Distributing Keys and Random Secrets with Constant Complexity*

Benny Applebaum† Benny Pinkas‡

Abstract

In the *Distributed Secret Sharing Generation* (DSG) problem n parties wish to obliviously sample a secret-sharing of a random value s taken from some finite field, without letting any of the parties learn s. *Distributed Key Generation* (DKG) is a closely related variant of the problem in which, in addition to their private shares, the parties also generate a public "commitment" g^s to the secret. Both DSG and DKG are central primitives in the domain of secure multiparty computation and threshold cryptography.

In this paper, we study the communication complexity of DSG and DKG. Motivated by large-scale cryptocurrency and blockchain applications, we ask whether it is possible to obtain protocols in which the communication per party is a constant that does not grow with the number of parties. We answer this question to the affirmative in a model where broadcast communication is implemented via a public bulletin board (e.g., a ledger). Specifically, we present a constant-round DSG/DKG protocol in which the number of bits that each party sends/receives from the public bulletin board is a constant that depends only on the security parameter and the field size but does not grow with the number of parties n. In contrast, in all existing solutions at least some of the parties send $\Omega(n)$ bits.

Our protocol works in the near-threshold setting. Given arbitrary privacy/correctness parameters $0 < \tau_p < \tau_c < 1$, the protocol tolerates up to $\tau_p n$ actively corrupted parties and delivers shares of a random secret according to some $\tau_p n$-private $\tau_c n$-correct secret sharing scheme, such that the adversary cannot bias the secret or learn anything about it. The protocol is based on non-interactive zero-knowledge proofs, non-interactive commitments and a novel secret-sharing scheme with special robustness properties that is based on Low-Density Parity-Check codes. As a secondary contribution, we extend the formal MPC-based treatment of DKG/DSG, and study new aspects of Affine Secret Sharing Schemes.

1 Introduction

Consider the following secure multiparty computation problem: n parties wish to obliviously sample a secret sharing of a random value s taken from some finite field $\mathbb{F}$ without letting any of

*This is the full version of a a paper that will appear in the proceedings of TCC'24 conference.

†Tel-Aviv University, Israel bennyap@post.tau.ac.il, Supported by ISF grant no. 2805/21 and by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

‡Aptos Labs and Bar Ilan University benny@pinkas.net

the parties learn the value of s. Roughly speaking, given a privacy threshold and a correctness threshold $t_p < t_c$, the protocol must ensure that any adversary $\mathscr{A}$ that actively (aka maliciously) corrupts up to t_p parties learns nothing about the secret s and cannot bias it, and that any set of t_c honest parties can recover the secret even in the presence of $\mathscr{A}$.

This task, hereafter referred to as *Distributed Secret Sharing Generation* (DSG), can be viewed as a natural extension of Verifiable Secret Sharing (VSS), with the difference being that in DSG the secret is obliviously sampled, whereas in VSS it is chosen by a designated dealer. DSG plays an important role in many secure multiparty computation protocols (MPC), especially in the online/offline setting. It is also closely related to the problem of *Distributed Key Generation* (DKG), in which, at the end of the sharing phase, the protocol publishes a commitment to the secret (e.g., the value g^s where g is a generator of a cyclic group of appropriate order).[1] (See, e.g. [44, 29, 14, 25, 28, 32, 49, 34, 37, 16, 40].) DKG protocols are typically employed to distribute a private key s and publish a corresponding public key for a threshold signature scheme or cryptosystem (e.g., ElGamal, ECDSA, Schnorr, and BLS). The rise of digital currencies and proof-of-stake blockchains have lead to the deployment of DSG protocols for large scale systems with hundreds and even thousands of users [33, 19]. As a result, an extensive body of research is currently devoted to the study of DSG protocols and their complexity. There are also many real-world implementations of DKG, e.g. [47, 46, 43, 21, 20, 50, 19, 4].

The communication complexity of DSG and DKG. In this work we study the communication complexity of DSG and DKG. Motivated by recent applications, we assume that the vast majority of the communication is performed via a public ledger. That is, in the distribution protocol parties write messages on a public bulletin board (BB) and read messages from the board, but cannot erase anything. The communication to the BB is non-anonymous and authenticated (e.g., via digital signatures and PKI). We also allow parties to send messages via private authenticated channels though such channels can be emulated over the public board assuming public-key encryption. Qualitatively, the BB can be simply viewed as an implementation of a broadcast channel. However it allows for a refined communication complexity measure. Specifically, we define the *upstream complexity* of a party as the number of bits that the party sends either to the BB or via private channels, the *downstream complexity* as the number of bits that a party reads from the BB or receives via private channels, and refer to the sum of the upstream complexity and downstream complexity as the *communication complexity* of the party. The *total communication complexity* is the sum of the communication complexity of all parties. For example, if Alice publishes 10 bits on the BB and Bob reads only the first 2 bits then Alice's communication complexity is 10 but Bob's communication complexity is only 2. When measuring the communication complexity of DSG and DKG we will focus on the distribution phase and ignore the communication complexity of the task of recovering the secret. Indeed, in all standard protocols (including the ones in this paper) one can release the secret s (or a committed version of it) to a receiver R via a single-round protocol with a linear communication complexity. If only a few parties need to recover the secret, the reduction in the communication is effective. For example, in the setting of threshold signatures, where a single client receives a signature on a document, only the client has to read $O(n)$ symbols from the BB

[1] There are several definitions and variants of DSG and DKG. In particular, sometimes the protocol is required to generate public commitments to the private shares, and, typically, one should be able to securely recover the secret in the exponent of a given group element h. To simplify the exposition, we postpone the formal definition, but mention for now that our protocols support these features.

whereas the n servers only have constant communication.

Existing solutions. The most common approach is to reduce the DSG problem to n parallel calls to verifiable secret sharing (VSS), where in each call a different party P_j deals shares of a random secret according to some linear secret sharing scheme (e.g., Shamir [48]), and where the final shares are defined by locally summing-up the received shares. One can optimize the protocol a little bit by using only $t_p + 1$ dealers. Assuming that t_p, t_c are both linear in n (which will be taken to be our default setting), the total communication complexity of this protocol is about $\Omega(n^2)$ field elements since $\Omega(n)$ parties must each communicate at least $\Omega(n)$ field elements (even if the protocol is only passively secure). General MPC solutions for computing the randomized sharing functionality also lead to a similar communication cost. One can slightly improve the communication by running the protocol over a small super-logarithmic size sub-committee that is chosen at random via collective random coin-tossing. Ignoring the cost of the coin-tossing protocol, this reduces the total communication to $\omega(n \log n)$ field elements although the communication overhead is unbalanced and some parties still have a communication complexity of $\Theta(n)$.[2] Finally, we mention that if many instances of DSG are needed then one can amortize the communication cost to $O(1)$ per instance (e.g., via the use of hyper-invertible matrices [6]). This approach is typically useful for MPC applications, but is less useful in the DKG setting when the protocol is being used to set-up a single private key. Our main goal is to understand whether a constant cost can be established in a non-amortized setting. That is, we ask:

> How does the communication complexity of DSG and DKG protocols scale with the number of parties n? Specifically, is it possible to design a protocol where the communication complexity of each party is a constant that is independent of n?

The information bottleneck. All existing protocols suffer from the following "information bottleneck" which affects their communication overhead: There are some (typically, $\Omega(n)$) parties whose input influences the outputs of $\Omega(n)$ parties. However, in a constant-round protocol with a constant communication overhead per party each party can only affect the output of a constant number of parties.

2 Our Results

We show that n parties can obliviously sample a secret sharing for a random secret with constant communication complexity per party. Formally, we prove the following theorem.

Theorem 2.1 (main theorem)**.** *Assuming the existence of NIZKs the following holds. For any constants $0 < \tau_p < \tau_c < 1$ and every field $\mathbb{F}$ of size super-polynomial in the number of parties, there exists a constant-round n-party DSG (resp., DKG) protocol over $\mathbb{F}$ with privacy threshold of $\tau_p n$ and recovery threshold of $\tau_c n$ such that each party sends and receives only a constant number of field elements and a constant number of commitments and NIZK proofs for constant-size relations. Moreover, each party computes only a constant number of arithmetic operations and cryptographic operations.*

[2] In the standard model where there is no external source of randomness (e.g., random beacon), coin-tossing protocols suffer from a quadratic downstream communication cost (since $\Omega(n)$ parties contribute randomness to the process and each party has to read these contributions). Also, in the standard model, this solution is restricted to non-adaptive adversaries that select the corrupted parties *before* the committee is established.

In contrast, existing protocols fail to achieve constant communication even in relaxed models, e.g., if the downstream complexity is ignored and broadcasting a bit is counted as a unit cost operation or if the upstream complexity is ignored and only downstream complexity is counted. We proceed with some comments.

1. (About the thresholds) The protocol guarantees that even if the adversary actively corrupts up to $t_p = \tau_p n$ parties, at the end of the protocol the (honest) parties hold shares of a random secret according to t_p-private t_c-correct secret-sharing scheme for $t_c = \tau_c n$. For example, we can assume that every un-corrupted party will be participating in the reconstruction procedure and so we can take $t_p = n - t_c$ (e.g., $t_p = n/3$ and $t_c = 2n/3$). We note that the theorem is still meaningful even when $t_p + t_c \neq n$. (For example, to support the case of some honest parties being offline and not participating in the reconstruction of the secret, the parameters can be set such that $t_p + t_c < n$, e.g., $t_p = n/3$ and $t_c = 0.5n$ to support up to $n/6$ honest parties being offline. As another example, to support the case of parties that are passively dishonest and thus leak their shares while still participating in the reconstruction, the parameters can set such that $t_p + t_c > n$, e.g., $t_p = 0.6n$ and $t_c = 0.7n$.)[3] It should be emphasized that, unlike Shamir-based schemes, we do not get an exact threshold of $t_c = t_p + 1$, rather we only get *near-threshold* secret sharing.

2. (The field size) The limitation on the field size being super-polynomial in n can be completely waived at the expense of allowing a large constant gap between the parameters τ_c and τ_p. That is, for any finite field (including the binary field) the theorem can be proved for some constants $\tau_c < \tau_p$. (See Remark 4.15.) We focus on large fields since this is the natural setting for DKG applications (e.g., when the secret is taken to be the private key of a DLOG-based system).

3. (Formalizing security) Despite the popularity of DKG, there is no single canonical definition for its security. Building on Katz [38] and Gennaro et al. [29], we formalize security via an MPC framework by presenting abstract DSG/DKG functionalities that are independent of any concrete secret sharing scheme (similarly to the abstract definition of a commitment functionality). We assume that the network is synchronous and consider computationally-bounded, rushing, non-adaptive adversaries. Simple variants of our protocol achieve adaptive security assuming secure erasures and perfect commitments. Our simulators are straight-line black-box, so given UC-secure building blocks the protocol can be proved to be UC-secure.

4. (Round complexity) The number of rounds is a constant that grows linearly with the privacy-to-correctness gap $\tau_c - \tau_p$. We can optimize the round complexity and get 3 rounds if we allow a larger correctness-to-privacy gap (i.e., settle on some universal constants $\tau_p < \tau_c$). For DSG, we can even reduce the round complexity to 2, assuming a public-key infrastructure. This two-round solution can be also applied to DKG at the expense of a slight relaxation of the functionality (see Section 6.2).

5. (Concrete communication) Using DLOG-based primitives (e.g., ElGamal commitments and RO-based NIZK), each party communicates a constant number of elements from $\mathbb{F}$ and the

[3] Formally, we capture this property via the use of a *mixed* adversary [24] that applies different types of corruptions. See Section 5.

underlying group $\mathbb{G}$, where the constant is determined by information-theoretic objects (the sparsity of some low-density parity-check matrices). A concrete instantiation is described in Section 7. It therefore seems likely that one can get competitive practical results, at least for some range of parameters. We leave this direction for future work.

6. (The cost of recovering of the secret) Our DSG protocol generates $O(n)$ public elements on the BB. To recover the secret (either directly or taken to the power of some public group element h) one has to read these values from the BB and receive $O(1)$ values from each of at least t_c honest parties. Other existing protocols typically suffer from a similar cost as they have to read some certificates for the validity of the submitted shares (e.g., commitments). However, unlike other protocols, in our setting such access to the BB is necessary even if the adversary remains silent and only valid honestly-generated shares are being used. Put differently, our protocol suffers from the non-standard caveat that the local shares of authorized coalitions of size t_c have no information about the secret, and recovery is possible only when these shares are accompanied by the public values that are published on the BB. Similarly, in the context of DKG, computing the "public key" g^s requires reading $\Omega(n)$ public values that are available on the BB. Of course, such a computation can be done once and for all, and can be verified later by anyone based on public values. So in terms of usability, this property does not seem very limiting. Interestingly, it turns out that this non-standard property is *necessary* for bypassing the aforementioned *information bottleneck*, as we prove in Appendix A.

2.1 Technical overview

To prove Theorem 2.1, we will try to design a special-purpose secret-sharing scheme (SS) that natively supports distributed sampling. This requirement is satisfied if the shares are independently distributed. On the other hand, the correctness requirement implies that shares must be highly correlated. To bypass this problem, we design a scheme in which the shares are sampled independently at random and the correlation is achieved by publishing global public information that depends on all the shares. For example, think of the following variant of Shamir's scheme where each party i locally samples a random field element y_i as its share. We can think of these shares as defining a polynomial f of degree $n-1$ for which $f(i) = y_i, \forall i \in [n]$. To add redundancy, the parties securely evaluate the polynomial f in additional $m = n - t$ points $n+1, ..., n+m$ and publish the resulting vector $y = (y_{n+1}, \ldots, y_{n+m})$ on the BB as a "public header". Given this information, every set of t parties can recover the secret $f(0)$. Now, our goal is to securely compute a function that takes a single field element from each party and publishes $O(n)$ field elements on the BB. At least in terms of information (ignoring secrecy), this may be doable by using $O(1)$ communication per party. While it is not clear how to do it securely, at least we do not face the previous information bottleneck.

The AFS abstraction. Let us abstract the above idea. Our goal is to design a secret-sharing with public header y that is available to all parties such that (1) a random sharing $x = (x_1, \ldots, x_n)$ can be sampled by letting each party sample her share x_i uniformly at random, and (2) the header y can be securely computed based on x. We note that any linear secret sharing (LSS) Σ can be brought to this form. To see this, observe that a random sharing $x = (x_1, \ldots, x_n) \in \mathbb{F}^n$ according to Σ is a random vector in the Kernel of some $m \times n$ "constraint" matrix M (i.e. $M \cdot x = 0$), and the secret s associated with x can always be written as some linear combination $v \in \mathbb{F}^n$ of the shares, i.e.,

$s = \sum_i v_i x_i$. Consider a new secret sharing scheme in which $x = (x_1, \ldots, x_n)$ are sampled uniformly at random, rather than be sampled subject to $M \cdot x = 0$, and where the public header $y \in \mathbb{F}^m$ is taken to be the "syndrome" $M \cdot x$. It is not hard to see that a set T of parties can reconstruct the secret in the new scheme (given y) if and only if it can reconstruct the secret in the original scheme Σ. Thus, any LSS, specified by M and v, gives rise to an *affine secret sharing scheme* (AFS) with similar privacy and correctness thresholds. More generally, the scheme remains secure for any fixing of the public header y when the vector of shares x is sampled uniformly subject to $Mx = y$. (See Section 4 for formal definitions and statements.)

Computing the public header efficiently. Our goal now is to find an MPC-friendly AFS such that the mapping $\mathscr{F} : x \mapsto y$ given by $y = Mx$ can be securely computed with low communication. (The output of $\mathscr{F}$ should also consist of commitments to the private shares x_i, but let us ignore this for simplicity.) The functionality $\mathscr{F}$ takes a single field element from each party and publishes $m = \Omega(n)$ elements, and generic protocols for this task consume $\Omega(n^2)$ communication even in the presence of a broadcast channel. To cope with this problem, we employ a concrete secret-sharing scheme in which the constraint matrix M is *sparse*, i.e., each row and column have only a constant number of ones. Such a scheme was recently suggested by [3] based on Low-Density-Parity-Check Codes (LDPC). We extend their construction and show that such secret-sharing schemes can achieve near-threshold parameters.[4] Since the matrix is sparse, each output y_i depends on a constant number of inputs, and each party i affects a constant number of outputs. In the passive (aka semi-honest) setting, this immediately leads to a highly efficient protocol for computing the public header y. For instance, to compute an output $y_i = x_1 + \ldots + x_d$, the d relevant parties collectively generate an additive sharing of zero, with shares $r_1, \ldots, r_d$ given to the d parties, and post to the BB the values $x_i + r_i$ that sum-up to y_i. (To generate a sharing of zero we let each relevant party additively share the value zero and take the sum of these shares.)

One can handle active (malicious) adversaries by applying the GMW compiler (or cheaper variants of it). That is, we let the parties publicly commit to their inputs and randomness, send private messages publicly via the use of public-key encryption, and use NIZK to prove the consistency of their messages with the committed values and the previous rounds. The communication per party remains constant. One may worry that the adversary chooses its shares in a non-uniform way, however, it is not hard to show that such an attack does not violate the security of the secret. (The adversary still has no control or knowledge about the secret.) A more serious problem arises when the adversary *aborts* some of the outputs. Indeed, if a corrupt party aborts then it is impossible to compute any output y_i that depends on the input of that party.

Handling aborts. Assuming an honest majority, a naive solution for aborts is to force parties to share their inputs at the beginning of the protocol, and later when a party aborts have the other parties reveal the corresponding input. This solution has a linear communication cost per party and is therefore not applicable in our context. Alternatively, since the aborts in our case are *identifiable* [36] (i.e., we can identify a corrupted party that misbehaves) we can *repeat* the computation for an aborted output y_i without the corrupted parties. It is possible to implement this solution with low communication. However, it can be shown that the adversary can force a linear number of

[4] Along the way, we prove that, over large fields, LDPC codes can approach the singleton bound – a result that may be of independent interest. See Remark 4.4.

rounds by corrupting only a single party in each "correction round".[5] To derive a constant round solution, we take a different route and require the underlying AFS to be *robust* against a bounded number of erasures of the *public header*. (This notion of robustness should not be confused with the standard notion of robust secret sharing tolerating faulty shares.)

Robust AFS. Roughly speaking, in robust AFS, we want the secret to be recoverable even if the adversary erases some subset $B \subset [m]$ of the entries of $y = (y_1, \ldots, y_m)$. (Think of $|B|$ as a small constant fraction of m.) Intuitively, this means that a subset of t_c honest parties T that holds the shares $(x_i)_{i \in T}$ should be able to recover the secret $s = \sum_i v_i x_i$ given only some of the public shares $(y_i)_{i \notin B}$. Unfortunately, when the matrix is sparse such a strong level of robustness cannot be achieved since the adversary can erase all the $O(1)$ equations in which, say, the first honest party participates. This means that an honest coalition that does not contain the first honest party cannot recover x_1 and thus cannot reconstruct the secret $s = \sum_i v_i x_i$. Indeed, erasures can effectively remove all the information about the shares of some of the (possibly honest) parties. We solve the problem by compromising on the following weaker notion of robustness: After the removal of B it should be possible to efficiently locate a set A of parties such that after their removal, the *residual scheme* (M', y', v') obtained by removing (more precisely zero-ing) the B entries of y, the A entries of v, and the $B \times A$ submatrix of M, still supports recovery for a sufficiently large correctness threshold t_c. This means that we can "sacrifice" the B entries of y and still recover the newly defined secret $s' = \sum_{i \notin A} v_i x_i$. Observe that the adversary effectively shifts the secret to s', moreover, the adversary (which is rushing) can choose which subset B to abort after seeing the entire vector of pubic shares y. The robustness definition takes this into account and guarantees that, even under such an attack, security holds (i.e., the secret remains private and independent of the adversary's attack). To make this approach work, we show that sparse matrices can be used to derive robust AFS. We also need to carefully define ideal functionalities that capture the adversary's behavior and show that, when instantiated with robust-AFS, they realize the abstract DSG and DKG functionalities.

