An abridged version of this paper appears in the proceedings of the 21^{st} International Conference on Cryptology in India — Indocrypt 2020. This is the full version.

Adaptively secure Threshold Symmetric-key Encryption

Pratyay Mukherjee

Visa Research
pratyay85@protonmail.com

October 26, 2020

Abstract

In a threshold symmetric-key encryption (TSE) scheme, encryption/decryption is performed by interacting with *any* threshold number of parties who hold parts of the secret-keys. Security holds as long as the number of corrupt (possibly colluding) parties stay below the threshold. Recently, Agrawal et al. [CCS 2018] (alternatively called DiSE) initiated the study of TSE. They proposed a *generic* TSE construction based on *any* distributed pseudorandom function (DPRF). Instantiating with DPRF constructions by Naor, Pinkas and Reingold [Eurocrypt 1999] (also called NPR) they obtained several efficient TSE schemes with various merits. However, their security models and corresponding analyses consider only *static* (and malicious) corruption, in that the adversary fixes the set of corrupt parties in the beginning of the execution before acquiring any information (except the public parameters) and is not allowed to change that later.

In this work we augment the DiSE TSE definitions to the *fully adaptive* (and malicious) setting, in that the adversary is allowed to corrupt parties *dynamically at any time* during the execution. The adversary may choose to corrupt a party depending on the information acquired thus far, as long as the total number of corrupt parties stays below the threshold. We also augment DiSE's DPRF definitions to support adaptive corruption. We show that their generic TSE construction, when plugged-in with an adaptive DPRF (satisfying our definition), meets our adaptive TSE definitions.

We provide an efficient instantiation of the adaptive DPRF, proven secure assuming decisional Diffie-Hellman assumption (DDH), in the random oracle model. Our construction borrows ideas from Naor, Pinkas and Reingold's [Eurocrypt 1999] statically secure DDH-based DPRF (used in DiSE) and Libert, Joye and Yung's [PODC 2014] adaptively secure threshold signature. Similar to DiSE, we also give an extension satisfying a *strengthened* adaptive DPRF definition, which in turn yields a *stronger* adaptive TSE scheme. For that, we construct a simple and efficient *adaptive* NIZK protocol for proving a specific commit-and-prove style statement in the random oracle model assuming DDH.

1 Introduction

Symmetric-key encryption is an extremely useful cryptographic technique to securely store sensitive data. However, to ensure the purported security it is of utmost important to store the *key* securely. Usually this is handled by using secure hardware like HSM, SGX, etc. This

approach suffers from several drawbacks including lack of flexibility (supports only a fixed operation), being expensive, prone to side-channel attacks (e.g. [KHF+20, LSG+20]) etc. Splitting the key among multiple parties, for example, by using Shamir's Secret Sharing [Sha79] provide a software-only solution. While this avoids a single-point of failure, many enterprise solutions, like Vault [vau], simply reconstructs the key each time an encryption/decryption is performed. This leaves the key exposed in the memory in the reconstructed form.

Threshold cryptography, on the other hand, ensures that the key is distributed at all time. A few recent enterprise solutions, such as [dya, por, sep], use threshold cryptographic techniques to provide software-only solutions without reconstruction for different applications. While studied extensively in the public-key setting [DF90, DDFY94, GJKR96, CG99, DK01, AMN01, SG02, Bol03, BBH06, GHKR08, BD10], threshold symmetric-key encryption has been an understudied topic in the literature. This changed recently with the work by Agrawal et al. [AMMR18a], that initiated the formal study of threshold symmetric-key encryption scheme (TSE). Their work, alternatively called DiSE, defines, designs and implements threshold (authenticated) symmetric-key encryption.

They put forward a generic TSE scheme based on any *distributed pseudorandom function* (DPRF). They provided two main instantiations of TSE based on two DPRF constructions from the work of Naor Pinkas and Reingold [NPR99] (henceforth called NPR):

(i) One (DP-PRF) based on any pseudorandom function, which is asymptotically efficient as long as $O(\binom{n}{t})$ is polynomial in the security parameter (when n is the total number of parties and t is the threshold).

(i) Another one (DP-DDH) based on DDH assumption (proven secure in the random oracle model) and is efficient for any n, t ($t \leq n$). By adding specialized NIZK to it, they constructed a stronger variant of DPRF– when plugged-in to the generic construction, a stronger TSE scheme is obtained.

A shortcoming of the DiSE TSE schemes is that they are secure only in a static (albeit malicious) corruption model, in that the set of corrupt parties is decided in the beginning and remains unchanged throughout the execution.[1] This is unrealistic in many scenarios where the adversary may corrupt different parties dynamically throughout the execution. For instance, an attacker may just corrupt one party and actively takes part in a few executions and based on the knowledge acquired through the corrupt party, decides to corrupt the next party and so on. Of course, the total number of corruption must stay below the threshold t, as otherwise all bets are off.

Our contribution. In this paper we augment DiSE to support adaptive corruption.[2] Our contributions can be summarized as follows:

1. We augment the security definitions of TSE, as presented in DiSE [AMMR18a], to support adaptive corruption. Like DiSE, our definitions are game-based. While some definitions (e.g. adaptive correctness, Def. 6.3) extend straightforwardly, others (e.g. adaptive

[1] We note that, DiSE's definition has a weak form of adaptivity, in that the corrupt set can be decided depending on the public parameters (but nothing else). Moreover, we do not have an adaptive attack against their scheme; instead their proof seems to rely crucially on the non-adaptivity.

[2] We remark that adaptive security is generically achieved from statically secure schemes using a standard complexity leveraging argument, in particular, just by guessing the corrupt set ahead of time. When $\binom{n}{t}$ is super-polynomial (in the security parameter), this technique naturally incurs a super-polynomial blow-up, for which super-polynomially hard assumptions are required. In contrast, all our constructions are based on polynomially hard assumptions.

authenticity, Def. 6.7) require more work. Intuitively, subtleties arise due to the dynamic nature of the corruption, which makes the task of tracking "the exact information" acquired by the attacker harder.

2. Similarly, we also augment the DPRF security definitions (following DiSE) to support adaptive corruption. We show that the generic DiSE construction satisfies our adaptive TSE definitions when instantiated with any such adaptively secure DPRF scheme. The proof of the generic transformation for adaptive corruption works in a fairly straightforward manner.

3. We provide new instantiations of efficient and simple DPRF constructions. Our main construction is based on the NPR's DDH-based DPRF (DP-DDH, as mentioned above) and the adaptive threshold signature construction by Libert, Joye and Yung [LJY14] (henceforth called LJY). It is proven secure assuming DDH in the random oracle model. The proof diverges significantly from DiSE's proof of DP-DDH. Rather, it closely follows the proof of the LJY's adaptive threshold signature. Similar to DiSE, we also provide a stronger DPRF construction by adding NIZK proofs – this yields a stronger version of TSE via the same transformation. However, in contrast to DiSE, we need the NIZK to support *adaptive* corruption. We provide a construction of *adaptive* NIZK for a special commit-and-prove style statement from DDH in the random oracle model – this may be of an independent interest. Finally, we observe that the adaptively secure construction obtained via a standard complexity leveraging argument on DP-PRF is asymptotically optimal. So, essentially the same construction can be used to ensure security against adaptive corruption without any additional machinery.

Efficiency compared to DiSE. Though we do not provide implementation benchmarks, it is not too hard to see the performance with respect to DiSE. Our DDH-based DPRF scheme is structurally very similar to DP-DDH– the only difference is in the evaluation, which requires two exponentiation operations instead of one used in DiSE. So, we expect the overall throughput of the TSE scheme may degrade by about a factor of 2. However, the communication remains the same because in our construction, like DP-DDH, only one group element is sent by each party.

For the stronger (NIZK-added) version, we expect again a factor of 2 slow down when instantiated with our adaptive NIZK construction (c.f. App 9) – we have 8 exponentiations as compared to DiSE's 4 in this case. In this case, however, there are also some degradation (around a factor of 2) in the communication due to presence of two additional elements of $\mathbb{Z}_p$ in the NIZK proof, compared to the (publicly verifiable) DiSE instantiation (Figure 5 of [AMMR18b]).[3]

Roadmap. We discuss related works in Sec 2 and technical overview in Sec. 3. After providing some basic notations and preliminaries in Sec. 4, we put forward our adaptive DPRF definitions in Sec. 5 and adaptive TSE definitions in Sec. 6. We give our DPRF constructions in Sec. 7. The full proof of our main construction is deferred to Supp B. In Sec 9 we provide our adaptive NIZK construction. Our TSE construction, that extends fairly straightforwardly from DiSE, is deferred to Supp 8. Supp A contains definitions etc. of additional cryptographic building blocks, mostly taken verbatim from DiSE.

[3] A privately verifiable version, similar to Fig. 6 of [AMMR18b] can be constructed analogously with similar efficiency. We do not elaborate on that.

2 Related Work

Threshold cryptography may be thought of as a special-case of general-purpose secure multi-party computation (MPC). Specific MPC protocols for symmetric encryption has been proposed in the literature [DK10, GRR+16, RSS17, dya], that essentially serve the same purpose. While they adhere to the standardized schemes, such as AES, their performance often falls short of the desired level. Theoretical solutions, such as universal thresholdizers [BGG+18] resolve the problem of threshold (authenticated) symmetric-key encryption generically, but are far from being practical. For a detailed discussion on threshold cryptography literature, and comparison with other approaches such as general MPC we refer to DiSE.

Adaptive DPRF has been constructed from lattice-based assumption by [LST18]. Their construction has an advantage of being in the standard model apart from being post-quantum secure. However, their construction is quite theoretical and is therefore not suitable for constructing practical TSE schemes. Our construction can be implemented within the same framework as DiSE with minor adjustments and will be similarly efficient (only with a factor of up to 2 degradation in the throughput, which is a reasonable price to pay for ensuring adaptive security).

Our approach to construct adaptive DPRF is similar to and inspired by the adaptive threshold signature scheme of LJY [LJY16]. Our proof strategy follows their footsteps closely. In fact, the LJY construction was mentioned as a plausible approach to construct adaptively secure DPRF in [LST18]. However, the threshold signature construction, as presented in LJY, becomes more cumbersome due to the requirement of public verification. In particular, it additionally requires bilinear maps and therefore is not suitable for using in the adaptive TSE setting. In this paper we provide a clean and simple exposition of the DPRF (that largely resembles the DiSE's), formalize the arguments and apply it to construct efficient adaptive TSE. Furthermore, we go one step further to combine it with adaptive NIZK to obtain a stronger version of adaptive DPRF and subsequently a stronger adaptive TSE.

3 Technical Overview

Technical Overview. Our main technical contribution is twofold: (i) definitions and (ii) constructions. DiSE definitions are game-based and hence they crucially rely on "tracking the exact amount of information" acquired by an attacker via playing security games with the challenger. In other words, the attacker is "allowed" to receive a certain amount of information and is only considered a "winner" if it is able to produce more than that. For standard (non-interactive) schemes, like CCA-secure encryption this is captured simply by requiring the adversary to submit challenge ciphertexts that are not queried to the decryption oracle (in other words the attacker is allowed to receive decryption of any ciphertexts that are *not* challenge ciphertexts). In the interactive setting this becomes challenging as already elaborated in DiSE. In the adaptive case this becomes even trickier as the corrupt set is not known until the very end.[4] Our general strategy is to "remember everything" (in several lists) the adversary has queried so far. Later in the final phase, when the entire corrupt set is known, the challenger "extracts" the exact amount of information learned by the adversary from the list and decides whether the winning condition is satisfied based on that.

[4]Note that, we assume a stronger erasure-free adaptive model, in that each party keeps its entire history from the beginning of execution in its internal state. Therefore, when the adversary corrupts a party, it gets access to the entire history. This compels the reduction to "explain" its earlier simulation of that party, before it was corrupt. In a weaker model, that assumes erasure, parties periodically removes their history.

Our DDH-based adaptive DPRF construction, inspired by the adaptive threshold signature of LJY, is based on a small but crucial tweak to the NPR's DP-DDH construction. Recall that, NPR's DP-DDH construction simply outputs $y := \mathsf{DP}_k(x) = \mathcal{H}(x)^k$ on input x, when k is the secret-key and $\mathcal{H} : \{0,1\}^\star \to G$ is a hash function (modeled as a random oracle) mapping to a cyclic group G; k is shared among n parties by a t-out-of-n Shamir's secret sharing scheme. Each party i outputs $y_i := \mathcal{H}(x)^{k_i}$ where k_i is the i-th share of k. The reconstruction is a public procedure, that takes many y_i's from a set S (of size at least t) and computes $\prod_{i \in S} y_i^{\lambda_{i,S}} = \mathcal{H}(x)^{\sum_{i \in S} \lambda_{i,S} k_i} = y$ where $\lambda_{i,S}$ is the Lagrange coefficient for party i for set S. The DPRF pseudorandomness requires that, despite learning multiple real outputs $\mathsf{DP}_k(x_1), \mathsf{DP}_k(x_2), \ldots$ (x_i's are called non-challenge inputs), the value $\mathcal{H}(x^\star)^k$ on a fresh $x^\star \neq x_i$ remains pseudorandom.

