Threshold Encryption with Internally Motivated Corruptions

Jan Bormet*[1] Hussien Othman [1] Benedikt Wagner [2]

[1] TU Darmstadt
jan.bormet@tu-darmstadt.de, hussien.othman@gmail.com
[2] Ethereum Foundation
benedikt.wagner@ethereum.org

Abstract

In recent years, threshold encryption has gained a lot of interest, particularly due to its potential use in encrypted mempools in blockchains. Standard security models allow the adversary to corrupt parties either statically (i.e., fixed at the onset of the game) or adaptively (i.e., via an oracle one-by-one, depending on keys and ciphertexts).

In this work, we observe that neither of these models captures the case in which a party decides to become corrupted *based on secret information*. For instance, in an encrypted mempool application with randomly rotating committees, an adversary may set up a smart contract that pays parties who reveal their decryption share, and parties decide whether to claim it based on, say, whether they are on the next committee. Such corruptions are not fixed in advance, but they are also not chosen solely by an external adversary based on public information.

We initiate the formal study of such *internally motivated corruptions and partial decryptions*. We introduce a security framework in which each party's corruption behavior may depend on its local secret state. That is, on a corruption query, the adversary can submit a *motivation function* and all parties for which this motivation function outputs 1 (on their secret information) are corrupted. A similar internally motivated behavior is allowed for releasing partial decryptions. We then study threshold encryption under this stronger notion of security. In particular, we show:

- *Negative Results:* We show that for certain classes of motivation functions and number of queries, *no* threshold encryption scheme can satisfy security. We also show a concrete practical attack with internally motivated corruptions against a scheme that has been proven secure with standard corruptions.
- *Positive Results:* We give two efficient classes of constructions from the (Bilinear) Diffie-Hellman assumptions in the random oracle model. The first is secure when partial decryptions on the challenge ciphertext are internally motivated. The second additionally allows internally motivated corruptions.

*This work was done while the first author was doing an internship at the Ethereum Foundation.

Contents

1 Introduction **3**
- 1.1 Our Contribution . . . 4
- 1.2 Related Work and Discussion . . . 6

2 Preliminaries **9**
- 2.1 Interpolation and Shamir Secret Sharing . . . 9
- 2.2 Computational Assumptions . . . 9
- 2.3 Cryptographic Primitives . . . 10

3 Security Definition: Internally Motivated Corruptions **11**
- 3.1 Threshold Encryption . . . 12
- 3.2 Internally Motivated Corruptions . . . 12
- 3.3 Simple Observations . . . 15

4 Attacks **16**
- 4.1 Attack with Many Partial Decryption Queries . . . 16
- 4.2 Attack with Decrypting Motivation Function . . . 17
- 4.3 Attack on Committee-based Schemes à la HSW25 . . . 18

5 Constructions with Internally Motivated Partial Decryptions **19**
- 5.1 Overview . . . 19
- 5.2 Statistical Core Lemma . . . 22
- 5.3 Construction . . . 27
- 5.4 Constant Parameter Sizes from Pairings . . . 37
- 5.5 Avoiding Simulation-Extractability . . . 46

6 Construction with Adaptive Internally Motivated Corruptions **47**
- 6.1 Overview . . . 47
- 6.2 Second Statistical Core Lemma . . . 49
- 6.3 Construction . . . 51

7 Tradeoff Discussion **58**

1 Introduction

Threshold cryptography [37, 38, 63] enables a cryptographic secret to be shared among n parties such that they can later jointly perform the task of a single party holding that secret, without ever explicitly reconstructing it. In this way, threshold cryptography removes a single point of failure and reduces the need to place trust in any individual party. One example of threshold cryptography is threshold *encryption* [37, 38]. Such a scheme allows a single party to encrypt a message, but the decryption process is thresholdized: sufficiently many parties can work together to decrypt, while too few parties will learn nothing about the message. Threshold encryption has recently attracted renewed interest, driven both by an ongoing NIST standardization effort [22] and by its potential use in encrypted mempools for blockchains [55, 17, 29, 69, 58]. In an encrypted mempool, transactions are encrypted using threshold encryption before being broadcast to the network. They are decrypted by a threshold committee only after they have been included in a block. As a result, other parties cannot inspect transaction contents before inclusion, which can mitigate forms of maximal extractable value (MEV)[1], such as front-running and sandwich attacks [42]. Motivated by this application, we study a new security model for threshold encryption.

Static and Adaptive Corruptions. The security of a t-out-of-n cryptosystem should hold even if up to $t-1$ parties are corrupted and behave maliciously. Here, one typically distinguishes between *static* and *adaptive* corruptions.[2] For static corruptions, the adversary decides which parties to corrupt at the onset of the security experiment, even before seeing any key material. Adaptive corruptions, on the other hand, model a stronger adversary and so far have been considered the strongest corruption model. Here, the adversary may choose whom to corrupt over the course of the experiment, with each corruption decision depending on the information revealed so far, including public keys, ciphertexts, and the outcomes of previous corruptions. Formally, this is modeled by giving the adversary access to a corruption oracle that it can call up to $t-1$ times at any point during the game. We argue below that, despite their strength, adaptive corruptions are still insufficient to capture realistic attack scenarios arising in encrypted mempools.

Bribery and Selling Shares. Consider again the application of threshold encryption to encrypted mempools. Learning the contents of pending transactions before their inclusion can be a substantial financial advantage. In particular, it enables an attacker to identify and exploit profitable MEV opportunities before others do. Since the resulting gains can be significant, an attacker may have a strong economic incentive to obtain decryption shares before the prescribed decryption time. A natural attack strategy is therefore to *bribe*[3] members of the threshold committee. For example, an attacker could deploy a smart contract that rewards committee members for providing valid decryption shares.[4] If the offered bribe exceeds a committee member's expected cost of deviating from the protocol, that member may be willing to sell its share. By collecting sufficiently many such shares, the attacker can decrypt transactions prematurely and subsequently exploit the revealed information for MEV. At the same time, the potentially large profits available from MEV outweigh the cost of bribery, making bribery attacks a realistic threat. The canonical approach to deal with this would be calibrating the threshold t such that bribing $\geq t$ parties is too expensive compared to MEV gains. By doing so, the protocol designer implicitly interprets the security guarantees under $< t$ corruptions as being sufficient for this setting.

Adaptive Corruptions Don't Model This. However, we observe that standard adaptive corruptions do *not* capture this attack scenario. The reason is that, in a bribery attack, the decision of *whether* a party becomes corrupted is not made solely by the adversary. Instead, each party may independently decide whether to accept or reject the bribe, and this decision may depend on private information known only to that party. Consider, for example, a protocol that selects a fresh random subcommittee in each round, where committee membership remains hidden from all parties except the selected members

[1] Maximal/miner extractable value denotes the profit that can be obtained by reordering, inserting, or censoring transactions during block production.

[2] Intermediate notions have also been considered, such as the recent notion of one-shot adaptive corruptions [52].

[3] In Section 1.2, we discuss how our work relates to works on traitor tracing, which are also motivated by bribery attacks.

[4] The attacker would generally not want the submitted shares to become publicly available. In principle, this can be addressed using encryption together with zero-knowledge proofs.

themselves. Whether a party belongs to the upcoming subcommittee can substantially affect both the value of its decryption share to the attacker and the party's incentive to accept a bribe. A selected party may therefore accept an offer that an unselected party would reject, even though the attacker cannot distinguish between the two when making the offer.

This behavior is not represented by the standard adaptive corruption model. Namely, an adaptive adversary must make each corruption choice without first learning the target party's internal state. Consequently, it cannot express corruption decisions that depend on the parties' private information.