Achieving constant rounds, constant communication and near-threshold. With the help of robust-AFS, we can run the GMW-based protocol for computing the headers $y = Mx$ and simply give up on the "erased" header $(y_i), i \in B$. This immediately yields a three-round protocol with low communication assuming that the adversary can erase up to b entries where b is the robustness parameter of the AFS. This limitation induces a very small (yet linear) bound on the privacy threshold t_p. In order to improve this and derive near-threshold scheme, we apply the robustness property in a less aggressive way. Specifically, the robustness parameter b is taken to be some small (linear in n) value, and we apply robustness only if there are less than b erasures. If the number of erasures is larger, we remove the parties that were publicly identified as cheaters and re-compute the missing entries. This is done repeatedly, and it can be shown that after a constant number of rounds (that depends on the t_p, t_c and b), the number of erasures is sufficiently small and robustness can be applied. This strategy is non-trivial to implement with constant downstream communication (since we do not even have enough bandwidth to publish the number of missing entries), nevertheless we realize this approach with constant communication by carefully

[5] Such a protocol has an "optimistic" constant round complexity (when there are no aborts), and a "pessimistic" linear round complexity. Moreover, if the adversary delays the protocol by $r \leq t_p$ rounds it must publicly reveal $\Omega(r)$ corrupted parties. Assuming some penalty mechanism, such a protocol may be acceptable in practice.

postponing some of the computation to a post-processing public-decoding phase that is invoked after the sharing phase as part of the reconstruction.

Organization. Following some preliminaries in Section 3, we devote Section 4 to the study of AFS including definitions, properties, and sparse constructions. In Section 5 we formalize DSG and DKG protocols in an MPC framework and show how to realize these notions based on appropriate protocols for robust-AFS. Communication-efficient protocols for distributing a secret according to a sparse robust-AFS are presented in Section 6 and a concrete instantiation of this protocol appears in Section 7.

3 Preliminaries

General notation. We let $[n]$ denote the set of integers $\{1,\ldots,n\}$. For an $m \times n$ matrix $M = (M_{j,i})_{j\in[m],i\in[n]}$ and sets $R \subset [m]$ and $C \subset [n]$, we let $M[R;C]$ denote the $m \times n$ matrix whose (j,i)th entry is $M_{j,i}$ if $(j,i) \in R \times C$ and zero otherwise. We also let $M[;C] := M[[m];C]$ be the matrix that agrees with M on the columns in C and takes the value zero in all other columns. The complement of a set $T \subset [n]$ is denoted by $\bar{T}$. For random variables X and Y, we write $X \equiv Y$ to denote that X and Y are identically distributed.

Cryptographic definitions and primitives. We use the standard notion of a *non-interactive commitment scheme* $\mathsf{Com}_{\mathsf{crscm}}(x;k)$ where crscm is a random reference string crscm, x is a message and k is a random commitment key k. (See Appendix B for a definition.) To simplify notation, we typically omit the reference string crscm from the description of the commitment algorithm. Such commitments can be constructed based on one-way functions [35, 42].

We employ *Non-interactive zero-knowledge proofs of knowledge* (NIZK). In particular, following [38], we rely on ID-based simulation-sound NIZK proof system (see also [45, 39]). Syntactically, this means that proofs are generated with respect to an identifier. Roughly, zero-knowledge requires that simulated proofs are indistinguishable from real proofs even for adaptively chosen statements. Simulation soundness requires that if an adversary who is given an access to simulated proofs with respect to a set of identities H, can generate a valid proof with respect to any identity outside H, then a valid witness can be extracted. The formal definition appears in Definition B.1.

We assume familiarity with standard MPC definitions (see, e.g., [30, 13]). Throughout the paper we let C denote the set of corrupted parties and H denote the set of honest parties.

The BB model. We assume that parties have an access to a public bulletin board (BB) that is abstracted as an array or dictionary with random access. The array is partitioned to sections, and each party has an exclusive write-once permission for her section, i.e., only party i can write an element to the ith sub-array and once an element was written to cell number j, this value remains unchanged forever, and so parties who read these cell will always *agree* on its value. We view the elements on the BB as publicly available to all the parties, that is, all the parties have read permission to all sections. Our protocols naturally define for every message an address (or a key) in which it is stored, and instruct each party which addresses to read from the BB in each

step.[6] (Malicious parties can, of course, read everything.) For the sake of clarity, when describing a protocol, we typically treat the BB as a broadcast channel (keeping the mapping between messages and their addresses implicit), and only later analyze the fine-grained communication and see how many elements a party reads/writes during the protocol.

4 Secret Sharing

4.1 Definitions and Basic Facts

Through the paper, we assume that $\mathbb{F}$ is a finite field or a family $\mathbb{F} = \{\mathbb{F}_n\}$ of finite fields whose size grows with the security parameter or the number of parties. In the latter case, we assume that field operations can be implemented in polynomial time, and keep the dependency in n implicit.

We use a slightly non-standard variant of the notion of Linear Secret Sharing schemes. Roughly, (1) we assume that the share of each party is a single field element and (2) we replace linearity with affineness. (See Remark 4.5 for an explanation about the usage of affineness.) In addition, for convenience, our definition is centered around the process of sampling a random secret sharing vector that corresponds to a random secret, as opposed to sharing a given secret. (This difference is mainly syntactic and one can easily move between these two variants.)

Definition 4.1 (AFS: Semantics)**.** *An n-party $(t_{\mathsf{p}}, t_{\mathsf{c}})$* Affine Secret Sharing *Scheme (AFS) over a finite field $\mathbb{F}$ is a pair (Σ, Rec) where Σ is a probability distribution of sampling shares over an affine subspace of $\mathbb{F}^n$ and Rec is a recovery algorithm that takes a subset $T \subset [n]$ and a vector of shares $x[T] = (x_i)_{i \in T} \in \mathbb{F}^{|T|}$ and outputs a secret $s \in \mathbb{F}$ with the following properties:*

- *t_{c}-Correctness: For every subset $T \subset [n]$ of size at least t_{c} (hereafter referred to as "authorized") it holds that*

$$\Pr_{x \xleftarrow{R} \Sigma} [\mathsf{Rec}(T, x[T]) = s(x)] = 1,$$

  where $s(x) = \mathsf{Rec}([n], x)$ is referred to as the secret associated with the vector of sharing x. Furthermore, for every fixing of T the mapping $\mathsf{Rec}(T, \cdot) : \mathbb{F}^{|T|} \to \mathbb{F}$ is a linear mapping.

- *t_{p}-Privacy: For every set $T \subset [n]$ of size at most t_{p} (hereafter referred to as "unauthorized"), we have*

$$(x[T], s(x)) \equiv (x[T], s'),$$

  where $x \xleftarrow{R} \Sigma$, and $s' \xleftarrow{R} \mathbb{F}$ is chosen independently and uniformly.

Standard representation. By default, we assume that the AFS works as follows:

- The AFS is specified by an $m \times n$ *constraint matrix* M, a column *offset vector* $y \in \mathbb{F}^m$, and a row vector $v \in \mathbb{F}^n$ referred to as the *extraction vector*.
- The sampling algorithm $\Sigma_{M,y,v}$ samples a uniform solution $x \in \mathbb{F}^n$ to the set of equations $Mx = y$. When y is the all zero vector the scheme is *linear* as opposed to affine.

[6] We note that this convention is aligned with modern blockchains (e.g., Ethereum, Solana, Aptos) that implement storage as a key-value store (RocksDB in the latter two), and support direct retrieval of data using keys. Thus the download channel of reading values from the BB is cheap and straightforward.

- The underlying secret $s(x) = \sum_i x_i v_i$ is taken to be the inner-product between the vector of shares x and the extraction vector v. (The terminology is borrowed from the notion of *randomness extractors*, i.e., we extract the secret from the randomness of the parties).
- The recovery algorithm expresses the missing shares as a linear combination of the existing shares, and outputs the multiplication of v by the vector of shares. More precisely, the recovery algorithm $\mathsf{Rec}_{M,y,v}(T, x[T])$ finds a row vector $\alpha \in \mathbb{F}^m$ such that $\alpha \cdot M[\ ;\bar{T}] = v[\bar{T}]$, and outputs $\sum_{i \in T} v_i x_i + \alpha \cdot (y - M[\ ;T] \cdot x[T])$.

  If there is no such vector α, i.e., $v[\bar{T}]$ is not in the row-span of $M[\ ;\bar{T}]$, the recovery algorithm fails. Note that both Σ and Rec can be computed efficiently by making $\mathrm{poly}(n)$ number of arithmetic operations over $\mathbb{F}$.

The following simple fact characterizes the correctness and privacy in linear algebraic terms. (This is a straightforward generalization of the well-known linear algebraic characterization of linear secert sharing to the affine setting).

Fact 4.2 (linear-algebraic characterization of privacy and correctness)**.** *Let $M \in \mathbb{F}^{m \times n}$, $y \in \mathbb{F}^m$ and $v \in \mathbb{F}^n$, and assume that the offset vector y is a vector in the image of the constraint matrix M. Let x be a uniformly chosen solution to the system $Mx = y$ and let $s(x) = \sum_i x_i v_i$ denote the random variable induced by the choice of x. For every set $T \subset [n]$, if $v[\bar{T}] \in \mathrm{rowspan}(M[\ ;\bar{T}])$ then*

$$\Pr_x[\mathsf{Rec}_{M,y,z}(T, x[T]) = s(x)] = 1,$$

and otherwise,

$$(x[T], s(x)) \equiv (x[T], s')$$

where s' is uniform over $\mathbb{F}$. Consequently, $(\Sigma_{M,y,v}, \mathsf{Rec}_{M,y,v})$ is t_c-correct (resp., t_p-private) if and only if for every set $T \subset [n]$ of size t_c (resp., t_p) the vector $v[\bar{T}]$ is spanned (resp., not spanned) by $M[\ ;\bar{T}]$.

We say that (M, y, v) is t_c-*correct* (resp., t_p-*private*) if the AFS given by $(\Sigma_{M,y,v}, \mathsf{Rec}_{M,y,v})$ is t_c-correct (resp., t_p-private). By Fact 4.2, the offset vector y plays no role in the privacy/correctness of the scheme as long as it is in the image of M. We will always assume that the offset vector y is in the image of the constraint matrix M, and accordingly refer to (M, v) as t_c-*correct* (resp., t_p-*private*) if (M, y, v) is t_c-correct (resp., t_p-private) for every y is in the image of M.

From codes to an affine secret sharing scheme (AFS). It is not hard to see that the correctness property can be based solely on the error correction properties of the constraint matrix M, regardless of the choice of v. Formally, define the *dual distance* of M, denoted by $\mathsf{dd}(M)$, to be the smallest number of linearly *dependent* columns of M (over $\mathbb{F}$). Note that this means that for every subset $T \subset [n]$ of size lesser than $\mathsf{dd}(M)$ the matrix $M[\ ;T]$ is of rank at least $|T|$ and so $v[\bar{T}] \in \mathrm{rowspan}(M[\ ;\bar{T}])$ for every vector $v \in \mathbb{F}^n$. Therefore, the tuple (M, y, v) is $(n - \mathsf{dd}(M) + 1)$-correct no matter how the extraction vector v is chosen. Privacy now boils down to the selection of the extraction vector. We say that the extraction vector v is t_p-*private* for M (over $\mathbb{F}$) if for every t_p-subset $T \subset [n]$, it holds that $v[\bar{T}] \notin \mathrm{rowspan}(M[\ ;\bar{T}])$. Then, we have the following immediate claim (whose proof is implicit in [3] and is closely related to the general transformation of [18]).

Claim 4.3. *For every $m \times n$ matrix M, vector $y \in \mathbb{F}^m$ in the image of M and extraction vector $v \in \mathbb{F}^n$ which is t_p-private for M, the tuple (M, y, v) is $(n - \mathsf{dd}(M) + 1)$-correct and t_p-private. Moreover, except with probability $|\mathbb{F}|^{-(n-m-t_p)}\binom{n}{t_p}$, a randomly chosen vector $v \xleftarrow{R} \mathbb{F}^n$ is t_p-private for M.*

Proof. The first part follows from the above discussion and Fact 4.2. The "Moreover" part, follows by noting that for any fixed t_{p}-subset $T \subset [n]$, the rank of $M[\ ;\bar{T}]$ is m, and therefore, the probability that $v[\bar{T}] \in \text{rowspan}(M[\ ;\bar{T}])$ is at most $|\mathbb{F}|^m/|\mathbb{F}|^{n-t_{\mathsf{p}}}$. By applying a union-bound over all possible t_{p}-subsets, we get a failure probability of $|\mathbb{F}|^{-(n-t_{\mathsf{p}}-m)}\binom{n}{t_{\mathsf{p}}}$, as required. □

Remark 4.4 (Near-threshold AFS)**.** *Assuming that the field size grows asymptotically with the number of parties (e.g., $|\mathbb{F}| > \omega(1)$) we can take $t_{\mathsf{p}} = (1-\varepsilon)(n-m)$ for an arbitrary small constant $\varepsilon > 0$, and still get a negligible failure probability of $2^{-\Omega(n)}$.*[7] *If the distance of the code approaches the singleton bound, i.e., $\mathsf{dd}(M) > (1-\varepsilon)m$, then $t_{\mathsf{c}} = (1+\varepsilon)(n-m)$. Altogether, we get an almost-tight privacy-to-correctness gap $t_{\mathsf{c}} - t_{\mathsf{p}} \leq 2\varepsilon(n-m)$.*

For small fields (including the case of the binary field), we cannot hope to get arbitrarily small gap [11]. Still for a code with constant relative distance and constant rate, we still get, except with negligible probability, some non-trivial constants $0 < t_{\mathsf{p}} < t_{\mathsf{c}} < 1$ that are bounded-away from zero and one.

Collections of AFS. A $(\tau_{\mathsf{p}}, \tau_{\mathsf{c}})$-AFS collection with error $\varepsilon(\cdot)$ is specified by a probabilistic polynomial-time algorithm Z that given 1^n samples an index $z = (M, y, v)$ such that, except with probability $\varepsilon(n)$, the pair $(\Sigma_z, \mathsf{Rec}_z)$ forms an n-party $(\tau_{\mathsf{p}} n, \tau_{\mathsf{c}} n)$-AFS. By default, we assume that the error parameter ε is negligible in n. We may also assume that Z samples only the constraint matrix M and the extraction vector v since any y in the image of M can be used. We say that an AFS collection is *sparse* if the number of non-zero elements in every row and column of the constraint matrix is bounded by a fixed constant that does not grow with n.

Remark 4.5 (Why should we use affine schemes?)**.** *By Fact 4.2, privacy and correctness depend only on the constraint matrix M and the extraction vector v, and any offset vector y (in the column span of M) can be used. For this reason, the standard choice in the literature is to focus on LSS (as opposed to AFS) and assume that y is the all-zero vector. Still, for computational efficiency, it will be beneficial to employ a non-zero y since, in some cases sampling x conditioned on $Mx = 0$ is more expensive than sampling a uniform x and setting $y = Mx$. In particular, in a distributed setting, each party can sample its own share x_i independently at random, and then the parties reveal y via MPC. This approach will be used in our DSG and DKG constructions. Getting back to the information bottleneck mentioned in the introduction, the use of a non-zero vector y is in fact necessary for achieving our results.*

Remark 4.6 (From affine to linear)**.** *In many applications of secret sharing affineness provides a sufficiently good substitute for linearity. Moreover, if this is not the case then one can easily turn an affine sharing x of a random secret s under the AFS $z = (M, y, v)$ into a linear sharing of a random secret s' under the linear secret sharing scheme $(M, 0)$. This can be done by letting each party i locally redefine its share to $x_i - x'_i$ where $x' \in \mathbb{F}^n$ is some canonical vector for which $Mx' = y$. It is not hard to verify that the resulting sharing vector $x - x'$ is a random sharing under the scheme $z' = (M, 0, v)$ of the shifted secret $s - s'$ where $s' = \sum_i x'_i v_i$ is the secret associated with x' under $z = (M, y, v)$. Moreover, if (M, y, v) is $(t_{\mathsf{p}}, t_{\mathsf{c}})$-AFS then the scheme $(M, 0, v)$ is an $(t_{\mathsf{p}}, t_{\mathsf{c}})$-LSS.*

4.2 AFS from Expanders

In this section we define a certain expansion property for matrices, use existing techniques (Fact 4.7) to sample matrices with this property, and prove (Theorem 4.8) that such expanders can be used

[7] If the field is exponentially large (which is a reasonable scenario in the context of threshold cryptography), we can even take $t_{\mathsf{p}} = (n-m)$.

to construct a near-threshold sparse-AFS.

Matrices, sparsity, and expansion. Let $M = (M_{j,i})_{j\in[m],i\in[n]}$ be an $m \times n$ matrix. We say that M is *d-sparse* if every column of M has at most d non-zero elements, and say that it is *(d,r)-sparse* if, in addition, every row of M has at most r elements. We say that M is *(ℓ,e) column-expanding* (or just expanding) if for every set S of at most ℓ columns, the submatrix $M[\ ;S]$ has at least $e\cdot|S|$ non-zero rows. Let $\eta_e(M)$ denote the largest ℓ for which M is (ℓ,e) expanding. Note that $\eta_e(M)$ is monotonically decreasing with e and, for d-sparse matrices, $\eta_e(M) = 0$ for any $e > d$. It is well known (see, e.g., [51, Problem 5.5.]) that for (d,r)-sparse matrices and every $a > d/2$,

$$\eta_a(M) \leq \mathsf{dd}(M) - 1 \leq \eta_1(M), \tag{1}$$

where the equation holds regardless of the choice of the finite field $\mathbb{F}$ over which the dual-distance is computed. That is, expansion beyond half-the-column-sparsity, $d/2$, guarantees good distance, whereas non-shrinkage (expansion of at least 1) is necessary for good distance. Jumping ahead, we note that for large fields and properly chosen matrices, non-shrinkage is also sufficient for good distance.

Collections of matrices. For constants $\mu \in (0,1)$ and $d,r \in \mathbb{N}$, a *collection of (μ,d,r)-matrices* is defined by a (possibly randomized) polynomial-time algorithm $\mathscr{M}$ that given 1^n outputs a (d,r)-sparse $\mu n \times n$ matrix over $\mathbb{F}$. We say that the collection is *(λ,e) expanding with error $\varepsilon(n)$* (resp., *has distance* dd *with error $\varepsilon(n)$*) if the resulting matrix is $(\lambda n,e)$ expanding (resp., has distance $\mathsf{dd}n$) except with probability $\varepsilon(n)$. By default, we assume that ε is a negligible function. The following constructions are based on [15, 1].

Fact 4.7 (expanding collections)**.** *For every constant $\mu \in (0,1)$, constant $\varepsilon > 0$, and constant $\lambda < \mu/(1+\varepsilon)$ there exist constants d,r, and a collection of (μ,d,r) binary matrices that are $(\lambda,1+\varepsilon)$ expanding with a negligible error probability.*

Also, for every constant $\mu \in (0,1)$, there exist constants d,r,λ and a collection of (μ,d,r) binary matrices that are $(\lambda,0.9d)$ expanding with zero error probability.

We note that the constant 0.9 in the second part of the fact was chosen arbitrarily, and could be replaced by any constant larger than 0.5.