To prove security, the key k must not be known to the reduction (as it would be replaced by a DDH challenge) except in the exponent (that is g^k); the reduction may sample at most $(t-1)$ random k_i's which leaves k completely undetermined. Let us assume a relatively simpler case when the attacker corrupts exactly $(t-1)$ parties. In this case, a static adversary declares the corrupt set C ahead of time and the reduction may pick only those k_i such that $i \in C$, which are given to the adversary to simulate corruption. The evaluation queries on non-challenge inputs are easily simulated using DDH challenges and extrapolating adequately in the exponent (those $(t-1)$ keys and the DDH challenge fully determines the key in the exponent). However, in the adaptive case the reduction can not do that because it does not know C until the very end– in the mean-time it has to simulate evaluation queries on non-challenge inputs. If the reduction tries to simulate them by sampling *any* $(t-1)$ keys, it may encounter a problem. For example, the reduction may end up simulating a party i's response y_i on a non-challenge input on an extrapolated key k_i, which is only known to the reduction in the exponent as g^{k_i}. Later, an adaptive adversary may corrupt party i, when the reduction has to return k_i– which is not known in the "clear".

The tweak we employ here is very similar to the one used in the adaptive threshold signature scheme of LJY [LJY14]. The DPRF construction will now have two keys u and v and the output on x will be $\mathsf{DP}_{u,v}(x) := w_1^u w_2^v$ where $\mathcal{H}(x) = (w_1, w_2) \in G \times G$. The main intuition is that the value $w_1^u w_2^v$ is not revealing enough information on (u, v). In our proof we program the random oracle on non-challenge inputs such that they are answered by $\mathcal{H}(x) = (g^{s_j}, g^{w s_j})$ for the same w. This change is indistinguishable to the attacker as long as DDH is hard. Once we are in this hybrid, information theoretically it is possible to argue that the attacker only gets information $\{s_1(u + vw), s_2(u + vw), \ldots\}$– this basically leaves (u, v) undetermined except $u + vw = k$ for given k and w. Hence, it is possible to handle adaptive queries as the original keys u, v are always known to the reduction.

For the case when the attacker corrupts $\ell < t - 1$ parties, the DiSE DP-DDH proof becomes significantly more complex. This is due to the fact that, in that case the attacker may ask evaluation queries on the challenge $x^\star$ too, albeit up to $g = (t - 1 - \ell)$ many of them (otherwise the attacker would have enough information to compute $\mathsf{DP}_k(x^\star)$). Now, it becomes hard to simulate the challenge and non-challenge evaluation queries together from the DDH challenge if it is not known ahead of time which g evaluation queries would be made on $x^\star$. To handle that, the DiSE proof first makes the non-challenge evaluation queries independent of the challenge evaluation queries– they went through q hybrids, where q is the total number of distinct evaluation queries. Each successive hybrids are proven indistinguishable assuming DDH; in each of these reductions the knowledge of C plays a crucial role. Our construction, on the other hand, takes care of this setting already due

to the adaptive nature. Specifically, when we switch all the random oracle queries on any no-challenge input x to $\mathcal{H}(x) = (g^{s_j}, g^{ws_j})$ for the same w, the answers to the evaluation queries on $x^\star$ (that are still being programmed with $\mathcal{H}(x) = (g^{s_j} g^{t_j})$ for uniform t_j) becomes statistically independent of them. So, no additional effort is needed to handle the case $\ell < t - 1$.

Similar to DiSE, we provide a stronger version of DPRF with a stronger *adaptive* correctness property; plugging-in with the generic DiSE construction we obtain a stronger variant of TSE scheme that too achieves *adaptive* correctness. Our stronger DPRF is obtained by adding a commit-and-prove technique similar to DiSE. However, in contrast to DiSE, which relies on trapdoor commitments, we only require statistically hiding commitment scheme. This is due to the fact that DiSE DPRF definition supports a weak form adaptivity that allows the adversary to choose the corrupt set based on the public parameters. The public parameters contain the commitments. Therefore, the reduction needs to produce the commitments in an "equivocal manner" before knowing C – when C is known, the reduction picks up the shares for corrupt parties and uses the trapdoor to open them to the committed values. Our construction, on the other hand, tackles adaptivity in a different way and hence trapdoors are not required. However, we need *adaptive* NIZKs for this strengthened construction. We provide a simple and efficient adaptive NIZK construction for the specific commit-and-prove statement in Section 9 based on Schnorr's protocol and Fiat-Shamir, which may be of independent interest.

4 Preliminaries

In this paper, unless mentioned otherwise, we focus on specific interactive protocols that consist of only two-rounds of non-simultaneous interactions: an initiating party (often called an initiator) sends messages to a number of other parties and gets a response from each one of them. In particular, the parties contacted do not communicate with each other. Our security model considers *adaptive* and *malicious* corruption.

Common notation. We use notations similar to DiSE. Let $\mathbb{N}$ denote the set of positive integers. We use $[n]$ for $n \in \mathbb{N}$ to denote the set $\{1, 2, \ldots, n\}$. A function $f : \mathbb{N} \to \mathbb{N}$ is negligible, denoted by negl, if for every polynomial p, $f(n) < 1/p(n)$ for all large enough values of n. We use $\mathsf{D}(x) =: y$ or $y := \mathsf{D}(x)$ to denote that y is the output of the *deterministic* algorithm D on input x. Also, $\mathsf{R}(x) \to y$ or $y \leftarrow \mathsf{R}(x)$ denotes that y is the output of the *randomized* algorithm R on input x. R can be derandomized as $\mathsf{R}(x; r) =: y$, where r is the explicit random tape used by the algorithm. For two random variables X and Y we write $X \approx_{\mathsf{comp}} Y$ to denote that they are computationally indistinguishable and $X \approx_{\mathsf{stat}} Y$ to denote that they are statistically close. Concatenation of two strings a and b is either denoted by $(a \| b)$ or (a, b). Throughout the paper, we use n to denote the total number of parties, t to denote the threshold, and κ to denote the security parameter. We make the natural identification between players and elements of $\{1, \ldots, n\}$.

We will use *Lagrange interpolation* for evaluating a polynomial. For any polynomial P, the i-th Lagrange coefficient for a set S to compute $P(j)$ is denoted by $\lambda_{j,i,S}$. Matching the threshold, we will mostly consider $(t-1)$-degree polynomials, unless otherwise mentioned. In this case, at least t points on P are needed to compute any $P(j)$.

Inputs and outputs. We write $[j : x]$ to denote that the value x is private to party j. For a protocol π, we write $[j : z'] \leftarrow \pi([i : (x, y)], [j : z], c)$ to denote that party i has two private inputs x and y; party j has one private input z; all the other parties have no private input; c is a common public input; and, after the execution, only j receives an output z'. We write $[i : x_i]_{\forall i \in S}$ or more compactly $[\![\mathbf{x}]\!]_S$ to denote that each party $i \in S$ has a private value x_i.

Network model. We assume that all the parties are connected by point-to-point secure and authenticated channels. We also assume that there is a known upper-bound on the time it takes to deliver a message over these channels.

Cryptographic primitives. We need some standard cryptographic primitives to design our protocols like commitments, secret-sharing, adaptive non-interactive zero-knowledge proofs, etc. For completeness we provide the formal definitions, mostly taken verbatim from DiSE, in Supp A.

5 Distributed Pseudo-random Functions: Definitions

We now present a formal treatment of **adaptively secure** DPRF. First we present the DPRF consistency which is exactly the same as in DiSE [AMMR18a] and is taken verbatim from there as it does not consider any corruption.

Definition 5.1 (Distributed Pseudo-random Function) A distributed pseudo-random function (DPRF) DP is a tuple of three algorithms $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{Combine})$ satisfying a consistency property as defined below.

- $\mathsf{Setup}(1^\kappa, n, t) \rightarrow ((sk_1, \ldots, sk_n), pp)$. The setup algorithm generates n secret keys (sk_1, sk_2, …, sk_n) and public parameters pp. The i-th secret key sk_i is given to party i.
- $\mathsf{Eval}(sk_i, x, pp) \rightarrow z_i$. The Eval algorithm generates pseudo-random shares for a given value. Party i computes the i-th share z_i for a value x by running Eval with sk_i, x and pp.
- $\mathsf{Combine}(\{(i, z_i)\}_{i \in S}, pp) =: z/\perp$. The $\mathsf{Combine}$ algorithm combines the partial shares $\{z_i\}_{i \in S}$ from parties in the set S to generate a value z. If the algorithm fails, its output is denoted by $\perp$.

Consistency. For any $n, t \in \mathbb{N}$ such that $t \leq n$, all $((sk_1, \ldots, sk_n), pp)$ generated by $\mathsf{Setup}(1^\kappa, n, t)$, any input x, any two sets $S, S' \subset [n]$ of size at least t, there exists a negligible function negl such that

$$\Pr[\mathsf{Combine}(\{(i, z_i)\}_{i \in S}, pp) = \mathsf{Combine}(\{(j, z'_j)\}_{j \in S'}, pp) \neq \perp] \geq 1 - \mathsf{negl}(\kappa),$$

where $z_i \leftarrow \mathsf{Eval}(sk_i, x, pp)$ for $i \in S$, $z'_j \leftarrow \mathsf{Eval}(sk_j, x, pp)$ for $j \in S'$, and the probability is over the randomness used by Eval.

Next we define the adaptive security of DPRF. This differs from the definition provided in DiSE as for both correctness and pseudorandomness adaptive corruption is considered.

Definition 5.2 ((Strong)-adaptive security of DPRF) Let DP be a distributed pseudorandom function. We say that DP is **adaptively** *secure* against malicious adversaries if it satisfies the **adaptive** pseudorandomness requirement (Def. 5.3). Also, we say that DP is **strongly-adaptively-secure** against malicious adversaries if it satisfies both the **adaptive** pseudorandomness and **adaptive** correctness (Def. 5.6) requirements.

A DPRF is **adaptively** *pseudorandom* if no **adaptive** adversary can guess the PRF value on an input for which it hasn't obtained shares from at least t parties. It is **adaptively** *correct* if no **adaptive** adversary can generate shares which lead to an incorrect PRF value. We define these properties formally below.

Definition 5.3 (Adaptive pseudorandomness) A DPRF $\mathsf{DP} := (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Combine})$ is *adaptively pseudorandom* if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that

$$|\Pr\left[\mathsf{PseudoRand}_{\mathsf{DP},\mathcal{A}}(1^\kappa, 0) = 1\right] \quad - \quad \Pr\left[\mathsf{PseudoRand}_{\mathsf{DP},\mathcal{A}}(1^\kappa, 1) = 1\right]| \qquad \leq \qquad \mathsf{negl}(\kappa),$$

where $\mathsf{PseudoRand}$ is defined below.

$\underline{\mathsf{PseudoRand}_{\mathsf{DP},\mathcal{A}}(1^\kappa, b)}$:

- *Initialization.* Run $\mathsf{Setup}(1^\kappa, n, t)$ to get $((sk_1, \ldots, sk_n), pp)$. Give pp to $\mathcal{A}$. Initialize the state of party-i as $\mathsf{st}_i := \{sk_i\}$. The state of each honest party is updated accordingly– we leave it implicit below. Initialize a list $L := \emptyset$ to record the set of values for which $\mathcal{A}$ may know the PRF outputs. Initialize the set of corrupt parties $C := \emptyset$.
- *Adaptive corruption.* At any point receive a new set of corrupt parties $\tilde{C}$ from $\mathcal{A}$. Give the states $\{\mathsf{st}_i\}_{i \in \tilde{C}}$ of these parties to $\mathcal{A}$ and update $C := C \cup \tilde{C}$. Repeat this step as many times as $\mathcal{A}$ desires.
- *Pre-challenge Evaluation.* In response to $\mathcal{A}$'s evaluation query $(\texttt{Eval}, x, i)$ return $\mathsf{Eval}(sk_i, x, pp)$ to $\mathcal{A}$. Repeat this step as many times as $\mathcal{A}$ desires. Record all these queries.
- *Build lists.* For each evaluation query on an input x, build a list L_x containing all parties contacted at any time in the game.
- *Challenge.* $\mathcal{A}$ outputs $(\texttt{Challenge}, x^\star, S^\star, \{(i, z_i^\star)\}_{i \in U^\star})$ such that $|S^\star| \geq t$ and $U^\star \in S^\star \cap C$. Let $z_i \leftarrow \mathsf{Eval}(sk_i, x^\star, pp)$ for $i \in S^\star \setminus U^\star$ and $z^\star := \mathsf{Combine}(\{(i, z_i)\}_{i \in S \setminus U} \cup \{(i, z_i^\star)\}_{i \in U}, pp)$. If $z^\star = \perp$, return $\perp$. Else, if $b = 0$, return $z^\star$; otherwise, return a uniformly random value.
- *Post-challenge evaluation and corruption.* Exactly same as the pre-challenge corruption and evaluation queries.
- *Guess.* When $\mathcal{A}$ returns a guess b' then do as follows:
  - if the total number of corrupt parties, $|C| \geq t$ then output 0 and stop;
  - if the challenge $x^\star$ has been queried for evaluation for at least $g := t - |C|$ honest parties, that is if $L_{x^\star} \cap ([n] \setminus C) \geq g$ then output 0 and stop;
  - otherwise output b'.