In this work, we formally model such stronger corruption scenarios and analyze which threshold schemes can achieve the resulting notions of security.

1.1 Our Contribution

In this work, we study a strong notion of corruptions (and partial decryptions), namely, *internally motivated corruptions and partial decryptions*. We show both negative results (i.e., attacks in this model) as well as constructions achieving this strong security property. The motivation for this stronger notion stems primarily from the aforementioned encrypted mempool application, as explained above.[5] However, we also view this new notion as being of theoretical interest in its own right, given that it strengthens the security notion of adaptive corruptions and high-threshold [9] chosen ciphertext security. Let us now summarize our results in more detail.

Our New Framework. We first introduce a general framework of security models that captures the setting described in the introduction, where an honest party's decision to become corrupted or release a partial decryption may depend on secret information available to that party but not yet known to the adversary.

To nevertheless give the adversary full control over this process, we allow it to specify *motivation functions* that model the decision-making behavior of honest parties. Concretely, when invoking the corruption oracle, the adversary submits a set $S \subseteq [n]$ and a function ψ. Each party $i \in S$ is corrupted if and only if $\psi(i, \mathsf{sk}_i) = 1$. Note that this covers adaptive corruptions as a special case ($|S| = 1$ and $\psi \equiv 1$). However, for more expressive motivation functions, the notion is stronger and finding constructions is more challenging. Intuitively, this is because the fact that a party remains uncorrupted may leak information about sk_i. The partial decryption oracle is modeled in a similar way, where the motivation function takes the secret key share and/or the partial decryption as input.

Our framework is parameterized by classes of admissible motivation functions, with a separate class for each oracle and each stage of the security game. Different choices of these classes yield different security notions, including existing and novel definitions. We refer to Section 3 for details.

Attacks. To explore the boundaries of our new notion and clarify the gap between it and existing security definitions, we present *three attacks* in Section 4. One is specific to a class of schemes, and two are generic:

1. The first attack is closely related to the *subcommittee setting* discussed in the introduction. We consider the committee-based silent-setup threshold encryption scheme of [52], as well as natural variants. At encryption time, the scheme samples a random subcommittee of size $M \ll t$ and encrypts to its members. Crucially, the subcommittee can be hidden, which allows the scheme to achieve a restricted form of adaptive security together with CCA1 security. We show that the scheme becomes insecure under internally motivated partial decryption queries on the challenge ciphertext. The attack is conceptually simple: the adversary submits a motivation function that causes precisely the M subcommittee members to release their partial decryptions. Determining whether a party belongs to the subcommittee requires access to its partial decryption, which is exactly why internally motivated queries are necessary for the attack.[6]

2. Beyond this scheme-specific result, we give *two generic attacks* showing that internally motivated security is unachievable without restricting the admissible class of motivation functions. Both

[5] We do not claim that our schemes are a perfect fit for the use in encrypted mempools. Instead, our paper introduces a new notion and they are (efficient) feasibility results for this notion.

[6] A similar attack is possible via internally motivated corruption queries.

attacks exploit the fact that motivation functions for partial decryption queries on non-challenge ciphertexts may depend on the parties' secret key shares. The outcomes of these queries can therefore leak partial information about those shares. In the first attack, the adversary issues many partial decryption queries and gradually reconstructs sufficiently many secret key shares bit by bit. The second attack achieves a similar goal using a single, more intricate query and $t-1$ standard static corruptions.

The key takeaway is that motivation functions for partial decryption queries on non-challenge ciphertexts cannot be allowed to arbitrarily depend on the parties' secret key shares. In other words, if parties are motivated to leak partial decryptions based on their secret key shares regularly, then security cannot be guaranteed.

Constructions. Motivated by the preceding attacks, we ask whether security remains achievable when internally motivated queries are allowed, provided that motivation functions for partial decryption queries on non-challenge ciphertexts do not depend on the secret key shares.

We answer this question affirmatively by presenting constructions over cyclic groups of prime order, secure in the random oracle model under the standard decisional Diffie-Hellman assumption, and summarized in Table 1:

1. **Internally Motivated Partial Decryptions.** Our first construction is secure against static corruptions while allowing internally motivated partial decryption queries, including such queries on the challenge ciphertext.[7] To avoid the generic attacks described above, the motivation functions may not depend on secret key shares; they may, however, depend on the partial decryptions themselves. We also present a variant with smaller ciphertexts and public keys, based on pairings and the Bilinear Decisional Diffie-Hellman assumption.

2. **Corruptions and Less Restrictions.** Our second construction additionally supports internally motivated adaptive corruptions, while continuing to allow internally motivated partial decryption queries. Here, only motivation functions for partial decryption queries on non-challenge ciphertexts are restricted so as to preclude the generic attacks. For all other queries, the motivation functions may be *arbitrary* and, in particular, may depend on the parties' secret key shares. As a result, our second construction comes very close to the impossibility implied by our generic attacks, which exploit non-challenge partial decryption oracle access with motivation functions that depend on secret key shares.[8]

While our constructions resemble some existing schemes [33, 9], establishing their security against internally motivated queries requires substantial *new proof techniques*. In particular, in our analysis, we develop two statistical core lemmas that capture the leakage induced by internally motivated queries.

These core lemmas contribute an additive statistical term $\binom{n}{t-1}/p^{uv}$ to the security bound. Here, p is the group order, and u and v allow us to compensate for the binomial in the bound. Roughly, $u \cdot v \approx O(1 + n/\lambda)$ is sufficient for standard choices of p. They also induce a *tradeoff* for ciphertext and public key size: public keys contain exactly v group elements in both schemes; ciphertexts contain $\Theta(u)$ elements in the first scheme and $\Theta(u^2)$ group elements in the second. In the pairing-based scheme, ciphertext overhead and key sizes are constant, and only computation and partial decryption size scales (linearly) with $\ell := u \cdot v$. The hidden constants are small, resulting in practical ciphertext and public key sizes for any reasonable choice of n, as we show in Section 7. For details of the schemes, including detailed technical overviews, we refer to Sections 5 and 6.

Limitation and Future Work. While our schemes achieve ciphertext and public key sizes that are sublinear in n, their main drawback is that the partial decryptions have size $O(1 + n/\lambda)$. One might instead consider the "naive" threshold encryption scheme that encrypts a separate secret share of the message to each party. Although this yields small partial decryptions, it comes at the cost of substantially

[7] Our model allows an arbitrary number of corruption queries and partial decryption queries on the challenge ciphertext, as long as the total number of resulting corrupted parties and partial decryptions learned does not exceed $t-1$.

[8] One may be able to circumvent the generic attacks in an even tighter manner by allowing motivation functions with secret keys on non-challenge partial decryptions, but restricting how they can depend on the secret key, or imposing joint bounds on corruptions and number of queries with such functions. We leave this as an open question.

Table 1: Overview of our positive results. We show which type of corruption and partial decryption queries on non-challenge ciphertexts and challenge ciphertexts are allowed, where "IM" stands for internally motivated queries with arbitrary motivation functions, and "IM-nosk" stands for internally motivated queries with motivation functions whose output does not depend on the secret key share but only on the partial decryption. Parameters u, v are tradeoff parameters such that $u \cdot v \approx O(1 + n/\lambda)$. A more detailed overview is in Table 2.

	Corruptions	Part. Dec. (non-chal.)	Part. Dec. (chal.)	Ciphertext	Pub. Key
Sec. 5	Static	IM-nosk	IM-nosk	$O(u)$	$O(v)$
Sec. 5 (pairings)	Static	IM-nosk	IM-nosk	$O(1)$	$O(1)$
Sec. 6	IM	IM-nosk	IM	$O(u^2)$	$O(v)$

larger public keys and ciphertexts, both of which scale linearly in n. Even more importantly, it is unclear whether this naive scheme is secure against internally motivated corruptions, showing non-triviality of our results. More generally, we leave as future work the analysis of existing threshold encryption schemes under internally motivated corruptions, with sublinear partial decryption size in particular.