Proof. Observe that it suffices to prove the statement without worrying about the row sparsity. Indeed, the average row sparsity must be d/μ and so, by Markov's inequality, for every $\alpha > 0$, all but α-fraction of the rows have weight at most $r = d/(\mu\alpha)$. By removing these heavy rows we get (d,r)-sparsity at the expense of a small constant degradation in the parameter λ. The second part of the theorem now follows immediately from the celebrated result of [15].

To prove the first part we rely on [1]. Since the statement in the original paper refers to a slightly different regime of parameters, we sketch the argument here. Consider a random $\mu n \times n$ binary matrix R that each of its columns is sampled independently at random so that each column contains d ones. Let p_ℓ denote the probability that there exists a *non-expanding set* of exactly ℓ columns, i.e., a set that fails to expand by a factor of $1+\varepsilon$. A standard calculation shows that

$$p_\ell \leq \left[c_{\varepsilon,\mu} \left(\frac{(1+\varepsilon)\ell}{\mu n} \right)^{d-2-\varepsilon} \right]^\ell,$$

where $c_{\varepsilon,\mu}$ is a constant that depends on ε and μ but is independent of d. By taking d to be a sufficiently large constant, we can guarantee that p_ℓ is negligible for every $\omega(1) < \ell < \mu n/(1+\varepsilon)$. However, for constant size ℓ's we get an inverse polynomial failure probability, which means that the overall failure probability $\sum_{\ell \le \mu n/(1+\varepsilon)} p_\ell$ is inverse polynomial in n. To reduce the error to be negligible, we use the construction from [1] that samples a sparse matrix M' such that, except with negligible probability $\gamma(n)$, there are no non-expanding sets of size smaller than ℓ_0 for some super-constant parameter $\ell_0 = \omega(1)$. Let M denote the matrix obtained by taking the union of M' with a random sparse matrix R (i.e., $M_{i,j} = M'_{i,j} \vee R_{i,j}$ for each i, j). Then M is a sparse matrix that does not have a non-expanding set of size smaller than $\mu n/(1+\varepsilon)$ except with probability $\gamma(n) + \sum_{\ell_0 < \ell \le \mu n/(1+\varepsilon)} p_\ell$ which is negligible in n. □

Theorem 4.8 (near-threshold sparse-AFS from expanders)**.** *For every constants $\tau_\mathsf{p} < \tau_\mathsf{c}$ there exists constants d, r such that for every field $\mathbb{F}$ of size super-polynomial $n^{\omega(1)}$, there exists a $(\tau_\mathsf{p}, \tau_\mathsf{c})$-AFS over $\mathbb{F}$ whose constraint matrix is (d, r)-sparse. Furthermore, except with negligible probability the dual distance of the constraint matrix is $(1-\tau_\mathsf{c})n$.*

Proof. Let $\mu \in (1-\tau_\mathsf{c}, 1-\tau_\mathsf{p})$ be a constant. Given 1^n, we sample a tuple (M, y, v) as follows. (1) Use the first part of Fact 4.7 to sample a sparse $\mu n \times n$ binary matrix which is $((1-\tau_\mathsf{c})\mu, 1+\varepsilon)$ expanding for some constant $\varepsilon > 0$. Next, replace each non-zero position by a uniformly chosen field element and let M denote the resulting matrix. It is shown in [52, Lemma 3.9] that, except with negligible probability $|\mathbb{F}|^{-1}$, the dual distance of M over $\mathbb{F}$, is at least as large as the expansion parameter $(1-\tau_\mathsf{c})n$. Sample a random reconstruction vector $v \xleftarrow{R} \mathbb{F}^n$ and take y to be an arbitrary vector in the image of M. By Remark 4.4, except with exponentially small probability, we get a $(\tau_\mathsf{c} n, \tau_\mathsf{p} n)$-AFS (since $\tau_\mathsf{p} < 1-\mu$ and $\tau_\mathsf{c} = 1 - \mathrm{dd}(M)/n$), as required. □

Remark 4.9 (LDPC codes that almost achieve the singleton bound)**.** *Our proof implicitly shows that when the field $\mathbb{F}$ is sufficiently large (say super-polynomial in n), for every $\varepsilon > 0$ there are d_ε-sparse $m \times n$ parity-check matrices whose distance Δ approaches the singleton bound, i.e., $\Delta > (1-\varepsilon)n$. Moreover, such codes can be efficiently sampled with negligible error probability. To the best of our knowledge, this result does not appear in the literature. For comparison, the work of [41, Thm. 2.14] shows that such codes can achieve the Gilbert-Varshamov bound when the sparsity grows with the field size.*

Remark 4.10 (sparse-AFS over small fields)**.** *One can efficiently construct matrices that achieve a constant rate and a constant distance even under constant size fields [26]. In fact, by using the second part of Fact 4.7 and the connection between expansion beyond half-the-column-sparsity $d/2$ in Eq. (1), one can get binary matrices that achieve constant distance over any finite field. By sampling a random extraction vector as in Claim 4.3, we get a $(\tau_\mathsf{p}, \tau_\mathsf{c})$-AFS, for some non-trivial constants $0 < \tau_\mathsf{p} < \tau_\mathsf{c} < 0$, that works over small fields whose constraint matrix is a sparse binary matrix. (In fact, by using the techniques of [3], we can get a single scheme that works universally over all finite fields that also enjoys several efficiency features in reconstruction.)*

Example 4.11 (sparse-AFS over large fields: Concrete numbers)**.** *Say that the field is of size at least 2^{100} (in threshold systems the size is typically larger, e.g., $\approx 2^{255}$ for Schnorr's signature). Consider the following examples:*

1. *Say that we have $n \ge 10$ parties and take an AFS matrix with $\mu \cdot n$ rows where $\mu = 0.5$. By standard*

expansion calculations, there are sparse matrices with $d = 8$ and $r \approx 16$, that achieve $\tau_c = 0.66$.[8] *By choosing a random extraction vector, we get $\tau_p = 0.4$ except with failure probability 2^{-100}. So we get a $(8,16)$-sparse $(0.4, 0.66)$-AFS. (This favorably compares to the canonical setting of $1/3$ corrupt vs $2/3$ honest that is used in many scenarios.)*

2. *As another data point, assume that the field $\mathbb{F}$ is of size at 2^{255} and that the number of parties $n > 50$. Then, by taking $\mu = 0.6$, we can get a $(4,10)$-sparse matrix with $\tau_p = 0.39$, $\tau_c = 0.9$.*

4.3 Robust AFS

Motivation. When a DKG is run, some participants might behave maliciously and corrupt some of the shares that are needed for reconstructing the newly distributed key. We need the AFS to be robust to such attempts. For concreteness, consider the following scenario. Given an AFS $z = (M, y, v)$, we distribute a vector of random shares $x \in \mathbb{F}^n$ by sampling a uniform solution to the system $Mx = y$. Then, an adversary who controls a t_p-subset $T \subset [n]$ of the parties gets his shares $x[T]$ and is allowed to corrupt the index z by erasing a small subset B of the entries of the vector y. (Think of B as a small constant fraction of n.) Intuitively, we want the secret to still be recoverable given only $(M[\bar{B};\ ], y[\bar{B}], v)$. Unfortunately, when the matrix is sparse this is impossible since the adversary can, for example, include in B and erase all the $O(1)$ equations in which the first honest party participates. In this case, an honest coalition that does not contain the first honest party has no information on x_1 and cannot reconstruct the secret $s = \sum_i v_i x_i$.

Indeed, erasures in z effectively remove all the information about the shares of some of the (possibly honest) parties. The key idea is to make sure that these "lost" parties will not affect the secret by zero-ing the corresponding entries of the extraction vector. This way the secret remains recoverable even without the missing shares. In more details, we compromise on the following weaker notion of robustness: After the removal of B, it should be possible to locate a set A of parties such that even if their shares are lost, the *residual scheme* $z_2 = (M[\bar{B};\bar{A}], y[\bar{B}], v[\bar{A}])$, namely the n-party scheme containing all the original shares except those of A that are effectively taken to be zero, still supports recovery for a sufficiently large correctness threshold t_c. That is, for $z_2 = (M[\bar{B};\bar{A}], y[\bar{B}], v[\bar{A}])$, the recovery algorithm Rec_{z_2} can t_c-recover the secret $s' = \sum_{i \in \bar{A}} x_i v_i$ even when the shares x are sampled according to Σ_z (i.e., as a uniform solution to $Mx = y$). Note that the secret associated with x is changed to s' since we use the restricted extraction vector $v[\bar{A}]$. So privacy now means that the secret s' should remain information-theoretically hidden given $x[T]$. That is, the restricted extraction vector $v[\bar{A}]$ should be t_p-private for the *original* matrix M. Jumping ahead, note that by erasing B, the adversary effectively shifts the shared secret from s to s'. Still this does not bias the output since s' is still uniform and since our DSG/DKG protocols will ensure that the choice of B is independent of the secret. We continue with a formal definition of robust AFS.

Definition 4.12 (robust AFS)**.** *Let M be an $m \times n$ matrix, y be a vector in the column span of M, and $v \in \mathbb{F}^n$. We say that the tuple $z = (M, y, v)$ is b-*robust *(t_p, t_c)-*AFS *if for every b-subset $B \subset [m]$ there exists a set $A = A(B) \subset [n]$, referred to as the* sacrificed set of B*, such that for*

$$z_1 = (M, y, v[\bar{A}]) \quad \textit{and} \quad z_2 = (M[\bar{B};\bar{A}], y[\bar{B}], v[\bar{A}])$$

[8] The calculation here is based on the probabilistic method (i.e., we bound the probability that a random sparse matrix fails to expand well). We ignore here the issue of finding explicit expanding matrices and note that this can be done via several existing techniques, e.g., [1].

the pair $(\Sigma_{z_1}, \mathsf{Rec}_{z_2})$ forms a $(t_\mathsf{p}, t_\mathsf{c})$-AFS. (We use this pair of algorithms since the secret is shared with Σ_{z_1} and recovered with Rec_{z_2}.) We further assume that if B is an empty set then A must be an empty set as well, and therefore every b-robust $(t_\mathsf{p}, t_\mathsf{c})$-AFS, for $b \geq 0$, is also $(t_\mathsf{p}, t_\mathsf{c})$-AFS, and a 0-robust $(t_\mathsf{p}, t_\mathsf{c})$-AFS is simply a $(t_\mathsf{p}, t_\mathsf{c})$-AFS.

*An AFS ensemble with an index sampler Z is β-*robust $(\tau_\mathsf{p}, \tau_\mathsf{c})$-AFS *if for all but negligible probability over $(M, y, v) \xleftarrow{R} Z(1^n)$, the AFS (M, y, v) is βn-robust $(\tau_\mathsf{p} n, \tau_\mathsf{c} n)$-AFS. We also require that the set A should be efficiently computable given (M, B).*

Importantly, the residual scheme is defined over n parties, and the definition guarantees that even after the erasure, every subset of t_c parties (possibly including parties in A) can recover the secret.

Lemma 4.13 (Sparse AFS are robust)**.** *Suppose that (M, y, v) is a $(t_\mathsf{p}, t_\mathsf{c})$-AFS and that M is a (d, r)-sparse matrix whose dual distance is $\Delta = n - t_\mathsf{c} + 1$. Then (M, y, v) is a b-robust $(t_\mathsf{p} - b \cdot r, t_\mathsf{c})$-AFS for every b. Furthermore, the set $A(B)$ is taken to be the columns whose support intersects with B, i.e., $A(B) = \{i : \exists j \in B, M[j, i] \neq 0\}$.*

Note that the lemma keeps the correctness parameter t_c unchanged, and the extra robustness property only affects the privacy thershold.

of Lemma 4.13. Fix a b-subset $B \subset [m]$ and let $A = A(B)$ as defined above. We begin by claiming that $(\star)$ the $\bar{B} \times \bar{A}$ sub-matrix L of M has distance of at least $\Delta = n - t_\mathsf{c} + 1$. For this it suffices to show that any set of $\Delta - 1$ columns $\{w_i\}_{i \in S}$ in L are linearly independent. To see this, recall that each column vector w_i is obtained from a column $\hat{w}_i$ of M via the projection $w_i = \hat{w}_i[\bar{B}]$ where the B-coordinates of w_i are known to be zero (otherwise $i \in A$ and w_i is not a column of L). This means that $\{w_i\}_{i \in S}$ is linearly independent if the M-vectors $\{\hat{w}_i\}_{i \in S}$ are linearly dependent, which is the case by the assumption on the distance of M.

Let $t'_\mathsf{p} = t_\mathsf{p} - b \cdot r$. We will now prove that $(\Sigma_{z_1}, \mathsf{Rec}_{z_2})$ forms a $(t'_\mathsf{p}, t_\mathsf{c})$-AFS where z_1, z_2 are defined as in Definition 4.12. Let $x \in \mathbb{F}^n$ be a random solution to the system $Mx = y$. Then, $x' = x[\bar{A}]$ is also a solution to the system defined by $M' = M[\bar{B}; \bar{A}]$ and $y' = y[\bar{B}]$). Let $v' = v[\bar{A}]$. Fix a set $T \subset [n]$ of size at least t_c. Given $x[T]$, the recovery algorithm Rec_{z_2} recovers the secret $s' = \sum_i v'_i x'_i$ if and only if $v'[\bar{T}]$ is spanned by the rows of $M'[\ ; \bar{T}]$. This condition is equivalent to the condition that $v'[\bar{T} \cap \bar{A}]$ is spanned by the rows of $M'[\ ; \bar{T} \cap \bar{A}]$ (since the A entries/columns are set to zero). This is indeed the case, since the set $\bar{T} \cap \bar{A}$ is of size at most $n - t_\mathsf{c}$ which is smaller than the distance, Δ, of the $\bar{A} \times \bar{B}$ sub-matrix L of M, as shown in $(\star)$.

Fix a t'_p-subset $T \subset [n]$. To show that s' is distributed independently of $x[T]$, it suffices to show that

$$v'[\bar{T}] = v[\bar{A} \cap \bar{T}] \text{ is not in } \mathrm{colspan}(M[\ ; \bar{T}]). \tag{2}$$

Taking $S := A \cup T$, it holds that $\bar{S} = \bar{A} \cap \bar{T}$, and so (2) holds if $v[\bar{S}] \notin \mathrm{colspan}(M[\ ; \bar{T}])$ which must be the case since $|S| = (t'_\mathsf{p} + |A|) \leq t'_\mathsf{p} + br = t_\mathsf{p}$ and since v is t_p-private for M by assumption. (The inequality $|A| \leq b \cdot r$ follows by the sparsity condition on the matrix). □

By combining Lemma 4.13 with Theorem 4.8 we derive the following corollary.

Corollary 4.14 (near-threshold robust sparse-AFS)**.** *For every constants $\tau_\mathsf{p} < \tau_\mathsf{c}$ there exists constants d, r such that for every field $\mathbb{F}$ of size super-polynomial $n^{\omega(1)}$, there exists a (d, r)-sparse AFS collection which is β-robust $(\tau_\mathsf{p} - r\beta, \tau_\mathsf{c})$-AFS over $\mathbb{F}$ for every $\beta \geq 0$.*

Remark 4.15 (robust sparse binary AFS)**.** *For constant-size fields (e.g., the binary field), a similar corollary can be obtained for some constants $\tau_p < \tau_c$, by combining Lemma 4.13 with Remark 4.10.*

Remark 4.16 (Comparison to [3])**.** *Binary LDPC codes were employed in [3] to obtain secret sharing schemes with highly efficient recovery algorithm (that makes only linear number of additions regardless of the underlying field). Our constructions are quite similar but our motivation (reducing the complexity of secure distributed sharing) is different, and accordingly we exploit "sparsity" in a different way. This leads to new variants (e.g., the use of non-binary sparse matrices) and to novel notions such as robustness that were not considered in [3].*

5 Distributed Secret-Sharing Generation

Following Katz [38] we define *distributed key generation in the discrete-logarithm setting* (hereafter referred to as DKG), and *distributed secret-sharing generation* DSG, via an MPC framework. While Katz's definitions are tailored to Shamir-based DKG, we will need slightly more general definitions that are compatible with general collections of AFS schemes. We begin with an abstract version that captures the desired security properties and move on to more concrete variants, formally captured by *canonical* protocols, that provide additional efficiency features.

5.1 DSG and DKG: Abstract Version

Syntactically, a DSG is a two-stage n-party protocol where the parties hold no input. The first phase, Share, distributes to each party a private share and generates some public information. At the second phase, Rec, the parties recover the secret $s \in \mathbb{F}$, where the field $\mathbb{F} = \mathbb{F}_p$ is implicitly specified as part of the parameters of the scheme. For technical reasons, it will be convenient to add a special party, a "client", whose role is to invoke the two stages of the protocol. The syntax of DKG is similar, except that after the sharing phase, the protocol reveals also the public key g^s as part of the public information where g generates a cyclic group $\mathbb{G}$ of order p that is given implicitly as part of the parameters of the scheme. In its most abstract form, the scheme should realize the following reactive ideal functionality (Functionality 1). The term "broadcasts" should be interpreted as writing a message on the public BB.

Functionality 1 ($\mathcal{F}_{\mathsf{dsg}}$ and $\mathcal{F}_{\mathsf{dkg}}$)**.** *The functionality has two phases that are invoked by the client:*

1. *Share phase: the functionality samples a secret $s \in \mathbb{F}$ and broadcasts the message "shared", and, in the case of $\mathcal{F}_{\mathsf{dkg}}$, also the value g^s.*
2. *Recovery phase: the functionality broadcasts the secret s.*
   (For $\mathcal{F}_{\mathsf{dkg}}$, we can consider a variant in which the client specifies a public group element h, and the functionality broadcasts the pair (h, h^s).)

We will say that a protocol (t_p, t_c)-*realizes* $\mathcal{F}_{\mathsf{dsg}}$ (resp., (t_p, t_c)-*realizes* $\mathcal{F}_{\mathsf{dkg}}$) if it realizes $\mathcal{F}_{\mathsf{dsg}}$ (resp., $\mathcal{F}_{\mathsf{dkg}}$) in the presence of a mixed adversary that controls the client, and corrupts up to t_p parties with an arbitrary mix of t_1 passive and t_2 active corruptions as long as $t_1 + t_2 \leq t_p$. In addition, at the reconstruction phase, the adversary is allowed to abort (i.e., "crash") additional t_3 honest parties as long as $n - (t_1 + t_3) \geq t_c$, i.e., at least t_c parties honestly participate in the reconstruction.

This definition implies that any set of t_c honest/passively corrupted parties can recover the secret even when the adversary submits faulty shares on behalf of the actively corrupted parties. Such a protocol is also private in the sense that during the sharing phase, an adversary controlling up to t_p parties cannot bias the distribution of the secret and cannot learn anything about s (except for what follows from g^s in the case of DKG). In particular, the above MPC-based definition implies the property-based definition of DKG from [29] in its strongest form. We always assume that $t_p < t_c$ and note that the definition is meaningful even when $t_c + t_p \neq n$ (due the use of a mixed adversary).