Remark 5.4 (Difference with static security [AMMR18a]) *The main differences with the static version, given in DiSE, are in the "Corruption" and the "Guess" phase. Corruption takes place at any time in the security game and the set of all corrupt parties is updated correspondingly. Now, we need to prevent the adversary to win trivially. For that, we maintain lists corresponding to each evaluation input (in DiSE definition only one list suffices) and in the end check that whether the adversary has sufficient information to compute the DPRF output itself. This becomes slightly trickier than the static case due to constant updating of the list of corrupt parties.*

Remark 5.5 (Comparing with definition of [LST18]) *Our pseudorandomness definition is stronger than the definition of Libert et al. [LST18], in that a malicious adversary is not allowed to supply malformed partial evaluations during the challenge phase. We handle this by attaching NIZK proofs.*

Next we define **adaptive** correctness, which is very similar to the static case with necessary adjustment in the "Corruption" phase.

Definition 5.6 (Adaptive correctness) A DPRF $\mathsf{DP} := (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Combine})$ is *adaptively correct* if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that the following game outputs 1 with probability at least $1 - \mathsf{negl}(\kappa)$.

- *Initialization.* Run $\mathsf{Setup}(1^\kappa, n, t)$ to get $((sk_1, \ldots, sk_n), pp)$. Give pp to $\mathcal{A}$. Initialize the state of party-i as $\mathsf{st}_i := \{sk_i\}$. The state of each honest party is updated accordingly– we leave it implicit below. Initialize the set of corrupt parties $C := \emptyset$.
- *Adaptive Corruption.* At any time, receive a new set of corrupt parties $\tilde{C}$ from $\mathcal{A}$, where $|C \cup \tilde{C}| < t$. Give the secret states $\{\mathsf{st}_i\}_{i \in \tilde{C}}$ of these parties to $\mathcal{A}$ and update $C := C \cup \tilde{C}$. Repeat this step as many times as $\mathcal{A}$ desires.
- *Evaluation.* In response to $\mathcal{A}$'s evaluation query $(\mathtt{Eval}, x, i)$ for some $i \in [n] \setminus C$, return $\mathsf{Eval}(sk_i, x, pp)$ to $\mathcal{A}$. Repeat this step as many times as $\mathcal{A}$ desires.
- *Guess.* Receive a set S of size at least t, an input $x^\star$, and shares $\{(i, z_i^\star)\}_{i \in S \cap C}$ from $\mathcal{A}$. Let $z_j \leftarrow \mathsf{Eval}(sk_j, x^\star, pp)$ for $j \in S$ and $z'_i \leftarrow \mathsf{Eval}(sk_i, x^\star, pp)$ for $i \in S \setminus C$. Also, let $z := \mathsf{Combine}(\{(j, z_j)\}_{j \in S}, pp)$ and $z^\star := \mathsf{Combine}(\{(i, z'_i)\}_{i \in S \setminus C} \cup \{(i, z_i^\star)\}_{i \in S \cap C}, pp)$. Output 1 if $z^\star \in \{z, \perp\}$; else, output 0.

6 Threshold Symmetric-key Encryption: Definitions

In this section, we provide the formal definitions of threshold symmetric-key encryption (TSE). We start by specifying the algorithms that constitute a TSE scheme, which is taken verbatim from DiSE [AMMR18a].

Definition 6.1 (Threshold Symmetric-key Encryption) A threshold symmetric-key encryption scheme TSE is given by a tuple $(\mathsf{Setup}, \mathsf{DistEnc}, \mathsf{DistDec})$ that satisfies the consistency property below.

- $\mathsf{Setup}(1^\kappa, n, t) \to (\llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$: Setup is a randomized algorithm that takes the security parameter as input, and outputs n secret keys $sk_1, \ldots, sk_n$ and public parameters pp. The i-th secret key sk_i is given to party i.

– $\mathsf{DistEnc}(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j : m, S], pp) \rightarrow [j : c/\perp]$: $\mathsf{DistEnc}$ is a distributed protocol through which a party j encrypts a message m with the help of parties in a set S. At the end of the protocol, j outputs a ciphertext c (or $\perp$ to denote failure). All the other parties have no output.

– $\mathsf{DistDec}(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j : c, S], pp) \rightarrow [j : m/\perp]$: $\mathsf{DistDec}$ is a distributed protocol through which a party j decrypts a ciphertext c with the help of parties in a set S. At the end of the protocol, j outputs a message m (or $\perp$ to denote failure). All the other parties have no output.

CONSISTENCY. For any $n, t \in \mathbb{N}$ such that $t \leq n$, all $(\llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$ output by $\mathsf{Setup}(1^\kappa)$, for any message m, any two sets $S, S' \subset [n]$ such that $|S|, |S'| \geq t$, and any two parties $j \in S, j' \in S'$, if all the parties behave honestly, then there exists a negligible function negl such that

$$\Pr\Big[[j' : m] \leftarrow \mathsf{DistDec}(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j' : c, S'], pp) \;\; | \;\; [j : c] \leftarrow \mathsf{DistEnc}(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j : m, S], pp)\Big] \geq 1 - \mathsf{negl}(\kappa),$$

where the probability is over the random coin tosses of the parties involved in $\mathsf{DistEnc}$ and $\mathsf{DistDec}$.

Next we define the security of a TSE scheme in presence of an **adaptive** and malicious adversary.

Definition 6.2 ((Strong)-Adaptive Security of TSE) Let TSE be a threshold symmetric-key encryption scheme. We say that TSE is **(strongly)-adaptively** secure against malicious adversaries if it satisfies the **(strong)-adaptive** correctness (Def. 6.3), **adaptive** message privacy (Def. 6.5) and **(strong)-adaptive** authenticity (Def. 6.7) requirements.

6.1 Adaptive Correctness

The **adaptive** correctness definition barely changes from the static version in DiSE, except the required adjustment in the corruption phase.

Definition 6.3 (Adaptive Correctness) A TSE scheme $\mathsf{TSE} := (\mathsf{Setup}, \mathsf{DistEnc}, \mathsf{DistDec})$ is *adaptively correct* if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that the following game outputs 1 with probability at least $1 - \mathsf{negl}(\kappa)$.

– *Initialization.* Run $\mathsf{Setup}(1^\kappa)$ to get $(\llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$. Give pp to $\mathcal{A}$. Initialize the state of party-i as $\mathsf{st}_i := \{sk_i\}$. The state of each honest party is updated accordingly– we leave it implicit below. Initialize the set of corrupt parties to $C := \emptyset$.

– *Adaptive Corruption.* At any time receive a new set of corrupt parties $\tilde{C}$ from $\mathcal{A}$, where $|C \cup \tilde{C}| < t$. Give the secret-states $\{\mathsf{st}_i\}_{i \in \tilde{C}}$ to $\mathcal{A}$. Repeat this as many times as $\mathcal{A}$ desires.

– *Encryption.* Receive $(\texttt{Encrypt}, j, m, S)$ from $\mathcal{A}$ where $j \in S \setminus C$ and $|S| \geq t$. Initiate the protocol $\mathsf{DistEnc}$ from party j with inputs m and S. If j outputs $\perp$ at the end, then output 1 and stop. Else, let c be the output ciphertext.

– *Decryption.* Receive $(\texttt{Decrypt}, j', S')$ from $\mathcal{A}$ where $j' \in S' \setminus C$ and $|S'| \geq t$. Initiate the protocol $\mathsf{DistDec}$ from party j' with inputs c, S' and pp.

– *Output.* Output 1 if and only if j' outputs m or $\perp$.

A *strongly-adaptively-correct* TSE scheme is a correct TSE scheme but with a different output step. Specifically, output 1 if and only if:

– If all parties in S' behave honestly, then j' outputs m; or,

– If corrupt parties in S' deviate from the protocol, then j' outputs m or $\perp$.

Remark 6.4 *Note that, an adaptive adversary may corrupt a party j right after the encryption phase such that the condition $j \in S \setminus C$ does not hold anymore. However, this does not affect the winning condition, because we just need the party j, who makes the encryption query, to output a legitimate ciphertext immediately – for which we need that party to be honest only within the encryption phase.*

6.2 Adaptive Message privacy

Similar to DiSE our definition is a CPA-security style definition additionally accompanied by an *indirect decryption* access to the attacker. However, due to adaptive corruption, handling indirect decryption queries become more subtle. We provide the formal definition below.

Definition 6.5 (Adaptive message privacy) A TSE scheme $\mathsf{TSE} := (\mathsf{Setup}, \mathsf{DistEnc}, \mathsf{DistDec})$ satisfies *message privacy* if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that

$$\left|\Pr\left[\mathsf{MsgPriv}_{\mathsf{TSE},\mathcal{A}}(1^\kappa, 0) = 1\right] - \Pr\left[\mathsf{MsgPriv}_{\mathsf{TSE},\mathcal{A}}(1^\kappa, 1) = 1\right]\right| \leq \mathsf{negl}(\kappa),$$

where $\mathsf{MsgPriv}$ is defined below.

$\mathsf{MsgPriv}_{\mathsf{TSE},\mathcal{A}}(1^\kappa, b)$:

– *Initialization.* Run $\mathsf{Setup}(1^\kappa, n, t)$ to get $(\llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$. Give pp to $\mathcal{A}$. Initialize the state of party-i as $\mathsf{st}_i := \{sk_i\}$. The state of each honest party is updated accordingly– we leave it implicit below. Initialize a list $L_{\mathsf{dec}} := \emptyset$.

– *Adaptive Corruption.* Initialize $C := \emptyset$ At any time receive a new set of corrupt parties $\tilde{C}$ from $\mathcal{A}$, where $|C \cup \tilde{C}| < t$. Give the secret-states $\{\mathsf{st}_i\}_{i \in \tilde{C}}$ to $\mathcal{A}$. Repeat this as many times as $\mathcal{A}$ desires.

– *Pre-challenge encryption queries.* In response to $\mathcal{A}$'s encryption query $(\texttt{Encrypt}, j, m, S)$, where $j \in S$ and $|S| \geq t$, run an instance of the protocol $\mathsf{DistEnc}$ with $\mathcal{A}$[5]. If $j \notin C$, then party j initiates the protocol with inputs m and S, and the output of j is given to $\mathcal{A}$. Repeat this step as many times as $\mathcal{A}$ desires.

– *Pre-challenge indirect decryption queries.* In response to $\mathcal{A}$'s decryption query $(\texttt{Decrypt}, j, c, S)$, where $j \in S \setminus C$ and $|S| \geq t$, party j initiates $\mathsf{DistDec}$ with inputs c and S. Record j in a list L_{dec}. Repeat this step as many times as $\mathcal{A}$ desires.

[5]Note that j can be either honest or corrupt here. So both types of encryption queries are captured.

– *Challenge.* $\mathcal{A}$ outputs $(\texttt{Challenge}, j^\star, m_0, m_1, S^\star)$ where $|m_0| = |m_1|$, $j^\star \in S^\star \setminus C$ and $|S^\star| \geq t$. Initiate the protocol $\mathsf{DistEnc}$ from party $j^\star$ with inputs m_b and $S^\star$. Give $c^\star$ (or $\perp$) output by $j^\star$ as the challenge to $\mathcal{A}$.

– *Post-challenge encryption queries.* Repeat pre-challenge encryption phase.

– *Post-challenge indirect decryption queries.* Repeat pre-challenge decryption phase.

– *Guess.* Finally, $\mathcal{A}$ returns a guess b'. Output b' if and only if (i) $j^\star \notin C$ and (ii) $L_{\mathsf{dec}} \cap C = \emptyset$; otherwise return a random bit.

Remark 6.6 *The main difference from the non-adaptive setting comes in the* Guess *phase, in that, the winning condition requires that neither (i) the initiator of the challenge query, (ii) nor any of the initiator of an indirect decryption query is corrupt. To handle the later we introduce a list L_{dec} which records identities of all parties who made an indirect decryption query.*

6.3 Adaptive Authenticity

Similar to DiSE, our authenticity definition follows a *one-more type notion.* To adapt the authenticity definition into the adaptive setting, we need to make sure to exactly track the information gained by adversary which is sufficient to produce valid ciphertexts. This leads to some subtleties in the adaptive case. We incorporate that below by "delaying" the counting of the number of honest responses per query.

Definition 6.7 (Adaptive authenticity) A TSE scheme $\mathsf{TSE} := (\mathsf{Setup}, \mathsf{DistEnc}, \mathsf{DistDec})$ satisfies *authenticity* if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that

$$\Pr\left[\mathsf{AUTH}_{\mathsf{TSE},\mathcal{A}}(1^\kappa) = 1\right] \leq \mathsf{negl}(\kappa),$$

where AUTH is defined below.

$\underline{\mathsf{AUTH}_{\mathsf{TSE},\mathcal{A}}(1^\kappa)}$:

– *Initialization.* Run $\mathsf{Setup}(1^\kappa, n, t)$ to get $(\llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$. Give pp to $\mathcal{A}$. Initialize the state of party-i as $\mathsf{st}_i := \{sk_i\}$. The state of each honest party is updated accordingly– we leave it implicit below. Initialize counter $\mathsf{ct} := 0$ and *ordered* lists $L_{\mathsf{act}}, L_{\mathsf{ctxt}} := \emptyset$. Below, we assume that for every query, the (j, S) output by $\mathcal{A}$ are such that $j \in S$ and $|S| \geq t$. Initialize $C := \emptyset$.

– *Adaptive Corruption.* At any time receive a new set of corrupt parties $\tilde{C}$ from $\mathcal{A}$, where $|C \cup \tilde{C}| < t$. Give the secret-states $\{\mathsf{st}_i\}_{i \in \tilde{C}}$ to $\mathcal{A}$. Repeat this as many times as $\mathcal{A}$ desires.