1.2 Related Work and Discussion

We address a gap in the security definitions for threshold encryption: while existing definitions guarantee that it is computationally hard to find a small coalition that can learn non-trivial information about the plaintext, our results show that they do not guarantee that every small coalition is unable to learn such information. This distinction becomes important in collusion and bribery attacks, where parties decide if they become corrupted or not by themselves, and may provide the adversary with information that helps it identify a small coalition capable of breaking security. We take a step towards addressing this gap by introducing the *internally motivated corruptions* (IMC) model and by developing techniques for proving security in this model. We next discuss related works and failed directions for proving security in our model.

Complexity Leveraging. Via *complexity leveraging*, security in the IMC model may be reduced to security against static corruptions. Roughly, the reduction may guess the subset of parties for which the motivation functions evaluate to 1. Since there are at most $\binom{n}{t-1}$ possible subsets of size at most $t-1$, the reduction incurs a loss of a factor of $\binom{n}{t-1}$. Consequently, to retain λ bits of security, the underlying hardness assumption, e.g., DDH, must provide approximately $\lambda + \log_2 \binom{n}{t-1}$ bits of security, which implies the use of non-standard groups. Our challenge, therefore, is to develop security proofs that avoid the need for complexity leveraging. Indeed, the constructions that we provide avoid a binomial loss related to the computational assumptions. They still have a binomial loss in one of the statistical terms, but this can be compensated via appropriate tradeoffs between ciphertext and key size, as we show. Importantly, *standard* groups for DDH can be used.

Security Models for Threshold Encryption. Brzuska et al. [23] present a systematic study of security models for threshold encryption. Our model is *not covered* by their framework. The main aspect that they distinguish is between two types of partial decryption queries: *Type I*, which allows partial decryptions of non-challenge ciphertexts, and *Type II*, which allows partial decryptions of the challenge ciphertext, provided that fewer than t decryption shares are revealed. They show that security against Type I queries does not imply security against Type II queries, via a construction in which the encryptor selects a *public* subcommittee and encrypts only to that subcommittee. They argue that it is secure (under static corruptions) against Type I queries but becomes insecure once Type II queries are allowed. In our attacks, we also consider a subcommittee-based construction, but one in which the decryption subcommittee remains hidden from the adversary, as proposed in [52]. Notably, [23] observes that simulation-based security can make reductions involving Type II queries more convenient. Very briefly, the simulator learns the plaintext of a challenge ciphertext only once the adversary obtains enough partial decryptions and key shares to reach the threshold t. However, this does not seem to

address internally motivated corruptions, since the simulator would need to predict the output of the motivation function. Boneh et al. [9] present threshold encryption constructions that achieve security against Type II partial decryption queries, as formalized by Brzuska et al. [23]. Their main motivation is *context-dependent* threshold decryption, where partial decryptions depend on a context and can only be combined if they have been computed for the same context. Our security notion captures attacks that are orthogonal to those addressed by context-dependent threshold decryption. Boneh et al. call security against Type II queries *high-threshold security*. In our framework of security models, both Type I and Type II can be covered by defining appropriate motivation function classes.

Leakage-Resilient Secret Sharing. A recent line of work [7, 59, 60, 56, 53] has studied the security of Shamir's secret sharing scheme against *local leakage attacks*. In this model, the adversary specifies a leakage function f_i for each party and learns $f_i(s_i)$, where s_i denotes the i-th party's share of the secret s. Each leakage function is restricted to output only a small number of bits. The strongest known positive result establishes security when the security threshold satisfies $t \geq 0.668n$. As Shamir's secret sharing scheme is a fundamental building block of threshold encryption and one could view the output of motivation functions as local leakage, this line of work seems closely related to our setting. Unfortunately, a formal connection between these two models remains unclear, as we argue now.

A natural idea would be to establish security in our model by reducing it to the leakage resilience of Shamir's scheme, where the reduction encodes the motivation function given by the adversary as a leakage function. Specifically, the reduction would forward the adversary's motivation functions to the challenger in the Shamir leakage-resilience game. From the resulting leakage bits, it could determine the set T of parties for which the motivation function evaluates to 1. In our model, however, the reduction must additionally provide the adversary with the partial decryptions, or secret shares, of all parties in T. To do so, it would need to obtain the corresponding secret shares by corrupting the parties in T in the Shamir security game. Unfortunately, existing leakage-resilience proofs for Shamir's scheme only allow the adversary to choose the corrupted parties *before* observing the leakage. In contrast, in our setting, the set T is determined by the leakage itself (i.e., the output of the motivation function). Thus, the existing proofs do not yield the desired reduction. Moreover, even if the existing proofs could be extended to support adaptive corruptions, the underlying security model still requires the adversary to commit to all leakage functions before observing any leakage. In contrast, our model allows the adversary to choose each leakage function adaptively, based on the leakage obtained from previous queries. Finally, the known analyses only establish security for relatively high reconstruction thresholds, namely $t \geq 0.668n$. In contrast, threshold encryption is typically instantiated with lower thresholds, such as $t = n/2$ or $t = n/3$.

On the other side, there are known attacks on Shamir's scheme when allowing leaking a few bits per share [59, 31, 62, 20]. One may wonder if such attacks can be used to break IMC security of threshold encryption. This does not seem to be the case generically: in the secret sharing setting, to break security, one needs to distinguish between sharings of two different secrets. In threshold encryption, however, the shared secret is the decryption key rather than the challenge message. Thus, the ability to distinguish between sharings of two secret keys does *not* immediately imply the ability to distinguish between encryptions.

We leave establishing a formal connection between our model and leakage-resilient secret sharing for future work.

Leakage-Resilient Public Key Encryption (LR-PKE). A line of work studies public key encryption secure under secret key leakage [61, 3, 4, 39, 21, 57, 51, 26, 27, 72]. Unlike our internal motivation notion, LR-PKE is mainly motivated by physical side-channels, such as timing or power analysis attacks, through which an attacker learns partial information about the secret key. It is tempting, however, to try to construct threshold encryption secure against internally motivated corruptions from LR-PKE in a naive manner: instantiate n independent LR-PKE instances, i.e., one per party, and to encrypt, (t, n)-secret-share the message m, encrypting each share m_i under the i-th party's public key pk_i.[9] The hope is that a reduction can simulate motivation functions via the leakage oracle provided by the LR-PKE security game. However, this approach is not only inefficient ($O(n)$ public key and ciphertext size), but also does not seem sufficient to achieve our notion. First, early works on LR-PKE [61, 3, 4, 39, 21, 57]

[9]This simple scheme, as described, is vulnerable against mix-and-match attacks, but can be made CCA2 secure using standard techniques.

only allow leakage oracle access in the indistinguishability game *before* the adversary sees the challenge ciphertext. This restriction is inherent, as an after-the-fact leakage query could just decrypt the challenge ciphertext and directly reveal, with a single bit of leakage, which message was encrypted. This is problematic because we want to allow adversaries to corrupt parties through motivation functions even after seeing the challenge ciphertext.

