynomial in λ, n, and 2^t, and so are $\mathsf{poly}(\lambda)$.*

In the interest of space, we give the formal proof in Appendix B.3.

4.2.1 Sending-Leaks Variant.

Our sending-leaks protocol is similar to Protocol 1, except that we have $2t+1$ receivers instead of $t+1$. A formal description of this modified protocol is in Protocol 5. In the interest of space, we defer this description, as well as the corresponding theorem and security proof to Appendix B.4.

5 Protocols from Non-Interactive Commitments

Our construction consists of a sequence of $t+1$ instantiations of a split-dealer VSS protocol, which is a modification of the protocol presented in [34]. To recall, each instance distinguishes between the following parties:

1. Party D, who acts as the dealer distributing the secrets (publishing commitments to the points of a bivariate degree-t polynomial and bilaterally sending to each receiver the openings to the horizontal and vertical projections), and sends its state to its counterpart D'.
2. $2t+1$ receivers R_i, who receive and verify the projections (they are of degree-t and consistent with the commitments); complain about the received values if applicable, and otherwise send these to all parties R'_j.
3. Party D' who obtains a state from D and uses it to reveal the projections of the receivers that complained. If D' cannot resolve a complaint, this instance is aborted (D' is deemed corrupt).
4. $t+1$ receivers R'_i who receive all the shares from each party R_j, as well as set their shares to the ones broadcast by D' (if the counterpart R_j complained), and publicly reveal all these shares.

To compose the instances, we organize the parties as follows:

- For $1 \leq i \leq t+1$, P_i executes the protocol of the dealer D in i-th instance. If additionally $i > 1$, P_i also executes the role R_{i-j} in j-th instance, where $j < i$.
- For $t+2 \leq i \leq 3t+2$, P_i executes the role R_{i-j} in j-th instance, where $j < i$. If additionally $i > 2t+2$, P_i also executes the role of the dealer D' in the $i-2t-2$-th instance.
- For $i = 3t+3$, P_i executes the role D' in the $(t+1)$-st instance.
- For $3t+4 \leq i \leq 4t+4$, P_i executes the protocol of R'_{i-3t-3} for each instance.

We formally describe the execution-leaks protocol in Protocol 2.

Theorem 5. *Assuming non-interactive perfectly binding commitments, there is an efficient $(t, n = 4t+4)$-computationally secure PASSO randomness generation protocol in the execution-leaks model.*

In the interest of space, we defer the formal proof to Appendix C.1.

5.0.1 Sending-Leaks Variant.

In the sending-leaks model, we similarly implement the behavior of each dealer using two parties – one responsible for the sharing of a secret, and one responsible for addressing the complaints. However, we not only have $2t+1$ parties R_i, but also $2t+1$ parties R'_j. Each R_i follows the procedure of round two, and if its shares verify, but it additionally sends its shares to *only* R'_i. Finally, each R'_i publishes the shares (it got from R_i) which verified correctly. Note that there are $t+1$ pairs (R_i, R'_i) that are both honest, which will publish projections that are consistent with the published commitments and therefore will be enough to reconstruct the bivariate polynomials.

Theorem 6. *Assuming non-interactive perfectly binding commitments, there is an efficient* $(t, n = 5t+4)$*-computationally secure PASSO randomness generation protocol in the sending-leaks model.*

We defer the formal protocol and discussion to Appendix D.3.

6 Protocols from One-Way Functions

We introduce a new building block called *Split-Dealer Verifiable Secret Sharing* in PASSO, which can be constructed using any signature scheme. Later, we will formally describe how this building block can be used to construct an efficient randomness generation protocol with $n = 5t+3$ for $t \in O(\mathsf{poly}(\lambda))$.

6.1 Split-Dealer Verifiable Secret Sharing in PASSO

We now describe our split-dealer verifiable secret sharing (SD-VSS) protocol, which requires n parties, divided into four categories: a dealer with an initial secret input s, a set of receivers, a resolver, and a set of reconstructors. The protocol satisfies the usual security guarantees, with the caveat that correctness and privacy only holds when *both* the dealer and resolver are honest. As we will show in Section 6.2, this is sufficient to obtain randomness generation.

A formal description of our SD-VSS scheme in the execution-leaks model can be found in Protocol 3. This scheme was already discussed informally at the beginning of Section 2.5. We defer its analysis to Appendix D.1.

In the sending-leaks model the protocol is almost the same as Protocol 3, except that we now require $2t+1$ reconstructors. Each receiver P_i now shares its state to the reconstructors, while signing each share. The reconstructors output all shares along with the signature of the corresponding P_i. The correctly signed shares are used to reconstruct the state of P_i, which in turn is used to compute the output as in the execution-leaks case. In the interest of space, we defer the formal description along with the security analysis to Appendix D.4.

6.2 PASSO SD-VSS-based Randomness Generation with $n = 5t+3$

To build a randomness generation protocol, we execute $t+1$ independent instances of our SD-VSS protocol, and set the final coin to the XOR of coin output of each SD-VSS instance that did not abort. To reduce the total number of parties, instead of naively repeating the SD-VSS protocol one instance after the other, we pipeline the protocol execution by starting the first instance with party P_1, the second instance with party P_2, and so on. This means that some parties may have to

simultaneously perform the role of a dealer in some i-th SD-VSS instance, that of a receiver in some k-th SD-VSS instance, etc. We give the full scheme in the execution-leaks model in Protocol 4. Its communication complexity is $O(n^4)$ (excluding any polynomial factors in the security parameter).

The protocol for the sending-leaks model is the same as the execution-leaks protocol except that parties P_i for $i \in [4t+4, 6t+4]$ act as reconstructors and we make use of the sending-leaks SD-VSS as our building block. We defer the full description to Appendix D.5.

Further Reduction of Number of Parties. We can further reduce the number of parties from $5t+4$ to $5t+3$ in execution-leaks and from $6t+4$ to $6t+3$ in sending-leaks. To do this, we observe that a dealer can be their own recipient similar to distributed key generation protocols [21], i.e., the recipients of P_1's sharing includes P_1 and $P_2, \ldots, P_{3t+1}$. Intuitively, this does not give the adversary any advantage in terms of its corruption budget as each dealer still performs a SD-VSS sharing to $3t+1$ parties. Thus, the receivers are now parties P_k for $k \in [4t+1]$, the resolver parties P_k for $k \in [3t+2, 4t+2]$, and the reconstructors are P_k for $k \in [4t+3.5t+3]$. For ease of presentation, we present our formal protocols without including this optimization and defer the formal proof of Theorem 7 to Appendix D.2.

Theorem 7. *Assuming the existence of digital signatures, there exists a $(t, n = 5t+3)$-computationally secure PASSO randomness generation protocol in the execution-leaks model.*

7 Lower Bounds for PASSO Protocols without Setup

We now discuss lower bounds on the number of parties of computationally secure PASSO randomness generation as a function of the corruption threshold. Such lower bounds were obtained for information-theoretic PASSO in [36]. They showed an impossibility result for $t = 1$ corruptions and $n = 4$ parties, which generalizes directly to an impossibility result for t corruptions and $n = 4t$ parties in the sending-leaks model (and hence to an $n \geq 4t+1$ lower bound for protocols in this model). However, contrary to what they claim, their impossibility result does not directly extend to $t > 1$ corruptions in the execution-leaks model.

We begin by showing a computational analog of their impossibility result for $t = 1$ corruptions (here the execution-leaks and sending-leaks models coincide).

Theorem 8. *For every PASSO protocol with $n = 3$ parties there exists an efficient* non-adaptive *execution-leaks adversary A corrupting $t = 1$ parties that achieves bias $\varepsilon \geq 0.01$. In particular, there is no $(t = 1, n = 3)$-computationally secure PASSO randomness generation protocol in either the execution-leaks or sending-leaks model.*

Note that the constant bias 0.01 suffices to show that sub-constant (in particular, negligible) bias cannot be achieved, and that the impossibility result also applies to adaptive adversaries. We defer Theorem 8's proof to Appendix E.1.

One take-away of this theorem combined with Theorem 1 is that $n = 4$ parties are necessary and sufficient for computationally secure PASSO randomness generation against $t = 1$ corruptions. In the information-theoretic setting, $n = 5$ parties are known to be necessary and sufficient [36].

Next, we show how to extend Theorem 8 to $t > 1$ corruptions and $n = 3t$ parties in the sending-leaks model. This implies that (t, n)-computationally secure PASSO protocols in the sending-leaks model require $n \geq 3t+1$ parties, for any $t \geq 1$. We formalize this below and defer the proof to Appendix E.2.

Theorem 9. *For every $t \geq 1$ and PASSO protocol with $n = 3t$ parties there exists an efficient* non-adaptive *sending-leaks adversary A corrupting t parties that achieves bias $\varepsilon \geq 0.01$. This means that there is no $(t, n = 3t)$-computationally secure PASSO randomness generation protocol in the sending-leaks model.*

As discussed in Section 2.3, it seems much harder to prove an analog of Theorem 9 in the execution-leaks setting. In the next result we crucially exploit the ability to choose corruption patterns other than consecutive blocks of parties to obtain a matching impossibility result in the execution-leaks setting with $t = 2$ corruptions.

Theorem 10. *For every PASSO protocol with $n = 6$ parties there exists an efficient* non-adaptive *execution-leaks adversary A corrupting $t = 2$ parties that achieves non-negligible bias. This means that there is no $(t = 2, n = 6)$-computationally secure PASSO protocol in the execution-leaks model.*

We defer the proof of this theorem to Appendix E.3. Combining this result with Theorem 1 allows us to conclude that $n = 7$ parties are necessary and sufficient for computationally secure PASSO randomness generation against $t = 2$ corruptions in the execution-leaks model. We leave it as an interesting open problem to generalize this result to arbitrary $t > 2$.

Acknowledgements

This work was supported by a Protocol Labs Cryptonet Network Grant "Stateless Distributed Randomness Generation". C. Liu-Zhang was supported in part by the ETH Zurich Leading House Research Partnership Grant RPG-072023-19. J. Ribeiro was also supported by NOVA LINCS (ref. UIDB/04516/2020) and by FCT/MECI through national funds and when applicable co-funded EU funds under UID/50008: Instituto de Telecomunicações. E. Masserova was also supported by the Defense Advanced Research Projects Agency under contracts No.HR001120C0086 and FA8750-17-1-0059 "Obfuscated Manufacturing for GPS (OMG)".