– *Encryption queries.* On receiving $(\texttt{Encrypt}, j, m, S)$ from $\mathcal{A}$, run the protocol $\mathsf{DistEnc}$ with m, S as the inputs of j. Append (j, S) into list L_{act}. If $j \in C$, then also append the ciphertext into the list L_{ctxt}.

– *Decryption queries.* On receiving $(\texttt{Decrypt}, j, c, S)$ from $\mathcal{A}$ run the protocol $\mathsf{DistDec}$ with c, S as the inputs of j. Append (j, S) into the list L_{act}.

– *Targeted decryption queries.* On receiving $(\texttt{TargetDecrypt}, j, \ell, S)$ from $\mathcal{A}$ for some $j \in S \setminus C$, run $\mathsf{DistDec}$ with c, S as the inputs of j, where c is the ℓ-th ciphertext in L_{ctxt}. Append (j, S) into L_{act}.

– *Forgery.* For each $j \in C$, for each entry (there can be multiple entries, which are counted as many times) $(j, S) \in L_{\mathsf{act}}$ increment ct by $|S \setminus C|$. Define $g := t - |C|$ and $k := \lfloor \mathsf{ct}/g \rfloor$. $\mathcal{A}$ outputs $((j_1, S_1, c_1), (j_2, S_2, c_2), \ldots, (j_{k+1}, S_{k+1}, c_{k+1}))$ such that $j_1, \ldots, j_{k+1} \notin C$ and $c_u \neq c_v$ for any $u \neq v \in [k+1]$ (ciphertexts are not repeated). For every $i \in [k+1]$, run an instance of $\mathsf{DistDec}$ with c_i, S_i as the input of party j_i. In that instance, all parties in S_i behave honestly. Output 0 if any j_i outputs $\perp$; else output 1.

A TSE scheme satisfies *strong-authenticity* if it satisfies authenticity but with a slightly modified AUTH: In the forgery phase, the restriction on corrupt parties in S_i to behave honestly is removed (for all $i \in [k+1]$).

Remark 6.8 *The above definition has some important differences with the static case [AMMR18a], because tracking the exact amount of information acquired by the adversary throughout the game for producing legitimate ciphertexts becomes tricky in presence of adaptive corruption. To enable this we keep track of pairs (j, S) for* any *encryption or decryption query made at any time irrespective of whether j is honest or corrupt at that time. This is because, an adaptive adversary may corrupt j at a later time. In the forgery phase, when the whole corrupt set C is known, then for each such pair the corresponding number of maximum possible honest responses $|S \setminus C|$ for that query is computed. For static corruption this issue does not come up as the corrupted set C is known in the beginning.*

7 Our DPRF constructions

In this section we provide several DPRF constructions against adaptive attackers. In Section 7.1 we provide our main DPRF construction. In Sec. 7.2, we provide an extension which is *strongly adaptively* secure. Finally in Sec. 7.3 we briefly argue the adaptive security naturally achieved by the DP-PRF construction.

7.1 Adaptively-secure DPRF

Our DDH-based DPRF is provided in Fig. 1. We prove the following theorem formally.

Parameters: Let $G = \langle g \rangle$ be a multiplicative cyclic group of prime order p in which the DDH assumption holds and $\mathcal{H}\{0,1\}^* \to G^2$ be a hash function modeled as a random oracle. Let SSS be Shamir's secret sharing scheme(Def. A.2).

– $\mathsf{Setup}(1^\kappa, n, t) \to (\llbracket \mathbf{sk} \rrbracket_{[n]}, pp)$: Sample $(u, v) \leftarrow_\$ \mathbb{Z}_p^2$ and get $(u_1, \ldots, u_n) \leftarrow \mathsf{SSS}(n, t, p, u)$ and $(v_1, \ldots, v_n) \leftarrow \mathsf{SSS}(n, t, p, v)$. Set $pp := (p, g, G)$ and $sk_i := (u_i, v_i)$ and give (sk_i, pp) to party i, for $i \in [n]$.

– $\mathsf{Eval}(sk_i, x, pp) \to z_i$: Parse $(u_i, v_i) := sk_i$; compute $(w_1, w_2) := \mathcal{H}(x)$ and $h_i := w_1^{u_i} w_2^{v_i}$; then output h_i.

– $\mathsf{Combine}(\{(i, h_i)\}_{i \in S}, pp) =: z/\perp$: If $|S| < t$, output $\perp$. Else output $\prod_{i \in S} h_i^{\lambda_{0,i,S}}$.

Figure 1: An adaptively secure DPRF protocol Π_{adap} based on DDH.

Theorem 7.1 *Protocol* Π_{adap} *in Figure 1 is an adaptively secure DPRF under the DDH assumption in the random oracle model.*

PROOF SKETCH. We need to show that the construction given in Fig. 1 satisfies the consistency and adaptive pseudorandomness. The consistency is straightforward from the construction. So below we only focus on adaptive pseudorandomness. In particular, we show that if there exists a PPT adversary $\mathcal{A}$ which breaks the adaptive pseudorandomness game, then we can build a polynomial time reduction that breaks the DDH assumption. The formal proof is provided in Suppl B. We give a sketch below.

Somewhat surprisingly our proof is significantly simpler than DiSE. This is because, since our construction is purposefully designed to protect against adaptive corruption, we can easily switch to a hybrid where the information obtained by the adverasry through all non-challenge evaluation queries are simulated using a key, that is *statistically* independent from the actual key. In contrast, DiSE's proof needs to carefully make the non-challenge evaluation query independent to reach a similar hybrid by crucially relying on the knowledge of corrupt set from the beginning.

For any PPT adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$ we briefly describe the hybrids below:

$\mathsf{PseudoRand}_{\mathcal{A}}(b)$**.** This is the real game in that the challenger chooses random $(s_i, t_i) \leftarrow_{\$} \mathbb{Z}_p^2$ for simulating the i-th random oracle query on any x_i as $\mathcal{H}(x_i) := (g^{s_i}, g^{t_i})$.

$\mathsf{Hyb}^1_{\mathcal{A}}(b)$ In the next hybrid experiment $\mathsf{Hyb}^1_{\mathcal{A}}(b)$ the only change we make is: the challenger guesses the challenge input $x^\star$ randomly (incurring a $1/q_{\mathcal{H}}$ loss for $q_{\mathcal{H}} = \mathsf{poly}(\kappa)$ distinct random oracle queries) and simulates the random oracle query on all $x_i \neq x^\star$ as $\mathcal{H}(x_i) := (g^{s_i}, g^{\omega s_i})$ where $\omega, s_1, s_2, \ldots$ are each sampled uniformly random from $\mathbb{Z}_p$. This implicitly sets $t_i := \omega s_i$. Note that, each query has the same ω, but a different s_i – this way the challenger ensures that the attacker does not learn any new information by making more queries. However, for $x^\star$ the random oracle is programmed as usual by sampling random $s^\star, t^\star$ values as $\mathcal{H}(x^\star) := (g^{s^\star}, g^{t^\star})$.

Claim 7.2 *Assuming DDH is hard in group G, we have that* $\mathsf{PseudoRand}_{\mathcal{A}}(b) \approx_{\mathsf{comp}} \mathsf{Hyb}^1_{\mathcal{A}}(b)$.

Given a DDH challenge $g^\alpha, g^\beta, g^\gamma$ where γ is either is equal to $\alpha\beta$ or uniform random in $\mathbb{Z}_p$ the reduction set $g^{s_i} := g^{\mu_i} \cdot g^{\alpha b_i}$ for uniform random $\mu_i, \sigma_i \in \mathbb{Z}_p^2$ and $g^{t_i} := g^{\mu_i \beta} \cdot g^{\gamma \sigma_i}$. Now, if $\gamma = \alpha\beta$, then implicitly (in the exponent) the challenger sets $s_i := \mu_i + \sigma_i \alpha$ and $t_i := \beta(\mu_i + \sigma_i \alpha)$, which implies that $\omega := \beta$ and $t_i := s_i \omega$. So Hyb_1 is perfectly simulated. On the other hand, when γ is uniform random, then $t_i := \mu_i \beta + \sigma_i \gamma$ which is uniform random in $\mathbb{Z}_p$ – this perfectly simulates Hyb_0.

$\mathsf{Hyb}^2_{\mathcal{A}}(b)$**.** In this hybrid we do not make any change from $\mathsf{Hyb}^1_{\mathcal{A}}(b)$ except that all non-challenge evaluation queries are responded with a key $k \leftarrow_{\$} \mathbb{Z}_p$ sampled uniformly at random, whereas the corruption query for party j are answered using randomly sampled u_j, v_j subject to $u_j + \omega v_j = k_j$ where k_j is the j-th Shamir's share of k. In particular, for a non-challenge evaluation query $\mathsf{Eval}(x_i, j)$, the challenger returns $g^{s_i k_j}$ where $\mathcal{H}(x_i) = (g^{s_i}, g^{\omega s_i})$. The challenge query and the evaluation queries on $x^\star$ are answered similar to $\mathsf{Hyb}^1_{\mathcal{A}}(b)$.

Claim 7.3 *We have that:* $\mathsf{Hyb}^1_{\mathcal{A}}(b) \approx_{\mathsf{stat}} \mathsf{Hyb}^2_{\mathcal{A}}(b)$

Note that, this statement is information theoretic and hence we assume that the adversary here can be unbounded. First we notice that, in both the hybrids an unbounded adversary learns values $\{s_1, s_2, \ldots\}$, ω from the random oracle responses. Furthermore, it learns at most $t-1$ pairs $\{u_j, v_j\}_{i \in C}$ where $|C| < t$, given which (u, v) remains statistically hidden. Now, the only difference comes in the non-challenge evaluation queries. In $\mathsf{Hyb}^1_{\mathcal{A}}(b)$, the adversary learns $\{u + \omega v\}$ from them whereas in $\mathsf{Hyb}^2_{\mathcal{A}}(b)$ it learns a random k. Now, we note that conditioned on the common values, $u + v\omega$ is uniformly random for randomly chosen u, v, because it is basically a universal hash function where ω is the input and (u, v) are uniform random keys. Hence the two distributions are statistically close.

Now, once we are in $\mathsf{Hyb}^2_{\mathcal{A}}(b)$, it is easy to observe that the response to the challenge query is uniformly random irrespective of b, which in turn implies that $\mathsf{Hyb}^2_{\mathcal{A}}(0) \approx_{\mathsf{stat}} \mathsf{Hyb}^2_{\mathcal{A}}(1)$, which concludes the proof of the theorem.

We provide the full proof in detail in Suppl B.

■

7.2 Strongly-adaptively secure DPRF

Adding adequate NIZK proofs and commitments we obtain an *adaptively-secure* DPRF which satisfies *adaptive* correctness too. However, we need to rely on *statistically hiding* and a specific *adaptive NIZK*[6] argument [GOS12] for a commit-and-prove style statement. We also provide a simple and efficient construction of adaptive NIZK argument system for the particular commit-and-prove statement in Appendix 9 based on Schnorr's protocol and Fiat-Shamir's transformation, which may be of independent interest. The protocol is proven secure assuming DDH in the random oracle model. The construction is provided in Fig. 2.

Formally we prove the following theorem. We skip the full proof as it is quite similar to the proof of Theorem 7.1 with the adequate changes. Instead we provide a sketch below remarking on the changes needed.

Theorem 7.4 *Protocol* $\Pi_{\mathsf{str\text{-}adap}}$ *in Figure 2 is a* strongly *adaptively secure DPRF under the DDH assumption in the random oracle model.*

Proof sketch. For strong adaptive security we need to show (i) adaptive pseudorandomness and (ii) adaptive correctness. First we discuss adaptive correctness.

The proof of adaptive correctness is very similar to the one provided in DiSE for the non-adaptive case except adequate changes in the the statement of the NIZK proof. Recall that, the main idea of the construction (Fig. 2) is to use commitments using which the Setup procedure publishes commitments of everyone's secret key. Later, when queried for an evaluation, each party, in addition to the evaluation, sends a NIZK proof stating that the evaluation is correctly computed using the actual keys (those are committed before). So, to violate adaptive correctness the attacker must break either the simulation-soundness of NIZK or the binding of the commitment scheme– rendering its task infeasible. Note that, the adversary does not gain anything by being adaptive in this case.

The adaptive pseudorandomness proof follows the footstep of the proof of Theorem 7.1. However, due to presence of the commitments and NIZKs some adjustments are required. In particular, we need to ensure that neither the commitments, nor the proofs give away more information, for which statistical hiding and adaptive zero-knowledge properties of them will

[6] Groth et al. [GOS12] alternatively calls them *zero-knowledge in erasure-free model.*

be used respectively. We describe the hybrids below and highlight the changes in red from the proof of Theorem 7.1.

For any PPT adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$ we briefly describe the hybrids below:

$\mathsf{PseudoRand}_{\mathcal{A}}(b)$**.** This is the real game in that the challenger chooses random $(s_i, t_i) \leftarrow_{\$} \mathbb{Z}_p^2$ for simulating the i-th random oracle query on any x_i as $\mathcal{H}(x_i) := (g^{s_i}, g^{t_i})$.