Later work [51, 26, 27, 72] avoids this issue and permits after-the-fact leakage by weakening the guarantee. Rather than full indistinguishability, they give an entropic guarantee that if a message starts with high entropy, leaking a bounded number of bits after-the-fact still leaves it with correspondingly high unpredictability. This guarantee, however, is vacuous in our setting: once $t-1$ other parties are corrupted, the known shares together with m_0 and m_1 leave only two candidate sharing polynomials[10] and thus only two candidate shares of m_b as encrypted messages in the target instance. As a result, there is no high-entropy message to protect, and a single bit of after-the-fact leakage suffices to recover it. Building on this entropic notion, these works do achieve indistinguishability, but only in the *split-state* leakage model, which restricts each leakage query to only access a distinct part of the secret key at a time. While this may be well-motivated in the context of side-channel leakage, we find it unrealistic in our setting to assume that an internally motivated party would access only part of its secret key share. As a result, LR-PKE does not appear to yield our notion directly, and new techniques are needed.

Zhang et al. [71] consider leakage-resilient *threshold* encryption, allowing bounded leakage on individual key shares. Their notion, however, is tailored to side-channel leakage and comes with several restrictions: They do not consider partial decryption oracles (neither on adversarial nor on the challenge ciphertext), and corruptions are static and fixed before the adversary can obtain any leakage. This makes the notion insufficient to model internally motivated corruptions, since whether a party is corrupted or issues a partial decryption cannot depend on its secret state.

Threshold Traitor Tracing. A recent line of work [13, 8, 25, 18, 24, 19, 16, 14] studies the problem of collusion between committee members and an adversary. In these works, the adversary may bribe committee members to assist in decrypting ciphertexts, and the goal is to identify traitors, thereby disincentivizing collusion. While the motivation is closely related to ours, our perspective is fundamentally different. Threshold traitor tracing seeks to *disincentivize* collusion by enabling the identification and punishment of colluding committee members, referred to as *traitors*. In contrast, our goal is to make such attacks *impossible* for small coalitions, i.e., those of size smaller than t, when each party decides locally and adaptively whether to participate in the attack. This distinction is important because, as discussed above, existing security definitions do *not* rule out the possibility that some coalitions of fewer than t members can decrypt. One may study combinations of our model with threshold traitor tracing in the future.

Variants of Threshold Encryption. Motivated by applications such as encrypted mempools [5, 40], recent works have explored some variants of threshold encryption, such as silent threshold encryption [48, 70, 52], batched threshold encryption [29, 17, 30, 2, 15, 11, 44, 50, 12, 1, 64], and context-dependent threshold encryption [9]. However, all of these constructions are proven secure under standard static or adaptive corruptions, which do not capture internally motivated corruptions.

Strong Adaptive Models. To the best of our knowledge, this is the first work to study a notion stronger than adaptive corruption in the setting of threshold cryptography. Stronger adversarial models have previously been considered in the context of collective coin flipping. In particular, Goldwasser et al. [49] introduce a stronger adversary that can observe the messages honest parties intend to send in the round before deciding whom to corrupt in that round. This captures the concern that the adversary can stop or alter messages sent by the parties depending on the message content. Using game-theoretic techniques, they show how to transform a *symmetric* one-round coin-flipping protocol that is secure against up to t adaptive corruptions into a protocol that is secure against strong adaptive adversaries corrupting up to $t/2$ parties. Our setting is different and captures a different strengthening of corruption, in that we strengthen the way corrupted parties are selected. Clearly, this requires different techniques.

[10] Each one is given by completing the shares with the evaluation point $(0, m_b)$ and interpolating.

2 Preliminaries

Vectors are row vectors by default. We denote the security parameter by λ, and write $\mathsf{negl}(\lambda)$ (resp. $\mathsf{poly}(\lambda)$) for functions that are negligible (resp. polynomial) in λ. We denote the number of parties by n. When describing asymptotic properties of our constructions, we ignore factors other than n in the O-notation. For integers $a, b \in \mathbb{N}$, we denote the set $\{1, \ldots, b\}$ by $[b]$ and the set $\{a, \ldots, b\}$ by $[a, b]$. We write $y \leftarrow A(x; r)$ to execute the algorithm A on input x with randomness r and assign the result to y. If A is executed with uniform randomness, we abbreviate $y \xleftarrow{\$} A(x)$ to indicate that A is probabilistic. For a set of indices S and a function f with domain S, we denote with $(f(i))_{i \in S}$ the indexed set $\{(i, f(i)) \mid i \in S\}$. Whenever such a set is interpreted as a vector, we use the canonical ordering of S, which will be clear from context. For example, $(\mathsf{sk}_i)_{i \in [n]}$ refers to the tuple of secret key shares $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n)$. We denote the bitwise XOR by $\oplus$. When we say that an adversary makes q queries to an oracle, we implicitly also include queries made by the game in which the adversary runs in this bound.

2.1 Interpolation and Shamir Secret Sharing

Denote with $\lambda_{i,S}^{(x)} = \prod_{j \in S \setminus \{i\}} (x - j)/(i - j)$ the Lagrange coefficient for interpolating a polynomial at position x using index i among the set of indices S. Then, given points (i, y_i) for $i \in S$, the polynomial $p(x) = \sum_{i \in S} \lambda_{i,S}^{(x)} \cdot y_i$ is of degree $|S| - 1$, and $p(i) = y_i$ for all $i \in S$. Henceforth, we denote the set of degree $< t$ polynomials over a prime order field $\mathbb{Z}_p$ as $\mathbb{Z}_p[X]_{\deg < t}$. We denote by $(x_i)_{i \in [n]} \xleftarrow{\$} \mathsf{Share}(x, n, t)$ a t-of-n Shamir secret sharing [66] of a value $x \in \mathbb{Z}_p$. Namely, the algorithm Share samples a random polynomial $f \in \mathbb{Z}_p[X]_{\deg < t}$ with $f(0) = x$ and outputs the evaluations $x_i = f(i), i \in [n]$. Any t of these suffice to reconstruct x via interpolation, whereas any $t - 1$ of these reveal no information about x.

2.2 Computational Assumptions

(Pairing) Groups. Our constructions rely on groups and pairings. A group ensemble $E = (\mathbb{G}, g, p)$ consists of a cyclic group $\mathbb{G}$ with generator g of prime order p, and is generated by a group generation algorithm $E \xleftarrow{\$} \mathcal{G}(1^\lambda)$. A bilinear pairing ensemble $E = (\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, p)$ consists of cyclic groups $\mathbb{G}_1$ with generator g_1, $\mathbb{G}_2$ with generator g_2, and $\mathbb{G}_T$ of prime order p with a pairing operation $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies bilinearity and non-degeneracy. It is generated by a group generation algorithm $E \xleftarrow{\$} \mathcal{G}(1^\lambda)$. For $A, B \in \mathbb{G}$ (or $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively), excluding the identity for A, we write $\mathsf{DLog}_A(B)$ to denote the unique element $x \in \mathbb{Z}_p$ such that $A^x = B$.

Assumption 1 (Decisional Diffie-Hellman (DDH))**.** Let $\mathcal{G}$ be a group generation algorithm. The Decisional Diffie-Hellman problem (DDH) is hard for $\mathcal{G}$ if for all PPT adversaries $\mathcal{A}$, the following function is negligible:

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathcal{G}}(1^\lambda) \coloneqq \left| \Pr\left[\mathcal{A}(1^\lambda, E, g^\alpha, g^\beta, g^{\alpha\beta}) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, E, g^\alpha, g^\beta, g^z) = 1\right] \right|,$$

where $E \coloneqq (\mathbb{G}, g, p) \xleftarrow{\$} \mathcal{G}(1^\lambda)$ and $\alpha, \beta, z \xleftarrow{\$} \mathbb{Z}_p$.