References

[1] Acharya, A., Hazay, C., Kolesnikov, V., Prabhakaran, M.: Scales. In: Kiltz, E., Vaikuntanathan, V. (eds.) Theory of Cryptography. pp. 502–531. Springer Nature Switzerland, Cham (2022)

[2] Acharya, A., Hazay, C., Kolesnikov, V., Prabhakaran, M.: Malicious security for SCALES - outsourced computation with ephemeral servers. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX. Lecture Notes in Computer Science, vol. 14928, pp. 3–38. Springer (2024). https://doi.org/10.1007/978-3-031-68400-5_1, https://doi.org/10.1007/978-3-031-68400-5_1

[3] Applebaum, B., Kachlon, E., Patra, A.: The round complexity of statistical MPC with optimal resiliency. In: Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023). pp. 1527–1536 (2023)

[4] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for noncryptographic fault-tolerant distributed computations. In: Annual Symposium on the Theory of Computing (1988)

[5] Benhamouda, F., Gentry, C., Gorbunov, S., Halevi, S., Krawczyk, H., Lin, C., Rabin, T., Reyzin, L.: Can a public blockchain keep a secret? In: Pass, R., Pietrzak, K. (eds.) Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12550, pp. 260–290. Springer (2020). https://doi.org/10.1007/978-3-030-64375-1_10, https://doi.org/10.1007/978-3-030-64375-1_10

[6] Bienstock, A., Escudero, D., Polychroniadou, A.: On linear communication complexity for (maximally) fluid MPC. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14081, pp. 263–294. Springer (2023). https://doi.org/10.1007/978-3-031-38557-5_9, https://doi.org/10.1007/978-3-031-38557-5_9

[7] Blum, E., Katz, J., Liu-Zhang, C.D., Loss, J.: Asynchronous byzantine agreement with subquadratic communication. In: Theory of Cryptography: 18th International Conference, TCC. pp. 353–380. Springer (2020)

[8] Blum, M.: Coin flipping by telephone a protocol for solving impossible problems. SIGACT News **15**(1), 23–27 (jan 1983). https://doi.org/10.1145/1008908.1008911

[9] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: International conference on the theory and application of cryptology and information security. pp. 514–532. Springer (2001)

[10] Braun, L., Damgård, I., Orlandi, C.: Secure multiparty computation from threshold encryption based on class groups. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14081, pp. 613–645. Springer (2023). https://doi.org/10.1007/978-3-031-38557-5_20, https://doi.org/10.1007/978-3-031-38557-5_20

[11] Cascudo, I., David, B.: SCRAPE: Scalable Randomness Attested by Public Entities. In: International Conference on Applied Cryptography and Network Security. pp. 537–556 (2017)

[12] Cascudo, I., David, B.: ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In: Advances in Cryptology – ASIACRYPT 2020. pp. 311–341. Springer International Publishing, Cham (2020)

[13] Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theoretical Computer Science **777**, 155–183 (2019). https://doi.org/10.1016/j.tcs.2019.02.001, in memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I

[14] Chopard, A., Hirt, M., Liu-Zhang, C.D.: On communication-efficient asynchronous mpc with adaptive security. In: Theory of Cryptography: 19th International Conference, TCC 2021. pp. 35–65. Springer (2021)

[15] Choudhuri, A.R., Goel, A., Green, M., Jain, A., Kaptchuk, G.: Fluid MPC: Secure multiparty computation with dynamic participants. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 94–123. Springer International Publishing, Cham (2021)

[16] Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient multiparty computations secure against an adaptive adversary. In: Stern, J. (ed.) Advances in Cryptology — EUROCRYPT '99. pp. 311–326. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)

[17] David, B., Deligios, G., Goel, A., Ishai, Y., Konring, A., Kushilevitz, E., Liu-Zhang, C.D., Narayanan, V.: Perfect MPC over layered graphs. In: Advances in Cryptology – CRYPTO 2023. pp. 360–392 (2023)

[18] Deligios, G., Goel, A., Liu-Zhang, C.: Maximally-fluid MPC with guaranteed output delivery. IACR Cryptol. ePrint Arch. p. 415 (2023), https://eprint.iacr.org/2023/415

[19] Deligios, G., Konring, A., Liu-Zhang, C.D., Narayanan, V.: Statistical layered MPC. In: Theory of Cryptography. pp. 362–394 (2025)

[20] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)

[21] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18. pp. 295–310. Springer (1999)

[22] Gentry, C., Halevi, S., Krawczyk, H., Magri, B., Nielsen, J.B., Rabin, T., Yakoubov, S.: YOSO: You Only Speak Once. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 64–93. Springer International Publishing, Cham (2021)

[23] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC 2008). pp. 197–206. Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1374376.1374407

[24] Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing. p. 25–32. STOC '89, Association for Computing Machinery, New York, NY, USA (1989). https://doi.org/10.1145/73007.73010

[25] Goyal, R., Hohenberger, S., Koppula, V., Waters, B.: A generic approach to constructing and proving verifiable random functions. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography. pp. 537–566. Springer International Publishing, Cham (2017)

[26] Goyal, V., Kothapalli, A., Masserova, E., Parno, B., Song, Y.: Storing and retrieving secrets on a blockchain. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13177, pp. 252–282. Springer (2022). https://doi.org/10.1007/978-3-030-97121-2_10, https://doi.org/10.1007/978-3-030-97121-2_10

[27] Goyal, V., Masserova, E., Parno, B., Song, Y.: Blockchains enable non-interactive MPC. In: Nissim, K., Waters, B. (eds.) Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13043, pp. 162–193. Springer (2021). https://doi.org/10.1007/978-3-030-90453-1_6, https://doi.org/10.1007/978-3-030-90453-1_6

[28] Guruswami, V., Rudra, A., Sudan, M.: Essential Coding Theory (2023), draft available at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book

[29] Hirt, M., Zikas, V.: Adaptively secure broadcast. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 466–485. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[30] Kelsey, J., Brandão, L.T.A.N., Peralta, R., Booth, H.: A reference for randomness beacons: Format and protocol version 2. Tech. rep., National Institute of Standards and Technology (2019), https://csrc.nist.gov/pubs/ir/8213/ipd

[31] Kolby, S., Ravi, D., Yakoubov, S.: Constant-round YOSO MPC without setup. IACR Commun. Cryptol. **1**(3), 30 (2024). https://doi.org/10.62056/AE5W4FE-3, https://doi.org/10.62056/ae5w4fe-3

[32] Kumaresan, R., Patra, A., Rangan, C.P.: The round complexity of verifiable secret sharing: The statistical case. In: Advances in Cryptology – ASIACRYPT 2010. pp. 431–447. Springer (2010)

[33] Lamport, L.: Constructing digital signatures from a one way function. Tech. Rep. CSL-98 (October 1979), https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/, this paper was published by IEEE in the Proceedings of HICSS-43 in January, 2010.

[34] Liu-Zhang, C.D., Masserova, E., Ribeiro, J., Soni, P., Thyagarajan, S.: Improved YOSO randomness generation with worst-case corruptions. In: Financial Cryptography and Data Security (FC 2024) (2024), available at https://fc24.ifca.ai/preproceedings/147.pdf

[35] Mahmoody, M., Pass, R.: The curious case of non-interactive commitments – on the power of black-box vs. non-black-box use of primitives. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. pp. 701–718. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[36] Nielsen, J.B., Ribeiro, J., Obremski, M.: Public randomness extraction with ephemeral roles and worst-case corruptions. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. pp. 127–147. Springer Nature Switzerland, Cham (2022), updated version available at https://eprint.iacr.org/2022/237.

[37] Patra, A., Choudhary, A., Rabin, T., Rangan, C.P.: The round complexity of verifiable secret sharing revisited. In: Halevi, S. (ed.) Advances in Cryptology – CRYPTO 2009. pp. 487–504. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[38] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology — CRYPTO '91. pp. 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg (1992)

[39] Rabin, M.O.: Transaction protection by beacons. Journal of Computer and System Sciences **27**(2), 256–267 (1983)

[40] Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991). https://doi.org/10.1007/BF00196725

[41] Waters, B.: Efficient identity-based encryption without random oracles. In: Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24. pp. 114–127. Springer (2005)

[42] Yao, A.C.: Theory and application of trapdoor functions. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982). pp. 80–91 (1982). https://doi.org/10.1109/SFCS.1982.45

Supplementary Material

A Basic Cryptographic Building Blocks

A.1 Digital Signatures

A *digital signature scheme* DS is a tuple of algorithms:

- A *key generation algorithm* $\mathsf{KGen}(1^\lambda)$ that takes the security parameter λ and outputs the verification/signing key pair $(\mathsf{vk}, \mathsf{sk})$.
- A *signing algorithm* $\mathsf{Sign}(\mathsf{sk}, m)$ which outputs a signature σ on input the secret key sk and the message m.
- A *verification algorithm* $\mathsf{Vf}(\mathsf{vk}, m, \sigma)$ with binary output. We say that σ is a *valid* signature of m under the verification key vk if $\mathsf{Vf}(\mathsf{vk}, m, \sigma) = 1$, and *invalid* otherwise.

We require standard unforgeability of the digital signature scheme.

Definition 3 (Unforgeability)**.** *A digital signature scheme* DS *is* unforgeable *if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that the probability of winning the following game is upper bounded by* $\mathsf{negl}(\lambda)$*, where* λ *is a security parameter:*

1. *The challenger runs* $\mathsf{KGen}(1^\lambda)$ *and obtains a verification/secret key pair* $(\mathsf{vk}, \mathsf{sk})$*.*
2. *The adversary* $\mathcal{A}$ *can adaptively choose messages* m *and query the challenger to learn a corresponding signature* $\mathsf{Sign}(\mathsf{sk}, m)$*. Let* $m_1, \ldots, m_q$ *be the messages queried by* $\mathcal{A}$*, for some integer* q*.*
3. *The adversary* $\mathcal{A}$ *chooses a fresh message* $m' \notin \{m_1, \ldots, m_q\}$ *and wins the game if they output a valid signature* σ *of* m' *under the verification key* vk*.*

Signature schemes have been constructed from a wide range of assumptions starting from one-way functions [33], to more structured algebraic assumptions like the discrete logarithm problem [20, 40], pairing-based problems [9, 41], and the Shortest Integer Solution problem (SIS) [23].

A.2 Non-Interactive Commitments

A *non-interactive commitment scheme* COM is a tuple of algorithms:

- A *commitment generation algorithm* $\mathsf{Commit}(m; r)$ that takes as input a message $m \in \{0,1\}^{\ell_m(\lambda)}$ (for some message space $\mathcal{M}$), and some randomness $r \in \{0,1\}^{\ell_r(\lambda)}$, and returns a commitment $\mathsf{com} \in \{0,1\}^{\ell_c(\lambda)}$. Here ℓ_m, ℓ_r, ℓ_c are some polynomials in λ, the security parameter.
- The *opening* of the commitment com. In our case, this is simply the message m and the randomness r.