Remark 5.1 (Relaxation)**.** *For completeness, we present here a relaxed variant of the definition, which is not used in our work. In some scenarios, it makes sense to relax the definition by requiring simulatability only for the sharing phase. Formally, we say that a two-phase protocol* Π weakly realizes $\mathcal{F}_{\mathsf{dsg}}$ *if for every adversary* $\mathcal{A}$ *there exists an efficient simulator* Sim *such that the random variable* $(\mathsf{View}^{\mathsf{Share}}_{\mathcal{A},\Pi}, \mathsf{Output}^{\mathsf{Rec}}_{\mathcal{A},\Pi})$*, consisting of the view of the adversary after the* sharing *phase and the output of the honest parties after the* recovery *phase, is computationally indistinguishable from the pair* (Sim, s) *where* Sim *is the output of the simulator and* $s \xleftarrow{R} \mathbb{F}$ *is a uniformly chosen secret. For the case of DKG, the simulator also gets* g^s *as an input. Indeed, the DKG variant of this definition is essentially equivalent to the property-based definition from [29].*

5.2 Canonical Schemes

While the above definition nicely captures the desired security properties of DSG and DKG, it misses some useful "efficiency" aspects such as non-interactive reconstruction or the ability to reconstruct shares via linear operations – a feature that is necessary for "reconstruction-in-the-exponent" in DLOG-based threshold systems. To capture these additional properties (which are common to most existing schemes), we introduce the notion of canonical schemes and focus throughout the paper on such schemes.[9]

Let E be a non-interactive commitment scheme. We say that a DSG is in *canonical* form if, at the end of the sharing phase, each honest party holds a share $x_i \in \mathbb{F}$ of the secret s according to some (t_p, t_c)-AFS that is specified by the public values $z = (M, y, v)$ that are known to all parties (e.g., broadcast during the protocol). In addition, at the end of the sharing phase all the parties learn commitments $(\alpha_i = E(x_i; \rho_i))_{i \in [n]}$ to all the shares. In this case, the recovery phase can be implemented by the following single-round *canonical recovery* protocol (Protocol 1).

[9] Alternatively, one could try to formalize an these properties as part of the ideal functionality (e.g., by letting the functionality distribute "handles" for the secret). We feel that the current solution is simpler and more intuitive.

Protocol 1 (Canonical Recovery Protocol Π_{Rec})**.** *We assume a common reference string* crspf*, public index $z = (M, y, v)$ and public share commitments $\alpha = (\alpha_i)_{i\in[n]}$. In addition, each honest party P_i holds (x_i, ρ_i) such that $\alpha_i = E(x_i; \rho_i)$.*

- ***R1:*** *Each party P_i broadcasts x_i and a NIZK π_i (with respect to* crspf*) that x_i and α_i satisfy the equality $\alpha_i = E(x_i; \rho_i)$ with respect to the witness ρ_i.*
- ***Output*** *Let (x'_i, α'_i) denote the values broadcasted by the ith party and let $T \subset [n]$ be the set of indices $i \in [n]$ for which the proof π'_i passes verification with respect to α'_i and x'_i. Compute the linear recovery algorithm $\mathsf{Rec}_{M,y,v}(T, x'[T])$ and output the result.*

If we strive for the weaker variant of DSG/DKG that is mentioned in Remark 5.1 then we can simply open the commitment in the recovery phase and avoid the NIZK. (See Footnote 16 in the proof of Lemma 5.3 for more details.)

Remark 5.2 (Canonical recovery in the exponent)**.** *The above protocol can be easily modified to allow the reconstruction of the secret s in the exponent of a public group element $h \in \mathbb{G}$ (which is broadcasted to all the parties by the "client"). In the first round, each party sends h^{x_i} together with NIZK that certifies that x_i is consistent with its commitment α_i. At this point any (possibly external) party can compute h^s by dropping the invalid elements (whose validity proofs fail) and by computing the* linear *reconstruction algorithm Rec_z "in the exponent". Thus canonical protocols efficiently support "reconstruction in the exponent", which is a crucial feature in the context of DLOG-based threshold cryptography. We emphasize that most threshold cryptography applications use recovery in the exponent, for example to compute BLS signatures or to compute a verifiable random function (VRF). The secret is used as the key for these functions. Furthermore, in these applications, when there are multiple invocations of the threshold function, the same secret is recovered multiple times in the exponent, using different random public bases. (One can capture this by defining the DSG/DKG functionality as a reactive multi-phase functionality in which sharing happens once during initialization and recovery-in-the-exponent can be called multiple times with different group elements h; Our protocols hold in this setting as well.)*

Given the above discussion, to realize a canonical DSG it suffices to implement a secure protocol for the sharing phase. This is formalized by Functionality 2 in Fig. 1. The ideal functionality $\mathscr{F}_{\mathsf{cdsg},b}$ is parameterized with a robustness parameter b and is implicitly parameterized by a non-interactive commitment scheme E and by a b-robust $(t_{\mathsf{p}}, t_{\mathsf{c}})$-AFS. The latter is specified by a public $m \times n$ constraint matrix M, and a public extraction vector $v \in \mathbb{F}^m$ such that for every $y \in \mathbb{F}^m$ in the image of M the AFS (M, y, v) is b-robust $(t_{\mathsf{p}}, t_{\mathsf{c}})$-AFS.[10] The non-robust variant is handled by taking the robustness parameter b to be zero. Jumping ahead, we will show later (Lemma 5.3) that security holds even if the adversary is allowed to choose her own shares based on the residual value of the offset y and to erase up to b of entries of the resulting offset vector.

To understand the definition, let us focus on the non-robust version where the adversary does not erase entries, i.e., $B = \emptyset$. Intuitively, security holds since for any fixing of y in the image of M, and any fixing for the shares of the corrupted parties, $x[\mathsf{C}]$, if we choose the shares of the honest parties $x[\mathsf{H}]$ uniformly at random subject to $Mx = y$, then the secret $s = \sum_i v_i x_i$ is uniformly distributed.

[10] Asymptotically, we may assume that M and v are sampled from some b-robust $(t_{\mathsf{p}}, t_{\mathsf{c}})$-AFS sampler $Z(1^n)$ during a one-time set-up phase; Such a phase is needed any way to set the field and the underlying cyclic group $\mathbb{G}$.

Functionality 2 ($\mathcal{F}_{\mathsf{cdsg},b}$). *The functionality gets the set of corrupt parties* C.

1. *(Sampling) The functionality samples random commitment keys for the honest parties* $(\rho_i)_{i\in\mathsf{H}}$ *and samples random field elements as shares for the honest parties* $x_\mathsf{H} = (x_i)_{i\in\mathsf{H}} \xleftarrow{R} \mathbb{F}^{|\mathsf{H}|}$. *Then it computes the residual offset vector* $y' = M_\mathsf{H} \cdot x_\mathsf{H} \in \mathbb{F}^m$ *where* M_H *is the restriction of* M *to the columns indexed by* H. *The adversary gets the commitments of the honest parties* $(\alpha_i)_{i\in\mathsf{H}}$ *where* $\alpha_i = E(x_i;\rho_i)$ *and the offset vector* y'. *(The constraint matrix* M *and the extraction vector* v *are assumed to be public.)*

2. *(Corruption) The adversary selects her own shares* $x_\mathsf{C} = (x_i)_{i\in\mathsf{C}}$, *her own commitment keys* $\rho_\mathsf{C} = (\rho_i)_{i\in\mathsf{C}}$ *and specifies an erasure subset* $B \subset [m]$ *of size at most* b. *The tuple* $(x_\mathsf{C}, \rho_\mathsf{C}, B)$ *is sent to the functionality.*

3. *The functionality merges* $x_\mathsf{H}, x_\mathsf{C}$ *to a single vector* $x \in \mathbb{F}^n$, *computes* $y = Mx = y' + M_\mathsf{C} \cdot x_\mathsf{C}$ *where* M_C *is the restriction of* M *to the columns indexed by* C. *Finally, the functionality broadcasts the erasure set* B, *the modified, erased, offset vector* $y[\bar{B}]$, *and the commitments* $(E(x_i;\rho_i))_{i\in[n]}$. *In addition, each honest party* i *privately receives* (x_i, ρ_i).

Figure 1: Functionality 8 – sharing phase for a canonical DSG.

Formally, we prove the following lemma.

Lemma 5.3. *Let* Π *be a protocol that* t_p*-realizes* $\mathcal{F}_{\mathsf{cdsg},b}$ *for some* b*-robust* $(t_\mathsf{p}, t_\mathsf{c})$*-AFS* (M, y) *and let* Π_{Rec} *denote the canonical recovery protocol. Then,* $(\Pi, \Pi_{\mathsf{Rec}})$*, viewed as a two-phase protocol,* $(t_\mathsf{p}, t_\mathsf{c})$*-realizes* $\mathcal{F}_{\mathsf{dsg}}$ *where we assume that* Π_{Rec} *is applied to the index* $(M[\bar{A};\bar{B}], y[\bar{B}], v[\bar{A}])$ *where* $B, y[\bar{B}]$ *are the public output of the first phase and* A *is the sacrificed set of* B *(which is computed based on* B *and* M *by using the specification of the AFS).*

sketch. We begin with a brief intuition under the simplifying assumption that the commitment scheme is perfectly hiding. (This assumption will be waived in actual proof.) In this case, by the privacy properties of the AFS, for any fixing of the view of the adversary after the sharing phase, the distribution of the secret is uniform and independent of the view. Furthermore, assume that in the reconstruction phase, the adversary aborts all but t_c parties that submit honest (uncorrupted) shares. Then, by the correctness and robustness properties, the secret is recovered properly. In the actual proof, we argue that (1) computationally hiding commitments suffice for achieving computationally close simulation and that (2) the binding properties and the NIZK guarantee that a computationally bounded adversary cannot modify its shares during the reconstruction. The full proof is deferred to Appendix C.1. □

5.3 From DSG to DKG

We reduce $\mathcal{F}_{\mathsf{dkg}}$ to canonical DSG, $\mathcal{F}_{\mathsf{cdsg}}$, by adding a single round of "reconstruction in the exponent" of the generator g. (Party i broadcasts g^{s_i} with a NIZK that proves consistency with α_i, and the valid elements can be combined into g^s as explained in Remark 5.2.) This allows everyone to recover the public key g^s without revealing any additional information on s. The communication

per party is constant (it depends on the security parameter but does not grow with n). Formally, we have the following lemma whose proof is deferred to Section C.2.

Lemma 5.4. *Let Π be a protocol that t_{p}-realizes $\mathscr{F}_{\mathsf{cdsg},b}$ for some b-robust $(t_{\mathsf{p}},t_{\mathsf{c}})$-AFS (M,y) and let Π'_{Rec} denote the "canonical recovery in the exponent" protocol from Remark 5.2. Then, the functionality $\mathscr{F}_{\mathsf{dkg}}$ is $(t_{\mathsf{p}},t_{\mathsf{c}})$-realized by the following two-phase protocol:*

- *(Sharing) Invoke Π and then apply Π'_{Rec} with the public group generator g and where the index of the AFS is taken to be $(M[\bar{A};\bar{B}],y[\bar{B}],v[\bar{A}])$ where $B,y[\bar{B}]$ are the public output of Π and A is the sacrificed set of B (which is computed based on B and M by using the specification of the AFS).*
- *(Reconstruction in the exponent) Given public group generator h, invoke Π'_{Rec} with the generator h and where the index of the AFS is taken to be $(M[\bar{A};\bar{B}],y[\bar{B}],v[\bar{A}])$ as defined above.*

The drawback of the above approach is that it adds a single round of communication in order to reconstruct the secret in the exponent. This can be avoided if we are willing to realize a weaker variant of the DKG functionality. The idea is to make a single call to the $\mathscr{F}_{\mathsf{cdsg}}$ functionality while setting the underlying commitment scheme in $\mathscr{F}_{\mathsf{cdsg}}$ to $E(x_i;\rho_i) := g^{x_i}$. We refer to this variant as *canonical DKG*, $\mathscr{F}_{\mathsf{cdkg}}$. Although E is not a valid commitment scheme (being deterministic it fails to satisfy semantic security), the values $(g^{x_i})_{i\in[n]}$ leak exactly the public key g^s which should be revealed anyway. Still, strictly speaking, $\mathscr{F}_{\mathsf{cdkg}}$ does not realize $\mathscr{F}_{\mathsf{dkg}}$ since the adversary can choose its inputs after seeing the "exponentiated shares" of the honest parties, and so the adversary can effectively shift the public key by an arbitrary shift $\Delta \in \mathbb{F}$. This issue is discussed by [29] who show that this variant suffices for typical applications of threshold cryptography (e.g., Schnorr's signatures). Intuitively, if the underlying hardness assumption (e.g., in-feasibility of extracting DLOG) holds over the un-shifted key, then it also holds with respect to the shifted public key since the shift Δ can be extracted from the adversary. We note that this variant can be formalized by a variant of the DKG ideal functionality in which the functionality first sends the public key g^s to the adversary who is allowed to shift it by a chosen Δ, and then forwards the shifted public key $g^{s+\Delta}$ to the honest parties.

6 Realizing Robust Canonical DSG

In Section 5 we showed that the task of realizing DSG/DKG reduces (with constant overhead) to the task of realizing the $\mathscr{F}_{\mathsf{cdsg},b}$ functionality. In this section we present two protocols that realize $\mathscr{F}_{\mathsf{cdsg},b}$. In both cases, each party reads/writes $O(1)$ elements from the BB such that at the end of the protocol each party holds her private output. In addition, everyone can recover the public outputs by reading the content of the BB. Our first "basic" protocol (Section 6.2) achieves a relatively low, yet constant, privacy threshold, and our second "extended" protocol (Section 6.3) provides a near-threshold result, namely an arbitrarily small gap between the privacy and correctness thresholds, τ_{p} and τ_{c}. We begin with some preliminaries (Section 6.1).

6.1 Notation and Tools

Notation. Let $M = (M_{j,i})_{j\in[m],i\in[n]}$ be a sparse $m \times n$ matrix. We focus on binary matrices though the following can be easily generalized to the non-binary case. The support of column number $i \in [n]$ is denoted by $R_i = \left\{ j \in [m] : M_{j,i} \neq 0 \right\}$ and the support of row number $j \in [m]$ is denoted by

$L_j = \{i \in [n] : M_{j,i} \neq 0\}$. We let $L_{j,-i}$ denote the set $L_j \setminus \{i\}$. The matrix M will be used as a mapping from vectors $x \in \mathbb{F}^n$ to vectors $y \in \mathbb{F}^m$ where $y = Mx$. Accordingly, each column of M corresponds to an input and each row corresponds to an output, and so $j \in R_i$ means that the output j is influenced by the input i and by the inputs in $L_{j,-i}$. For a set of inputs $I \subset [n]$, we let $R(I) = \cup_{i \in I} R_i$ denote the set of outputs that are affected by inputs in I. Throughout the section, $E_\rho(x)$ is taken to be a non-interactive commitment scheme that is specified as part of the description of $\mathscr{F}_{\mathsf{cdsg},b}$. The algorithm E takes a field element $x \in \mathbb{F}$ and a key ρ as input, and outputs some "tag".

Tools. We will need non-interactive commitments $\mathsf{Com}_{\mathsf{crscm}}(x;k)$, and for clarity we distinguish between these commitments and the "internal" commitments E that is specified by $\mathscr{F}_{\mathsf{cdsg},b}$. We will also need an ID-based simulation-sound NIZK proof system for the following relations: The E-relation, defined wrt the tagging-algorithm E, via

$$\mathscr{R}_E := \{(\alpha, (\rho, x)) : \alpha = E_\rho(x)\},$$

and an additive commitment relation, about a value y being equal to a linear combination of committed values. For a coefficient vector $v \in \mathbb{F}^\kappa$, it is defined as

$$\begin{aligned} \mathscr{R}_v \quad = \quad & \{(\mathsf{crscm}, y, \alpha, (c_i)_{i\in[\kappa]}), (x, \rho, (x_i, k_i)_{i\in[\kappa]} : \\ & \forall i \in [\kappa], c_i = \mathsf{Com}_{\mathsf{crscm}}(x_i; k_i), \alpha = E_\rho(x), y = x + \sum_i v_i \cdot x_i\}. \end{aligned}$$

To simplify notation, we typically omit the CRS crscm from the subscript of the commitment and from the relations.

6.2 The Basic Protocol

Protocol 2 in Fig. 2 describes a basic DSG protocol that realized $\mathscr{F}_{\mathsf{cdsg},b}$. Intuitively, the protocol $\Pi_{M,E}$ securely computes the mapping $x = (x_1, \ldots, x_n) \mapsto y = Mx$ for an arbitrary security threshold t, except that it allows the adversary to abort outputs that depend on the adversary's inputs.[11] Since the matrix is sparse and each column contains at most d non-zero elements, an adversary that corrupts t parties can only abort at most dt outputs. As a result, if (M, v) is a b-robust $(t_\mathsf{p}, t_\mathsf{c})$-AFS then $\Pi_{M,E}$ realizes $\mathscr{F}_{\mathsf{cdsg},b}$ with security threshold of $t'_\mathsf{p} = \min(t_\mathsf{p}, b/d)$. We also note that the protocol publicly identifies some of the corrupted parties (i.e., the "auxiliary output" C') – a feature that is not needed under the definition of $\mathscr{F}_{\mathsf{cdsg},b}$. To match the syntax of $\mathscr{F}_{\mathsf{cdsg},b}$ we can always assume that C' is dropped from the output. The following theorem is proved in Appendix D.

Theorem 6.1. *Suppose that M is a (d, r)-sparse constraint matrix that together with a recovery vector v forms a b-robust $(t_\mathsf{p}, t_\mathsf{c})$-AFS. Then, $\Pi_{M,E}$ t'_p-realizes the functionality $\mathscr{F}_{\mathsf{cdsg},b}$ for $t'_\mathsf{p} = \min(t_\mathsf{p}, b/d)$.*

Remark 6.2 (The complexity of $\Pi_{M,E}$)**.** *The protocol has 3 rounds of interaction and each party sends (either privately or to the BB via broadcast) at most $O(d \cdot r \cdot \max(\kappa, \log|\mathbb{F}|))$ bits where d and r are the maximal number of non-zero elements in a column of M and the maximal number of non-zero elements in a row of M, respectively. Similarly, each party P_i receives at most $O(d \cdot r \cdot \max(\kappa, \log|\mathbb{F}|))$ bits via*

[11] We emphasize again that we used affine secret sharing in order to let each participant choose its share x_i at random, and have the system publish $y = Mx$ to enable recovering the joint secret. If, instead, we were using secret sharing where $y = Mx = 0$ then the participants would have needed to coordinate their share generation, to ensure that $Mx = 0$.

Protocol 2 (The basic protocol $\Pi_{M,E}$). *We assume a common reference string* $\mathsf{crs} = (\mathsf{crspf}, \mathsf{crscm})$. *Each party* $P_i, i \in [n]$ *locally samples random* (x_i, ρ_i) *and proceeds as follows.*

- ***R1:*** *(Sending Randomizers) For every output* $o \in R_i$ *influenced by* i *and every* $j \in L_{o,-i}$, *party* P_i *samples a random mask* $r_{o,i,j} \xleftarrow{R} \mathbb{F}$ *and a random commitment key* $k_{o,i,j}$, *broadcasts the commitment* $c_{o,i,j} = \mathsf{Com}(r_{o,i,j}; k_{o,i,j})$, *and sends the opening* $(r_{o,i,j}, k_{o,i,j})$ *of the commitment to the* j*th party* P_j *over a private channel.*

- ***R2:*** *(Resolving Private Inconsistencies) For every* $o \in R_i, j \in L_{o,-i}$, *if* P_i *does not receive from* P_j *an opening for the published commitment* $c_{o,j,i}$, *or receives an opening that is inconsistent with* $c_{o,j,i}$, *then* P_i *broadcasts a "complaint"* (j, o). *This complaint asserts that* $r_{o,j,i}$ *and* $k_{o,j,i}$ *should be set to zero and* $c_{o,j,i} = \mathsf{Com}(0;0)$. *(We assume that* $i, j \in R_o$, *and if this is not the case, ignore the complaint.)*[a]

- ***R3:*** *(Computing shares of the vector* y*: Local Sums and Output Tags) Party* P_i *broadcasts* $\alpha_i = E_{\rho_i}(x_i)$ *together with a NIZK* π_i *for consistency. In addition, for every output* $o \in R_i$, *party* P_i *broadcasts the value*

$$y_{o,i} = x_i + \sum_{j \in L_{o,-i}} r_{o,i,j} - \sum_{j \in L_{o,-i}} r_{o,j,i},$$

  together with a NIZK $\pi_{o,i}$ *that the committed values in* $\alpha_i, (c_{o,i,j}, c_{o,j,i})_{j \in L_{o,-i}}$ *satisfy the above linear equation about* $y_{o,i}$.