Assumption 2 ((k,ℓ)-Multi Decisional Diffie-Hellman ((k,ℓ)-mDDH / (k,ℓ)-mXDH))**.** Let $\mathcal{G}$ be a group generation algorithm. For $k, \ell \in \mathbb{N}$, the (k,ℓ)-Multi Decisional Diffie-Hellman problem ((k,ℓ)-mDDH) is hard for $\mathcal{G}$ if for all PPT adversaries $\mathcal{A}$, the following function is negligible:

$$\begin{aligned}\mathsf{Adv}^{(k,\ell)\text{-}\mathsf{mDDH}}_{\mathcal{A},\mathcal{G}}(1^\lambda) \coloneqq \Big| &\Pr\left[\mathcal{A}(1^\lambda, E, (g^{\alpha_i})_{i \in [k]}, (g^{\beta_j})_{j \in [\ell]}, (g^{\alpha_i \beta_j})_{i \in [k], j \in [\ell]}) = 1\right] \\ &- \Pr\left[\mathcal{A}(1^\lambda, E, (g^{\alpha_i})_{i \in [k]}, (g^{\beta_j})_{j \in [\ell]}, (g^{z_{i,j}})_{i \in [k], j \in [\ell]}) = 1\right] \Big|,\end{aligned}$$

where $E \coloneqq (\mathbb{G}, g, p) \xleftarrow{\$} \mathcal{G}(1^\lambda)$ and $(\alpha_i)_{i \in [k]} \xleftarrow{\$} \mathbb{Z}_p^k, (\beta_j)_{j \in [\ell]} \xleftarrow{\$} \mathbb{Z}_p^\ell$, and $(z_{i,j})_{i \in [k], j \in [\ell]} \xleftarrow{\$} \mathbb{Z}_p^{k \times \ell}$.

If $\mathcal{G}$ generates a pairing ensemble $E \coloneqq (\mathbb{G}, g, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$, we refer to it as the (k,ℓ)-mXDH assumption.

Observe that (k,ℓ)-mDDH and (k,ℓ)-mXDH follow from DDH and XDH, respectively, by a standard hybrid argument with a loss of $k \cdot \ell$.[11] In particular, $(1,1)$-mDDH is identical to DDH and $(1,1)$-mXDH is also called XDH.

[11] We can get a tighter bound via random self-reducibility of DDH, but we are not interested in tightness in this work.

Assumption 3 (Bilinear Decisional Diffie-Hellman (BDDH))**.** Let $\mathcal{G}$ be a group generation algorithm. The Bilinear Decisional Diffie-Hellman problem (BDDH) is hard for $\mathcal{G}$ if for all PPT adversaries $\mathcal{A}$, the following function is negligible:

$$\mathsf{Adv}^{\mathsf{BDDH}}_{\mathcal{A},\mathcal{G}}(1^\lambda) \coloneqq \left|\Pr\left[\mathcal{A}(1^\lambda, E, \mathsf{view}, e(g_1,g_2)^{\alpha\beta\gamma}) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, E, \mathsf{view}, e(g_1,g_2)^{z}) = 1\right]\right|,$$

where $E \coloneqq (\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$, $\alpha, \beta, \gamma, z \xleftarrow{\$} \mathbb{Z}_p$, and $\mathsf{view} \leftarrow \left(g_1^\alpha, g_1^\beta, g_2^\beta, g_2^\gamma\right)$.

2.3 Cryptographic Primitives

Definition 2.1 (NIZK-(PoK)). Let R be an NP-relation of statement-witness pairs $(\chi,\omega) \in \mathsf{R}$. Let $\mathcal{L}_\mathsf{R} = \{\chi \colon \exists\omega \text{ s.t. } (\chi,\omega) \in \mathsf{R}\}$ denote the language of true statements with respect to R. A non-interactive zero-knowledge proof system (of knowledge) PS_R for R is a tuple of PPT algorithms $\mathsf{PS}_\mathsf{R} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$. Given the security parameter λ, the Setup algorithm outputs public parameters pp, which are implicit input to all subsequent algorithms.
- $\pi \xleftarrow{\$} \mathsf{Prove}(\chi,\omega)$. For a statement-witness pair (χ,ω), the Prove algorithm returns a proof π.
- $1/0 \xleftarrow{\$} \mathsf{Verify}(\chi,\pi)$. Given a statement χ and a proof π, the Verify algorithm outputs 1, iff π is valid for χ.

A proof system must satisfy *correctness*: For all $\lambda \in \mathbb{N}$, all $(\chi,\omega) \in \mathsf{R}$, and all $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$ it holds that $\Pr[\mathsf{Verify}(\chi, \mathsf{Prove}(\chi,\omega)) = 1] = 1$.

To model proofs of knowledge, we require a NIZK-PoK to satisfy *simulation-extractability*, following the definition of simulation-extractability of [23], but strengthening it to require non-malleability of proofs similarly to [68, 35].

$\mathsf{SimExt}^{\mathsf{PS}_\mathsf{R},\mathsf{Sim}}_{\mathcal{A}}(1^\lambda)$

$b \xleftarrow{\$} \{0,1\}; Q \leftarrow \emptyset$
if $b = 0$
  $\mathsf{pp} \xleftarrow{\$} \mathsf{PS}_\mathsf{R}.\mathsf{Setup}(1^\lambda)$
else
  $(\mathsf{pp}, \mathsf{st}_\mathsf{Sim}) \xleftarrow{\$} \mathsf{Sim}.\mathsf{Setup}(1^\lambda)$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^{\mathsf{Prove}}_b, \mathcal{O}^{\mathsf{Verify}}_b}(1^\lambda, \mathsf{pp})$
return $b' \stackrel{?}{=} b$

Oracle $\mathcal{O}^{\mathsf{Prove}}_b(\chi,\omega)$

if $(\chi,\omega) \notin \mathsf{R}$ **then return** $\bot$
if $b = 0$
  $\pi \xleftarrow{\$} \mathsf{PS}_\mathsf{R}.\mathsf{Prove}(\chi,\omega)$
else
  $(\pi, \mathsf{st}_\mathsf{Sim}) \xleftarrow{\$} \mathsf{Sim}.\mathsf{Prove}(\chi, \mathsf{st}_\mathsf{Sim})$
  $Q \leftarrow Q \cup \{(\chi,\pi)\}$
return π

Oracle $\mathcal{O}^{\mathsf{Verify}}_b(\chi,\pi)$

if $b = 0$ **then**
  return $\mathsf{Verify}(\chi,\pi)$
else
  if $\mathsf{Verify}(\chi,\pi) = 0$ **then return** 0
  if $(\chi,\pi) \in Q$ **then return** 1
  $(\omega, \mathsf{st}_\mathsf{Sim}) \xleftarrow{\$} \mathsf{Sim}.\mathsf{Ext}(\chi,\pi,\mathsf{st}_\mathsf{Sim})$
  if $(\chi,\omega) \notin \mathsf{R}$ **then return** $\bot$
  return 1

Figure 1: The simulation-extractability game SimExt for NIZK-PoKs.

Definition 2.2 (Simulation-Extractability). A proof system PS_R for R is simulation-extractable if there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ext})$ such that for all PPT adversaries $\mathcal{A}$ the following advantage is negligible:

$$\mathsf{Adv}^{\mathsf{SimExt}}_{\mathcal{A},(\mathsf{PS}_\mathsf{R},\mathsf{Sim})}(1^\lambda) \coloneqq \left|\Pr\left[\mathsf{SimExt}^{\mathsf{PS}_\mathsf{R},\mathsf{Sim}}_{\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2}\right|,$$

where SimExt is defined in Figure 1.