We will require two standard properties of non-interactive commitments: perfect binding and computational hiding.

Definition 4 (Perfectly binding commitment)**.** *A non-interactive commitment scheme* COM *is* perfectly binding *if for all* $m_0, m_1 \in \{0,1\}^{\ell_m(\lambda)}$ *such that* $m_0 \neq m_1$ *it holds that*

$$\{\mathsf{Commit}(m_0; r_0)\}_{r_0 \in \{0,1\}^{\ell_r(\lambda)}} \bigcap \{\mathsf{Commit}(m_1; r_1)\}_{r_1 \in \{0,1\}^{\ell_r(\lambda)}} = \varnothing.$$

Definition 5 (Computationally hiding commitment)**.** *A non-interactive commitment scheme* COM *is* computationally hiding *if for every polynomially bounded function* $\alpha(\cdot)$ *and every PPT adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that the probability of winning the following game is upper bounded by* $1/2 + \mathsf{negl}(\lambda)$*, where* λ *is the security parameter:*

1. *The adversary* $\mathcal{A}(1^\lambda)$ *samples distinct messages* $m_0, m_1 \in \{0,1\}^{\alpha(\lambda)}$ *and sends them to the challenger;*
2. *The challenger samples a bit* $b \leftarrow\!\!\$\ \{0,1\}$*, computes a commitment* $\mathsf{com} = \mathsf{Commit}(m_b; r)$ *to* m_b*, and sends* com *to* $\mathcal{A}$*.*
3. *The adversary* $\mathcal{A}$ *outputs a bit* b' *and wins if and only if* $b' = b$*.*

As mentioned before, non-interactive commitments can be instantiated from a variety of concrete assumptions including factoring [8, 42, 24], more recently from LWE and LPN [25], and even from the general assumption of injective one-way functions. While black-box separations between general one-way functions and non-interactive commitments are known [35], non-interactive commitments are fundamental and one of the weakest complexity-theoretic cryptographic assumptions.

B Missing Proofs and Protocols from Section 4

B.1 Proof of Lemma 1

We say that a vector $v \in \{0,1\}^\ell$ *evades* a set $S \subseteq [\ell]$ if $v_S = 0$. Every such v of weight t evades exactly $\binom{\ell-t}{t-1}$ sets $S \subseteq [\ell]$ of size $t-1$. This means that at most $m \cdot \binom{\ell-t}{t-1}$ such sets are evaded by at least one of the m rows of M. On the other hand, for M to be t-sharing it must be the case that every set $S \subseteq [\ell]$ of size $t-1$ (of which there $\binom{\ell}{t-1}$ choices) is evaded by some row of M, and so we must have

$$m \cdot \binom{\ell - t}{t-1} \geq \binom{\ell}{t-1}.$$

This yields the desired lower bound on m.

B.2 Proof of Lemma 2

Consider the matrix $M \in \{0,1\}^{\binom{2t-1}{t} \times (2t-1)}$ where the rows of M correspond to all $(2t-1)$-bit vectors of weight exactly t. It suffices to check that for all subsets of $t-1$ columns of M there exists an index i on which they are all 0.

Fix any subset $S \subseteq [2t-1]$ of size $t-1$. Since $[\ell] \setminus S$ has size t, there is a row of M whose support lies outside S. Therefore, the columns of M indexed by S are all 0 on this row.

B.3 Proof of Theorem 4

Before we prove this theorem, we state and prove the following lemma that will be useful in our analysis.

Lemma 3. *If the adversary corrupts some P_i with $i \in [2t-1]$ such that for some $j \in [m]$ we have $M_{jk} = 0$ for all $k \in [i-1]$ and $M_{ji} = 1$, then either the adversary's commitment com_j receives a $(\texttt{Complain}, j)$ or a valid opening is broadcast by some P_h for $h \in [2t+1, 3t+1]$.*

Proof of Lemma 3. Consider index $i \in [2t-1]$, such that party P_i is corrupt and any index $j \in [m]$ such that $M_{jk} = 0$ for all $k \in [i-1]$ and $M_{ji} = 1$. The adversary broadcasts commitment com_j, and the openings to any party it wishes to. However, note that there exists an honest party P_k for $k \in [i, 2t-1]$ and $M_{jk} = 1$, which if it did not receive the valid opening, it would broadcast a message $(\texttt{Complain}, j)$. On the other hand, if the adversary sends valid opening of the commitment com_j to P_k, then P_k would send the opening to all recipients $\{P_{2t+1}, \ldots, P_{3t+1}\}$. Note that there is at least 1 honest $P_h \in \{P_{2t+1}, \ldots, P_{3t+1}\}$ that receives the opening of com_j from P_k and outputs that during its execution. This ensures that the opening of the adversarial commitment will be broadcast by an honest P_h. □ □

Proof of Theorem 4. Let A be an arbitrary computationally-bounded adaptive adversary with a budget of t corruptions. We begin by noting that to argue that Protocol 1 is (t, n)-computationally secure it suffices to consider adversaries A that corrupt party P_{2t} with probability 1. Indeed, in any execution of Protocol 1 where party P_{2t} behaves honestly it holds that

1. The value of the final string is fixed at the end of the execution of P_{2t};
2. The string broadcast by P_{2t} is uniformly random and independent of the actions of parties $P_1, \ldots, P_{2t-1}$ and of the adversary's behavior up to the execution of P_{2t}.

Since the final string is obtained by XORing the strings correctly committed to by parties $P_1, \ldots, P_{2t-1}$ and the string broadcast by P_{2t}, combining these two observations shows that any execution of Protocol 1 where party P_{2t} is honest yields a uniformly random final string.

Therefore, from here onwards we may assume that the adaptive adversary A always corrupts party P_{2t}. The proof follows a hybrid argument, where hybrids are simulated executions of appropriate variants of Protocol 1 interacting with adversary A. More precisely, the simulator simulates the operations of the honest parties, while receiving all the broadcast messages and those that were sent privately by the adversary, and then outputs the final string produced by this simulation. The starting hybrid corresponds to a true execution of the protocol against adversary A, the final hybrid corresponds to the execution of a protocol that clearly produces a uniformly random final string, and we will show that the distributions of the final strings output by any two consecutive hybrids are computationally indistinguishable.

More precisely, we consider the following hybrids.

Hybrid_0: This hybrid corresponds to the execution of the real protocol interacting with adversary A.

Hybrid_1: This hybrid execution is the same as Hybrid_0, except that the simulator exits the execution of the protocol after executing party P_{2t} and sets the final string appropriately. This is done by looking at the random coins committed to by the honest parties and the openings shared by the adversary. Importantly, the parties $\{P_{2t+1}, \ldots, P_{3t+1}\}$ need not be executed as the final string is already determined.

Claim 1. *The outputs of* Hybrid_0 *and* Hybrid_1 *are identically distributed.*

Proof. This follows from the fact that the commitments are perfectly binding and from Lemma 3. Note that apart from P_{2t}, the adversary has a budget of $t-1$ corruptions. Therefore, there is at least one honest role that receives the opening of the adversary's commitment and can complain (kill the value) or forward a valid opening (thereby considering the value into the final randomness). That is, there is at least one honest P_k for each corrupted P_i for $i \in [2t-1]$ such that $M_{ji} = 1$ and $M_{jk} = 1$ where $i < k < 2t-1$. Here we let $M_{jk'} = 0$ for $k' \in [i-1]$. This justifies the application of Lemma 3 and ensures the coin is determined by the end of P_{2t}'s execution. □

$\underline{\mathsf{Hybrid}_2}$: The hybrid execution is the same as Hybrid_1, except for the following. The simulator does not broadcast $(\texttt{Complain}, \cdot)$ on behalf of honest parties for the commitments broadcast by honest parties. In other words, the honest parties do not complain about other honest parties' commitments.

Claim 2. *The outputs of* Hybrid_1 *and* Hybrid_2 *are identically distributed.*

Proof. The final string distributions in both hybrids is identical as honest parties behave according to the protocol specification. This means that when they commit to values and open them, they will do so correctly. Moreover, honest parties wouldn't complain if the opening was indeed a valid one. Therefore, honest commitments will not receive complaints from other honest parties. □

$\underline{\mathsf{Hybrid}_3}$: The hybrid repeats the following procedure a maximum of $T = 2\lambda \cdot 2t \cdot 2^{2t}$ times, stopping if one of these repetitions succeeds or if it reaches the maximum number T of repetitions, in which case it outputs the special symbol $\perp$ as its final string s. The procedure starts by sampling indices $i^\star \in [2t-1]$ and $j^\star \in [m]$ uniformly at random. Intuitively, $i^\star$ and $j^\star$ are guesses for a dealer and a corresponding sharing such that all participants are honest. The existence of indices with these properties is guaranteed by the definition of the t-sharing matrix M. If $M_{ji} = 0$ or $M_{jk} = 1$ for some $k < i$, the hybrid aborts and the restarts the procedure. Otherwise, the hybrid behaves like Hybrid_2, except that if at some point in its execution it holds that some party $P_{k'}$ with $M_{j^\star k'} = 1$ is dishonest, then the hybrid aborts and restarts the procedure. If this procedure succeeds, then Hybrid_3 outputs the corresponding final string and stops.

Claim 3. *The outputs of* Hybrid_2 *and* Hybrid_3 *are* $\mathsf{negl}(\lambda)$*-close in statistical distance. Furthermore,* Hybrid_3 *runs in time* $\mathsf{poly}(\lambda)$.

Proof. We begin by arguing about the running time of Hybrid_3. Each execution of the procedure runs in time $\mathsf{poly}(\lambda)$ and this is repeated at most $T = \lambda \cdot 2t \cdot 2^{2t} = \mathsf{poly}(\lambda)$ times, since $2^{2t} \leq \mathsf{poly}(\lambda)$ by hypothesis.

To see that the output distributions of Hybrid_2 and Hybrid_3 are $\mathsf{negl}(\lambda)$-close in statistical distance, first note that if one of the iterations of the procedure executed by Hybrid_3 succeeds, then its output is identically distributed to that of Hybrid_2. Therefore, the statistical distance between the outputs of Hybrid_2 and Hybrid_3 is upper bounded by the probability that all T repetitions of the procedure in Hybrid_3 fail and this hybrid outputs $\perp$. The probability of a given execution of the procedure succeeding is at least $\frac{1}{2t \cdot m} \geq \frac{1}{2t \cdot 2^{2t}}$. This holds since each execution is independent of the choice of $i^\star$ and $j^\star$ up until it is aborted, there are at most $2t$ choices for $i^\star$ and $m \leq 2^t$ choices for $j^\star$, and a good choice of $(i^\star, j^\star)$ always exists by the definition of the t-sharing matrix M. Since

different executions of the procedure are independent and $T = 2\lambda \cdot 2t \cdot 2^{2t}$, a Chernoff bound yields that the probability that Hybrid_3 outputs $\perp$ is $\mathsf{negl}(\lambda)$. □

Hybrid_4: This hybrid behaves like Hybrid_3 except for the following change. In each execution of the procedure, the hybrid sets the commitment $\mathsf{com}_{j^\star}$ of $P_{i^\star}$ to be a commitment to 0. If the procedure does not abort, the final string is computed right after the execution of P_{2t} as before, but choosing a random string $s_{j^\star}$ as opening for $P_{i^\star}$'s commitment $\mathsf{com}_{j^\star}$. Clearly, the opening is not correct, note that this opening is not sent to the adversary if for all $k' \in [i, 2t-1]$ with $M_{jk} = 1$ the party $P_{k'}$ is honest.