- ***Private outputs:*** *The private output of the* i*th party is taken to be its private inputs* (x_i, ρ_i).

- ***Public output:*** *The broadcasted values define the public outputs of the protocol via the following decoding procedure. If, for some party* i, *the proof* π_i *fails to verify, set* $\alpha_i = E_0(0)$ *and replace* π_i *with a valid proof. In addition, initialize* $\mathsf{C}', B = \emptyset$. *For every output* $o \in [m]$:

  - ***If*** *there exists* $j \in L_o$ *for which the proof* $\pi_{o,j}$ *fails to verify insert* o *to* B *and* j *to* C'.
  - ***Otherwise****, set* $y_o = \sum_{j \in L_o} y_{o,j}$.

  Set $B, (y_o)_{o \notin B}$ *and* $(\alpha_i)_{i \in [n]}$, *as the public verification information of the DSG. (Treat* C' *as auxiliary output.)*

[a] Note that an adversary can issue false complaints and force $r_{o,j,i}$ and $k_{o,j,i}$ to be zero. In the proof, we show that this is not an issue (essentially since $r_{o,j,i}$ is being used to pad information known to the adversary anyway).

Figure 2: The basic protocol $\Pi_{M,E}$.

point-to-point communication and has to read a similar amount of bits from the BB (basically, only the commitments sent by parties that influence an output that is also influenced by P_i). Since $r = d = O(1)$, the communication per party is a constant that does not grow with the number of parties. The computational complexity per party is $O(rd) = O(1)$ cryptographic operations/field operations during the execution of the protocol. The final public decoding costs $O(n)$ downstream communication and $O(n)$ operations and it can be postponed to the recovery phase.

Variants: The above protocol can be tweaked in many ways to optimize different goals. We mention some of these variants.

- **Two-Round version**: If we assume a PKI we can reduce the round complexity to two rounds as follows. Recall that in the first round P_i broadcasts a commitment $c_{o,i,j} = \mathsf{Com}(r_{o,i,j}; k_{o,i,j}))$ and sends its private opening $(r_{o,i,j}, k_{o,i,j})$ to the jth party P_j. Instead, we let P_i broadcast an encryption of $r_{o,i,j}$ encrypted under the public-key of P_j together with a NIZK for the validity of the ciphertext. Now we can remove the "complaining round" **R2** and proceed directly to **R3**. (The NIZK relations in **R3** should be updated, e.g., instead of proving consistency with commitments one has to prove consistency with a ciphertext.) The security proof goes through assuming that the underlying encryption is perfectly correct. Conveniently, the first round of this protocol is independent of the inputs (x_i, ρ_i) and so it can be invoked as an "offline" round.

- **Adaptive security**: Assuming *secure erasures* and *trapdoor commitments* (e.g., Pedersen commitments), the protocol can be made adaptively secure by letting each party erase the randomness used for the NIZKs.

- **Abstraction:** The protocol can be viewed as a concrete instantiation of the following more general approach. Say that a set of n parties holding inputs $x_1, \ldots, x_n \in X$ wishes to securely compute m functions $f_1, \ldots, f_m$ such that the output is delivered to all the parties and where each function f_i depends on a small set of inputs $S_i \subset [n]$. Then, we can get such a protocol (with refined identifiable abort) based on protocols $\Pi_1, \ldots, \Pi_m$ for individually computing the functionalities $f_1, \ldots, f_m$ where protocol Π_i is defined over the parties S_i. Assuming a PKI and NIZK, the idea is to start by letting each party commit to its randomness and input, then run the protocols $\Pi_1, \ldots, \Pi_m$ where private messages are sent encrypted over public channels together with NIZKs that certify consistency with the committed inputs and committed randomness. This variant of the well-known GMW compiler [31] inherits the security properties of the underlying protocols, and leads to a communication complexity which is essentially the sum of the complexities of the Π_i protocols. Furthermore, by using the "undeniable transmission" mechanism from [5, 2], the public key encryption scheme can be replaced by standard commitments. The actual version that appears above is obtained by tailoring this approach to the special case of linear functions while exploiting the simple structure of the standard secure-addition protocol.

- **Relaxing E:** The proof of Theorem 6.1 does not make use of the hiding property of the underlying commitment E. This means that Protocol $\Pi_{M,E}$ realizes the functionality $\mathscr{F}_{\mathsf{cdsg},b}$ even if the underlying function E is binding but not necessarily hiding. Of course, in such a case Lemmas 5.3 and 5.4 do not hold. Still, this observation allows us to employ the pro-

tocol where $E(x) = g^x$ and realize the canonical DKG protocol (with the caveats discussed in Section 5.3).

6.3 Improving Security by Limiting Aborts

In order to achieve near-threshold results (i.e., an arbitrarily small gap between the privacy τ_{p} and correctness τ_{c} thresholds), we need to limit the number of aborts. Recall that the basic protocol $\Pi_{M,E}$ aborts each output that depends on an input from a party that was publicly identified as being corrupted. To limit the number of aborts, we invoke the basic protocol $\Pi_{M,E}$ and then try to recover the aborted outputs by using an additional sub-protocol. (Protocol 3 in Fig. 3.)

Informally, the idea is to remove the set of publicly corrupted parties, and to re-compute the corresponding outputs over the inputs of the other parties as if all the inputs of the publicly corrupted parties were taken to be zero. This approach means that all the outputs $o \in R_{\mathsf{C}'}$ that are affected by publicly corrupted parties (including valid ones) have to be re-computed. Fortunately, re-computing such values is quite simple given the information that was gathered in $\Pi_{M,E}$: All that is needed is to reveal the randomizers $r_{o,i,j}$ that correspond to a pair (i,j) consisting of a party $i \notin \mathsf{C}'$ and a publicly corrupted party $j \in \mathsf{C}'$ (or vice versa). Since each such value was already committed to, and the honest party from the pair knows it, that party can simply open the corresponding commitment.

A potential difficulty is that malicious parties that acted honestly in $\Pi_{M,E}$ and were not detected, may decide not to collaborate in this new sub-protocol. Namely, these parties will not open in the new sub-protocol the commitments that the protocol requires them to open. This behavior will be detected during the new sub-protocol, the corresponding parties will be added to the set of publicly corrupted parties, and as a result their inputs will be set to zero and additional outputs will need to be computed. This process might cascade over multiple iterations of running the sub-protocol, if in each iteration a new party is identified as being corrupted, and as a result its inputs must be set to zero. Fortunately, it can be shown that the amount of communication is still constant per party. Moreover, if the number of aborts is linear and is equal to $b = \beta n$ for some small constant β, then the round complexity will be constant as well. The resulting protocol, $\Pi_{M,E,b}$, is described in Protocol 3 in Fig. 3. It is parameterized with the number b of outputs that the adversary is allowed to abort (which corresponds to the robustness parameter).

To simplify the presentation, the protocol description ignores communication complexity limitations. It will be also convenient to drop the distinction between online protocol operations and public-decoding operations that will be post-processed after the execution (e.g., as part of the recovery phase) based on publicly available values that appear on the BB. Still, we highlight such public operations by the label "**All:**" that indicates that the following operations can be computed based on public values. We will later explain how to obtain a communication-efficient variant of the protocol.

Analysis. It is not hard to verify that C' contains only corrupted parties, that $B \subseteq R(\mathsf{C}')$, and that if the procedure halts then $|B| \leq b$. We also prove an upper-bound on the number of iterations needed for the procedure to halt.

Claim 6.3. *The sub-protocol Π_2 halts after at most $1 + |\mathsf{C}| \cdot d/(b+1)$ iterations.*

Proof. At the end of each iteration, if an output o is in B, then there must exist at least one new publicly-corrupted input $i \in L_o$ that influences o. Since such a party i can influence at most d

Protocol 3 (The extended protocol $\Pi_{M,E,b}$)**.** *Execute Protocol 2, and compute but do not output the values* $\mathsf{C}', B, (y_o)_{o \notin B}, (\alpha_i)_{i \in [n]}$ *as in the last step of the protocol. Initialize an empty set* $\mathscr{Z}$*, and apply the following sub-protocol* Π_2 *as long as the set* B *is larger than* b*:*

1. *Every party* $P_i, i \notin \mathsf{C}'$ *does the following: For every output* $o \in R_i$ *and every publicly corrupted* $j \in L_{o,-i} \cap \mathsf{C}'$*, broadcast all the private randomizers* $r_{o,i,j}, r_{o,j,i}$ *and their commitment keys* $k_{o,i,j}, k_{o,j,i}$ *that were sent to/from the corrupted party* j*. (If these values were sent in the previous iterations there is no need to send them again.) We say that the randomizers are successfully revealed if the opening is consistent with the commitments* $c_{o,i,j}, c_{o,j,i}$*.*

2. ***All:*** *For every output* o *that depends on some publicly corrupted input* $j \in \mathsf{C}' \setminus \mathscr{Z}$*,* ***If*** *for every* $i \in L_o \setminus \mathsf{C}'$ *and* $j \in L_o \cap \mathsf{C}'$ *the randomizers* $r_{o,i,j}, r_{o,j,i}$ *were successfully revealed by* P_i*,* ***Then*** *call* o tentatively ready *and set a* tentative value

$$y'_o = \sum_{i \in L_o \setminus \mathsf{C}'} \left(y_{o,i} - \sum_{j \in L_{o,-i} \cap \mathsf{C}'} r_{o,i,j} + r_{o,j,i} \right).$$

3. ***All:*** *For every publicly corrupted party* $j \in \mathsf{C}'$*,* ***if*** *all the outputs that are influenced by* j *are tentatively ready* ***do****: (a) insert* j *to* $\mathscr{Z}$ *and redefine* $\alpha_j = E_0(0)$*; and (b) update all the affected outputs* $o \in R_j$ *to be* $y_o = y'_o$*.*

4. ***All:*** *Insert to* C' *every party* j *that did not successfully reveal one of its randomizers. (Such a party is now publicly corrupted.) Insert to* B *all the outputs that are influenced by these new publicly-corrupted parties.*

All: Output $B, (y_o)_{o \notin B}$ *and* $(\alpha_i)_{i \notin \mathsf{C}'}$*. (Auxiliary output:* C'*.)*

Figure 3: The extended protocol $\Pi_{M,E,b}$.

outputs, we discover at least $(b+1)/d$ new corrupt parties in each iteration (except for the last one), and the number of iterations is at most $1 + |\mathsf{C}| \cdot d/(b+1)$. □

Hence, when the robustness parameter is $b = 0$, we need a linear number of iterations, and when $b = \beta n$ for a constant β, only a constant number of $O(1/\beta)$ iterations is needed. In fact, even if $b = 0$, the proof shows that the number of rounds scales linearly with the number of (identifiable) corrupted parties. So an adversary can only slow down the process at the expense of revealing the identities of corrupted parties. (Specifically, in an optimistic execution path where all the parties behave honestly, the above extension adds no overhead.)

Intuitively, the security of the protocol relies on the following observations: (1) The information revealed during Π_2 (i.e., the randomizers adjacent to the publicly corrupted parties) does not violate privacy since it is already known to the adversary; and (2) Assuming that the adversary cannot violate the binding of the commitments, the outputs $(y_o)_{o \notin B}$ are consistent with the inputs $(x_i)_{i \in \mathsf{H}}, (x'_i)_{i \in \mathsf{C}}$ where x'_i is either the witness used to generate π_i (for parties that weren't caught cheating), or zero otherwise. Formally, in Appendix E we prove the following theorem.

Theorem 6.4. *Suppose that M is a (d,r)-sparse constraint matrix that together with a recovery vector v forms a b-robust $(t_\mathsf{p},t_\mathsf{c})$-AFS. Then, the protocol $\Pi_{M,E,b}$ t_p-realizes the functionality $\mathscr{F}_{\mathsf{cdsg},b}$.*

Remark 6.5 (Reducing the Communication)**.** *Let d and r be the maximal number of non-zero elements in a column of M and the maximal number of non-zero elements in a row of M, respectively. Observe that each party needs to communicate at most $d(r-1)$ openings during the protocol (since this is the number of randomizers that are "adjacent" to her). So the upstream communication per party is constant. To obtain constant downstream communication, we split the protocol into an online part and a post-processing public-decoding part. In the online part, we apply the first step of the protocol for $T = 1+t_\mathsf{p}\cdot d/(b+1)$ iterations while updating the set C'. (Below we show that this can be done with constant downstream complexity.) In the decoding phase, we iteratively repeat over Steps 2–4 while in each iteration i we use the values that were computed in the ith iteration of the online phase. We terminate the post-processing public-decoding once B is smaller than b, which, by Claim 6.3, takes at most T iterations.*[12]

Let us get back to the online part and analyze the downstream complexity. Assuming that $b=\Omega(n)$, the number of iterations is constant. We will show that the downstream complexity of every party P_i is also constant. Call P_j a neighbor of P_i if they both influence a common output o. Recall that in each iteration P_i has to check, for each of her neighbors P_j, whether P_j publicly cheated, i.e., if $j\in\mathsf{C}'$. Let us denote by C'_k the set of parties that publicly cheated for the first time at the kth iteration where C'_0 is the set of parties that publicly cheated in $\Pi_{M,E}$. At the first iteration, the communication cost of checking if P_j is in C'_0 is constant (since it suffices to check the validity of the proofs sent by P_j during $\Pi_{M,E}$). For $k>0$, the party P_j is in C'_k if (a) P_j is supposed to open a commitment; and (b) the opening is either invalid or was not sent. Condition (b) is easy to verify with $O(1)$ communication (by accessing the opening and verifying against the commitment). Condition (a) boils down to checking whether P_j has a neighbor that publicly cheated in the $k-1$ iteration. Denoting by c_k the downstream communication needed for checking if a party is in C'_k, we have that $c_k = O(D\cdot c_{k-1})$ where $D=d(r-1)$ is the maximal number of neighbors of a party. It follows that the communication in the kth iteration is $O(D^k)$ which is still constant since the number of rounds is constant. Furthermore, the computational complexity of each party is constant as well. (See Section F for a more detailed description.)

By combining Theorem 6.4 with Lemma 5.3 (or Lemma 5.4 for the case of DKG) and with Corollary 4.8, we derive the following corollary (formal version of the main theorem).

Corollary 6.6 (near-threshold DSG and DKG)**.** *Assuming the existence of NIZKs the following holds. For every constants $\tau_\mathsf{p}<\tau_\mathsf{c}$ and every field $\mathbb{F}$ of size super-polynomial $n^{\omega(1)}$, there exists a protocol that $(\tau_\mathsf{p},\tau_\mathsf{c})$ realizes the $\mathscr{F}_\mathsf{dsg}$ functionality (resp., $\mathscr{F}_\mathsf{dkg}$ functionality) over $\mathbb{F}$ in which each party sends and receives only a constant number of field elements and commitments/NIZKs and computes a constant number of arithmetic and cryptographic operations. Moreover, the sharing phase has a constant number of rounds.*

7 Instantiating the Protocols

We describe here a straightforward instantiation of the protocols, using El-Gamal based commitments [27] and simple proofs. The purpose of this instantiation is to demonstrate feasibility rather than optimize efficiency. Accordingly, we will focus on the cryptographic components and leave

[12] In the online phase, we do not update the size of B since this is communication expensive, and therefore just iterate T times.

the instantiation of the combinatorial part of the protocol (i.e., the AFS scheme) for future optimizations. (Some preliminary non-optimized suggestion is given in Remark 7.2 at the end of this section.)

The protocols work over a finite cyclic group $\mathbb{G}$ of order p, and will deal a secret $s \in \mathbb{F}$, where $\mathbb{F} = \mathbb{F}_p$ is a field. The values g, h are generators of $\mathbb{G}$. We assume that the decisional Diffie-Hellman assumption (DDH) holds in $\mathbb{G}$, and that no one knows $\log_g h$. El-Gamal's commitment is defined as follows: given a value x and a key r, both in $\mathbb{F}$, the commitment to x is $\mathsf{Com}(x;r) = (g^r, g^x h^r)$. This commitment is perfectly binding and computationally hiding. The zero-knowledge proofs are obtained by using standard Sigma protocols that can be compiled into NIZK via the aid of Random Oracles (e.g., [22, 8, 23]). (As usual, the identifier of a party should appear as part of the input to the Random Oracle).[13]

7.1 Instantiating the Sharing Phase

The CRS contains $\mathbb{G}, \mathbb{F}_p$ as well as two generators g, h of $\mathbb{G}$. We assume that the AFS is specified by a matrix M and an extraction vector v. Each party P_i samples random inputs (x_i, ρ_i). We instantiate the Basic Protocol (Protocol 2) as follows.

R1: (Sending Randomizers) For every output $o \in R_i$ and every $j \in L_{o,-i}$, party P_i samples in $\mathbb{F}$ a random mask $r_{o,i,j}$ and a random commitment key $k_{o,i,j}$, and broadcasts the commitment $c_{o,i,j} = \mathsf{Com}(r_{o,i,j}; k_{o,i,j}) = (g^{k_{o,i,j}}, g^{r_{o,i,j}} h^{k_{o,i,j}})$. P_i also sends the opening $(r_{o,i,j}, k_{o,i,j})$ of the commitment to P_j.

R2: (Resolving Private Inconsistencies) For every $o \in R_i, j \in L_{o,-i}$, if P_i does not receive from P_j an opening that is consistent with the published commitment $c_{o,j,i}$, then P_i broadcasts a "complaint" (j, o). This complaint asserts that $r_{o,j,i}$ and $k_{o,j,i}$ should be set to zero and $c_{o,j,i} = \mathsf{Com}(0;0)$.

R3: (Local Sums and Output Tags) P_i broadcasts $\alpha_i = E_{\rho_i}(x_i) = \mathsf{Com}(x_i, \rho_i)$. The ZKPOK π_i for proving the consistency of this value is standard. (Publish a random commitment $\alpha'_i = \mathsf{Com}(x'_i, \rho'_i)$ and, given a challenge $\beta \in \mathbb{F}$, send the pair $(x'_i + \beta x_i, \rho'_i + \beta \rho_i)$. To verify check that the sent pair is a valid opening of the commitment $\alpha_i^{\beta} \cdot \alpha'_i$.)

In addition, for every output $o \in R_i$, party P_i broadcasts the value

$$y_{o,i} = x_i + \sum_{j \in L_{o,-i}} r_{o,i,j} - \sum_{j \in L_{o,-i}} r_{o,j,i}.$$

To prove that $y_{o,i}$ agrees with the committed value in α_i and the committed randomizers in $(c_{o,i,j}, c_{o,j,i})_{j \in L_{o,-i}}$, we first use the linear homomorphism of El-Gamal's commitments to combine all the commitments of the right-hand side into a single commitment $c_{o,i} = (c_{o,i}[1], c_{o,i}[2])$ and then prove the knowledge of a key $r_{o,i}$ that opens the commitment to $y_{o,i}$. The latter can be done via the Chaum-Pedersen Sigma protocol [17] for proving that $(g, h, c_{o,i}[1], c_{o,i}[2]/g^{y_{o,i}})$ is a DH tuple.

(Optimization) P_i needs to broadcast a value $y_{o,i}$ and a proof $\pi_{o,i}$ for every output $o \in R_i$. To reduce communication and improve run time through multi-exponentiations, P_i can batch these proofs by using random coefficients ρ_o (chosen, again, by the random oracle) for every $o \in R_i$, and proving that the commitment $\prod_{o \in R_i} c_{o,i}^{\rho_o}$ opens to $\sum_{o \in R_i} \rho_o y_{o,i}$.