$\mathsf{Hyb}^1_{\mathcal{A}}(b)$ In the next hybrid experiment $\mathsf{Hyb}^1_{\mathcal{A}}(b)$ the only change we make is: the challenger guesses the challenge input $x^\star$ randomly (incurring a $1/q_{\mathcal{H}}$ loss for $q_{\mathcal{H}} = \mathsf{poly}(\kappa)$ distinct random oracle queries) and simulates the random oracle query on all $x_i \neq x^\star$ as $\mathcal{H}(x_i) := (g^{s_i}, g^{\omega s_i})$ where $\omega, s_1, s_2, \ldots$ are each sampled uniformly random from $\mathbb{Z}_p$. This implicitly sets $t_i := \omega s_i$. Note that, each query has the same ω, but a different s_i – this way the challenger ensures that the attacker does not learn any new information by making more queries. However, for $x^\star$ the random oracle is programmed as usual by sampling random $s^\star, t^\star$ values as $\mathcal{H}(x^\star) := (g^{s^\star}, g^{t^\star})$.

Claim 7.5 *Assuming DDH is hard in group G, we have that* $\mathsf{PseudoRand}_{\mathcal{A}}(b) \approx_{\mathsf{comp}} \mathsf{Hyb}^1_{\mathcal{A}}(b)$.

This claim follows analogously to Theorem 7.1.

$\mathsf{Hyb}^{1.5}_{\mathcal{A}}(b)$**.** In this hybrid the only changes made are in the NIZK proofs– the challenger sends simulated NIZK proofs instead of the actual NIZK proofs to the attacker for *all* honest evaluation/challenge requests. Hence, the NIZK proofs are made independent of the witnesses $(u_i, v_i, \rho_i^1, \rho_i^2)$.

From the adaptive zero-knowledge property of NIZK we conclude that $\mathsf{Hyb}^1_{\mathcal{A}}(b) \approx_{\mathsf{comp}} \mathsf{Hyb}^{1.5}_{\mathcal{A}}(b)$. Note that, adaptive zero-knowledge is crucial here as the attacker may corrupt a party, on behalf of which a simulated proof has already been sent earlier. The corruption request is simulated by providing a randomness along with the witnesses $((u_i, v_i, \rho_i^1, \rho_i^2)$ for party i in this case) to explain the simulated proof.

$\mathsf{Hyb}^2_{A}(b)$**.** In this hybrid we do not make any change from $\mathsf{Hyb}^1_{\mathcal{A}}(b)$ except that all non-challenge evaluation queries are responded with an independent key $k \leftarrow_{\$} \mathbb{Z}_p$ sampled uniformly at random, whereas the corruption query for party j are answered using randomly sampled u_j, v_j subject to $u_j + \omega v_j = k_j$ where k_j is the j-th Shamir's share of k. In particular, for a non-challenge evaluation query $\mathsf{Eval}(x_i, j)$, the challenger returns $g^{s_i k_j}$ where $\mathcal{H}(x_i) = (g^{s_i}, g^{\omega s_i})$. Challenge queries and the evaluation query on $x^\star$ are answered similar to $\mathsf{Hyb}^1_{\mathcal{A}}(b)$.

Claim 7.6 *We have that:* $\mathsf{Hyb}^1_{\mathcal{A}}(b) \approx_{\mathsf{stat}} \mathsf{Hyb}^2_{\mathcal{A}}(b)$

To prove this information theoretic claim we use that the fact that the commitments are *statistically hiding* and hence they do not reveal any additional information. Furthermore, in both the hybrids the proofs are simulated and hence they are too statistically independent of the witnesses. The rest of argument follows analogous to the proof of Theorem 7.1, and therefore we omit the details.

7.3 PRF-based Construction.

The PRF-based construction of NPR (also used in DiSE) can be easily shown to be adaptive secure using complexity leveraging, just by guessing the corrupt set ahead of time in the security game. This, of course, incurs a loss proportional to $O(\binom{n}{t})$ in the security as there are that many possible corrupt sets. However, since the complexity of this scheme is anyway proportional to $O(\binom{n}{t})$, this natural solution is asymptotically optimal in this case. As a result the DP-PRF construction can be used to protect against adaptive corruption readily without any change in certain settings, in particular when $\binom{n}{t}$ is a polynomial in the security parameter.

■

Parameters: Let $G = \langle g \rangle$ be a multiplicative cyclic group of prime order p in which the DDH assumption holds, $\mathcal{H} : \{0,1\}^* \to G^2$ and $\mathcal{H}' : \{0,1\}^* \to \{0,1\}^{\mathsf{poly}(\kappa)}$ be two hash functions modeled as random oracles. Let SSS be Shamir's secret sharing scheme (Def. A.2), $\mathsf{SCom} := (\mathsf{Setup}_{\mathsf{com}}, \mathsf{Com})$ be a **statistically hiding** commitment scheme (Def. A.1)and **adaptive** $\mathsf{NIZK} := (\mathsf{Prove}^{\mathcal{H}'}, \mathsf{Verify}^{\mathcal{H}'})$ be a simulation-sound adaptive NIZK proof system (Def. A.3).

- $\mathsf{Setup}(1^\kappa, n, t) \to ([\![\mathbf{sk}]\!]_{[n]}, pp)$. Sample $(u, v) \leftarrow_\$ \mathbb{Z}_p^2$ and get $(u_1, \ldots, u_n) \leftarrow \mathsf{SSS}(n, t, p, u)$ and $(v_1, \ldots, v_n) \leftarrow \mathsf{SSS}(n, t, p, v)$. Run $\mathsf{Setup}_{\mathsf{com}}(1^\kappa)$ to get pp_{com}. Compute commitments $\gamma_i := \mathsf{Com}(u_i, pp_{\mathsf{com}}; \rho_i^1)$ and $\delta_i := \mathsf{Com}(v_i, pp_{\mathsf{com}}; \rho_i^2)$ by picking ρ_i^1, ρ_i^2 at random. Set $pp = (p, g, G, (\gamma_1, \delta_1) \ldots, (\gamma_n, \delta_n), pp_{\mathsf{com}})$, $sk_i := (u_i, v_i, \rho_i^1, \rho_i^2)$ and give sk_i to party i, for $i \in [n]$.
- $\mathsf{Eval}(sk_i, x, pp) \to z_i$. Compute $w_1, w_2 := H(x)$ and $h_i := w_1^{u_i} w_2^{v_i}$. Run $\mathsf{Prove}^{\mathcal{H}'}$ with the statement stmt_i: $\{\exists u_i, v_i, \rho_i^1, \rho_i^2$ s.t. $h_i = w_1^{u_i} w_2^{v_i} \wedge \gamma_i = \mathsf{Com}(u_i, pp_{\mathsf{com}}; \rho_i^1) \wedge \delta_i = \mathsf{Com}(v_i, pp_{\mathsf{com}}; \rho_i^2)\}$ and witness $(u_i, v_i, \rho_i^1, \rho_i^2)$ to obtain a proof π_i. Output $((w_1, w_2, h_i), \pi_i)$.
- $\mathsf{Combine}(\{(i, z_i)\}_{i \in S}, pp) =: z/\perp$. If $|S| < t$, output $\perp$. Else, parse z_i as $((w_1, w_2, h_i), \pi_i)$ and check if $\mathsf{Verify}^{\mathcal{H}'}(\mathsf{stmt}_i, \pi_i) = 1$ for all $i \in S$. If check fails for any i, output $\perp$. Else, output $\prod_{i \in S} h_i^{\lambda_{0,i,S}}$.

Figure 2: A *strongly* adaptively secure DPRF protocol $\Pi_{\mathsf{str\text{-}adap}}$based on DDH. Differences from $\Pi_{\mathsf{adap}}$are highlighted in blue.

8 Adaptively secure TSE Construction

In this section, we show that, plugging in *any* **adpatively**-secure (strong) DPRF to the DiSE TSE construction one can obtain an **adaptively** secure (strongly-correct) TSE scheme. A full description of the DiSE construction is provided verbatim in Figure 3 for completeness. Our result can be formalized in the following theorem. The proof is very similar to the non-adaptive case and therefore we just mention the adjustments needed to augment it to the adaptive setting and skip the details.

Theorem 8.1 *The TSE scheme DiSE of Figure 3 is (strongly)-adaptive-secure if the underlying DPRF* DP *is (strongly)-adaptive-secure.*

Proof. We show each property of DiSE separately.

Ingredients:

- An (n,t)-DPRF protocol $\mathsf{DP} := (\mathsf{DP.Setup}, \mathsf{Eval}, \mathsf{Combine})$ (Def. 5.1).
- A pseudorandom generator PRG of polynomial stretch.
- A commitment scheme $\Sigma := (\Sigma.\mathsf{Setup}, \mathsf{Com})$

$\underline{\mathsf{Setup}(1^\kappa, n, t) \to \left(\llbracket \mathbf{sk} \rrbracket_{[n]}, pp\right)}$: Run $\mathsf{DP.Setup}(1^\kappa, n, t)$ to get $((rk_1, \ldots, rk_n), pp_{\mathsf{DP}})$ and $\Sigma.\mathsf{Setup}(1^\kappa)$ to get pp_{com}. Set $sk_i := rk_i$ for $i \in [n]$ and $pp := (pp_{\mathsf{DP}}, pp_{\mathsf{com}})$.

$\underline{\mathsf{DistEnc}(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j : m, S], pp) \to [j : c/\perp]}$: To encrypt a message m with the help of parties in S:

- Party j computes $\alpha := \mathsf{Com}(m, pp_{\mathsf{com}}; \rho)$ for a randomly chosen ρ and sends α to all parties in S.
- For every $i \in S$, party i runs $\mathsf{Eval}(sk_i, j \| \alpha, pp)$ to get z_i, and sends it to party j.
- Party j runs $\mathsf{Combine}(\{(i, z_i)\}_{i \in S}, pp)$ to get w or $\perp$. In the latter case, it outputs $\perp$. Otherwise, it computes $e := \mathsf{PRG}(w) \oplus (m \| \rho)$ and then outputs $c := (j, \alpha, e)$.

$\underline{\mathsf{DistDec}(\llbracket \mathbf{sk} \rrbracket_{[n]}, [j' : c, S], pp) \to [j' : m/\perp]}$: To decrypt a ciphertext c with the help of parties in S:

- Party j' first parses c into (j, α, e). Then it sends $j \| \alpha$ to all the parties in S.
- For $i \in S$, party i receives x and checks if it is of the form $j^\star \| \alpha^\star$ for some $j^\star \in [n]$. If not, then it sends $\perp$ to party j'. Else, it runs $\mathsf{Eval}(sk_i, x, pp)$ to get z_i, and sends it to party j'.
- Party j' runs $\mathsf{Combine}(\{(i, z_i)\}_{i \in S}, pp)$ to get w or $\perp$. In the latter case, it outputs $\perp$. Otherwise, it computes $m \| \rho := \mathsf{PRG}(w) \oplus e$ and checks if $\alpha = \mathsf{Com}(m, pp_{\mathsf{com}}; \rho)$. If the check succeeds, it outputs m; otherwise, it outputs $\perp$.

Figure 3: DiSE: our threshold symmetric-key encryption protocol.

Consistency. Consistency remains unchanged from the static case.

Lemma 8.2 ((Strong)-adaptive-correctness) *DiSE is an adaptively correct* TSE *scheme.*

Proof sketch. The adaptive (strong)-correctness definition of TSE (Def. 6.3) is very similar to the static case. The only change takes place in the corruption, where an adaptive adversary may dynamically corrupt parties. Adaptive corruption does not alter the proof of correctness which relies on the binding of the commitment scheme. For the strong correctness we need to rely on the adaptive correctness of the underlying DPRF scheme. Again, we notice that the adaptive correctness of DPRF (Def. 5.6) is very similar to its static correctness as provided in DiSE, except the corruption capabilities. So, any adaptive corruption query of a TSE adversary can be simulated using the adaptive corruption query of the DPRF scheme. The rest of the proof follows from the static case.

■

Lemma 8.3 (Adaptive message privacy) *If* DP *is a adaptively secure DPRF (Def. 5.2), then DiSE is an adaptively message-private* TSE *scheme.*

Proof sketch. Let us recall the intuition of static message privacy from [AMMR18a]. The challenge ciphertext $c^\star$ has the form $(j^\star, \alpha^\star, e^\star)$ where $e^\star = \mathsf{PRG}(w^\star) \oplus (m_b \| \rho^\star)$, $\alpha^\star = \mathsf{Com}(m_b, pp_{\mathsf{com}}; \rho^\star)$ and $w^\star$ is the output of DPRF DP on $j^\star \| \alpha^\star$. Clearly, due to the masking with PRG, it is computationally hard guess b with probability significantly better than $\frac{1}{2}$ if $w^\star$ is indistinguishable from random.

Now, in the adaptive case, the adaptive pseudorandomness property of DP ensures this as long as an adaptive adversary $\mathcal{A}$ has no way of evaluating the DPRF on $j^\star \| \alpha^\star$ itself, as due to hiding of the commitment scheme Σ no information about m_b is leaked from $\alpha^\star$. So it boils down to the unpredictability of the DPRF output on $j^\star \| \alpha^\star$. Note that, the definition (Def. 6.5) of adaptive message privacy ensures that $j^\star$ is not corrupt at any point of time by checking $j^\star \notin C$. So, it is not possible to learn the DPRF output on $j^\star \| \alpha^\star$ by corrupting $j^\star$ after the challenge phase and accessing its past state. Alternatively, the attacker may hope to win the game by using the indirect decryption queries initiated from an honest party, but corrupting the initiator later. However, this is taken care of by ensuring that any initiator of the indirect decryption query may not be corrupt at any time (see the "guess" phase of Def. 6.5). All other potential attack strategies are dealt with similar to the static case as provided in DiSE.