Claim 4. *The output of* Hybrid_4 *is uniformly random. Furthermore, the outputs of* Hybrid_3 *and* Hybrid_4 *are computationally indistinguishable.*

Proof. To see the first statement, note that the commitment com_{j^*} has no information about $s_{j^\star}$. Additionally, the adversary has committed to its corruption pattern, values, commitments, and choice of recipients independently of $s_{j^\star}$. Since the output of Hybrid_4 is obtained via an XOR with $s_{j^\star}$ and $s_{j^\star}$ is independent of values broadcast during the protocol, the output of Hybrid_4 is uniformly random.

The second statement follows from the computational hiding property of the commitment scheme COM. More precisely, let $\mathcal{D}'$ be an efficient distinguisher that distinguishes between the outputs of Hybrid_3 and Hybrid_4 with non-negligible advantage. We will use it to construct an efficient distinguisher $\mathcal{D}$ that breaks the hiding property of COM with the same non-negligible advantage. The distinguisher $\mathcal{D}$ chooses a random value $s_{j^\star}$ and sends $(s_{j^\star}, 0)$ as the two challenge messages to the challenger in the hiding game. It then receives a commitment com_{j^*}. The distinguisher proceeds as Hybrid_4, except that the $j^\star$-th commitment broadcast by party $P_{i^\star}$ is set to be com_{j^*}. The final output s is computed as in Hybrid_3 using $s_{j^\star}$ as the supposed opening of the commitment com_{j^*}. The distinguisher $\mathcal{D}$ feeds the output s to $\mathcal{D}'$ and returns whatever bit b that $\mathcal{D}'$ outputs. Notice that if the challenger sets $\mathsf{com}_{j^\star}$ to be the commitment to $s_{j^\star}$, then the random coin s is set according to Hybrid_3. On the other hand, if $\mathsf{com}_{j^\star}$ was set to be a commitment to 0, then the output s is set according to Hybrid_4. Therefore, distinguisher $\mathcal{D}$ is able to win the hiding game with the same non-negligible advantage as that of $\mathcal{D}'$. Given the computationally hiding property of COM, we arrive at a contradiction and conclude that the outputs of Hybrid_4 and Hybrid_3 are computationally indistinguishable. □

Combining the claims above shows that the outputs of Hybrid_0 (the real protocol) and Hybrid_4 (which produces a uniformly random string) are computationally indistinguishable. We conclude that Protocol 1 is (t, n)-computationally secure, as desired.

□

B.4 Sending-Leaks Protocol for $n = 4t$ Parties

We formally describe our sending-leaks protocol for $n = 4t$ parties in Protocol 5. Its security guarantees are stated in the theorem below.

Theorem 11. *If* COM *is a perfectly binding and computationally hiding non-interactive commitment scheme, then Protocol 5 is a* $(t, n = 4t)$*-computationally secure PASSO randomness generation protocol in the sending-leaks model, provided that* $n \leq \mathsf{poly}(\lambda)$ *and* $2^t \leq \mathsf{poly}(\lambda)$ *with* λ *the security*

parameter. Furthermore, its computational and communication complexities are also polynomial in λ*,* n*, and* 2^t*, and so are* $\mathsf{poly}(\lambda)$*.*

The proof of Theorem 11 proceeds exactly like the proof of Theorem 4, except that we use the following lemma in place of Lemma 3.

Lemma 4. *If the adversary corrupts some party* P_i *where* $i \in [2t-1]$*, such that for some* $j \in [m]$*,* $M_{jk} = 0$ *for* $k \in [i-1]$ *and* $M_{ji} = 1$*, then either the adversary's commitment* com_j *receives a* $(\texttt{Complain}, j)$ *or a valid opening is reconstructed after all parties* P_h *for* $h \in [2t, 4t]$ *have completed their execution.*

Proof of Lemma 4. Consider index $i \in [2t-1]$, such that party P_i is corrupt and any index $j \in [m]$ such that $M_{jk} = 0$ for all $k \in [i-1]$ and $M_{ji} = 1$. The adversary broadcasts commitment com_j, and the openings to any party it wishes to. However, note that there exists an honest party P_k for $k \in [i, 2t-1]$ and $M_{jk} = 1$, which if it did not receive the valid opening, it would broadcast a message $(\texttt{Complain}, j)$. On the other hand, if the adversary sends a valid opening of the commitment com_j to party P_k, then party P_k would send $t+1$-out of $2t+1$ sharing of the opening to all recipients $\{P_{2t}, \ldots, P_{4t}\}$. That is party P_h for $h \in [2t, 4t]$ receives the $(h-(2t-1))$-th share of the opening. Note that there are at least $t+1$ honest parties in the set $\{P_{2t}, \ldots, P_{4t}\}$ that receive the share of the opening of com_j from P_k and output that during their execution. This ensures that the opening of the adversarial commitment will be reconstructed after P_{4t} has completed its execution. □

C Missing Proofs of Section 5

C.1 Proof of Theorem 5

Note that since the adversary corrupts up to t parties and there are $t+1$ instances, there exists an instance where both D_i and D'_i are honest.

First, observe that for this particular instance we prove that before party $3t+4$ the corresponding bivariate polynomial F^i is unknown. This is because at most t parties R_j are corrupt, and therefore only up to t projections of the bivariate polynomial F^i are initially known to the adversary. Furthermore, note that honest receivers R_j do not complain about D_i, since an honest D_i always commits to a degree-t bivariate polynomial F^i, and the points that are sent private are consistent with the commitments. Therefore, any point that is publicly opened by an honest D'_i belongs to a projection that is known to the adversary. Moreover, at the end of the protocol, the client will reconstruct $F^i(0,0)$. This is because there is at least one honest recipient R'_j that holds openings corresponding to $t+1$ projections and that are consistent with the published commitments.

Second, observe that at the start of party $3t+4$ any instance j where either D_j or D'_j were corrupted, and where the instance did not abort (D'_j was not publicly deemed corrupt), has a fixed bivariate polynomial F^j and the client will reconstruct $F^j(0,0)$. This is because the honest receivers hold consistent projections of degree-t with the published commitments. Moreover, by the first point, the value $F^j(0,0)$ was fixed independently of the honest instance F^i, due to the hiding property of the commitments.

From the two points above, since the final coin is computed as the XOR of instances that were successful (where the dealer was not deemed corrupt), and in at least one instance the random

value was chosen by an honest dealer and the adversary's behavior is independent of this random value, the coin has negligible bias.

D Missing Proofs and Protocols from Section 6

D.1 Security Proof of SD-VSS in the Execution-Leaks Model

We now prove the security properties of the execution-leaks version of our SD-VSS from Section 6.1.

Lemma 5 (Correctness)**.** *If the dealer and the resolver are honest, the sharing phase in Protocol 3 does not fail and the reconstructed secret after executing both the sharing and consequently the reconstruction phase is s.*

Proof. First, we show that the sharing phase does not fail. This is because all points signed by the dealer lie on the bivariate polynomial F, and any polynomial broadcasted by the dealer or point signed by the dealer is consistent with F. That is, any message `Complain` with a point and a dealer's signature that is broadcasted by a receiver lies on F, and any polynomial broadcasted by the resolver as a response to a message `ComplainPoly` lies on F as well.

Moreover, consider each i-th honest receiver, for $i \in [1, 3t+1]$. The receiver correctly received a signed degree-t projection, so he did not broadcast any `ComplainPoly` message. Moreover, for at least $2t+1$ points, the receiver sent this point triply signed by the dealer, the j-th receiver and his own signature to all the reconstructors at step $3b$.

Now we show that the reconstructed secret is s. In the reconstruction phase, the $t+1$ reconstructors broadcast all the received points and signatures that have been received, for each receiver. Since there is at least one honest reconstructor, this reconstructor will broadcast points corresponding to the $t+1$ consistent valid projections (with the polynomial F) from honest receivers, uniquely determine the secret s. Moreover, any projection broadcasted by the resolver as a result of a `ComplainPoly` also lies on F. Finally, note that the adversary cannot contribute any projection that is not consistent with F because the projection must be signed by each of the honest receivers. □

Lemma 6 (Privacy)**.** *If the dealer and the resolver are honest, no information about the secret s is revealed before the reconstruction phase is started in Protocol 3.*

Proof. The view of the adversary contains only the projections of F with indices that belong to corrupted receivers. Since there are at most t corrupted receivers, this information is statistically independent of the secret s.

Note that any broadcasted message of the form `ComplainPoly` comes from a corrupted receiver and therefore the resolver broadcasts a projection that was already known to the adversary. Similarly, any broadcasted message `Complain` containing a point $F(i,j)$ at step $3a$ or $3c$,[7] is only broadcasted when either the i-th or the j-th receiver is corrupted (and therefore the point was already known by the adversary). □

Lemma 7 (Verifiability)**.** *At the end of the sharing phase in Protocol 3, if it succeeded, there is only one secret s' that will be reconstructed upon executing the reconstruction phase.*

[7] Note that step $3b$ does not reveal any information, since we consider the execution-leaks model and messages in this step are sent to the reconstructors, which are executed only in the reconstruction phase.

Proof. Let $i, j \in [1, 3t+1]$ with $i < j$. We first show that if the sharing phase succeeds, the projections corresponding to every pair of i-th and j-th honest receivers (taken into account in the Secret Reconstruction phase) are consistent. We divide four cases.

1. Both projections are broadcasted by the resolver as a result of corresponding `ComplainPoly` messages. In this case, since the sharing succeeds, the protocol prescribes that both polynomials need to be consistent.

2. Noone broadcasted a `ComplainPoly` message and therefore no projection is broadcasted by the resolver. In this case, both honest receivers received correctly signed degree-t projections from the dealer. Moreover, the i-th receiver sent the point $f_i(j)$ signed by the dealer and his own signature to the j-th receiver. Since the sharing phase succeeds, we have that both projections are consistent, i.e. $f_i(j) = f_j(i)$, and therefore the point $f_j(i)$ and the signatures from the dealer, the i-th receiver and the j-th receiver are sent to every reconstructor at step $3b$. Moreover, at least one (honest) reconstructor will broadcast this information.