[13] While Fiat-Shamir is somewhat problematic when it is needed to ensure adaptive knowledge-extraction based simulation soundness (SS) (see, e.g., [9]), we can rely on the weaker variants mentioned in Remark B.2. Specifically, we only need weak non-adaptive SS for Theorems 6.1 and 6.4 (by Remark D.3), and only SS without knowledge extraction for Lemmas 5.3 and 5.4 (by Remark C.2 and since we use perfectly-binding commitments). It is known that Fiat-Shamir compiled Sigma protocols satisfy these notions, see, e.g., [10, 7].

Computing the Outputs: The private output of the ith party is taken to be its private inputs (x_i, ρ_i). The public outputs can be computed by all parties based on the information that was broadcasted as follows.[14] If for some party i, the proof π_i fails to verify, set $\alpha_i = E_0(0)$ and replace π_i with a valid proof. (This is trivial to do, since we know that $\alpha_i = (g^0, g^0 h^0)$.) In addition, initialize $C', B = \emptyset$. For every output $o \in [m]$: **If** there exists $j \in L_o$ for which the proof $\pi_{o,j}$ fails to verify insert o to B and j to C'. **Otherwise**, set $y_o = \sum_{j \in L_o} y_{o,j}$. **Output** $B, (y_o)_{o \notin B}$ and $(\alpha_i)_{i \in [n]}$. (One can also output C' and apply some penalty mechanism to the parties in C'.)

Recovering the public key for a DKG: In a DKG application, as defined in Functionality 1, the sharing protocol must publish g raised to the power of the secret. This can be implemented using the method described in the next section (Section 7.2) for recovering the secret in the exponent, by replacing the computation of $H(m)$ to the power of the secret with computing g to the power of the secret. (As mentioned in Section 5.3, this step is not needed if the parties run a *canonical DKG*, set $E_{\rho_i}(x_i) = g^{x_i}$, and take care of the subtleties discussed in Section 5.3.)

Remark 7.1 (Other variants)**.** *Assuming a PKI (instantiated with, say, El-Gamal encryption), one can turn the above protocol into a two-round protocol as described in Section 6.2. We can also realize the extended protocol from Section 6.3 on top of the current instantiation in a straightforward way. (Recall that the extension only requires openning commitments for randomizers).*

Remark 7.2 (Consrete instantiation of the robust-AFS)**.** *Recall that in the second item in Example 4.11 we showed that when $|\mathbb{F}| \geq 2^{255}$ and $n > 50$, we can get a $(4, 10)$-sparse matrix with $\tau_p = 0.39$, $\tau_c = 0.9$ (by taking $\mu = 0.6$). For (relative) robustness of $\beta = 0.029$, we get, by Lemma 4.13, a privacy threshold of $\tau'_p = 0.39 - 10 \cdot 0.029 = 0.1$. Plugging this into the extended protocol (Protocol 3), yields a DKG with a privacy threshold of $\tau'_p = 0.1$, correctness threshold of $\tau_c = 0.9$, optimistic round complexity of, say, 2, and additional $\tau'_p \cdot d/\beta < 14$ rounds in the worst case.*

7.2 Instantiating the Recovery Phase

Recovering the secret itself:

- Each party broadcasts its private input x_i, and proves in ZK via the Chaum-Pedersen proof that he knows an opening ρ_i that opens the commitment α_i to g^{x_i}.
- Each party that wishes to compute the output verifies all published proofs.
- Let $T \subset [n]$ be the set of indices $i \in [n]$ for which theses proofs pass verification. A party that wishes to compute the output does the following. (1) retrieve the public outputs B and $(y_o)_{o \notin B}$, compute the set A based on B and M using the robust-AFS algorithm[15] and set

$$M' = M[\bar{A}; \bar{B}], \quad y' = y[\bar{B}], \quad \text{and } v' = v[\bar{A}] \tag{3}$$

  Next, (2) call the linear recovery algorithm $\mathsf{Rec}_{M',y',v'}(T, x[T])$ and output the result. (Recall that the recovery algorithm expresses the missing shares as a linear combination of the existing shares, and outputs the multiplication of v' by the vector of shares.) Since all computation is deterministic and is based on data that was publicly broadcast, anyone can individually compute the recovery algorithm and arrive at the same result.

[14] This computation can be postponed to the recovery phase.

[15] In our AFS constructions, this is simply the set A of columns in M whose support intersects with B.

Recovering the secret in the exponent: Many applications of threshold cryptography do not recover the secret itself. Instead, the participants receive an input m, compute $H(m) \in \mathbb{G}$, and must jointly compute $H(m)$ raised to the power of the shared secret. This can be done in the following way:

- Each P_i broadcasts $s_i = H(m)^{x_i}$, and proves in ZK that it knows x_i, ρ_i such that $\alpha_i = (g^{\rho_i}, g^{x_i} \cdot h^{\rho_i})$ and $s_i = H(m)^{x_i}$. (One can easily design an appropriate Sigma protocol for this statement, see, e.g., [12, Section 19.5.3].)
- Each party that wishes to compute the output verifies all these proofs that were published.
- Let $T \subset [n]$ be the set of indices $i \in [n]$ for which theses proofs pass verification. Each party that wishes to compute the output defines (M', y', v') as in Eq. (3) and computes the linear recovery algorithm $\mathsf{Rec}_{M',y',v'}(T, x[T])$ *in the exponent* and outputs the result. Namely, Rec searches for a row vector $\alpha \in \mathbb{F}^m$ such that $\alpha \cdot M'[\ ; \bar{T}] = v'[\bar{T}]$, and outputs $\prod_{i \in T} s_i^{v'_i} \cdot H(m)^{\alpha \cdot y'}$. (In other words, we know that $\sum_{j \in \bar{T}} v'_j x_j = \alpha \cdot y'$, and therefore $\prod_{i \in [n]} H(m)^{x_i v'_i} = \prod_{i \in T} s_i^{v'_i} \cdot H(m)^{\alpha \cdot y'}$.)

  As before, all computation is deterministic and is based on data that was publicly broadcast, and therefore anyone can individually compute the recovery algorithm and arrive at the same result.

Acknowledgements. We thank Noga Ron-Zewi for helpful discussions and the anonymous TCC reviewers for their comments.

References

[1] Benny Applebaum and Eliran Kachlon. Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. *SIAM J. Comput.*, 52(6):1321–1368, 2023.

[2] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Verifiable relation sharing and multi-verifier zero-knowledge in two rounds: Trading nizks with honest majority - (extended abstract). In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, pages 33–56, 2022.

[3] Benny Applebaum, Oded Nir, and Benny Pinkas. How to recover a secret with o(n) additions. In *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, pages 236–262, 2023.

[4] Renas Bacho, Christoph Lenzen, Julian Loss, Simon Ochsenreither, and Dimitrios Papachristoudis. GRandLine: Adaptively secure DKG and randomness beacon with (log-)quadratic communication complexity. Cryptology ePrint Archive, Paper 2023/1887, 2023.

[5] Michael Backes, Aniket Kate, and Arpita Patra. Computational verifiable secret sharing revisited. In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 590–609. Springer, 2011.

[6] Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure MPC with linear communication complexity. In *Theory of Cryptography Conference, TCC 2008*, volume 4948 of *LNCS*, pages 213–230. Springer, 2008.

[7] Mihir Bellare. Lectures on NIZKs: A concrete security treatment, 2021.

[8] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.

[9] David Bernhard, Marc Fischlin, and Bogdan Warinschi. Adaptive proofs of knowledge in the random oracle model. *IET Inf. Secur.*, 10(6):319–331, 2016.

[10] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. *IACR Cryptol. ePrint Arch.*, page 771, 2016.

[11] Andrej Bogdanov, Siyao Guo, and Ilan Komargodski. Threshold secret sharing requires a linear-size alphabet. *Theory Comput.*, 16:1–18, 2020.

[12] Dan Boneh and Victor Shoup. A graduate course in applied cryptography, 2023. Version 6.

[13] Ran Canetti. Universally composable security. *J. ACM*, 67(5):28:1–28:94, 2020.

[14] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *CRYPTO '99*, volume 1666 of *LNCS*, pages 98–115. Springer, 1999.

[15] Michael R. Capalbo, Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 659–668. ACM, 2002.

[16] Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to dkg and yoso. *Cryptology ePrint Archive*, 2023.

[17] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.

[18] Ronald Cramer, Ivan Bjerre Damgård, Nico Döttling, Serge Fehr, and Gabriele Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 313–336. Springer, 2015.

[19] Sourav Das, Benny Pinkas, Alin Tomescu, and Zhuolun Xiang. Distributed randomness using weighted vrfs. Cryptology ePrint Archive, Paper 2024/198, 2024. `https://eprint.iacr.org/2024/198`.

[20] DKGPG developers. DKGPG: Distributed Key Generation for Pretty Good Privacy (PGP). https://www.nongnu.org/dkgpg/, 2017. Accessed: February 14, 2024.

[21] drand. drand: Distributed Randomness Beacon Service. https://github.com/drand/drand, 2020. Accessed: February 14, 2024.

[22] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[23] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2005.

[24] Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 121–136. Springer, 1998.

[25] Pierre-Alain Fouque and Jacques Stern. One round threshold discrete-log key generation without private channels. In *International Workshop on Public Key Cryptography*, pages 300–316. Springer, 2001.

[26] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21–28, 1962.

[27] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985.

[28] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure applications of pedersen's distributed key generation protocol. In *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003*, volume 2612 of *LNCS*, pages 373–390. Springer, 2003.

[29] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, 2007.

[30] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[31] Shafi Goldwasser, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[32] Jens Groth. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive*, 2021.

[33] Jens Groth and Victor Shoup. Design and analysis of a distributed ecdsa signing service. Cryptology ePrint Archive, Paper 2022/506, 2022. https://eprint.iacr.org/2022/506.

[34] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In *EUROCRYPT 2021*, volume 12696 of *LNCS*, pages 147–176. Springer, 2021.

[35] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[36] Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *CRYPTO 2014*, volume 8617 of *LNCS*, pages 369–386. Springer, 2014.

[37] Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. Non-interactive vss using class groups and application to dkg. *Cryptology ePrint Archive*, 2023.

[38] Jonathan Katz. Round optimal robust distributed key generation. *IACR Cryptol. ePrint Arch.*, page 1094, 2023.

[39] Jonathan Katz, Rafail Ostrovsky, and Michael O. Rabin. Identity-based zero knowledge. In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, volume 3352 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2004.

[40] Chelsea Komlo, Ian Goldberg, and Douglas Stebila. A formal treatment of distributed key generation, and new constructions. *IACR Cryptol. ePrint Arch.*, page 292, 2023.

[41] Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. LDPC codes achieve list decoding capacity. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 458–469. IEEE, 2020.

[42] Moni Naor. Bit commitment using pseudo-randomness. In *CRYPTO '89*, volume 435 of *LNCS*, pages 128–136. Springer, 1989.

[43] Orbs-Network. DKG-on-EVM: A Distributed Key Generation Protocol for Ethereum Virtual Machine. https://github.com/orbs-network/dkg-on-evm, 2018. Accessed: February 14, 2024.

[44] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[45] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, 2001.

[46] Philipp Schindler. ethdkg: An Ethereum-based DKG Implementation. https://github.com/PhilippSchindler/ethdkg, 2020. Accessed: February 14, 2024.

[47] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. Ethdkg: Distributed key generation with ethereum smart contracts. Cryptology ePrint Archive, Paper 2019/985, 2019. https://eprint.iacr.org/2019/985.

[48] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[49] Nibesh Shrestha, Adithya Bhat, Aniket Kate, and Kartik Nayak. Synchronous distributed key generation without broadcasts. *IACR Cryptol. ePrint Arch.*, page 1635, 2021.

[50] H. Stamer. Gnosis dkg: A Distributed Key Generation Library. https://github.com/gnosis/dkg, 2018. Accessed: February 14, 2024.

[51] Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012.

[52] Lior Zichron. Locally computable arithmetic pseudorandom generators. Master thesis, Tel Aviv University, 2017. Available at https://www.bennyapplebaum.sites.tau.ac.il/_files/ugd/f706bf_501515c9cd7744c498935684bd1648a2.pdf.

A The Necessity of Public Headers

Fix some constants $0 < \tau_{\mathsf{p}} < \tau_{\mathsf{c}} < 1$ and let Π be a $(\tau_{\mathsf{p}} n, \tau_{\mathsf{c}} n)$ DSG protocol that takes randomness r_i from each party P_i and delivers a share x_i to each party. We assume that there is no *public header*, i.e., the shares $(x_i)_{i \in T}$ of any t_{c}-subset T uniquely determine the secret s. We also assume that there is no external source of randomness, i.e., the secret is fully determined by the randomness $(r_1, \ldots, r_n)$. We prove that in this case, the protocol suffers from the information bottleneck mentioned in the introduction. That is, there are $\Omega(n)$ parties ("influencers") whose inputs influence the output of at least $\Omega(n)$ parties. Formally, consider a digraph G over n parties where we put an edge from P_i to P_j, denoted $i \to j$, if there exists a pair of input vectors $r = (r_k)_{k \in [n]}$ and $r' = (r'_k)_{k \in [n]}$ that differ only in their ith location (i.e., $r_k = r'_k$ for $k \neq i$) such that the share y_j delivered to P_j in an honest execution with the inputs r is different from the share y'_j that is delivered to P_j in an honest execution with the inputs r'. We will show that there are $\Omega(n)$ influential nodes with out-degree of $\Omega(n)$.

For every authorized set S of size at least t_{c}, let $N_S = \{i : \exists j \in S, i \to j\}$ denote the set of parties that influence the shares of S. By correctness, the secret depends only on the random tape of the parties that belong to $N^* = \bigcap_S N_S$. (If a party $i \notin N^*$ affects the secret, then there exists a recovery set S whose shares do not depend on the randomness of i and so correctness is violated.) Put differently, each party i in N^* talks to a set M_i that contains at least a single party in each recovery set. We conclude that i talks to at least $|M_i| > n - t_{\mathsf{c}} = \Omega(n)$ parties. Otherwise, if $|M_i| < n - t_{\mathsf{c}}$ party i misses the complement $\bar{M}_i$ of M_i which is a set of size t_{c} and so it is an authorized set. Finally, by privacy, N^* must be larger than $t_{\mathsf{p}} = \Omega(n)$, and the claim follows.

Note that the above argument applies even when the protocol achieves only a weak level of semi-maliciously security. (That is, security holds only against corrupted parties that choose their inputs arbitrarily but follow the protocol honestly.)

B Omitted Preliminaries

Non-Interactive commitment. A non-interactive commitment $\mathsf{Com}_{\mathsf{crscm}}(x; k)$ is an efficient algorithm that takes a random reference string crscm sampled from some efficiently samplable distri-

bution $D(1^\kappa)$, a message x and a random commitment key k and outputs a commitment string α. **Hiding** asserts that for any pair of messages x, x', and randomly chosen crscm and key k the commitments $\mathsf{Com}_{\mathsf{crscm}}(x;k)$ and $\mathsf{Com}_{\mathsf{crscm}}(x';k)$ are computationally indistinguishable. **Binding** asserts that for a randomly chosen crscm, except with negligible probability, no efficient algorithm can find a pair of messages $x \neq x'$ and a pair of keys k, k' for which $\mathsf{Com}_{\mathsf{crscm}}(x;k) = \mathsf{Com}_{\mathsf{crscm}}(x';k)$. Such commitments can be constructed based on one-way functions [35, 42].

The following formalization of NIZK is taken from [38].

Definition B.1 (Non-Interactive Zero-Knowldge)**.** *For an NP-relation R a NIZK is a tuple of efficient algorithms $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{KE})$ that satisfy the following requirements.*

Completeness *asserts that for all $(x,w) \in R$, all identities $i \in [n]$ and every security parameter κ it holds that*

$$\Pr_{\mathsf{crspf} \xleftarrow{R} \mathsf{CRSGen}(1^\kappa)} [\mathsf{V}(\mathsf{crspf}, i, x, \mathsf{P}(\mathsf{crspf}, i, x, w)) = 1] = 1,$$

Adaptive, multi-theorem zero-knowledge *asserts that for every efficient adversary $\mathscr{A}$ the following quantity is negligible*

$$\left| \Pr_{\mathsf{crspf} \xleftarrow{R} \mathsf{CRSGen}(1^\kappa)} [\mathscr{A}^{\mathsf{P}^*(\mathsf{crspf}, \cdot)}(\mathsf{crspf})] - \Pr_{(\mathsf{crspf}, \mathsf{tdpf}) \xleftarrow{R} \mathsf{Sim}_1(1^\kappa)} [\mathscr{A}^{\mathsf{Sim}_2^*(\mathsf{tdpf}, \cdot)}(\mathsf{crspf})] \right|,$$

where $\mathsf{P}^*(\mathsf{crspf}, i, x, w)$ returns $\mathsf{P}(\mathsf{crspf}, i, x, w)$ if $(x,w) \in R$ and $\perp$ otherwise and $Sim_2^*(\mathsf{tdpf}, i, x)$ returns $Sim_2(\mathsf{tdpf}, i, x)$ if $(x,w) \in R$ and $\perp$ otherwise.

Identity-based simulation soundness *(SS) asserts every efficient adversary $\mathscr{A}$ wins in the following game with at most negligible probability.*

1. *The Challenger samples $(\mathsf{crspf}, \mathsf{tdpf}) \xleftarrow{R} \mathsf{Sim}_1(1^\kappa)$ and a challenge bit $b \xleftarrow{R} \{0,1\}$ and sends crspf to the adversary who specifies a set of "honest" identities $\mathsf{H} \subset [n]$.*

2. *The adversary is given an access to a prover oracle and a verification oracle. The former takes an input (i,x,w) and returns a simulated proof $\mathsf{Sim}_2(\mathsf{tdpf}, i, x)$ if $i \in \mathsf{H}$ and $(x,w) \in R$; otherwise, it returns $\perp$. The verification oracle takes an input (i,x,π) and returns $\perp$ if $i \in \mathsf{H}$ or $\mathsf{V}(\mathsf{crspf}, i, x, \pi) = 0$. Otherwise it returns 1 if either $b = 1$ and the witness extractor $\mathsf{KE}(\mathsf{tdpf}, i, x, \pi)$ extracts a valid witness w such that $(x,w) \in R$, or if $b = 0$.*

3. *At the end, the adversary outputs b' and wins the game if $b' = b$.*

Remark B.2 (weaker variants)**.** *The above formulation is somewhat strong and is adopted for the sake of simplicity. We will later show that the following weaker variants of* simulation soundness *suffices for our purposes:*

- ***Identity-based simulation soundness without extraction****: defined similarly to SS except that we do not require knowledge extraction. Accordingly, the condition "$\mathsf{KE}(\mathsf{tdpf}, i, x, \pi)$ extracts a valid witness w such that $(x,w) \in R$" is replace with the* validity condition *there is no witness w for which $(x,w) \in R$.*

- *Alternatively, we will use* ***Non-Adaptive identity-based simulation soundness*** *that is defined similarly to SS except that the adversary operates in two phases: First she calls the prover oracles (multiple times), and then she makes a multiple parallel calls to the verification oracle. Notably, the*

calls to the verification oracle are non-adaptive*, i.e., independent of its answers. A closely related notion (without identifiers) was studied in [10] and is referred to as* Simulation Sound Extractability*.*

C Proofs of Lemmas 5.3 and 5.4

C.1 Proof of Lemma 5.3

By standard MPC composition theorems (e.g., [30, 13]), it suffices to prove that the 2-phase functionality $\mathscr{F}_{\mathsf{dsg}}$ is $(t_{\mathsf{p}}, t_{\mathsf{c}})$-realized by the protocol Π_{dsg} that makes a single call to the ideal functionality $\mathscr{F}_{\mathsf{cdsg},b}$ followed by a canonical recovery protocol. Indeed, let $\mathscr{A}$ be an adversary that attacks Π_{dsg}. Then, we construct a straight-line black-box simulator Sim that attacks $\mathscr{F}_{\mathsf{dsg}}$ as follows.