Remark 2.3 (NIZK-PoKs in the Random Oracle Model). When we think about NIZK-PoKs, we may also think about them in the random oracle model [6]. In this case, we implicitly allow the simulator to observe and program the random oracle. We use the definition with public parameters (aka common reference string), but other notions work as well for our context, as long as straight-line extraction is given.

One of our constructions does not rely on proofs of knowledge and thus avoids simulation-extractability. For this, we define the notion of simulation-soundness, following [65]. Simulation-soundness requires that a malicious prover cannot convince a verifier of a false statement, even if the prover gets access to a simulated proof for a (possibly false) statement. Note that discrete logarithm equality (DLEQ) proofs satisfy simulation-soundness in the random oracle model [46, 43].

$\underline{\mathsf{ZK}^{\mathsf{PS_R},\mathsf{Sim}}_{\mathcal{A}}(1^\lambda)}$

$b \xleftarrow{\$} \{0,1\}; Q \leftarrow \emptyset$
if $b = 0$
  $\mathsf{pp} \xleftarrow{\$} \mathsf{PS_R.Setup}(1^\lambda)$
else
  $(\mathsf{pp}, \mathsf{st_{Sim}}) \xleftarrow{\$} \mathsf{Sim.Setup}(1^\lambda)$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_b^{\mathsf{Prove}}}(1^\lambda, \mathsf{pp})$
return $b' \stackrel{?}{=} b$

$\underline{\text{Oracle } \mathcal{O}_b^{\mathsf{Prove}}(\chi, \omega)}$

if $(\chi, \omega) \notin \mathsf{R}$ **then return** $\bot$
if $b = 0$
  $\pi \xleftarrow{\$} \mathsf{PS_R.Prove}(\chi, \omega)$
else
  $(\pi, \mathsf{st_{Sim}}) \xleftarrow{\$} \mathsf{Sim.Prove}(\chi, \mathsf{st_{Sim}})$
  $Q \leftarrow Q \cup \{(\chi, \pi)\}$
return π

$\underline{\mathsf{SimSnd}^{\mathsf{PS_R},\mathsf{Sim}}_{\mathcal{A}}(1^\lambda)}$

$(\mathsf{pp}, \mathsf{st_{Sim}}) \xleftarrow{\$} \mathsf{Sim.Setup}(1^\lambda)$
$(\chi, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}(1^\lambda, \mathsf{pp})$
$\pi \xleftarrow{\$} \mathsf{Sim.Prove}(\chi, \mathsf{st_{Sim}})$
$(\tilde{\chi}, \tilde{\pi}) \xleftarrow{\$} \mathcal{A}(\mathsf{st}_{\mathcal{A}}, \pi)$
if $(\chi, \pi) = (\tilde{\chi}, \tilde{\pi})$ **then return** 0
return $\mathsf{PS.Verify}(\tilde{\chi}, \tilde{\pi}) = 1 \wedge (\tilde{\chi} \notin \mathcal{L}_{\mathsf{R}})$

Figure 2: The zero-knowledge game ZK and the simulation-soundness game SimSnd.

Definition 2.4 (Zero-Knowledge and Simulation-Soundness). A proof system $\mathsf{PS_R}$ for R is (1) zero-knowledge and (2) simulation-sound if there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Prove})$ such that for all PPT adversaries $\mathcal{A}$ the following two are negligible:

$$\mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{A},(\mathsf{PS_R},\mathsf{Sim})}(1^\lambda) \coloneqq \left| \Pr\left[\mathsf{ZK}^{\mathsf{PS_R},\mathsf{Sim}}_{\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2} \right|, \tag{1}$$

$$\mathsf{Adv}^{\mathsf{SimSnd}}_{\mathcal{A},(\mathsf{PS_R},\mathsf{Sim})}(1^\lambda) \coloneqq \Pr\left[\mathsf{SimSnd}^{\mathsf{PS_R},\mathsf{Sim}}_{\mathcal{A}}(1^\lambda) = 1\right], \tag{2}$$

where ZK and SimSnd are defined in Figure 2.

Definition 2.5 (Public Key Encryption). A public key encryption scheme PKE is a tuple of algorithms $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax.

- $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$. Given the security parameter λ, the Setup algorithm outputs public parameters pp that are implicit input to all subsequent algorithms.
- $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$. Given the security parameter λ, the KGen algorithm outputs a public key pk and a secret key sk.
- $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m)$. Given the public key pk and a message m, the encryption algorithm outputs a ciphertext c.
- $m/\bot \leftarrow \mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, c)$. Given the public key pk, the secret key sk, and a ciphertext c, the Dec algorithm outputs a message m or $\bot$ upon failure.

We denote the message space with $\mathcal{M}$ and the ciphertext space with $\mathcal{C}$. Both may implicitly depend on the security parameter and the public parameters.

Definition 2.6 (Correctness). We say that a public key encryption scheme PKE is correct if for all $\lambda \in \mathbb{N}$, all $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, all messages $m \in \mathcal{M}$, $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$, all $c \in \mathsf{Enc}(\mathsf{pk}, m)$, it holds that $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, c) = m$.

3 Security Definition: Internally Motivated Corruptions

In this section, we introduce our new security definition for threshold encryption. Intuitively, our model captures scenarios in which a party's decision to become corrupted or to reveal its decryption share may depend on information that is unavailable to the adversary, such as the party's own secret key share. As discussed in the introduction, such scenarios come up in bribery settings, e.g., if whether a party is in a relevant committee depends on a secret key or partial decryption, and the party decides to sell their share or not depending on this.

3.1 Threshold Encryption

We first define syntax and correctness for t-of-n threshold encryption, with a non-interactive decryption protocol.

Definition 3.1 (Threshold Encryption). A threshold encryption scheme TE is a tuple of algorithms $\mathsf{TE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{PDec}, \mathsf{Combine})$ with the following syntax.

- $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$. Given the security parameter λ, the Setup algorithm outputs public parameters pp that are implicit input to all subsequent algorithms.
- $(\mathsf{pk}, (\mathsf{sk}_i)_{i\in[n]}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda, n, t)$. Given the security parameter λ, the number of parties n, and the security threshold t, the KGen algorithm outputs a public key pk and a secret key share sk_i for every party $i \in [n]$.
- $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m)$. Given the public key pk and a message m, the encryption algorithm outputs a ciphertext c.
- $d_i/\perp \xleftarrow{\$} \mathsf{PDec}(\mathsf{sk}_i, c)$. Given the secret key share sk_i of party $i \in [n]$ and a ciphertext c, the PDec algorithm outputs a partial decryption d_i or $\perp$ upon failure.
- $m/\perp \leftarrow \mathsf{Combine}(\mathsf{pk}, S, (d_i)_{i\in S}, c)$. Given the public key pk and partial decryptions $(d_i)_{i\in S}$ on behalf of a set of parties $S \subseteq [n]$ and a ciphertext c, the $\mathsf{Combine}$ algorithm outputs a message m or $\perp$ upon failure.

We denote the message space with $\mathcal{M}$ and the ciphertext space with $\mathcal{C}$. Both may implicitly depend on the security parameter and the public parameters.