3. Only the i-th receiver broadcasts a `ComplainPoly`, and as a result the resolver broadcasts the projection f_i. In this case, the i-th receiver did not send any message to the j-th receiver at step $2d$, and therefore the j-th receiver broadcasts a message `Complain` with his own point $f_j(i)$. Since the sharing phase succeeds, it must be case that $f_i(j) = f_j(i)$. (Note that $2t+1$ triply signed points of f_j are sent to each of the reconstructors.)

4. Only the j-th receiver broadcasts a `ComplainPoly`, and as a result the resolver broadcasts the projection f_j. In this case, the i-th receiver sent the point $f_i(j)$ signed by the dealer and his own signature to the j-th receiver. The j-th receiver receives this point with correct signatures and broadcasts a message `Complain` with the point $f_i(j)$ signed by the dealer and the i-th receiver. Since the sharing phase succeeds, this point is consistent with the broadcasted polynomial f_j. (Note that $2t+1$ triply signed points of f_i are sent to each of the reconstructors.)

The projections corresponding to honest receivers are sufficient to uniquely define a bivariate polynomial F', which in turn defines a value s'. Furthermore, note that any valid projection for a corrupted receiver is also consistent with F'. Either the projection was broadcasted as a result of `ComplainPoly` (in which case it is consistent with the honest receiver's projections), or it is signed by at least $2t+1$ receivers ($t+1$ are honest, and therefore these uniquely define a consistent projection with F'). □

D.2 Proof of Theorem 7

Proof. The stacking mechanism executes $t+1$ instances of verifiable secret sharing. In order to see that the output coin is uniform distributed, we make the following observations. First, there is at least one instance of verifiable secret sharing in which the respective dealer and resolver are both honest. This is simply because there are $t+1$ SD-VSS instances, with at most t parties being corrupted, and each party in the protocol can execute at most one dealer or resolver role (but not both). Second, by the privacy of SD-VSS that has an honest dealer and resolver (see Lemma 6), no information about his shared value is revealed before any of the reconstructors are executed. Moreover, by the correctness of SD-VSS (see Lemma 5), the sharing phase of this SD-VSS instance is successful (and will be publicly reconstructed by the reconstructors). Third, by the verifiability

property of SD-VSS (see Lemma 7), for each other instance, either the sharing phase failed, or there is a fixed value that will be reconstructed by the reconstructors. Importantly, this value is fixed at the end of the sharing phase, independently of any value that has been distributed in an instance of SD-VSS that has an honest dealer and resolver (since in such an instance the adversary obtains no information, in a statistical sense, about the shared value).

Given that the output coin is computed as the sum of the values that are publicly reconstructed, and one of the values is distributed in an instance of SD-VSS with honest dealer and resolver, the output is uniformly random. □

D.3 Protocol from Non-Interactive Commitments in the Sending-Leaks Model

We formally describe the protocol sending-leaks protocol in Protocol 6. The proof of the corresponding Theorem 6 is analogous to that of the execution-leaks variant (see Appendix C.1), except that the secrecy of the honest dealer does not break during the reconstruction phase as the at most t corrupt reconstructors are not sufficient to recover any information about the degree-t bivariate polynomial. At the same time, we know that there are $(t+1)$ honest pairs (R_i, R'_i) which will publish projections that are consistent with the published commitments and therefore will be enough to reconstruct the bivariate polynomials.

D.4 SD-VSS in the Sending-Leaks Model

We describe our SD-VSS protocol in the sending-leaks model in Protocol 7. Same as in the execution-leaks model, the communication complexity is $O(n^3)$ (excluding any polynomial factors in the security parameter). The resulting protocol can be proven to achieve the same lemma statements.

The following lemmas formally argue that our VSS protocol in the sending-leaks model satisfies *correctness*, *privacy*, and *verifiability*.

Lemma 8 (Correctness)**.** *If the dealer and the resolver are honest, the sharing phase in Protocol 7 does not fail and the reconstructed secret after executing both the sharing and consequently the reconstruction phase is s.*

Proof. First, we show that the sharing phase does not fail. This is because all points signed by the dealer lie on the bivariate polynomial F, and any polynomial broadcasted by the dealer or point signed by the dealer is consistent with F. That is, any message `Complain` with a point and a dealer's signature that is broadcasted by a receiver lies on F, and any polynomial broadcasted by the resolver as a response to a message `ComplainPoly` lies on F as well.

Moreover, consider each i-th honest receiver, for $i \in [1, 3t+1]$. The receiver correctly received a signed degree-t projection, so he did not broadcast any `ComplainPoly` message. Moreover, the receiver sends a signed share of at least $2t+1$ points, each point triply signed by the dealer, the j-th receiver and himself, to each reconstructors at step $3b$.

Now we show that the reconstructed secret is s. In the reconstruction phase, the $t+1$ reconstructors broadcast all the correctly signed shares they received from each receiver. For each receiver, only the messages signed by this receiver are used to reconstruct the values it submitted. As there are at least $t+1$ honest reconstructors, each share of each honest receiver will be reconstructed correctly. The shares of $t+1$ honest receivers (where some of them might have broadcast

the `ComplainPoly`) uniquely determine the secret s. Finally, note that the adversary cannot contribute any projection that is not consistent with F because the projection must be signed by the dealer. □

Lemma 9 (Privacy)**.** *If the dealer and the resolver are honest, no information about the secret s is revealed before the reconstruction phase is started in Protocol 7.*

Proof. The view of the adversary contains only the projections of F with indices that belong to corrupted receivers. Since there are at most t corrupted receivers, this information is statistically independent of the secret s.

Note that any broadcasted message of the form `ComplainPoly` comes from a corrupted receiver and therefore the resolver broadcasts a projection that was already known to the adversary. Similarly, any broadcasted message `Complain` containing a point $F(i, j)$ at step $3a$ or $3c$, is only broadcasted when either the i-th or the j-th receiver is corrupted (and therefore the point was already known by the adversary). Finally, step $3b$ does not reveal any information, as we are using a $(t + 1, 2t + 1)$ secret sharing scheme, and $t + 1$ reconstructors are honest. □

Lemma 10 (Verifiability)**.** *At the end of the sharing phase in Protocol 7, if it succeeded, there is only one secret s' that will be reconstructed upon executing the reconstruction phase.*

Proof. Let $i < j$, $i, j \in [1, 3t + 1]$. We first show that if the sharing phase succeeds, the projections corresponding to every pair of i-th and j-th honest receivers (taken into account in the Secret Reconstruction phase) are consistent. We divide four cases.

1. Both projections are broadcasted by the resolver as a result of corresponding `ComplainPoly` messages. In this case, since the sharing succeeds, the protocol prescribes that both polynomials need to be consistent.

2. No one broadcasted a `ComplainPoly` message and therefore no projection is broadcasted by the resolver. In this case, both honest receivers received correctly signed degree-t projections from the dealer. Moreover, the i-th receiver sent the point $f_i(j)$ signed by the dealer and his own signature to the j-th receiver. Since the sharing phase succeeds, we have that both projections are consistent, i.e. $f_i(j) = f_j(i)$, and therefore the point $f_j(i)$ and the signatures from the dealer, the i-th receiver and the j-th receiver are secret-shared to the reconstructors at step $3b$, and each such share is signed. The $t + 1$ honest reconstructors are sufficient to recover this information, while no adversarial reconstructor can suggest a wrong share for P_j, as the share must be signed.

3. Only the i-th receiver broadcasts a `ComplainPoly`, and as a result the resolver broadcasts the projection f_i. In this case, the i-th receiver did not send any message to the j-th receiver at step $2d$, and therefore the j-th receiver broadcasts a message `Complain` with his own point $f_j(i)$. Since the sharing phase succeeds, it must be case that $f_i(j) = f_j(i)$. (Note that $2t + 1$ triply signed points of f_j are secret-shared to the reconstructors.)

4. Only the j-th receiver broadcasts a `ComplainPoly`, and as a result the resolver broadcasts the projection f_j. In this case, the i-th receiver sent the point $f_i(j)$ signed by the dealer and his own signature to the j-th receiver. The j-th receiver receives this point with correct signatures and broadcasts a message `Complain` with the point $f_i(j)$ signed by the dealer

and the i-th receiver. Since the sharing phase succeeds, this point is consistent with the broadcasted polynomial f_j. (Note that $2t+1$ triply signed points of f_i are secret-shared to the reconstructors.)

The projections corresponding to honest receivers are sufficient to uniquely define a bivariate polynomial F', which in turn defines a value s'. Furthermore, note that any valid projection for a corrupted receiver is also consistent with F'. Either the projection was broadcasted as a result of `ComplainPoly` (in which case it is consistent with the honest receiver's projections), or it is signed by at least $2t+1$ receivers ($t+1$ are honest, and therefore these uniquely define a consistent projection with F'). □

The sharing is considered to have failed if the polynomial output by the resolver does not have degree-t or it is inconsistent with any point broadcasted by a receiver that is correctly signed by the dealer at steps $3a$ or $3c$, or other projection polynomial broadcasted by the resolver at this step.

(Protocol 7) Reconstruction phase:

- If P_i is a reconstructor and the sharing phase did not fail, do the following. Consider the shares $s_{j,k}(i-(3t+3))$ and signatures on these shares $\mathsf{DS.Sign}_{\mathsf{sk}^j}(s_{j,k}(i-(3t+3)))$ by the j-th receiver. If the j-th receiver did not broadcast `ComplainPoly`, broadcast these shares and signatures.
- **Secret Reconstruction:** The secret s can be reconstructed by any party C. First, for each receiver j who did not broadcast `ComplainPoly`, each point $k \in [3t+1]$, C reconstructs the value $s_{j,k}$ using any $t+1$ shares of this value with correct signatures from P_{j+1}. Then, C reconstructs the secret s using valid projections for $t+1$ receivers. A projection for the j-th receiver is considered valid if 1) it was broadcasted by the resolver as a result of a `ComplainPoly` message, or 2) a reconstructor broadcasted triply-signed points $\{f_j(k), \sigma^D_{j,k}, \sigma^k_j, \sigma^j_k\}_k$by the j-th receiver, form a degree-t polynomial and there are at least $2t+1$ points which are all either correctly triply signed, or were contained in one of the `ComplainPoly` messages.

D.5 SD-VSS-based Randomness Generation in the Sending-Leaks Model

We describe our SD-VSS sending-leaks scheme in Protocol 8. As in the execution-leaks version, the communication complexity of the sending-leaks construction is $O(n^4)$ (excluding any polynomial factors in the security parameter).