■

Lemma 8.4 ((Strong) adaptive authenticity) *If* DP *is an (strongly)-adaptively secure DPRF, then DiSE is a* TSE *scheme that satisfies (strong) adaptive-authenticity.*

Proof sketch. Again, let us recall the intuition from the static case [AMMR18a]. If there are two forged ciphertexts $c_1 = (j, \alpha, e_1)$, $c_2 = (j, \alpha, e_2)$ with the same (j, α), the underlying message-randomness pair (m_1, ρ_1) and (m_2, ρ_2) recovered from c_1 and c_2, respectively, must be different (this also uses the consistency of the DPRF). Binding of Σ, ensures that decryption of one of c_1, c_2 fails, and subsequently AUTH outputs 0. So, in order to succeed the attacker must output $k+1$ ciphertexts each with a unique (j, α). So, it boils down to showing that the attacker can not acquire more information than producing k valid ciphertexts.

In the adaptive setting it becomes trickier to track the exact amount of information acquired by the adversary before outputting forgery. However, in the definition (Def. 6.7) we handle this by delaying the tracking until the very end when the set of the entire corrupt set is known. This is enabled by recording all pairs (j, S) used in a query in the list L_{act}. In fact, the list L_{act} may contain multiple entries (j, S) as the attacker may use the same set to compute multiple valid ciphertexts. As discussed in the preceding paragraph, storing this information is sufficient to count the exact number of possible ciphertexts the attacker can produce. The rest of the proof follows similar to the static counterpart.

To argue strong adaptive authenticity, one needs to additionally rely on adaptive correctness of the underlying DPRF– this can be easily adapted from the static case.

■

■

9 Our Adaptive Non-interactive Zero-knowledge.

Here we show a simple and efficient construction of adaptive NIZK for the specific commit-and-prove statement required for our construction $\Pi_{\mathsf{str\text{-}adap}}$. We provide the relevant definitions in Supp A.3.

Consider a group G of prime order p where discrete log is hard. Let g be a generator. Let $\mathsf{SCom} := (\mathsf{Setup}_{\mathsf{com}}, \mathsf{Com})$ be a the Pederson's commitment scheme where $\mathsf{Setup}_{\mathsf{com}}$ returns (g, h) such that $h = g^x$ and $\mathsf{Com}(m; r) := g^m h^r$. It is easy to see that this is a statistically hiding commitment scheme. Let $\mathcal{H} : \{0,1\}^\star \to G^2$ and $\mathcal{H}' : \{0,1\}^* \to \mathbb{Z}_p$ be hash functions modeled as random oracles. We construct a NIZK proof system for the relation

$R_{\mathsf{com\text{-}prov}} = \{x, z, c_1, c_2 \ : \ \exists\ (k_1, k_2, \rho_1, \rho_2)$ such that $\forall\ i \in \{1,2\},\ c_i = \mathsf{Com}(k_i; \rho_i)$ and $z = w_1^{k_1} w_2^{k_2}$ where $(w_1, w_2) := \mathcal{H}(x)\}$.

The main idea is to use a Schnorr's proof [Sch90] along with Fiat-Shamir transformation [FS87]. Recall that, the Schnorr proof system can be used prove a knowledge of exponent, in this case knowledge of k_1, k_2 for which $z = w_1^{k_1} w_2^{k_2}$. Nevertheless, this does not prove anything about the individual k_i. Separately, the commitment, when instantiated with Pederson's, can also be thought of as knowledge of exponents of k_1, ρ_1 and k_2, ρ_2 individually and can be proven using Schnorr-like scheme–this is possible due to the homomorphic property of the commitment scheme, by which $\mathsf{Com}(m_1; r_1)$ and $\mathsf{Com}(m_2; r_2)$ can be combined to produce a commitment of $\mathsf{Com}(m_1 + m_2; r_1 + r_2)$. These two separate proofs are bound together (that is, the same k_i are used) by using the same challenge e for verification.

$\mathsf{Prove}^{\mathcal{H},\mathcal{H}'}((x, z, c_1, c_2), (k_1, k_2, \rho_1, \rho_2))$: The prover works as follows:

- Sample randomnesses $v_1, v_2, \hat{v}_1, \hat{v}_2$ from $\mathbb{Z}_p$.
- Let $(w_1, w_2) := \mathcal{H}(x)$.
- Compute $t := w_1^{v_1} w_2^{v_2}$; $\hat{t}_1 := \mathsf{Com}(v_1; \hat{v}_1)$ and $\hat{t}_2 := \mathsf{Com}(v_2; \hat{v}_2)$.
- Generate the challenge (Fiat-Shamir) $e := \mathcal{H}'(t, \hat{t}_1, \hat{t}_2)$.
- Compute $u_i := v_i + ek_i$ and $\hat{u}_i := \hat{v}_i + e\rho_i$ for all $i \in \{1, 2\}$.
- Output $\pi := ((t, \hat{t}_1, \hat{t}_2), e, (u_1, u_2, \hat{u}_1, \hat{u}_2))$.

$\mathsf{Verify}^{\mathcal{H},\mathcal{H}'}(s := (x, z, c_1, c_2), \pi := ((t, \hat{t}_1, \hat{t}_2), e, (u_1, u_2, \hat{u}_1, \hat{u}_2))$ The verifier computes $(w_1, w_2) := \mathcal{H}(x)$ and then checks the following and output 1 if and only if all of them succeeds, and 0 otherwise:

- $e = \mathcal{H}'(t, \hat{t}_1, \hat{t}_2)$.
- $w_1^{u_1} w_2^{u_2} = t z^e$.
- $h^{\hat{u}_1} g^{u_1} = \hat{t}_1 c_1^e$.
- $h^{\hat{u}_2} g^{u_2} = \hat{t}_2 c_2^e$.

Lemma 9.1 *The above protocol is a adaptive NIZK argument system in the random oracle model assuming DDH.*

Proof. Perfect completeness is obvious. The simulation soundness follows from a standard Fiat-Shamir rewinding argument in ROM. We show the adaptive zero-knowledge, for which we construct simulators $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ as follows:

- $\underline{\mathcal{S}_1}$. This algorithm simulates fresh random oracle queries on x for $\mathcal{H}$ by sampling $\alpha, \beta \leftarrow_{\$} \mathbb{Z}_p^2$, storing (x, α, β) in table Q_1 and finally returning the pair (g^α, g^β). Furthermore, it also simulates fresh random oracle queries for $\mathcal{H}'$ by returning a uniform random value in G and storing the input-output pair in Q_2. Repeating queries are simulated using the tables Q, Q_2 appropriately. Furthermore, $\mathcal{S}_1$ can be asked by $\mathcal{S}_2$ or $\mathcal{S}_3$ to program both $\mathcal{H}$, $\mathcal{H}'$ with specific input-output pairs– if that pair is already defined (queried by $\mathcal{A}$ earlier), then $\mathcal{S}_1$ fails to program.

– $\underline{\mathcal{S}_2.}$ This algorithm, on input (x, z, c_1, c_2) works as follows:
  - sample uniform random $e, u_1, u_2, \hat{u}_1, \hat{u}_2$ from $\mathbb{Z}_p$ and define $\rho_{\mathcal{S}} := (e, u_1, u_2, \hat{u}_1, \hat{u}_2)$;
  - compute $(w_1, w_2) := \mathcal{H}(x)$;
  - set $t := w_1^{u_1} w_2^{u_2} z^{-e}$, $\{\hat{t}_i := h^{\hat{u}_i} g^{u_i} c_i^e\}_{i \in \{1,2\}}$;
  - ask $\mathcal{S}_1$ to program $\mathcal{H}'$ for input $(t, \hat{t}_1, \hat{t}_2)$ and output e;
  - returns $\pi := ((t, \hat{t}_1, \hat{t}_2), e, (u_1, u_2, \hat{u}_1, \hat{u}_2))$.

– $\underline{\mathcal{S}_3.}$ This algorithm, on input statement (x, z, c_1, c_2), witness $(k_1, k_2, \rho_1, \rho_2)$, and $\mathcal{S}_2$'s randomness $\rho_{\mathcal{S}} = (e, u_1, u_2, \hat{u}_1, \hat{u}_2)$ works as follows:
  - use $\mathcal{S}_2((x, z, c_1, c_2); \rho_{\mathcal{S}})$ to generate $\pi = ((t, \hat{t}_1, \hat{t}_2), e, (u_1, u_2, \hat{u}_1, \hat{u}_2))$ as above;
  - then compute $v_i := u_i - e k_i$ and $\hat{v}_i := \hat{u}_i - e\rho_i$ for $i \in \{1, 2\}$;
  - output $r := (v_1, v_2, \hat{v}_1, \hat{v}_2)$

It is straightforward to see that the above simulators indeed satisfy the adaptive zero-knowledge property. This concludes the proof.

■

References

[AMMR18a] Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee, and Peter Rindal. DiSE: Distributed symmetric-key encryption. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1993–2010, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[AMMR18b] Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee, and Peter Rindal. DiSE: Distributed symmetric-key encryption. Cryptology ePrint Archive, Report 2018/727, 2018. https://eprint.iacr.org/2018/727.

[AMN01] Michel Abdalla, Sara Miner, and Chanathip Namprempre. Forward-secure threshold signature schemes. In *Cryptographers Track at the RSA Conference*, pages 441–456. Springer, 2001.

[BBH06] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 226–243, San Jose, CA, USA, February 13–17, 2006. Springer, Heidelberg, Germany.

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 201–218, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany.

[CG99] Ran Canetti and Shafi Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 90–106, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th Annual ACM Symposium on Theory of Computing*, pages 522–533, Montréal, Québec, Canada, May 23–25, 1994. ACM Press.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[DK01] Ivan Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 152–165, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

[DK10] Ivan Damgård and Marcel Keller. Secure multiparty AES. In Radu Sion, editor, *FC 2010: 14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 367–374, Tenerife, Canary Islands, Spain, January 25–28, 2010. Springer, Heidelberg, Germany.

[dya] Dyadic Security. https://www.dyadicsec.com.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012: 13th International Conference in Cryptology in India*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79, Kolkata, India, December 9–12, 2012. Springer, Heidelberg, Germany.

[FS87] 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*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

[GHKR08] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Threshold RSA for dynamic and ad-hoc groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 88–107, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[GJKR96] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 354–371, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

[GRR+16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. MPC-friendly symmetric key primitives. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 430–443, Vienna, Austria, October 24–28, 2016. ACM Press.

[KHF+20] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: exploiting speculative execution. *Commun. ACM*, 63(7):93–101, 2020.

[LJY14] Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM Symposium Annual on Principles of Distributed Computing*, pages 303–312, Paris, France, July 15–18, 2014. Association for Computing Machinery.

[LJY16] Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares. *Theor. Comput. Sci.*, 645:1–24, 2016.

[LSG+20] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, Mike Hamburg, and Raoul Strackx. Meltdown: reading kernel memory from user space. *Commun. ACM*, 63(6):46–56, 2020.

[LST18] Benoît Libert, Damien Stehlé, and Radu Titiu. Adaptively secure distributed PRFs from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 391–421, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

[por] Porticor Cloud Security. http://www.porticor.com/. Acquired by Intuit.

[RSS17] Dragos Rotaru, Nigel P. Smart, and Martijn Stam. Modes of operation suitable for computing on encrypted data. Cryptology ePrint Archive, Report 2017/496, 2017. http://eprint.iacr.org/2017/496.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[sep] Sepior. https://sepior.com.

[SG02] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, March 2002.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[vau] Vault by HashiCorp. https://www.vaultproject.io/.

Supplementary Material

A Cryptographic primitives used as building blocks

We include definitions of some well-known primitives for completeness, mostly taken verbatim from [AMMR18b].

A.1 Commitment

Definition A.1 (Commitment Scheme) *A (non-interactive) commitment scheme Σ consists of two PPT algorithms $(\mathsf{Setup}_{\mathsf{com}}, \mathsf{Com})$ which satisfy hiding and binding properties:*

- $\mathsf{Setup}_{\mathsf{com}}(1^\kappa) \to pp_{\mathsf{com}}$: *It takes the security parameter as input, and outputs some public parameters.*
- $\mathsf{Com}(m, pp_{\mathsf{com}}; r) =: \alpha$: *It takes a message m, public parameters pp_{com} and randomness r as inputs, and outputs a commitment α.*

Hiding. A commitment scheme $\Sigma = (\mathsf{Setup}_{\mathsf{com}}, \mathsf{Com})$ is statistically/computationally hiding if for all unbounded/PPT adversaries $\mathcal{A}$, all messages m_0, m_1, there exists a negligible function negl such that for $pp_{\mathsf{com}} \leftarrow \mathsf{Setup}_{\mathsf{com}}(1^\kappa)$,

$$|\Pr[\mathcal{A}(pp_{\mathsf{com}}, \mathsf{Com}(m_0, pp_{\mathsf{com}}; r_0)) = 1] - \Pr[\mathcal{A}(pp_{\mathsf{com}}, \mathsf{Com}(m_1, pp_{\mathsf{com}}; r_1)) = 1]| \leq \mathsf{negl}(\kappa),$$

where the probability is over the randomness of $\mathsf{Setup}_{\mathsf{com}}$, random choice of r_0 and r_1, and the coin tosses of $\mathcal{A}$. Unless mentioned explicitly we assume that a commitment is computationally hiding, which suffices for construction Π_{adap} (Fig. 1). However for the construction $\Pi_{\mathsf{str\text{-}adap}}$ (Fig. 2) we need the commitment scheme to be statistically hiding. We can use, for example Pederson's commitment for that. We briefly describe Pederson's commitment in Sec. 9.