Definition 3.2 (Correctness). We say that a threshold encryption scheme TE is correct if for all $\lambda, n, t \in \mathbb{N}$ with $0 < t \leq n$, all $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, all messages $m \in \mathcal{M}$, $(\mathsf{pk}, (\mathsf{sk}_i)_{i\in[n]}) \in \mathsf{KGen}(1^\lambda, n, t)$, all $c \in \mathsf{Enc}(\mathsf{pk}, m)$, all sets $S \subseteq [n]$ with $|S| \geq t$, and all $d_i \in \mathsf{PDec}(\mathsf{sk}_i, c)$ with $i \in S$ it holds that

$$\mathsf{Combine}(\mathsf{pk}, S, (d_i)_{i\in S}, c) = m.$$

Remark 3.3 (Robustness and Verifiability of Partial Decryptions). In this paper, we do not explicitly model robustness, i.e., the property that one can decrypt an honest ciphertext even in the presence of malicious partial decryptions, provided sufficiently many of them are honest. Note, however, that robustness can be added to our constructions by attaching a NIZK proof to partial decryptions, attesting to correct computation, and publishing individual public keys. Combining t or more correct partial decryptions and weeding out invalid ones guarantees robustness by the scheme's correctness and soundness of NIZKs. Some of our proofs rely on the fact that PDec is deterministic, which can be accommodated by sampling the randomness for the NIZK proof from a PRF evaluated on the inputs of PDec.

3.2 Internally Motivated Corruptions

We now introduce our new security model for threshold encryption.

CCA Security. Our starting point is the standard notion of indistinguishability under chosen-ciphertext attacks. The adversary specifies two messages m_0, m_1 from the message space and receives a challenge ciphertext c^* encrypting m_b, where b is a bit chosen uniformly at random. Its goal is to guess b with probability significantly greater than $1/2$. In addition, the adversary is given access to a partial decryption oracle $\mathcal{O}^{\mathsf{PDec}}$. Before receiving c^*, it may query $\mathcal{O}^{\mathsf{PDec}}$ on arbitrary ciphertexts. After receiving c^*, it may continue querying the oracle on any ciphertext $c \neq c^*$. For each query, the adversary may choose a party and receives that party's decryption share for the queried ciphertext. For convenience, we split this oracle into two: $\mathcal{O}_0^{\mathsf{PDec}}$, which is available before seeing the challenge ciphertext c^*, and $\mathcal{O}_1^{\mathsf{PDec}}$, which is available afterward. Furthermore, we may provide the adversary with access to a partial decryption oracle for the challenge ciphertext itself, denoted by $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. As long as the adversary does not obtain too many decryption shares through this oracle, security need not be trivially compromised. Note that in [9], this setting with such an oracle $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ is called the *high-threshold setting*.

Static and Adaptive Corruptions. We additionally allow the adversary to corrupt up to $t-1$ parties. We distinguish between *static* and *adaptive* corruptions. In the static setting, the adversary specifies a corruption set $\mathsf{Cor}_{\mathsf{sk}} \subseteq [n]$ before seeing the public parameters or public key. It then receives the secret key shares sk_i for all $i \in \mathsf{Cor}_{\mathsf{sk}}$, together with the public key. In the adaptive setting, the adversary is then also given access to a corruption oracle $\mathcal{O}^{\mathsf{cor}}$. Upon querying a party $i \in [n]$, the oracle returns the corresponding secret key share sk_i. Again, we split this oracle into $\mathcal{O}_0^{\mathsf{cor}}$ (before c^*) and $\mathcal{O}_1^{\mathsf{cor}}$ (after c^*).

Internally Motivated Corruptions. The main novelty of our model is that we allow corruptions to depend on the secret key shares themselves. When invoking the corruption oracle $\mathcal{O}^{\mathsf{cor}}$, the adversary specifies a set $S \subseteq [n]$ of parties that may potentially become corrupted, together with a *motivation function*[12] ψ. This function takes as input the public information (e.g., the public key), a party index $i \in [n]$, and, crucially, the corresponding secret key share sk_i, and outputs a bit. For each party $i \in S$, the corruption is carried out if and only if the function outputs 1. Intuitively, the motivation function models any internal decision process by which a party decides whether to become corrupted based on its private information. Note that standard adaptive corruptions are captured as a special case: the adversary may simply specify a singleton set $S = \{i\}$ together with a motivation function that always outputs 1.

To model that parties may reveal or sell their decryption shares based on secret information, we make an analogous modification to the partial decryption oracle. In this case, the motivation function receives the corresponding partial decryption share as an additional input and determines whether it is revealed to the adversary.

Motivation Function Classes. As we shall see, security in this model cannot be achieved in full generality. In particular, it may be necessary to restrict the motivation functions that the adversary is allowed to specify. To this end, we parameterize the model by *function classes*. For example, the class $\mathcal{F}_1^{\mathsf{PDec}}$ contains all functions (or, more formally, circuits) that the adversary may submit to $\mathcal{O}_1^{\mathsf{PDec}}$, namely to the partial decryption oracle after receiving the challenge ciphertext c^*.

Definition 3.4 (Motivation Function Classes). A motivation function class $\mathcal{F} = (\mathcal{F}_\lambda)_{\lambda \in \mathbb{N}}$ is an ensemble of sets of *circuits* of the following form. For each $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{F}_\lambda$ have the signature

$$C(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i, d_i, c) \to 1/0,$$

i.e., circuits C take as input public parameters pp, a public key pk, a participant index i, a secret key share sk_i, a partial decryption d_i, a ciphertext c, and output a bit. Note that the input domain of circuits in $\mathcal{F}_\lambda$ may depend on the security parameter λ. When running such a circuit without specifying certain parameters in a manner that is clear from context, we omit them from the inputs and imply that they are set to $\perp$. For example[13], we denote $C(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i)$ as a shorthand for omitting c and d_i, meaning $C(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i, \perp, \perp)$. Note that motivation functions can access random oracles, which formally means that they have access to gates evaluating random oracles.

The Classes $\mathcal{F}_*$, $\mathcal{F}_{\mathsf{no\text{-}sk}}$, and $\mathcal{F}_{\mathsf{accept}}$. Throughout this paper, we will often need the motivation function classes $\mathcal{F}_*$, $\mathcal{F}_{\mathsf{no\text{-}sk}}$, and $\mathcal{F}_{\mathsf{accept}}$. $\mathcal{F}_*$ denotes the unrestricted function class containing all[14] circuits with the given signature. $\mathcal{F}_{\mathsf{no\text{-}sk}}$ consists of all motivation functions that do not take a secret key share sk_i as input, or more precisely, where the input wires associated with sk_i are fully disconnected from any gates. Note that membership of a circuit in $\mathcal{F}_{\mathsf{no\text{-}sk}}$ can be efficiently decided by inspecting the circuit. Finally, $\mathcal{F}_{\mathsf{accept}}$ denotes the function class containing only the constant function that always outputs 1 and makes no random oracle queries.

Query Bounds and Batches. We further parameterize our model by query bounds for the individual oracles. Here, we distinguish between the number of oracle queries, denoted by uppercase letters, and the total number of party indices involved in these queries, denoted by lowercase letters. These quantities may differ because the adversary is allowed to submit sets of parties $S \subseteq [n]$ in a single query. We use ∞ to denote that the adversary can make an unlimited number of queries to an oracle, but we note that this still means the number of queries is polynomially bounded, as the adversary is assumed to be PPT.

[12] Formally, we assume that the adversary specifies such functions by providing a *circuit*.
[13] This example is relevant for the oracle $\mathcal{O}^{\mathsf{cor}}$ in Figure 3.
[14] Of course, a polynomially-bounded adversary can only ever output polynomial-sized circuits.