Our protocol leads to the following result.

Theorem 12. *Assuming the existence of digital signatures, there exists an efficient $(t, n = 6t+3)$-computationally secure PASSO randomness generation protocol in the sending-leaks model.*

Proof. The proof for correctness and verifiability is similar to the argument for the execution-leaks model (see Appendix D.2). The only change is in arguing privacy. Notice that no t reconstructors can recover the secret of an honest dealer as the information is secret shared among the parties of $\mathcal{P}'$ using a $(t+1)$-out of-$(2t+1)$ secret sharing. Therefore, in the worst-case, the adversary only

has access to t shares and therefore a honest secret is information-theoretically hidden from its view. □ □

E Missing Proofs from Section 7

E.1 Proof of Theorem 8

Assume for the sake of contradiction that there exists a protocol which satisfies the conditions stated above. Let $f(x_1, x_2, x_3)$ denote the coin output of the protocol. Now, consider party P_3. As we are in the plain model without setup, P_3 is able to efficiently compute x_3 in its head using the messages sent by the honest P_1 and P_2 and thus compute $f(x_1, x_2, x_3)$. Therefore, P_3 should not be able to change the outcome of the coin too much. More formally, consider sampling honest x_1, x_2, x_3. Then, it must hold that

$$\Pr[f(x_1, x_2, \bot) = 1 - b | f(x_1, x_2, x_3) = b] \leq \frac{4\varepsilon}{1 + 2\varepsilon}$$

for $b \in \{0, 1\}$, where the probability is taken over the honest sampling of x_1, x_2, and x_3. To see this, consider $b = 0$ without loss of generality. If this inequality does not hold, an adversary who corrupts P_3 can efficiently sample x_3 honestly, and, if the result is 0, the adversary outputs $\bot$ for P_3. In this efficient attack, the final coin is 1 with probability at least

$$\left(\frac{1}{2} - \varepsilon\right) + \left(\frac{1}{2} + \varepsilon\right) \cdot \Pr[f(x_1, x_2, \bot) = 1 | f(x_1, x_2, x_3) = 0] > 1/2 + \varepsilon, \tag{1}$$

which contradicts the security of the protocol. From Equation (1), it follows that

$$\Pr[f(x_1, x_2, \bot) \neq f(x_1, x_2, x_3)] \leq 2 \cdot \left(\frac{1}{2} + \varepsilon\right) \cdot \frac{4\varepsilon}{1 + 2\varepsilon} = 4\varepsilon. \tag{2}$$

Due to Equation (2), not only does P_3 have little control over changing the outcome of the coin, but also P_2. This is because P_2 can now efficiently compute the output of the coin by setting P_3's message to $\bot$. Consider the following: Suppose that an honest P_1 outputs a public x_1 and sends private messages $s_{1,2}, s_{1,3}$. Given the messages x_1 and $s_{1,2}$, sample two values x_2^1, x_2^2, along with the corresponding private messages $s_{2,3}^1$ and $s_{2,3}^2$ (each in an honest way). Sample x_3^1 (resp. x_3^2) using $x_1, x_2^1, s_{1,3}, s_{2,3}^1$ (resp. $x_1, x_2^2, s_{1,3}, s_{2,3}^2$) in an honest way. Combining Equation (2) with a union bound over the two simulated runs shows that

$$\Pr[f(x_1, x_2^1, x_3^1) = f(x_1, x_2^1, \bot), f(x_1, x_2^2, x_3^2) = f(x_1, x_2^2, \bot)] \geq 1 - 8\varepsilon.$$

Furthermore, it must hold that

$$\Pr[f(x_1, x_2^1, x_3^1) = f(x_1, x_2^1, \bot), f(x_1, x_2^1, x_3^1) = f(x_1, x_2^2, \bot), f(x_1, x_2^1, \bot) \neq f(x_1, x_2^2, \bot)] \leq \varepsilon. \tag{3}$$

Otherwise, an adversary corrupting P_2 would be able to efficiently bias by sampling two messages x_2^1, x_2^2, each in an honest way, computing $f(x_1, x_2^1, \bot)$ and $f(x_1, x_2^2, \bot)$, and picking the one that corresponds to final coin 1. By combining Equations (2) and (3) with a union bound, we conclude that

$$\Pr[f(x_1, x_2^1, x_3^1) = f(x_1, x_2^1, \bot), f(x_1, x_2^1, x_3^1) = f(x_1, x_2^2, \bot), f(x_1, x_2^1, \bot) = f(x_1, x_2^2, \bot)] \geq 1 - 9\varepsilon.$$

This means that party P_1 fully determines the output of the protocol with probability at least $1 - 9\varepsilon$. In this case, P_1 also can also efficiently compute the value of the coin. Thus, an adversary corrupting P_1 can conduct the following efficient attack: Sample two sets of messages $x_1^1, s_{1,2}^1, s_{1,3}^1$ and $x_1^2, s_{1,2}^2, s_{1,3}^2$, compute the coin by emulating P_2 and setting P_3's message to $\perp$. If either of the coins is 1, output the corresponding set $x_1^i, s_{1,2}^i, s_{1,3}^i$, where $i \in \{1, 2\}$. As these are two independent runs of the protocol and we assume (for the sake of contradiction) that the protocol has bias at most ε, with probability at least $\frac{1}{2} - \varepsilon^2$ we get that the two runs result in a different coin. Thus, P_1's attack succeeds with probability at least $(\frac{1}{2} - \varepsilon^2) - 2 \cdot 9\varepsilon > 0.01 \geq \varepsilon$, which leads to a contradiction.

E.2 Proof of Theorem 9

For the sake of contradiction, say that such a protocol Π exists. We will show that given such a protocol, it is possible to obtain a PASSO protocol Π' for $n' = 3$ parties and $t' = 1$ corruptions, which would contradict Theorem 8.

We start by splitting the n parties into three groups $P^1 := \{P_1, \ldots, P_t\}$, $P^2 := \{P_{t+1}, \ldots, P_{2t}\}$, and $P^3 := \{P_{2t+1}, \ldots, P_{3t}\}$. Now, in the protocol Π' for $n' = 3$ parties P_1', P_2', P_3', we have the first party P_1' do the following: locally execute the protocol Π for each of the parties P_1 up to P_t one after the other, publish a concatenation of the corresponding public messages $x_1, \ldots, x_t$, and forward private messages that are to be received by parties in P^2 to the party P_2' (similarly, forward private messages that are to be received by parties in P^3 to the party P_3'). Party P_2' then similarly executes the protocol Π for each of the parties P_{t+1} up to P_{2t}, while publishing the concatenation of the messages $x_{t+1}, \ldots, x_{2t}$, and forwarding private messages designated for parties in P^3 to P_3'. Finally, party P_3' executes the protocol Π for each of the parties P_{2t+1} up to P_{3t}, while publishing the concatenation of the messages $x_{2t+1}, \ldots, x_{3t}$. Note that if protocol Π is secure, protocol Π' is secure as well: Corrupting a party in Π' corresponds to corrupting a block of parties in Π (crucially, the adversary obtains *all* private messages designated for a corrupt party in Π' before starting to execute this party, as in the sending-leaks model the adversary obtains the messages designated to the corrupt parties by the time they are sent). As each block is of size t and Π is secure for t corruptions, we get that Π' is a secure protocol for $n' = 3$ parties and $t' = 1$ corruptions, thus contradicting Theorem 8.

E.3 Proof of Theorem 10

Towards a contradiction, suppose that there exists such a protocol. Let f be the deterministic polynomial-time function which computes the coin based on the public broadcasts from the protocol. In other words, if $X_1, X_2, \ldots, X_6$ are published by parties $P_1, P_2, \ldots, P_6$, respectively, then the coin value is $f(X_1, X_2, \ldots, X_6)$. From this assumption, we know that

$$|\Pr[f(X_1, \ldots, X_6) = 0] - \Pr[f(X_1, \ldots, X_6) = 1]| \leq \mathsf{negl}(\lambda)$$

for any PPT execution-leaks adversary that corrupts at most $t = 2$ parties.

We begin by showing that we may assume that if the public values $X_1, \ldots, X_4$ of parties $P_1, \ldots, P_4$, respectively, are sampled honestly according to the protocol, then P_5 and P_6 both publish $\perp$. We corrupt P_5 and P_6 and analyze two PPT attacks. First, consider an attack where P_5 behaves honestly (publishing X_5) and P_6 generates its potential public value X_6 honestly. If $f(X_1, \ldots, X_6) = 1$ (which P_6 can compute), then P_6 outputs X_6. Otherwise, P_6 outputs $\perp$. Since

the protocol is secure, the following holds except with negligible probability

$$f(X_1, \ldots, X_5, X_6) = f(X_1, \ldots, X_5, \perp) \ . \tag{4}$$

Next, consider an attack where P_5 first generates its potential public value X_5 honestly. From Equation (4), we conclude that P_5 can efficiently predict the value the coin will take if he decides to publish X_5, except with negligible probability, by computing $f(X_1, \ldots, X_5, \perp)$ locally. If $f(X_1, \ldots, X_5, \perp) = 1$, then P_5 publishes X_5 and P_6 publishes $\perp$. Otherwise, P_5 and P_6 both publish $\perp$. Again by the security of the protocol against $t = 2$ corruptions, we must have the following, except with negligible probability

$$f(X_1, \ldots, X_4, \perp, \perp) = f(X_1, \ldots, X_4, X_5, X_6) \ . \tag{5}$$

We now corrupt P_1 and P_4 with the aim of arguing that the private message $s_{1,4}$ from P_1 to P_4 can be taken to be $s_{1,4} = \perp$, so long as P_1 samples its public value and the private messages to P_2 and P_3 honestly. Consider the attack where P_1 behaves honestly except that it sends $s_{1,5} = s_{1,6} = \perp$ and P_4 behaves as follows: First, P_4 samples X_4 honestly and X'_4 honestly *but assuming that the private message* $s_{1,4} = \perp$. From Equation (5), if P_4 publishes X_4 then the value of the coin will be $f(X_1, \ldots, X_4, \perp, \perp)$, which P_4 can efficiently compute locally, except with negligible probability. If $f(X_1, \ldots, X_4, \perp, \perp) = 1$, then P_4 publishes X_4. Otherwise, P_4 publishes X'_4. By the security of the protocol, we must have

$$f(X_1, \ldots, X_4, X_5, X_6) = f(X_1, X_2, X_3, X'_4, X'_5, X'_6) \tag{6}$$

except with negligible probability, where X'_5 and X'_6 are sampled honestly based on X'_4 (and on private messages $s_{1,5} = s_{1,6} = \perp$).