1. (Setup) The simulator Sim uses the first-phase simulator Sim_1 for the NIZK system to sample CRS/trapdoor pairs, $(\mathsf{crspf}, \mathsf{tdpf})$ and sends $\mathsf{crs} = (\mathsf{crspf}, \mathsf{crscm})$ to $\mathscr{A}$ who responds by specifying a set of corrupted parties $\mathsf{C} \subset [n]$ of size at most t_{p} where $\mathsf{C}' \subset \mathsf{C}$ are only passively corrupted.

2. The simulator samples a random vector y' in the image of M_{H} (i.e., $y' = M_{\mathsf{H}} \cdot x'_{\mathsf{H}}$ for randomly sampled x'_{H}). For each honest party $i \in \mathsf{H}$, sample a fresh zero commitment $\alpha_i = E_{\rho_i}(0)$ where ρ_i is a random commitment key. Send y' and $(\alpha_i)_{i \in \mathsf{H}}$ to $\mathscr{A}$.

3. The adversary responds with $x_{\mathsf{C}} = (x_i)_{i \in \mathsf{C}}$, $\rho_{\mathsf{C}} = (\rho_i)_{i \in \mathsf{C}}$ and a set $B \subset [m]$ of size at most b. The simulator computes $y = y' + M_{\mathsf{C}} \cdot x_{\mathsf{C}}$ and $\alpha_i = E_{\rho_i}(x_i)$ for $i \in \mathsf{C}$ and returns $B, y[\bar{B}], (\alpha_i)_{i \in [n]}$. In addition, the simulator invokes the sharing phase of $\mathscr{F}_{\mathsf{dsg}}$.

4. If the client (possibly controlled by the adversary) invokes the recovery phase of Π_{dsg}, then the recovery phase of $\mathscr{F}_{\mathsf{dsg}}$ is also invoked and we get the secret s. We compute the set A based on the set B specified by the adversary, and sample shares for the honest parties $x_{\mathsf{H}} = (x_i)_{i \in \mathsf{H}}$ subject to $y' = M_{\mathsf{H}} \cdot x_{\mathsf{H}}$ and subject to $v[\bar{A}] \cdot x = s$ where $\cdot$ stands for inner-product and $x \in \mathbb{F}^n$ is the vector of shares obtained by concatenating x_{H} with x_{C} that was chosen by the adversary. We send to the adversary the messages $(x_i, \pi_i)_{i \in \mathsf{H}}$ that the honest parties send in the recovery phase where π_i is a simulated (fake) NIZK for the commitment relation $\mathscr{R} = \{((\alpha, x), \rho) : \alpha = E_\rho(x)\}$.[16] The adversary responds with some values $(x'_i, \pi'_i)_{i \in \mathsf{C}}$ where for passively corrupted parties $i \in \mathsf{C}'$ it holds that $x'_i = x_i$ and π'_i is honestly generated (and passes verification). The adversary selects a subset of the honest parties $\mathsf{H}' \subset \mathsf{H}$ such that $|\mathsf{H}'| + |\mathsf{C}'| \geq t_{\mathsf{c}}$ and resets the messages of the honest parties $\mathsf{H} \setminus \mathsf{H}'$ outside H' to $\perp$. We terminate the simulation by outputting the adversary's view.

Let $\mathscr{H}_1$ denote the output of the simulated experiment that consists of the output of the simulator (when interacting with $\mathscr{F}_{\mathsf{dsg}}$) concatenated with the secret s that the $\mathscr{F}_{\mathsf{dsg}}$ delivers to the

[16] Here we see why we cannot just open the commitment in the recovery phase: The committed value is unknown to the simulator when the commitment is generated. We further note that the problem is avoided if one uses the weaker variant of DSG/DKG that is mentioned in Remark 5.1. In this case, it suffices to simulate only the sharing phase, and so we can just open the commitments in the recovery protocol. The indistinguishability property specified in Remark 5.1 then follows by the hiding and binding properties of the commitment. Unlike the current construction, this variant can be constructed without employing NIZK and without a CRS.

honest parties in the reconstruction phase. Let $\mathscr{H}_R$ be the output of the real experiment that corresponds to the case where $\mathscr{A}$ attacks the two-phase protocol $\Pi_{\mathsf{dsg}} = (\mathscr{F}_{\mathsf{cdsg},b}, \Pi_{\mathsf{Rec}})$. Specifically, $\mathscr{H}_R$ consists of the output of $\mathscr{A}$ concatenated with the public output s' of Π_{Rec}. To show that $\mathscr{H}_I$ is computationally indistinguishable from $\mathscr{H}_R$ we will use several hybrids.

The hybrid $\mathscr{H}_1$ is identical to the $\mathscr{H}_R$, except that the CRS is generated with a trapdoor and the proofs $(\pi_i)_{i\in\mathsf{H}}$ in the recovery phase are generated by using the NIZK simulator. By the security of the NIZK proofs, $\mathscr{H}_1$ is computationally indistinguishable from $\mathscr{H}_R$.

The hybrid $\mathscr{H}_2$ is identical to the $\mathscr{H}_1$, except that commitments $(\alpha_i)_{i\in\mathsf{H}}$ generated by $\mathscr{F}_{\mathsf{cdsg},b}$ are replaced by commitments to zeroes. By the hiding property of the commitments, $\mathscr{H}_2$ is computationally indistinguishable from $\mathscr{H}_1$.

To complete the proof of the theorem, it suffices to prove the following claim.

Claim C.1. *The output of $\mathscr{H}_2$ is statistically close to the output of the ideal experiment $\mathscr{H}_I$.*

From now on, we focus on the proof of the claim. First, observe that the first message that the adversary receives $y', (\alpha_i)_{i\in\mathsf{H}}$ is distributed identically in both experiments. Let us fix these values. Consequently, the values $B, x_{\mathsf{C}}, \rho_{\mathsf{C}}$ sent by the adversary are also distributed identically in both experiments, and we can fix them as well, and move on to the recovery phase.

We show that in both experiments the vector of honest shares x_{H} is distributed identically. Recall that in $\mathscr{H}_2$ the vector x_{H} is uniform subject to (1) $y' = M_{\mathsf{H}} \cdot x_{\mathsf{H}}$ whereas in $\mathscr{H}_I$ the vector x_{H} is uniform subject to (1) and to (2) $v[\bar{A}] \cdot x = s$ where s is uniformly distributed. Let X denote the set of solutions to (1), and for any fixing of s, let X_s denote the subset of X that satisfies (2) for this choice of s. By the t_{p}-privacy of the scheme $(M, y, v[\bar{A}])$, the sets X_s form a partition of X into sets of equal size, and therefore the vector x_{H} is uniformly distributed over X in both cases.

Let us fix x_{H} in both experiments. Since the view of the adversary is now identical in both experiments, it remains to show that the secret s that is delivered by the ideal functionality in $\mathscr{H}_I$ equals the secret s' that is recovered in $\mathscr{H}_2$. Let us condition on the good event, G, that $\forall i \in \mathsf{C}$ if the tuple (x'_i, π'_i, α_i) passes verification then $x'_i = x_i$. In this case, the secret s' is obtained by applying the recovery algorithm $\mathsf{Rec}_{(M[\bar{A};\bar{B}], y[\bar{B}], v[\bar{A}])}$ over a vector of shares in the support of $\Sigma_{M,y,v[\bar{A}]}$. Since the vector consists of at least t_{c} shares, recovery succeeds and $s' = v[\bar{A}] \cdot x = s$, as required.

Finally, we argue that the good event G happens with all but negligible probability. Indeed, if G happens we can use the knowledge extraction algorithm to retrieve, except with negligible probability, a witness ρ'_i such that $E_{\rho'_i}(x'_i) = \alpha_i = E_{\rho_i}(x_i)$ and violate the binding property of the commitment. This completes the proof of the claim and the proof of the lemma. □

Remark C.2 (Relaxing the proof-of-knowledge property)**.** *A closer look at the proof of Claim C.1 shows that if the commitment E is perfectly binding (as opposed to computationally binding) then the knowledge-extractor is not needed and standard soundness suffices. Since this is the only use of knowledge extraction in the proof, we conclude that the Lemma 5.3 holds even when the underlying NIZK satisfies Identity-based simulation soundness without extraction, provided that E is perfectly binding.*

C.2 Proof of Lemma 5.4

The proof of Lemma 5.4 is very similar to the proof of Lemma 5.4, we highlight the main differences.

By standard MPC composition theorems (e.g., [30, 13]), it suffices to prove that the 2-phase functionality $\mathscr{F}_{\mathsf{dkg}}$ is $(t_{\mathsf{p}}, t_{\mathsf{c}})$-realized by the protocol Π_{dkg} whose first phase consists of a single call

to the ideal functionality $\mathscr{F}_{\mathsf{cdsg},b}$ followed by a canonical recovery in the exponent protocol for the generator g (with input as in Lemma 5.4) and its second phase consists of a canonical recovery in the exponent protocol for an element h. Letting $\mathscr{A}$ be an adversary that attacks Π_{dkg}, we construct a straight-line black-box simulator Sim that attacks $\mathscr{F}_{\mathsf{dkg}}$ as follows.

1. The first three steps are exactly as in the proof of Lemma 5.4 (Section C.1). Except that at the end of Step 3, when the client invokes the sharing phase of $\mathscr{F}_{\mathsf{dkg}}$, the functionality returns the value g^s. We proceed as follows.

4. We compute the set A based on the set B specified by the adversary (in Step 3), and sample a vector of exponentiated shares $(g_i = g^{x_i})_{i\in\mathsf{H}}$ for the honest parties subject to $y' = M_{\mathsf{H}} \cdot x_{\mathsf{H}}$ and subject to $v[\bar{A}] \cdot x = s$ where $\cdot$ stands for inner-product and $x \in \mathbb{F}^n$ is the vector of shares obtained by concatenating x_{H} with the vector x_{C} that was chosen by the adversary in the third step. This is done efficiently via the following standard technique. View the constraints as a linear system over the formal variables (x_{H}, s). Since the system is consistent (as follows from the analysis of the simulation) and since s participates in a single constraint, we can treat s as a "free variable" and locate a set of additional free variables $x_F, F \subset \mathsf{H}$ such that every non-free variable $x_i, i \in \mathsf{H} \setminus F$ can be written as a linear combination v_i of (s, x_F). Now to sample a solution in the exponent of g, we sample x_F at random, set $g_i = g^{x_i}$ for free variables $i \in F$, and for non-free variables, $i \in \mathsf{H} \setminus F$, set $g_i = (g^s, g_F)^{v_i}$. (Here we use a power-product notation: for a vector of group elements $\alpha = (\alpha_1, \ldots, \alpha_k)$ and vector of field elements $v = (v_1, \ldots, v_k)$ we write α^v to denote $\prod_i \alpha_i^{v_i}$.) We send to the adversary the messages $(g_i, \pi_i)_{i\in\mathsf{H}}$ that the honest parties send in the recovery phase where π_i is a simulated (fake) NIZK for the relation $\mathscr{R} = \left\{((\alpha, \beta, g), (\rho, x)) : \alpha = E_\rho(x), \beta = g^x\right\}$. The adversary responds with some values $(g_i, \pi_i')_{i\in\mathsf{C}}$ where for passively corrupted parties $i \in \mathsf{C}'$ it holds that $g_i = g^{x_i}$ and π_i' is an honestly-generated proof (that passes verification).

5. When the client (possibly controlled by the adversary $\mathscr{A}$) invokes the recovery phase of Π_{dkg} with group element h, we call the recovery phase of $\mathscr{F}_{\mathsf{dkg}}$ and get h^s. We proceed exactly as in the previous step with the same values of the variables x_{H}. That is, set $h_i = h^{x_i}$ for a free variable $i \in F$ and $h_i = (h^s, h_F)^{v_i}$ for a non-free variable $i \in \mathsf{H} \setminus F$, and send to the adversary the messages $(h_i, \pi_i)_{i\in\mathsf{H}}$ that the honest parties send in the recovery phase where π_i is a simulated (fake) NIZK for the relation $\mathscr{R}$. The adversary responds with some values $(h_i, \pi_i')_{i\in\mathsf{C}}$ where for passively corrupted parties $i \in \mathsf{C}'$ it holds that $h_i = g^{x_i}$ and π_i' is an honestly-generated proof (that passes verification). The adversary also selects a subset of the honest parties $\mathsf{H}' \subset \mathsf{H}$ such that $|\mathsf{H}'| + |\mathsf{C}'| \geq t_{\mathsf{c}}$ and resets the messages of the honest parties $\mathsf{H} \setminus \mathsf{H}'$ outside H' to $\perp$. We terminate the simulation by outputting the adversary's view. We terminate the simulation by outputting the adversary's view.

The analysis of the simulator is similar to the analysis of Lemma 5.4 (see Section C.1). We note that Remark C.2 applies here as well.

D Proof of Theorem 6.1

We describe a simulator Sim that interacts with an adversary $\mathscr{A}$. We note that whenever $\mathscr{A}$ passively corrupts a party P_i our simulator also passively corrupts P_i, thus security against mixed adversaries follows.

- **Setup:** The simulator Sim uses the first-phase simulator Sim_1 for the NIZK system to sample CRS/trapdoor pairs, $(\mathsf{crspf}, \mathsf{tdpf})$. The simulator also samples a CRS crscm for the commitment. We send $\mathsf{crs} = (\mathsf{crspf}, \mathsf{crscm})$ to $\mathscr{A}$ who specifies a set $\mathsf{C} \subset [n]$ of corrupted parties of size at most t'_{p}.

- **R1:** The simulator generates the randomizers of the honest parties just like in the protocol; That is, for each $i \notin \mathsf{C}$, $o \in R_i$ and $j \in L_{o,-i}$: Sample a random mask $r_{o,i,j} \xleftarrow{R} \mathbb{F}$ and a random commitment key $k_{o,i,j}$, send to $\mathscr{A}$ the "broadcast values" $c_{o,i,j} = \mathsf{Com}(r_{o,i,j}; k_{o,i,j})$, and, for every $j \in \mathsf{C}$, the opening $(r_{o,i,j}, k_{o,i,j})$ as private-channel messages. The adversary $\mathscr{A}$ returns the tuple of (supposedly) committed randomizers

$$(c_{o,i,j})_{i\in\mathsf{C}, o\in R_i, j\in L_{o,-i}}, (r_{o,i,j}, k_{o,i,j})_{i\in\mathsf{C}, o\in R_i, j\in L_{o,-i}\setminus\mathsf{C}}.$$

- **R2:** Raise complaints of honest parties on corrupted parties as in the protocol; That is, for each $i \notin \mathsf{C}$, for every $o \in R_i, j \in L_{o,-i} \cap \mathsf{C}$, if the opening $k_{o,i,j}$ (generated by the adversary) is inconsistent with the published commitment $c_{o,j,i}$, send to $\mathscr{A}$ a "complaint" (i, j, o) and update $r_{o,j,i}$ to zero and $c_{o,j,i} = \mathsf{Com}(0;0)$, $k_{o,j,i} = 0$. The adversary responds with a (possibly empty) list of complaints, for each such complaint (i, j, o) if $i \in \mathsf{C}, i, j \in R_o$ update $r_{o,j,i}$ to zero.

- **R3:** The simulator calls the ideal functionality $\mathscr{F}_{\mathsf{cdsg},b}$ and receives $(\alpha_i)_{i\in\mathsf{H}}$ and a residual offset vector y'. The simulator chooses some arbitrary $x'_{\mathsf{H}} = (x'_i)_{i\in\mathsf{H}}$ that satisfies the equation $M_{\mathsf{H}} x'_{\mathsf{H}} = y'$ where M_{H} is the matrix M restricted to the columns in H.

  For every honest party $i \notin \mathsf{C}$:

  - Generate a fake proof $\pi_i = \mathsf{Sim}_2(\alpha_i, \mathsf{tdpf})$ that certifies that the tag α_i (received from the ideal functionality) satisfies the relation $\mathscr{R}_E$.
  - For every output $o \in L_i$, compute

$$y_{o,i} = x'_i + \sum_{j\in L_{o,-i}} r_{o,i,j} - \sum_{j\in L_{o,-i}} r_{o,j,i},$$

    just like in the original protocol.
  - Generate a fake proof $\pi_{o,i}$ that the committed values in $\alpha_i, (c_{o,i,j}, c_{o,j,i})_{j\in L_{o,-i}}$ satisfy the above linear equation, by using the second-phase simulator Sim_2 and the trapdoor tdpf.

  Send to $\mathscr{A}$ the tuple $(y_{o,i}, \pi_{o,i}, \alpha_i, \pi_i)$, $\forall i \notin \mathsf{C}, o \in R_i$ and get back the tuple pairs $(y_{o,i}, \pi_{o,i}, \alpha_i, \pi_i)$, $\forall i \in \mathsf{C}, o \in R_i$.

- **Generating outputs:** If for some $i \in \mathsf{C}$ the proof π_i fails to verify, set $x_i = 0, \rho_i = 0$. Else, use the knowledge extractor to extract (x_i, ρ_i). Next, compute the sets C' and B just like in the protocol. Formally, we iterate over every output $o \in [m]$, if there exists $j \in L_o$ for which the proof $\pi_{o,j}$ fails to verify, insert o to B, and j to C'. Finally, call the ideal functionality with B and $(x_i, \rho_i)_{i\in\mathsf{C}}$. Recall that the functionality computes $y = y' + M_{\mathsf{C}} \cdot x_{\mathsf{C}}$ and returns to all the parties the tuple $(B, y[\bar{B}], (\alpha_i)_{i\in\mathsf{H}}, (\alpha_i = E_{\rho_i}(x_i))_{i\in\mathsf{C}})$.

It is not hard to see that $\mathsf{C}' \subseteq \mathsf{C}$ and that B is of size at most $t'_{\mathsf{p}} \cdot d \leq b$ and so the simulator sends legal input to the ideal functionality. Fix the random input of the honest parties in the protocol

to some value $x = (x_i)_{i \notin \mathsf{C}}$ and $\rho = (\rho_i)_{i \notin \mathsf{C}}$, and let us condition on the event that the same input is chosen by the ideal functionality $\mathscr{F}_{\mathsf{cdsg},b}$. It suffices to show that the output of the simulated experiment $\mathscr{H}_S$ (that consists of the output of $\mathscr{A}$ concatenated with the public output that the ideal functionality delivers to the honest parties) is computationally indistinguishable from the real experiment $\mathscr{H}_\Pi$ (that consists of the output of $\mathscr{A}$ concatenated with the public output of the honest parties in the execution of $\Pi_{M,E}$). We define a sequence of hybrids.

$\mathscr{H}_1$**:** Identical to to the simulator, except that in **R1**, for every output $o \in [m]$ and every pair of honest parties $i, j \in L_o \setminus \mathsf{C}$ that influence o, we let $c_{o,i,j} = \mathsf{Com}(0; k_{o,i,j})$ be a random commitment of zero. (All other commitments remain unchanged.) Since these commitments are never opened, by the hiding property, we get that $\mathscr{H}_3$ is computationally indistinguishable from $\mathscr{H}_S$.

$\mathscr{H}_2$**:** Identical to $\mathscr{H}_1$, except that in **R3**, we replace x'_i with x_i for every honest party $i \in \mathscr{H}$. Note that that after this modification, x' is never used.