Binding. A commitment scheme $\Sigma = (\mathsf{Setup}_{\mathsf{com}}, \mathsf{Com})$ is binding if for all PPT adversaries $\mathcal{A}$, if $\mathcal{A}$ outputs m_0, m_1, r_0 and r_1 $((m_0, r_0) \neq (m_1, r_1))$ given $pp_{\mathsf{com}} \leftarrow \mathsf{Setup}_{\mathsf{com}}(1^\kappa)$, then there exists a negligible function negl such that

$$\Pr[\mathsf{Com}(m_0, pp_{\mathsf{com}}; r_0) = \mathsf{Com}(m_1, pp_{\mathsf{com}}; r_1)] \leq \mathsf{negl}(\kappa),$$

where the probability is over the randomness of $\mathsf{Setup}_{\mathsf{com}}$ and the coin tosses of $\mathcal{A}$.

A.2 Secret Sharing

Definition A.2 (Shamir's Secret Sharing) *Let p be a prime. An (n, t, p, s)-Shamir's secret sharing scheme is a randomized algorithm SSS that on input four integers n, t, p, s, where $0 < t \leq n < p$ and $s \in \mathbb{Z}_p$, outputs n shares $s_1, \ldots, s_n \in \mathbb{Z}_p$ such that the following two conditions hold for any set $\{i_1, \ldots, i_\ell\}$:*

- *if $\ell \geq t$, there exists fixed (i.e., independent of s) integers $\lambda_1, \ldots, \lambda_\ell \in \mathbb{Z}_p$ (a.k.a. Lagrange coefficients) such that $\sum_{j=1}^{\ell} \lambda_j s_{i_j} = s \mod p$;*

– *if $\ell < t$, the distribution of $(s_{i_1}, \ldots, s_{i_\ell})$ is uniformly random.*

Concretely, Shamir's secret sharing works as follows. Pick $a_1, \ldots, a_{t-1} \leftarrow_\$ \mathbb{Z}_p$. Let $f(x)$ be the polynomial $s + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_{t-1} \cdot x^{t-1}$. Then s_i is set to be $f(i)$ for all $i \in [n]$.

A.3 Adaptive NIZK

Let R be an efficiently computable binary relation. For pairs $(s, w) \in R$, we refer to s as the statement and w as the witness. Let L be the language of statements in R, i.e. $L = \{s : \exists w \text{ such that } R(s, w) = 1\}$. We define **adaptive** non-interactive zero-knowledge arguments of knowledge in the random oracle model based on the work of Faust et al. [FKMV12], but augmented to the adaptive case similar to [GOS12].

Definition A.3 (Adaptive Non-interactive Zero-knowledge Argument of Knowledge) *Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{\mathsf{poly}(\kappa)}$ be a hash function modeled as a random oracle. An* **adaptive** NIZK *for a binary relation R consists of two PPT algorithms* Prove *and* Verify *with oracle access to $\mathcal{H}$ defined as follows:*

– $\mathsf{Prove}^{\mathcal{H}}(s, w)$ *takes as input a statement s and a witness w, and outputs a proof π if $(s, w) \in R$ and $\perp$ otherwise.*

– $\mathsf{Verify}^{\mathcal{H}}(s, \pi)$ *takes as input a statement s and a candidate proof π, and outputs a bit $b \in \{0, 1\}$ denoting acceptance or rejection.*

These two algorithms must satisfy the following properties:

– ***Perfect completeness**: For any $(s, w) \in R$,*

$$\Pr\left[\mathsf{Verify}^{\mathcal{H}}(s, \pi) = 1 \mid \pi \leftarrow \mathsf{Prove}^{\mathcal{H}}(s, w)\right] = 1.$$

– ***Adaptive Zero-knowledge**: There must exist a triple of PPT simulators $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ such that for all PPT adversary $\mathcal{A}$,*

$$\left|\Pr[\mathcal{A}^{\mathcal{H}, \mathrm{PR}^{\mathcal{H}}(\cdot,\cdot)}(1^\kappa) = 1] - \Pr[\mathcal{A}^{\mathcal{S}_1(\cdot), \mathrm{SR}(\cdot,\cdot)}(1^\kappa) = 1]\right| \leq \mathsf{negl}(\kappa)$$

for some negligible function negl, where

  – $\mathrm{PR}^{\mathcal{H}}$*, on input a statement-witness pair (s, w), samples a randomness r, runs $\pi \leftarrow \mathrm{Prove}^{\mathcal{H}}(s, w)$ and returns a proof-randomness pair (π, r).*

  – *$\mathcal{S}_1$ simulates the random oracle $\mathcal{H}$;*

  – SR*, on input a statement-witness pair (s, w), samples a randomness $\rho_{\mathcal{S}}$, runs $\pi \leftarrow \mathcal{S}_2(s; \rho_{\mathcal{S}})$ and then $r \leftarrow \mathcal{S}_3(s, w, \rho_{\mathcal{S}})$. It returns a simulated proof-randomness pair (π, r) if $(s, w) \in R$ and $\perp$ otherwise;*

  – *$\mathcal{S}_1$, $\mathcal{S}_2$ $\mathcal{S}_3$ may share states.*

– ***Simulation soundness**: There must exist a PPT simulator $\mathcal{S}_1$ such that for all PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}^{\mathcal{A}}$ such that*

$$\Pr\Big[(s, w) \notin R \text{ and } \mathsf{Verify}^{\mathcal{H}}(s, \pi) = 1 \mid (s, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_1(\cdot)}(1^\kappa); w \leftarrow \mathcal{E}^{\mathcal{A}}(s, \pi, Q)\Big] \leq \mathsf{negl}(\kappa)$$

for some negligible function negl, where

– $\mathcal{S}_1$ *is like above;*

– Q *is the list of (query, response) pairs obtained from* $\mathcal{S}_1$.

Fiat-Shamir transform. Let $(\mathsf{Prove}, \mathsf{Verify})$ be a three-round public-coin honest-verifier zero-knowledge interactive proof system (a sigma protocol) with unique responses. Let $\mathcal{H}$ be a function with range equal to the space of the verifier's coins. In the random oracle model, the proof system $(\mathsf{Prove}^{\mathcal{H}}, \mathsf{Verify}^{\mathcal{H}})$ derived from $(\mathsf{Prove}, \mathsf{Verify})$ by applying the Fiat-Shamir transform satisfies the zero-knowledge and argument of knowledge properties defined above. See Definition 1, 2 and Theorem 1, 3 in Faust et al. [FKMV12] for more details. (They actually show that these properties hold even when adversary can ask for proofs of false statements.)

B Detailed proof of Theorem 7.1

Since consistency is obvious, here we provide a full proof of adaptive pseudorandomness of the construction given in Fig. 1. For a sketch we refer to Sec. 7.1.

Comparison with the DiSE DPRF proof of static case. Somewhat surprisingly our proof is simpler than the DiSE DPRF proof (Appendix C.4 of [AMMR18b]) even if our construction provides a strictly stronger guarantee than theirs. This is, in fact, due to the fact that our construction has a feature by which the response to the evaluation queries on the non-challenge values are made independent of the responses to the challenge queries easily (the same feature is used to achieve resilience against adaptive corruption). In DiSE DPRF, one needs to carefully use the knowledge of corrupt set to reach a similar hybrid.

Formally, we prove a reduction from DDH assumption in the random oracle model. We will go through a number of hybrid games. For a fixed $b \in \{0,1\}$ and a PPT adversary $\mathcal{A}$ starting from the real game $\mathsf{PseudoRand}_{\mathcal{A}}(b)$ we will be reaching a hybrid $\mathsf{Hyb}^2_{\mathcal{A}}(b)$. Then we shall show that $\mathsf{Hyb}^2_{\mathcal{A}}(0)$ and $\mathsf{Hyb}^2_{\mathcal{A}}(1)$ are statistically indistinguishable. We highlight the difference with between successive hybrids in *red*. We start by describing the real game $\mathsf{PseudoRand}_{\mathcal{A}}(b)$ concretely for our scheme Π_{adap}.

$\underline{\mathsf{PseudoRand}_{\mathcal{A}}(b):}$

1. Give the public parameters $pp := (p, g, G)$ (G is a cyclic group of order p and g is a generator of G) to $\mathcal{A}$.

2. Program the random oracle $\mathcal{H}$ as follows: Initialize $\mathcal{L}_{\mathcal{H}} := \emptyset$. For random oracle call with input x:

   1. If there exists a tuple $(x, s, t, w_1, w_2) \in \mathcal{L}_H$, output (w_1, w_2).
   2. Otherwise, choose $(s, t) \leftarrow_{\$} \mathbb{Z}_p^2$ and set $w_1 := g^s$ and $w_2 := g^t$. Update $\mathcal{L}_{\mathcal{H}} := \mathcal{L}_{\mathcal{H}} \cup (x, s, t, w_1, w_2)$ and output (w_1, w_2).

   Give random oracle access to $\mathcal{A}$.

3. Choose a couple of $(t-1)$-degree random polynomials f_1, f_2. Define $u_i := f_1(i)$ and $v_i := f_2(i)$ for $i \in [n] \cup \{0\}$. Initialize states for each party $i \in [n]$ as $\mathsf{st}_i := \{u_i, v_i\}$. Initialize $C := \emptyset$. At any point on receiving a new set of corrupt parties $\tilde{C}$ from $\mathcal{A}$, send the corresponding secret states $\{\mathsf{st}_i\}_{i \in \tilde{C}}$ to $\mathcal{A}$ and update $C := C \cup \tilde{C}$. For simplicity

we assume that the final set C is equal to $\{1, \ldots, \ell\}$, which is without loss of generality because we will not be using this information in the proof (otherwise handling the adaptive corruption would have been trivial).

4. On an evaluation query $(\texttt{Eval}, x, i)$ for an honest i, return $h_i := w_{1,i}^{u_i} w_{2,i}^{v_i}$ where $\mathcal{H}(x) = (w_{1,i}, w_{2,i}) \in G^2$; update $\texttt{st}_i := \texttt{st}_i \cup \{x\}$. Build a list L_x if this is the first evaluation query on x, or update exiting L_x by appending i to it.

5. On the challenge query $(\texttt{Challenge}, x^\star, S^\star, g_1^\star, \ldots, g_u^\star)$ for $m \leq \ell$ (without loss of generality assume that $S^\star \cap C = [m]$):

   1. Set $g_i^\star := w^{\star u_i}_{1,i} w^{\star v_i}_{2,i}$ for $i \in S^\star \setminus C$.
   2. Depending on b do as follows:
      1. If $b = 0$ then compute $z^\star := \prod_{i \in S^\star} g_i^{\star \lambda_{0,i,S}}$.
      2. Else, choose a random $z^\star \leftarrow_\$ G$.
   3. Send $z^\star$ to $\mathcal{A}$.

6. Continue answering evaluation queries as before.

7. Receive a guess b' from $\mathcal{A}$; then do as follows:

   – if the total number of corrupt parties, $|C| \geq t$ then output 0 and stop;
   – if the challenge $x^\star$ has been queried for evaluation for at least $g := t - |C|$ honest parties, that is if $L_{x^\star} \cap ([n] \setminus C) \geq g$ then output 0 and stop;
   – otherwise output b'.

In the next hybrid experiment $\mathsf{Hyb}^1_{\mathcal{A}}(b)$ the only change we make is: the challenger guesses the challenge input $x^\star$ randomly (incurring a $1/q_{\mathcal{H}}$ loss for $q_{\mathcal{H}} = \mathsf{poly}(\kappa)$ distinct random oracle query) and simulates the random oracle query on $x_i \neq x^\star$ as $\mathcal{H}(x_i) := (g^{s_i}, g^{\omega s_i})$ where $\omega, s_1, s_2, \ldots$ are sampled uniformly random from $\mathbb{Z}_p$. This implicitly sets $t_i := ws_i$. Note that, each query has the same ω, but a different s_i – this way the challenger ensures that the attacker does not learn any new information by making more queries. However, for $x^\star$ the random oracle is programmed as usual by sampling random $s^\star, t^\star$ values as $\mathcal{H}(x^\star) := (g^{s^\star}, g^{t^\star})$. We formally describe it below:

$\underline{\mathsf{Hyb}^1_{\mathcal{A}}(b) :}$

1. Give the public parameters $pp := (p, g, G)$ (G is a cyclic group of order p and g is a generator of G) to $\mathcal{A}$. Suppose $\mathcal{A}$ makes $q_{\mathcal{H}}$ distinct random oracle queries in total.