Next, corrupt P_2 and P_3 and consider the following attack. Party P_2 behaves honestly, except that it also forwards the private messages $s_{2,4}, s_{2,5}, s_{2,6}$, which it sent to P_4, P_5, P_6, respectively, to P_3. By Equation (6), P_3 can efficiently predict the final coin if he publishes X_3 except with negligible probability because (1) P_3 can efficiently sample P_4's public value X'_4 and private messages $s_{4,5}$ and $s_{4,6}$ based on X_1, X_2, X_3 and the private messages $s_{1,4} = \perp$ and $s_{2,4}$, and (2) P_3 can efficiently sample X'_5 and X'_6 based on X_1, X_2, X_3, X'_4 and the private messages $s_{1,5} = s_{1,6} = \perp$, $s_{2,5}, s_{2,6}$, and $s_{4,5}, s_{4,6}$. Party P_3 runs two independent honest samplings of X_3 and the messages $s_{3,i}$ for $i > 3$ – denote them by $X^j_3, (s^j_{3,i})_{i>3}$ for $j \in \{1, 2\}$. If P_3 predicts coin value 1 when publishing X^1_3 and sending $s^1_{3,i}$ for $i > 3$, it publishes this value and sends these messages to the corresponding later parties. Otherwise, P_3 publishes X^2_3 and sends $s^2_{3,i}$ for $i > 3$. The security of the protocol then implies that

$$f(X_1, X_2, X^1_3, X^1_4, X^1_5, X^1_6) = f(X_1, X_2, X^2_3, X^2_4, X^2_5, X^2_6) \tag{7}$$

except with negligible probability, where X^j_4, X^j_5, X^j_6 are generated honestly conditioned on X^j_3 for $j \in \{1, 2\}$.

Finally, we corrupt P_1 and P_2. The attack proceeds as follows: Party P_1 samples potential public values X^j_1, X^j_2 and potential private messages $(s^j_{1,i})_{i>1}$ and $(s^j_{2,i})_{i>2}$, for $j \in \{1, 2\}$. By Equation (7), P_1 can efficiently predict the value of the coin in the j-th run of the protocol based solely on the samples above, assuming that P_2 indeed publishes X^j_2 and sends private messages $(s^j_{2,i})_{i>2}$. Therefore, P_1 can check whether the j-th run leads to coin value 1, publish X^j_1, send private messages $(s^j_{1,i})_{i>1}$, and additionally tell P_2 to publish X^j_2 and send private messages $(s^j_{2,i})_{i>2}$.

Since the two protocol runs are honest and independent, the correctness of the protocol ensures that there is a run leading to coin value 1 with probability at least $3/4 - \mathsf{negl}(\lambda)$, and, as we argued above, P_1 can predict which run has this property.

Protocol 2 Ex. Leaks Rand. Gen. from any Non-Interactive Commitment

Sharing phase:

Each D_i, $i \in [t+1]$ does the following:

1. D_i chooses a random bivariate degree-t (on both variables) polynomial $F^i(x, y)$.
2. D_i commits to each point of $F^i(x, y)$ via $\mathsf{com}_{x,y} \leftarrow \mathsf{Commit}(F^i(x, y); r_{x,y})$ for random coins $r_{x,y}$, and publicly broadcasts $\mathsf{com}_{x,y}$ for each $x, y \in [1, 2t+1]$.
3. D_i sends the opening information of the x-th horizontal and vertical projections points: $\{(F^i(x, y), r_{x,y})\}_{y\in[2t+1]}$ and $\{(F^i(y, x), r_{y,x})\}_{y\in[2t+1]}$ to each P_x, $x \in [2t+1]$; and sends all opening information to D'_i.

Each R_i, $i \in [2t+1]$ does the following:

1. For each dealer D_j, R_i checks whether the received openings against the published commitments, and also that the received projections are of degree-t.
2. R_i sends its private state to every R'_j.

Each D'_i, $i \in [t+1]$ broadcasts openings of all points corresponding to parties who complained about D_i. If an opening is inconsistent with its commitment, or openings corresponding to a projection don't form a degree-t polynomial, D'_i is deemed corrupt.

Reconstruction phase:

Each R'_i, $i \in [t+1]$ does the following:

1. If R_i complained about D_j, and D'_j was not deemed corrupt, R'_i sets the points of the i-th projections to the values broadcasted by D'_j.
2. R'_i outputs all points obtained for non-corrupt dealers.

Client C does the following to compute the coin:

1. For each D'_i who was not deemed corrupt, C uses any $t+1$ projections that pass the verification check against the corresponding published commitments to reconstruct a bivariate polynomial F^i. Let the value $s_i = F^i(0, 0)$.
2. Let H denote the index set of dealers D'_i which were not deemed corrupt. C outputs $\bigoplus_{i\in H} s_i$.

Protocol 3 Split-Dealer Verifiable Secret Sharing from Digital Signatures in the Execution-Leaks Model.

The protocol is described from party P_i's view, for some $i \in [1, 4t+4]$. We call P_1 the dealer, $P_2, \ldots, P_{3t+2}$ the receivers, P_{3t+3} the resolver and $P_{3t+4}, \ldots, P_{4t+4}$ the reconstructors. Let s be the secret input of the dealer P_1. Let $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ denote a digital signatures scheme.

Sharing phase:

- If P_i is the dealer (i.e., $i = 1$):
  1. Sample a symmetric polynomial $F(x, y)$ such that $F(0, 0) = s$.
  2. Sample $\left(\mathsf{vk}^D, \mathsf{sk}^D\right) \leftarrow \mathsf{DS.KGen}(1^\lambda)$. Publicly broadcast the public key vk^D. Also privately send the polynomial F to the resolver (party P_{3t+3}).
  3. Let $f_j := F(j, y)$, $j \in [3t+1]$, be the j-th vertical projection of F. Privately send to the j-th receiver all points of f_j signed, i.e. send to party P_{j+1} the pairs $\{f_j(y), \sigma^D_{j,y}\}_{y \in [3t+1]}$, where $\sigma^D_{j,y} = \mathsf{DS.Sign}_{\mathsf{sk}^D}(f_j(y))$.
- If P_i is the j-th receiver (i.e. $i = j + 1$ and $j \in [3t+1]$):
  1. Sample $\left(\mathsf{vk}^j, \mathsf{sk}^j\right) \leftarrow \mathsf{DS.KGen}(1^\lambda)$ and publicly broadcast the public key vk^j.
  2. For the messages received from the dealer (P_1), $\{f_j(y), \sigma^D_{j,y}\}_{y \in [3t+1]}$:
     (a) Check that the signatures are correct according to the broadcasted public key vk^D, i.e. $\mathsf{DS.Vf}_{\mathsf{vk}^D}(f_j(y), \sigma^D_{j,y}) = 1$ for each point.
     (b) Check that the points $\{f_j(y)\}_{y \in [3t+1]}$ lie on a degree-t polynomial.
     (c) If any check fails, publicly broadcast a message `ComplainPoly`.
     (d) Otherwise, privately send the k-th evaluation point signed by the dealer and his own key to the k-th receiver, for each remaining future receiver. That is, for each receiver $j < k < 3t+2$, send $(f_j(k), \sigma^D_{j,k}, \sigma^j_k)$ to the k-th receiver, where $\sigma^j_k = \mathsf{DS.Sign}_{\mathsf{sk}^j}(f_j(k))$.
  3. For the message $(f_k(j), \sigma^D_{k,j}, \sigma^k_j)$ received from the k-th receiver (i.e., P_{k+1}) with $k < j$: check that the signatures are valid, i.e. $\mathsf{DS.Vf}_{\mathsf{sk}^D}(f_k(j)) = \mathsf{DS.Vf}_{\mathsf{sk}^k}(f_k(j)) = 1$. We now divide three cases:
     (a) If the received signatures are incorrect (or no message was received) and no message `ComplainPoly` is being broadcasted at step $2b$, broadcast a complaint message with the received point from the dealer and its signature: $(\texttt{Complain}, k, f_j(k), \sigma^D_{j,k})$.
     (b) If the received signatures are correct, no message `ComplainPoly` is being broadcasted at step $2b$, do the following. If the points are not consistent $f_k(j) \neq f_j(k)$ broadcast a complaint with both points signed by the dealer: $(\texttt{Complain}, k, f_k(j), \sigma^D_{k,j}, f_j(k), \sigma^D_{j,k})$ and the sharing phase is failed. Otherwise, if the points are consistent, send privately the point with the signatures from the dealer, the k-th receiver and the j-th receiver $(f_j(k), \sigma^D_{j,k}, \sigma^k_j, \sigma^j_k)$ to every reconstructor (parties $P_{3t+4}, \ldots, P_{4t+4}$).
     (c) If the received signatures are correct but message `ComplainPoly` is being broadcasted at step $2b$, broadcast a complaint message with the received point from the P_{k+1} and both signatures: $(\texttt{Complain}, k, f_k(j), \sigma^D_{k,j}, \sigma^k_j)$.
- If P_i is the resolver: For each message `ComplainPoly` from the j-th receiver, publicly broadcast the projection polynomial f_j.

The sharing is considered to have failed if the polynomial output by the resolver does not have degree-t or it is inconsistent with any point broadcasted by a receiver that is correctly signed by the dealer at steps $3a$ or $3c$, or other projection polynomial broadcasted by the resolver at this

(Protocol 3) Reconstruction phase:

- If P_i is a reconstructor and the sharing phase did not fail, do the following. Consider the triply-signed points $\{f_j(k), \sigma^D_{j,k}, \sigma^k_j, \sigma^j_k\}_k$ by the j-th receiver. If the j-th receiver did not broadcast `ComplainPoly`, broadcast all the received the triply-signed points.
- **Secret Reconstruction:** The secret s can be reconstructed by any party that takes any valid projections for $t+1$ receivers. A projection for the j-th receiver is considered valid if 1) it was broadcasted by the resolver as a result of a `ComplainPoly` message, or 2) a reconstructor broadcasted triply-signed points $\{f_j(k), \sigma^D_{j,k}, \sigma^k_j, \sigma^j_k\}_k$by the j-th receiver, form a degree-t polynomial and there are at least $2t+1$ points which are all either correctly triply signed, or were contained in one of the `ComplainPoly` messages.

Protocol 4 Randomness Generation from PASSO SD-VSS, Execution-Leaks.

The first $t+1$ parties are *dealers*, parties P_i for $i \in [2, 4t+2]$ are *receivers*, P_i for $i \in [3t+3, 4t+3]$ are *resolvers*, and P_i for $i \in [4t+4, 5t+4]$ are called *reconstructors*.