Claim D.1. *$\mathscr{H}_2$ is distributed identically to $\mathscr{H}_1$.*

Proof. By definition, for each output $o \in [m]$, it holds that the "residual honest sum" $y'_o = \sum_{i \in L_o \setminus \mathsf{C}} x'_i$ equals to the "residual honest sum" $y'_o = \sum_{i \in L_o \setminus \mathsf{C}} x_i$ computed over x. Thus the proof follows from the information-theoretic security of the standard secure-sum protocol. We sketch the details for completeness.

Fix all the randomness in the experiment up to the computation of $y_{o,i}$, except for the choice of $(r_{o,i,j})$ for all $o \in [m]$ and honest $i, j \in L_o \setminus \mathsf{C}$. It suffices to show that the vector $(y_{o,i})_{o \in [m], i \in L_o \setminus \mathsf{C}}$ is identically distributed in both experiments. (Since this is the only value that depends on x'.) Fix some $o \in [m]$ and honest $i \in L_o$, and let us subtract from $y_{o,i}$ the fixed values $z_{o,i} = \sum_{j \in L_{o,-i} \cap \mathsf{C}} r_{o,i,j} - \sum_{j \in L_{o,-i} \cap \mathsf{C}} r_{o,j,i}$, and show that the resulting values $y'_{o,i} = y_{o,i} - z_{o,i}$ are distributed identically in both experiments. Indeed, it is not hard to verify that, for every $o \in [m]$, the vector $(y'_{o,i})_{i \in L_o \setminus \mathsf{C}}$ is just a fresh additive secret sharing of the value y'_o. Furthermore, these sharings are statistically independent across the o's. The claim follows. □

$\mathscr{H}_3$**:** Identical to $\mathscr{H}_2$, except that in **R1**, we switch back the commitments $c_{o,i,j}$ to $\mathsf{Com}(r_{o,j}; k_{o,i,j})$ for every output $o \in [m]$ and every pair of honest parties $i, j \in L_o \setminus \mathsf{C}$ that influence o. (All other commitments remain unchanged.) By the hiding property of the commitment, we get that $\mathscr{H}_5$ is computationally indistinguishable from $\mathscr{H}_2$.

$\mathscr{H}_4$**:** Identical to $\mathscr{H}_3$, except that in **R3** for each honest party $i \in \mathsf{H}$ and $o \in R_i$, we generate the proof $\pi_{o,i}$ honestly (like in the protocol) based on the witnesses $\rho_i, (k_{o,i,j}, k_{o,j,i})_{j \in L_{o,-i}}$. By the zero-knowledge property of the NIZK system we get that $\mathscr{H}_4$ is computationally indistinguishable from $\mathscr{H}_3$.

$\mathscr{H}_5$**:** Identical to $\mathscr{H}_4$, except that in **R3** for each honest party $i \in \mathsf{H}$, the proof π_i is computed honestly by using the prover and the witness (x_i, ρ_i). By the zero-knowledge property of the NIZK system we get that $\mathscr{H}_5$ is computationally indistinguishable from $\mathscr{H}_4$.

$\mathscr{H}_6$**:** Identical to $\mathscr{H}_5$, except that we ignore the response of the ideal functionality in **R3** and define the vector $y' := M_{\mathsf{H}} x_{\mathsf{H}}$ and set $\alpha_i := E_{\rho_i}(x_i)$ for every $i \in \mathsf{H}$. Recall that we fixed the inputs $(x_i, \rho_i)_{i \notin \mathsf{C}}$ of the honest parties, and so, in both experiments the value α_i is taken to be $E_{\rho_i}(x_i)$ for every $i \in \mathsf{H}$ and $y' := M_{\mathsf{H}} \cdot x_{\mathsf{H}}$. The modification does not change the output of $\mathscr{A}$, and $\mathscr{H}_6$ is identically distributed to $\mathscr{H}_5$.

$\mathscr{H}_7$**:** Identical to $\mathscr{H}_6$, except that we remove the second call to the ideal functionality in the final stage, and define the output of the honest parties to be $B, (y_o)_{o \notin B}, (\alpha_i)_{i \in [n]]}$ computed as in the real-world experiment by protocol $\Pi_{M,E}$.

Claim D.2. *$\mathscr{H}_7$ is statistically close to $\mathscr{H}_6$.*

Proof. First, observe that in both experiments, for $i \in \mathsf{C}$ the value of α_i is taken to be $E_0(0)$ if the proof π_i is invalid. If the proof is valid, then α_i is taken to be the value sent by the adversary at the end of **R3** unless the knowledge extractor fails to extract a valid witness (x_i, ρ_i) from a valid proof π_i, which happens with negligible probability. Let us condition on the event that such failures do not happen, and let us fix the values x_i of the adversary (which is either extracted from a valid proofs or taken to be zero for invalid proofs).

Observe that the sets C', B are identically computed in both experiments, and so it remains to show that, for every $o \notin B$ the output $y_o = \sum_{j \in L_o} y_{o,j}$, as computed as in the real-world experiment, equals to the value $\sum_{i \in L_o} x_i$ (as computed by the ideal functionality) except with negligible probability. For this it suffices to show that for every $i \in L_o$, the good event G_i, defined by

$$y_{o,i} = x_i + \sum_{j \in L_{o,-i}} r_{o,i,j} - \sum_{j \in L_{o,-i}} r_{o,j,i}, \tag{4}$$

happens with all but negligible probability. For every honest party $i \notin \mathsf{C}$, the event G_i holds with probability 1, and so we focus on $i \in \mathsf{C}$. Recall that $o \notin B$ which means that the proof $\pi_{o,i}$ passes verification. Therefore, except with negligible probability, we can therefore use the knowledge-extractor KE to extract a vector of openings to the commitments $\alpha_i, (c_{o,i,j}, c_{o,j,i})_{j \in L_{o,-i}}$ that satisfy the linear equation (4). If G_i does not happen, then, for at least one of these commitments, we get two different valid openings, violating the binding property of the commitment scheme. The claim follows. □

Finally, observe that the only difference between the real experiment $\mathscr{H}_\Pi$ and $\mathscr{H}_7$, is that in $\mathscr{H}_\Pi$ we honestly sample the CRS crspf of the proof system. By the indistinguishability of the CRS, it follows that the two ensembles are computationally indistinguishable, completing the proof of Theorem 6.1. □

Remark D.3 (Relaxation: Non-Adaptive NIZK)**.** *Note that the adversary generates all her proofs in one shot (at the end of round **R3**) and after that she does not get to see new proofs of honest parties. We can therefore employ NIZK with* non-adaptive *identity-based simulation soundness as discussed in Remark B.2.*

E Proof of Theorem 6.4

We describe a simulator for Protocol 3. (A close variant of this simulator corresponds to the communication-optimized variant mentioned in Remark 6.5). Again, when the simulator is type-preserving (if the adversary passively corrupts a party, then the simulator also passively corrupts

the party) and so the simulation automatically applies to mixed adversaries. The simulator is similar to the simulator of the basic protocol $\Pi_{M,E}$ (Section D) except that the last step ("Generating outputs") is extended as follows.

1. If for some $i \in \mathsf{C}$ the proof π_i fails to verify, set $x_i = 0, \rho_i = 0$. Else, use the knowledge extractor to extract (x_i, ρ_i). Next, compute the sets C' and B just like in the protocol $\Pi_{M,E}$. Formally, we iterate over every output $o \in [m]$, if there exists $j \in L_o$ for which the proof $\pi_{o,j}$ fails to verify, insert o to B, and j to C'.

2. Initialize an empty set $\mathscr{Z}$, and iterate the sub-protocol Π_2 as long as the set B is larger than b while honestly emulating the role of honest parties. Specifically, in each iteration, the simulator reveals the randomizers that are adjacent to honest parties and publicly corrupted parties C'. (These randomizers were already defined in the steps **R1** and **R2** of the simulation and were already leaked to the adversary.) Receive from $\mathscr{A}$ randomizers and commitment keys on behalf of some of the parties in C, check these values and update the the sets $\mathscr{Z}$, $\mathsf{C}', \mathscr{Z}$ and B based on the public values just like in the original protocol.

3. Finally, after the iterations ends, for every $i \in \mathscr{Z}$ set $x_i = 0, \alpha_i = E_0(0)$, call the ideal functionality with B and $(x_i, \rho_i)_{i \in \mathsf{C}}$. Recall that the functionality computes $y = y' + M_{\mathsf{C}} \cdot x_{\mathsf{C}}$ and returns to all the parties the tuple $(B, y[\bar{B}], (\alpha_i)_{i \in \mathsf{H}}, (\alpha_i = E_{\rho_i}(x_i))_{i \in \mathsf{C}}$. Terminate with the output of $\mathscr{A}$.

It is not hard to see that $\mathsf{C}' \subseteq \mathsf{C}$. Also, by definition, the set B is of size at most b and so the simulator sends legal input to the ideal functionality. Fix the random input of the honest parties in the protocol to some value $x = (x_i)_{i \notin \mathsf{C}}$ and $\rho = (\rho_i)_{i \notin \mathsf{C}}$, and let us condition on the event that the same input is chosen by the ideal functionality $\mathscr{F}_{\mathsf{cdsg},b}$. It suffices to show that the output of the simulated experiment $\mathscr{H}_S$ (that consists of the output of $\mathscr{A}$ concatenated with the public output that the ideal functionality delivers to the honest parties) is computationally indistinguishable from the real experiment $\mathscr{H}_\Pi$ (that consists of the output of $\mathscr{A}$ concatenated with the public output of the honest parties in the execution of $\Pi_{M,E}$). The proof uses the same hybrids used in the proof of the basic protocol (Section D) except for the last hybrid $\mathscr{H}_7$ which is modified as follows.

$\mathscr{H}_7$**:** Identical to $\mathscr{H}_6$, except that we remove the call to the ideal functionality in the final stage, and define the output of the honest parties to be $B, (y_o)_{o \notin B}, (\alpha_i)_{i \in [n]]}$ computed as in the real-world experiment by protocol $\Pi_{M,E,b}$.

Claim E.1. *$\mathscr{H}_7$ is statistically close to $\mathscr{H}_6$.*

Proof. It suffices to show that the value returned by the functionality is statistically close to the output of the honest parties as computed in $\mathscr{H}_7$. Call party P_i, $i \in \mathsf{C}$ *public cheater* (in short PC), if she publicly cheats during the protocol, and *decent* otherwise. If P_i is PC then, in both experiments, α_i is taken to be $E_0(0)$. If P_i is decent then, in both experiments, α_i is taken to be the value sent by the adversary at the end of **R3** unless the knowledge extractor fails to extract a valid witness (x_i, ρ_i) from a valid proof π_i, which happens with negligible probability. Let us condition on the event that such failures do not happen, and let us fix the values x_i of the adversary (which are either extracted from valid proofs or taken to be zero for invalid proofs).

Observe that the sets C' and B are identically computed in both experiments and so it remains to show that, for every $o \notin B$ the output y_o, as computed as in the real-world experiment, equals to the value $\sum_{i \in L_o} x_i$ (as computed by the ideal functionality) except with negligible probability. For

an output $o \notin B$ that remains unchanged during Π_2, the argument is identical to the argument in the proof of the basic protocol (specifically, Claim D.2).

We move on to the case of $o \notin B$ whose value y_o was updated during the emulation of Π_2. Each of these values is computed in $\mathscr{H}_6$ as

$$y_o = \sum_{i \in L_o \setminus \mathsf{C}'} \left(y_{o,i} - \sum_{j \in L_{o,-i} \cap \mathsf{C}'} r_{o,i,j} + r_{o,j,i} \right),$$

where C' is the set of PC and $r_{o,i,j}$ together with $k_{o,i,j}$ are valid openings for $c_{o,i,j}$ that were published by P_i. Since $x_i = 0$ for $i \in \mathsf{C}'$, the output delivered by the ideal functionality can be written as $\sum_{i \in L_o \setminus \mathsf{C}'} x_i$. To prove that this value equals to y_o, it suffices to show that for every $i \in L_o \setminus \mathsf{C}'$, the good event G_i, defined by

$$y_{o,i} = x_i + \sum_{j \in L_{o,-i}} r_{o,i,j} - \sum_{j \in L_{o,-i}} r_{o,j,i}, \tag{5}$$

happens with all but negligible probability. For every honest party $i \notin \mathsf{C}$, the event G_i holds with probability 1, and so we focus on a decent party P_i, $i \in \mathsf{C} \setminus \mathsf{C}'$. Assume that G_i does not happen. Since the proof $\pi_{o,i}$ passes verification, we can use the knowledge-extractor KE to extract (except with negligible probability) a vector of openings to the commitments $\alpha_i, (c_{o,i,j}, c_{o,j,i})_{j \in L_{o,-i}}$ that satisfies the linear equation (5). This means that, for at least one of these commitments, we get two different valid openings, violating the binding property of the commitment scheme. The claim follows. □

Finally, observe that the only difference between the real experiment $\mathscr{H}_\Pi$ and $\mathscr{H}_7$, is that in $\mathscr{H}_\Pi$ we honestly sample the CRS crspf of the proof system. By the indistinguishability of the CRS, it follows that the two ensembles are computationally indistinguishable, completing the proof of Theorem 6.4. □

We note that Remark D.3 applies here as well. Indeed, no proofs are generated during the extended protocol (after the basic part ends).

F Communication-efficient variant of Protocol 3

We formally describe the communication efficient variant of Protocol 3 as explained in Remark 6.5. Below we say that P_i and P_j are neighbors if P_i and P_j influence a common output o, i.e., $R_i \cap R_j$ is non-empty. Note that this neighborhood relation implicitly defines an undirected graph over the parties $P_1, \ldots, P_n$ whose maximal-degree D is at most $d \cdot (r-1) = O(1)$. We further assume that t_{p} is taken to $\tau_{\mathsf{p}} n$ fro some constant $\tau_{\mathsf{p}} < 1$ and that $b = \beta n$ for some constant $\beta < 1$. We let $T := 1 + |\mathsf{C}| \cdot d/(b+1)$ and note that under our setting of parameters, T is upper-bounded by a constant that does not grow with n. The protocol is described in Figure 4.

Efficiency. During the entire online phase, P_i writes at most D values since she only reveals randomizers that are adjacent to her. The downstream complexity is at most $\sum_{\ell=1}^{T} D \cdot c_\ell$ where c_ℓ is the downstream complexity of a single call to **IsCorrupt**$(\cdot, \ell)$. Noting that c_0 is $O(d)$ and $c_\ell \leq D c_{\ell-1}$ we get that $c_\ell \leq d \cdot D^T$ since d, D and T are constants the total downstream complexity is constant as well.

Correctness. We claim that protocol $\Pi'_{M,E,b}$ t_{p}-realizes the functionality $\mathscr{F}_{\mathsf{cdsg},b}$ just like $\Pi_{M,E,b}$. Formally, consider a variant of $\Pi_{M,E,b}$, denoted by Π_1, in which the number of iteration is taken to be T but the final output $B, (y_o)_{o \notin B}$ and $(\alpha_i)_{i \notin \mathsf{C}'}$ is taken to be the output that is recorded at the end of the first iteration in which the set B is smaller than b. It is not hard to show that Theorem 6.4 applies also to Π_1 (via essentially the same proof that appears in Section E). Now consider a variant of Π', denoted by Π'_1, in which each call to the subroutine **IsCorrupt**(P_j, ℓ) is replaced by checking whether P_j is in C' which is defined as in the offline phase of Π'_1. Clearly, Π_1 is identical to Π'_1. (We only postponed some operations to the offline phase but these operations do not affect the online part anyway.) So it suffices to show that Π'_1 is identical to $\Pi'_{M,E,b}$. This follows by the correctness of **IsCorrupt** which can be edtablished by induction on the number of iterations.

Protocol 4 (Communication-efficient extended protocol $\Pi'_{M,E,b}$). ***Online:*** *Execute Protocol 2 without the offline (public output) phase. Next, each party P_i initializes a flag $f_{i,j}$ for each of its neighbors P_j that will indicate whether P_i knows that P_j was publicly cheating. We initialize all the flags to false.* ***For*** $\ell = 1, \ldots, T$ *every party P_i does the following.*

- ***Revealing randomizers:*** *For every output $o \in R_i$ and every neighbor $j \in L_{o,-i}$ if $f_{i,j}$ is false check if P_j is a* new publicly-corrupted *party by calling the subroutine* **IsCorrupt**$(j, \ell - 1)$ *defined below. If the result is positive we say that P_i is* triggered *by P_j in round ℓ with respect to output o and do the following:*
  - *P_i writes on the BB all the private randomizers $r_{o,i,j}, r_{o,j,i}$ and their commitment keys $k_{o,i,j}, k_{o,j,i}$ that were sent to/from the corrupted party j. In addition, P_i sets $f_{i,j}$ to true.*

The IsCorrupt subroutine: *Given an index $j \in [n]$ of a party and an iteration number ℓ the subroutine* **IsCorrupt**(j, ℓ) *returns* ***true*** *if, during iteration ℓ, party P_j publicly cheated for the first time. The subroutine is defined as follows.*

- **IsCorrupt**(j, ℓ)*:*
  1. *For $\ell = 0$: output true if P_j publicly cheated in the basic protocol $\Pi'_{M,E,b}$. That is, for every $o \in R_j$, read from the BB the proof $\pi_{o,j}$ and verify its validity.* ***If*** *any of these proofs fail, output* ***true****;* ***Else****, output* ***false****.*
  2. *For $\ell > 1$: for every output $o \in R_j$ and every neighbor $k \in L_{o,-j}$:*
     If **IsCorrupt**$(k, \ell - 1)$ *holds but P_j, who was triggered by P_k in iteration $\ell - 1$, did not reveal consistent randomizers in iteration ℓ,* ***then*** *output* ***true****;* ***Else****, output* ***false****.*

Offline decoding: *The public outputs are obtained by applying the protocol $\Pi_{M,E,b}$ on the values that were written on the BB so far. Formally, compute the values $\mathsf{C}', B, (y_o)_{o \notin B}$ as in the last step of the basic protocol based on the transcript of the basic protocol. Then repeat the following steps as long as the set B is larger than b:*

1. *In the ℓth iteration, read the transcript of the ℓth iteration of the online part.*
2. *For every output o that depends on some publicly corrupted input $j \in \mathsf{C}' \setminus \mathscr{X}$,* ***If*** *for every $i \in L_o \setminus \mathsf{C}'$ and $j \in L_o \cap \mathsf{C}'$ the randomizers $r_{o,i,j}, r_{o,j,i}$ were successfully revealed by P_i,* ***Then*** *call o* tentatively ready *and set a* tentative value

$$y'_o = \sum_{i \in L_o \setminus \mathsf{C}'} \left(y_{o,i} - \sum_{j \in L_{o,-i} \cap \mathsf{C}'} r_{o,i,j} + r_{o,j,i} \right).$$

3. *For every publicly corrupted party $j \in \mathsf{C}'$,* ***if*** *all the outputs that are influenced by j are tentatively ready* ***do****: (a) insert j to $\mathscr{X}$ and redefine $\alpha_j = E_0(0)$; and (b) update all the affected outputs $o \in R_j$ to be $y_o = y'_o$.*
4. *Insert to C' every party j who was triggered but did not successfully reveal all its randomizers. (Such a party is now publicly corrupted.) Insert to B all the outputs that are influenced by these new publicly-corrupted parties.*

Output *$B, (y_o)_{o \notin B}$ and $(\alpha_i)_{i \notin \mathsf{C}'}$. (Auxiliary output: C'.)*

Figure 4: The communication-efficient extended protocol $\Pi'_{M,E,b}$.