2. Guess $j \leftarrow_\$ [q_{\mathcal{H}}]$ and let the j-th query be on $x^\star$. Choose a random value $\omega \leftarrow_\$ \mathbb{Z}_p$. Then program the random oracle $\mathcal{H}$ as follows: Initialize $\mathcal{L}_{\mathcal{H}} := \emptyset$. For random oracle call on x do as follows:

   1. If there exists a tuple $(x, s, t, w_1, w_2) \in \mathcal{L}_H$, output (w_1, w_2).
   2. Otherwise,
      – if $x \neq x^\star$: choose $s \leftarrow_\$ \mathbb{Z}_p$ and define $t := s\omega \bmod p$; set $w_1 := g^s$ and $w_2 := g^t$. Update $\mathcal{L}_{\mathcal{H}} := \mathcal{L}_{\mathcal{H}} \cup (x, s, t, w_1, w_2)$ and output (w_1, w_2). ;

– otherwise if $x = x^\star$ choose $s^\star, t^\star \leftarrow_\$ \mathbb{Z}_p$; set $w_1^\star := g^{s^\star}$ and $w_2^\star := g^{t^\star}$. Update $\mathcal{L}_\mathcal{H} := \mathcal{L}_\mathcal{H} \cup (x^\star, s^\star, t^\star, w_1^\star, w_2^\star)$ and output $(w_1^\star, w_2^\star)$.

Give random oracle access to $\mathcal{A}$.

3. Choose a couple of $(t-1)$-degree random polynomials f_1, f_2. Define $u_i := f_1(i)$ and $v_i := f_2(i)$ for $i \in [n] \cup \{0\}$. Initialize states for each party $i \in [n]$ as $\mathsf{st}_i := \{u_i, v_i\}$. Initialize $C := \emptyset$. At any point on receiving a new set of corrupt parties $\tilde{C}$ from $\mathcal{A}$, send the corresponding secret states $\{\mathsf{st}_i\}_{i \in \tilde{C}}$ to $\mathcal{A}$ and update $C := C \cup \tilde{C}$. For simplicity we assume that the final set C is equal to $\{1, \ldots, \ell\}$, which is without loss of generality because we will not be using this information in the proof (otherwise handling the adaptive corruption would have been trivial).

4. On an evaluation query $(\texttt{Eval}, x, i)$ for an honest i, return $h_i := w_1^{u_i} w_2^{v_i}$ where $\mathcal{H}(x) = (w_1, w_2) \in G^2$; update $\mathsf{st}_i := \mathsf{st}_i \cup \{x\}$. Build a list L_x if this is the first evaluation query on x, or update exiting L_x by appending i to it.

5. On the challenge query $(\texttt{Challenge}, x, S^\star, g_1^\star, \ldots, g_m^\star)$ for $m \leq \ell$ if the guess is wrong, that is $x \neq x^\star$ then output a random guess, otherwise: (without loss of generality assume that $S^\star \cap C = [m]$):

   1. Set $g_i^\star := {w^\star}_1^{u_i} {w^\star}_2^{v_i}$ for $i \in S^\star \setminus C$.
   2. Depending on b do as follows:
      1. If $b = 0$ then compute $z^\star := \prod_{i \in S^\star} g_i^{\star \lambda_{0,i,S}}$.
      2. Else, choose a random $z^\star \leftarrow_\$ G$.
   3. Send $z^\star$ to $\mathcal{A}$.

6. Continue answering evaluation queries as before.

7. Receive a guess b' from $\mathcal{A}$; then do as follows:

   – if the total number of corrupt parties, $|C| \geq t$ then output 0 and stop;
   – if the challenge $x^\star$ has been queried for evaluation for at least $g := t - |C|$ honest parties, that is if $L_{x^\star} \cap ([n] \setminus C) \geq g$ then output 0 and stop;
   – otherwise output b'.

We prove the following claim:

Claim B.1 *Assuming DDH is hard in group G, we have that $\mathsf{PseudoRand}_\mathcal{A}(b) \approx_{\mathsf{comp}} \mathsf{Hyb}^1_\mathcal{A}(b)$ for any PPT adversary $\mathcal{A}$ and any $b \in \{0,1\}$.*

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that distinguishes between $\mathsf{PseudoRand}_\mathcal{A}(b)$ and $\mathsf{Hyb}^1_\mathcal{A}(b)$ for some $b \in \{0,1\}$ with non-negligible probability. We construct a PPT reduction $\mathcal{B}$, which breaks DDH given oracle access to $\mathcal{A}$ with non-negligible probability too (as long as $q_\mathcal{H}$ is a polynomial). Given a DDH challenge $g^\alpha, g^\beta, g^\gamma$ where γ is either is equal to $\alpha\beta \bmod p$ or uniform random in $\mathbb{Z}_p$ $\mathcal{B}$ simulates the random oracle queries for the evaluation phase $x \neq x^\star$ as follows:

Initialize $\mathcal{L}_\mathcal{H} := \emptyset$. For random oracle call with input x:

1. If there exists a tuple $(x, \star, \star, w_1, w_2) \in \mathcal{L}_H$, output (w_1, w_2).

2. Otherwise set $w_1 := g^{\mu} \cdot g^{\alpha\sigma}$ for uniformly chosen $\mu, \sigma \in \mathbb{Z}_p^2$ and $w_2 := g^{\mu\beta} \cdot g^{\gamma\sigma}$. Update $\mathcal{L}_{\mathcal{H}} := \mathcal{L}_{\mathcal{H}} \cup (x, \star, \star, w_1, w_2)$ and output (w_1, w_2)

Now, if $\gamma = \alpha\beta$, then implicitly (in the exponent) the challenger sets $s := \mu + \sigma\alpha$ and $t := \beta(\mu + \sigma\alpha)$ for unknown values $s := \log_g(w_1)$ and $t := \log_g(w_2)$. By change of variable we can write $\omega = \beta$ and $t := s\omega$. So the distribution of $\mathcal{A}$'s view in this case is identical to the view of $\mathcal{A}$ in $\mathsf{Hyb}^1_{\mathcal{A}} b$. On the other hand, when γ is uniform random in $\mathbb{Z}_p$, then the unknown value $t := \mu\beta + \sigma\gamma$ which is uniform random in $\mathbb{Z}_p$. So, in this case both s and t are uniform random and hence $\mathcal{A}$'s view is identical to the its view in $\mathsf{PseudoRand}_{\mathcal{A}}(b)$. Hence we can conclude that, if $\mathcal{A}$ distinguishes the above experiments with non-negligible probability, $\mathcal{B}$ will break DDH with non-negligible probability too. This concludes the proof of this claim. ■ In the next hybrid $\mathsf{Hyb}^2_{\mathcal{A}}(b)$, we do not make

any change from $\mathsf{Hyb}^1_{\mathcal{A}}(b)$ except that all non-challenge evaluation queries are responded with an independent key $k \leftarrow_{\$} \mathbb{Z}_p$ sampled uniformly at random, whereas the corruption query for party i are answered using randomly sampled u_i, v_i subject to $u_i + \omega v_i = k_i$ where k_i is the i-th Shamir's share of k. In particular, for a non-challenge evaluation query $\mathsf{Eval}(x, i)$, the challenger returns $g^{s_i k}$ where $\mathcal{H}(x) = (g^{s_i}, g^{\omega s_i})$. We describe this hybrid formally below:

$\underline{\mathsf{Hyb}^2_{\mathcal{A}}(b) :}$

1. Give the public parameters $pp := (p, g, G)$ (G is a cyclic group of order p and g is a generator of G) to $\mathcal{A}$. Suppose $\mathcal{A}$ makes $q_{\mathcal{H}}$ distinct random oracle queries in total.

2. Guess $j \leftarrow_{\$} [q_{\mathcal{H}}]$ and let the j-th query be on $x^\star$. Choose a random value $\omega \leftarrow_{\$} \mathbb{Z}_p$. Then program the random oracle $\mathcal{H}$ as follows: Initialize $\mathcal{L}_{\mathcal{H}} := \emptyset$. For random oracle call on x do as follows:

   1. If there exists a tuple $(x, s, t, w_1, w_2) \in \mathcal{L}_H$, output (w_1, w_2).
   2. Otherwise,
      - if $x \neq x^\star$: choose $s \leftarrow_{\$} \mathbb{Z}_p$ and define $t := s\omega \bmod p$; set $w_1 := g^s$ and $w_2 := g^t$. Update $\mathcal{L}_{\mathcal{H}} := \mathcal{L}_{\mathcal{H}} \cup (x, s, t, w_1, w_2)$ and output (w_1, w_2). ;
      - otherwise if $x = x^\star$ choose $s^\star, t^\star \leftarrow_{\$} \mathbb{Z}_p$; set $w_1^\star := g^{s^\star}$ and $w_2^\star := g^{t^\star}$. Update $\mathcal{L}_{\mathcal{H}} := \mathcal{L}_{\mathcal{H}} \cup (x^\star, s^\star, t^\star, w_1^\star, w_2^\star)$ and output $(w_1^\star, w_2^\star)$.

   Give random oracle access to $\mathcal{A}$.

3. Choose a $(t-1)$-degree random polynomials f. Define $k_i := f(i)$ for $i \in [n] \cup \{0\}$. Initialize states for each party $i \in [n]$ as $\mathsf{st}_i := \{u_i, v_i\}$ where u_i, v_i are random in $\mathbb{Z}_p$ subject to $k_i = u_i + \omega v_i$. Initialize $C := \emptyset$. At any point on receiving a new set of corrupt parties $\tilde{C}$ from $\mathcal{A}$, send the corresponding secret states $\{\mathsf{st}_i\}_{i \in \tilde{C}}$ to $\mathcal{A}$ and update $C := C \cup \tilde{C}$. For simplicity we assume that the final set C is equal to $\{1, \ldots, \ell\}$, which is without loss of generality because we will not be using this information in the proof (otherwise handling the adaptive corruption would have been trivial).

4. On a non-challenge evaluation query $(\mathtt{Eval}, x, i)$ for an honest i, return $h_i := g^{k_i s}$ where $(x, s, t, w_1, w_2) \in \mathcal{L}_{\mathcal{H}}$; on a evaluation query on the challenge, $(\mathtt{Eval}, x^\star, i)$ for an honest i, return $g^{u_i s^\star} g^{v_i t^\star}$. update $\mathsf{st}_i := \mathsf{st}_i \cup \{x\}$. Build a list L_x if this is the first evaluation query on x, or update exiting L_x by appending i to it.

5. On the challenge query $(\texttt{Challenge}, x, S^\star, g_1^\star, \ldots, g_u^\star)$ for $u \leq \ell$ if the guess is wrong, that is $x \neq x^\star$ then output a random guess, otherwise: (without loss of generality assume that $S^\star \cap C = [u]$):

   1. Sample $u^\star{}_i, v^\star{}_i \leftarrow_\$ \mathbb{Z}_p^2$. Set $g_i^\star := w^{\star u^\star{}_i}_1 w^{\star v^\star{}_i}_2$ for $i \in S^\star \setminus C$.
   2. Depending on b do as follows:
      1. If $b = 0$ then compute $z^\star := \prod_{i \in S^\star} g_i^{\star \lambda_{0,i,S}}$.
      2. Else, choose a random $z^\star \leftarrow_\$ G$.
   3. Send $z^\star$ to $\mathcal{A}$.

6. Continue answering evaluation queries as before.

7. Receive a guess b' from $\mathcal{A}$; then do as follows:

   – if the total number of corrupt parties, $|C| \geq t$ then output 0 and stop;
   – if the challenge $x^\star$ has been queried for evaluation for at least $g := t - |C|$ honest parties, that is if $L_{x^\star} \cap ([n] \setminus C) \geq g$ then output 0 and stop;
   – otherwise output b'.

We prove the following claim:

Claim B.2 *We have that* $\mathsf{Hyb}^1_{\mathcal{A}}(b) \approx_{\mathsf{stat}} \mathsf{Hyb}^2_{\mathcal{A}}(b)$ *for any (possibly unbounded) adversary* $\mathcal{A}$ *any any* $b \in \{0, 1\}$

Proof. Note that, this statement is information theoretic and hence we assume that the adversary here may be unbounded. First we notice that, in both the hybrids an unbounded adversary learns values $\{s_1, s_2, \ldots\}$, ω from the random oracle responses. Furthermore, it learns at most $t-1$ pairs $\{u_i, v_i\}_{i \in C}$ (u_i, v_i are independent and random) where $|C| < t$, given which (u, v) remains statistically hidden. Now, the only difference comes in the evaluation queries. In Hyb_1, the adversary is answered with the key $\{u + \omega v\}$ whereas in Hyb_2 it is answered with a uniform random k. Now, we note that conditioned on the given values, namely $\{s_1, s_2, \ldots\}$, ω, $u + \omega v$ is uniformly random for randomly chosen u, v – another way to see this will be as a universal hash function where ω is the input and (u, v) are uniform keys (here we rely on statistical property of secret-sharing, in that given at most $t-1$ values u_i, v_i, (u, v) remains statistically hidden). Hence the two distributions are statistically close. ■

Now, we note that, in the hybrid $\mathsf{Hyb}^2_{\mathcal{A}}(b)$, the value $z^\star$ is uniformly random irrespective of b. This follows from the fact that each pair $u^\star{}_i, v^\star{}_i$ are uniformly chosen. Therefore we can conclude that, $\mathsf{Hyb}^2_{\mathcal{A}}(0) \approx_{\mathsf{stat}} \mathsf{Hyb}^2_{\mathcal{A}}(1)$. This concludes the proof of the theorem.