We pipeline $t+1$ instances of SD-VSS given in Protocol 3, where i-th instance starts with party P_i. In the i-th instance of the execution-leaks SD-VSS:

- For $k = i$, party P_k plays the role of the dealer D of Protocol 3.
- For $k \in [i+1, 3t+i+1]$, party P_k plays the role of the $(k-i)$-th receiver of Protocol 3.
- For $k = 3t+2+i$, party P_k plays the role of the resolver of Protocol 3.
- For $k \in [4t+4, 5t+4]$, party P_k plays the role of $(k-4t-3)$-th reconstructor of Protocol 3.

Protocol 5 Randomness Generation using $n = 4t$ roles in the Sending-Leaks Model.

We have a t-sharing matrix $M \in \{0,1\}^{m \times \ell}$ according to Lemma 2, where $\ell = 2t-1$ and $m = \binom{2t-1}{t}$.

1. For $i \in [2t-1]$:
   (a) For $j \in [m]$, party P_i does the following:
      i. If $\forall k \in [i-1], M_{jk} = 0$ and $M_{ji} = 1$,
         A. Choose value $s_j \leftarrow\!\!\$\ \{0,1\}^{\ell_m(\lambda)}$ and generate $\mathsf{com}_j \leftarrow \mathsf{Commit}(s_j; r_j)$ for random coins $r_j \in \{0,1\}^{\ell_r(\lambda)}$.
         B. Broadcast com_j and send the opening (s_j, r_j) to all parties P_k where $k \in [i+1, \ell]$ and $M_{jk} = 1$.
         C. It generates a $t+1$-out of-$2t+1$ sharing of (s_j, r_j) and send the k-th share privately to party P_k where $k \in [2t, 4t]$.
      ii. Else if $M_{ji} = 1$,
         A. Receive (s_j, r_j) from party P_k for some $k \in [i-1]$ and $M_{jk} = 1$.
         B. Broadcast $(\texttt{Complain}, j)$ if nothing was received or if $\mathsf{Commit}(s_j; r_j) \neq \mathsf{com}_j$.
         C. Else, generate a $t+1$-out of-$2t+1$ sharing of (s_j, r_j) and send the k-th share privately to party P_k where $k \in [2t, 4t]$.
2. Party P_{2t} samples $s^* \leftarrow\!\!\$\ \{0,1\}^{\ell_m(\lambda)}$ and broadcasts s^*.
3. For $i \in [2t, 4t]$, party P_i does the following:
   (a) For any $j \in [m]$, if no message $(\texttt{Complain}, j)$ was seen, then receive shares from different parties for the opening of com_j and output the shares.

Let $C \subseteq [m]$ be the set of indices such that for any $j \in C$, no message $(\texttt{Complain}, j)$ was seen and there are $t+1$ shares output by the parties $P_{2t}, \ldots, P_{4t}$ that reconstruct the opening (s_j, r_j) such that $\mathsf{Commit}(s_j; r_j) = \mathsf{com}_j$. Let the final randomness be set as $s = \bigoplus_{j \in C} s_j \oplus s^*$.

Protocol 6 Sending Leaks Randomness Generation from any Non-Interactive Commitment

Sharing phase:

Each D_i, $i \in [t+1]$ does the following:

1. D_i chooses a random bivariate degree-t (on both variables) polynomial $F^i(x, y)$.
2. D_i commits to each point of $F^i(x, y)$ via $\mathsf{com}_{x,y} \leftarrow \mathsf{Commit}(F^i(x, y); r_{x,y})$ for random coins $r_{x,y}$, and publicly broadcasts $\mathsf{com}_{x,y}$ for each $x, y \in [1, 2t+1]$.
3. D_i sends the opening information of the x-th horizontal and vertical projections points: $\{(F^i(x, y), r_{x,y})\}_{y\in[2t+1]}$ and $\{(F^i(y, x), r_{y,x})\}_{y\in[2t+1]}$ to each P_x, $x \in [2t+1]$; and sends all opening information to D'_i.

Each R_i, $i \in [2t+1]$ does the following:

1. For each dealer D_j, R_i checks whether the received openings against the published commitments, and also that the received projections are of degree-t.
2. R_i sends its private state to its counterpart R'_i.

Each D'_i, $i \in [t+1]$ broadcasts openings of all points corresponding to parties who complained about D_i. If an opening is inconsistent with its commitment, or openings corresponding to a projection don't form a degree-t polynomial, D'_i is deemed corrupt.

Reconstruction phase:

Each R'_i, $i \in [2t+1]$ does the following:

1. If R_i complained about D_j, and D'_j was not deemed corrupt, R'_i sets the points of the i-th projections to the values broadcasted by D'_j.
2. R'_i outputs all points obtained for non-corrupt dealers.

Client C does the following to compute the coin:

1. For each D'_i who was not deemed corrupt, C uses any $t+1$ projections that pass the verification check against the corresponding published commitments to reconstruct a bivariate polynomial F^i. Let the value $s_i = F^i(0, 0)$.
2. Let H denote the index set of dealers D'_i which were not deemed corrupt. C outputs $\bigoplus_{i\in H} s_i$.

Protocol 7 Split-Dealer Verifiable Secret Sharing from Digital Signatures in the Sending-Leaks Model.

The protocol is described from the point of view of P_i, for some $i \in [1, 5t+4]$. We call P_1 the dealer, $P_2, \ldots, P_{3t+2}$ the receivers, P_{3t+3} the resolver and $P_{3t+4}, \ldots, P_{5t+4}$ the reconstructors. Let s be the secret input of the dealer P_1. Let $\mathsf{DS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ denote a digital signatures scheme.

Sharing phase:

- If P_i is the dealer (i.e., $i = 1$):
  1. Sample a symmetric polynomial $F(x, y)$ such that $F(0, 0) = s$.
  2. Sample $\left(\mathsf{vk}^D, \mathsf{sk}^D\right) \leftarrow \mathsf{DS.KGen}(1^\lambda)$. Publicly broadcast the public key vk^D. Also privately send the polynomial F to the resolver (party P_{3t+3}).
  3. Let $f_j := F(j, y)$, $j \in [3t+1]$, be the j-th vertical projection of F. Privately send to the j-th receiver all points of f_j signed, i.e. send to P_{j+1} the pairs $\{f_j(y), \sigma^D_{j,y}\}_{y \in [3t+1]}$, where $\sigma^D_{j,y} = \mathsf{DS.Sign}_{\mathsf{sk}^D}(f_j(y))$.
- If P_i is the j-th receiver (i.e. $i = j+1$ and $j \in [3t+1]$):
  1. Sample $\left(\mathsf{vk}^j, \mathsf{sk}^j\right) \leftarrow \mathsf{DS.KGen}(1^\lambda)$ and publicly broadcast the public key vk^j.
  2. For the messages received from the dealer (party P_1), $\{f_j(y), \sigma^D_{j,y}\}_{y \in [3t+1]}$, do the following:
     (a) Check that the signatures are correct according to the broadcasted public key vk^D, i.e. $\mathsf{DS.Vf}_{\mathsf{vk}^D}(f_j(y), \sigma^D_{j,y}) = 1$ for each point.
     (b) Check that the points $\{f_j(y)\}_{y \in [3t+1]}$ lie on a degree-t polynomial.
     (c) If any check fails, publicly broadcast a message `ComplainPoly`.
     (d) Otherwise, privately send the k-th evaluation point signed by the dealer and his own key to the k-th receiver, for each remaining future receiver. That is, for each receiver $j < k < 3t+2$, send $(f_j(k), \sigma^D_{j,k}, \sigma^j_k)$ to the k-th receiver, where $\sigma^j_k = \mathsf{DS.Sign}_{\mathsf{sk}^j}(f_j(k))$.
  3. For the message $(f_k(j), \sigma^D_{k,j}, \sigma^k_j)$ received from the k-th receiver (party P_{k+1}) with $k < j$: check that the signatures are valid, i.e. $\mathsf{DS.Vf}_{\mathsf{sk}^D}(f_k(j)) = \mathsf{DS.Vf}_{\mathsf{sk}^k}(f_k(j)) = 1$. We now divide three cases:
     (a) If the received signatures are incorrect (or no message was received) and no message `ComplainPoly` is being broadcasted at step $2b$, broadcast a complaint message with the received point from the dealer and its signature: $(\texttt{Complain}, k, f_j(k), \sigma^D_{j,k})$.
     (b) If the received signatures are correct, no message `ComplainPoly` is being broadcasted at step $2b$, do the following. If the points are not consistent $f_k(j) \neq f_j(k)$ broadcast a complaint with both points signed by the dealer: $(\texttt{Complain}, k, f_k(j), \sigma^D_{k,j}, f_j(k), \sigma^D_{j,k})$ and the sharing phase is failed. Otherwise, if the points are consistent, let $s_{j,k}$ denote the point with the signatures from the dealer, the k-th receiver and the j-th receiver: $(f_j(k) = s_{j,k}, \sigma^D_{j,k}, \sigma^k_j, \sigma^j_k)$. Share $s_{j,k}$ using $(t+1, 2t+1)$ Shamir's secret sharing, let $s_{j,k}(m)$ denote the m-th share. Send the share $s_{j,k}(m)$, along with its own signature on it $\mathsf{DS.Sign}_{\mathsf{sk}^j}(s_{j,k}(m))$ to the reconstructor P_{3t+3+m}, for $m \in [2t+1]$.
     (c) If the received signatures are correct but message `ComplainPoly` is being broadcasted at step $2b$, broadcast a complaint message with the received point from the P_k and both signatures: $(\texttt{Complain}, k, f_k(j), \sigma^D_{k,j}, \sigma^k_j)$.
- If P_i is the resolver: For each message `ComplainPoly` from the j-th receiver, publicly broadcast the projection polynomial f_j.

Protocol 8 Randomness Generation from PASSO SD-VSS in the Sending-Leaks Model.

The first $t+1$ parties are the *dealers*, parties P_i for $i \in [2, 4t+2]$ are *receivers*, parties P_i for $i \in [3t+3, 4t+3]$ are *resolvers* and parties P_i for $i \in [4t+4, 6t+4]$ are called *reconstructors*. We pipeline $t+1$ instances of SD-VSS given in Protocol 7, where i-th instance starts with party P_i. In the i-th instance of the execution-leaks SD-VSS:

- For $k = i$, party P_k plays the role of the dealer D of Protocol 7.
- For $k \in [i+1, 3t+i+1]$, party P_k plays the role of the $(k-i)$-th receiver of Protocol 7.
- For $k = 3t+2+i$, party P_k plays the role of the resolver of Protocol 7.
- For $k \in [4t+4, 6t+4]$, party P_k plays the role of $(k-4t-3)$-th reconstructor of Protocol 7.