Game $\mathsf{IND\text{-}IMC\text{-}CCA}^{\mathsf{TE}}_{\mathcal{A}}(1^\lambda, n, t, \mathcal{F}, q, Q)$

// Parse $\mathcal{F}$ as $(\mathcal{F}_0^{\mathsf{cor}}, \mathcal{F}_1^{\mathsf{cor}}, \mathcal{F}_0^{\mathsf{PDec}}, \mathcal{F}_1^{\mathsf{PDec}}, \mathcal{F}^{\mathsf{chal}})$

// Parse q as $(q_0^{\mathsf{cor}}, q_1^{\mathsf{cor}}, q_0^{\mathsf{PDec}}, q_1^{\mathsf{PDec}}, q^{\mathsf{chal}})$

// Parse Q as $(Q_0^{\mathsf{cor}}, Q_1^{\mathsf{cor}}, Q_0^{\mathsf{PDec}}, Q_1^{\mathsf{PDec}}, Q^{\mathsf{chal}})$

$(\mathsf{Cor_{sk}}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(1^\lambda, n, t)$ // $\mathsf{Cor_{sk}} \subseteq [n]$

$\mathsf{Cor_{pd}} \leftarrow \emptyset; c^* \leftarrow \bot$

$\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$

$(\mathsf{pk}, (\mathsf{sk}_i)_{i\in[n]}) \xleftarrow{\$} \mathsf{TE.KGen}(1^\lambda, n, t)$

$((m_0, m_1), \mathsf{st}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_0^{\mathsf{cor}}, \mathcal{O}_0^{\mathsf{PDec}}}(\mathsf{st}, \mathsf{pp}, \mathsf{pk}, (\mathsf{sk}_i)_{i\in\mathsf{Cor_{sk}}})$

// $m_0, m_1 \in \mathcal{M}$, $|m_0| = |m_1|$

$b \xleftarrow{\$} \{0,1\}$

$c^* \xleftarrow{\$} \mathsf{TE.Enc}(\mathsf{pk}, m_b)$

$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_1^{\mathsf{cor}}, \mathcal{O}_1^{\mathsf{PDec}}, \mathcal{O}^{\mathsf{PDec\text{-}chal}}}(\mathsf{st}, c^*)$

assert $|\mathsf{Cor_{sk}} \cup \mathsf{Cor_{pd}}| < t$

assert $q \geq (0,0,0,0,0) \wedge Q \geq (0,0,0,0,0)$

return $b \stackrel{?}{=} b'$

Oracle $\mathcal{O}_d^{\mathsf{cor}}(S, \psi)$

assert $S \subseteq [n]$

assert $\psi \in \mathcal{F}_d^{\mathsf{cor}}$

$q_d^{\mathsf{cor}} \leftarrow q_d^{\mathsf{cor}} - |S|$

$Q_d^{\mathsf{cor}} \leftarrow Q_d^{\mathsf{cor}} - 1$

$S' \leftarrow \emptyset$

for $i \in S$ **do**

  if $\psi(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i) = 1$ **then**

    $S' \leftarrow S' \cup \{i\}$

$\mathsf{Cor_{sk}} \leftarrow \mathsf{Cor_{sk}} \cup S'$

return $(\mathsf{sk}_i)_{i\in S'}$

Oracle $\mathcal{O}_d^{\mathsf{PDec}}(S, \varphi, c)$

assert $c \neq c^*$ // always holds for $d = 0$

assert $S \subseteq [n]$

assert $\varphi \in \mathcal{F}_d^{\mathsf{PDec}}$

$q_d^{\mathsf{PDec}} \leftarrow q_d^{\mathsf{PDec}} - |S|$

$Q_d^{\mathsf{PDec}} \leftarrow Q_d^{\mathsf{PDec}} - 1$

$S' \leftarrow \emptyset$

for $i \in S$ **do**

  $d_i \xleftarrow{\$} \mathsf{TE.PDec}(\mathsf{sk}_i, c)$

  if $\varphi(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i, d_i, c) = 1$ **then**

    $S' \leftarrow S' \cup \{i\}$

return $(d_i)_{i\in S'}$

Oracle $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S, \varphi)$

assert $S \subseteq [n]$

assert $\varphi \in \mathcal{F}^{\mathsf{chal}}$

$q^{\mathsf{chal}} \leftarrow q^{\mathsf{chal}} - |S|$

$Q^{\mathsf{chal}} \leftarrow Q^{\mathsf{chal}} - 1$

$S' \leftarrow \emptyset$

for $i \in S$ **do**

  $d_i \xleftarrow{\$} \mathsf{TE.PDec}(\mathsf{sk}_i, c^*)$

  if $\varphi(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i, d_i, c^*) = 1$ **then**

    $S' \leftarrow S' \cup \{i\}$

$\mathsf{Cor_{pd}} \leftarrow \mathsf{Cor_{pd}} \cup S'$

return $(d_i)_{i\in S'}$

Figure 3: The IND-IMC-CCA security game. If any assertion fails, the game returns 0. Note that while circuits in $\mathcal{F}_0^{\mathsf{cor}}$ and $\mathcal{F}_1^{\mathsf{cor}}$ may depend on a partial decryption and ciphertext as input, we implicitly set these inputs to $\bot$ when running ψ.

Definition 3.5 ($(\mathcal{F}, q, Q)$-IND-IMC-CCA Security). We say that a threshold encryption scheme TE is $(\mathcal{F}, q, Q)$-IND-IMC-CCA secure for a tuple of motivation function classes $\mathcal{F} = (\mathcal{F}_0^{\mathsf{cor}}, \mathcal{F}_1^{\mathsf{cor}}, \mathcal{F}_0^{\mathsf{PDec}}, \mathcal{F}_1^{\mathsf{PDec}}, \mathcal{F}^{\mathsf{chal}})$, a vector of individual query bounds $q = (q_0^{\mathsf{cor}}, q_1^{\mathsf{cor}}, q_0^{\mathsf{PDec}}, q_1^{\mathsf{PDec}}, q^{\mathsf{chal}}) \in (\mathbb{N} \cup \{0, \infty\})^5$, and oracle query bounds $Q = (Q_0^{\mathsf{cor}}, Q_1^{\mathsf{cor}}, Q_0^{\mathsf{PDec}}, Q_1^{\mathsf{PDec}}, Q^{\mathsf{chal}}) \in (\mathbb{N} \cup \{0, \infty\})^5$ if for all PPT adversaries $\mathcal{A}$ and all $n \in \mathsf{poly}(\lambda)$, and all $0 < t \leq n$, the following advantage is negligible in λ:

$$\mathsf{Adv}^{\mathsf{IND\text{-}IMC\text{-}CCA}}_{\mathcal{A},\mathsf{TE}}(1^\lambda, n, t, \mathcal{F}, q, Q) := \Pr\left[\mathsf{IND\text{-}IMC\text{-}CCA}^{\mathsf{TE}}_{\mathcal{A}}(1^\lambda, n, t, \mathcal{F}, q, Q) = 1\right] - \frac{1}{2},$$

where the IND-IMC-CCA game is defined in Figure 3. Note that we do not take an absolute value here.

Remark 3.6 Observe that the challenger needs to verify that any motivation function given to $\mathcal{O}_d^{\mathsf{cor}}$, $\mathcal{O}_d^{\mathsf{PDec}}$, and $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ is valid, i.e., is a member of the respective motivation function class. Hence, IND-IMC-CCA security may be non-falsifiable if membership is not efficiently decidable for function classes in $\mathcal{F}$. We make sure that all motivation function classes in this paper are efficiently decidable.

3.3 Simple Observations

We make some simple observations about our security model and relate it to existing security models. Proofs are trivial and omitted for most of the observations. Throughout, we let TE be any threshold encryption scheme.

Relation to Known Models. We now explain for which choices of q, Q, and $\mathcal{F}$ we get security models equivalent to known models:

- **Static Corruptions.** If we set $q_0^{\mathsf{cor}} = q_1^{\mathsf{cor}} = Q_0^{\mathsf{cor}} = Q_1^{\mathsf{cor}} = 0$, then no corruption oracle call is allowed, but the initial corruptions before seeing the public key are still allowed. We obtain what is commonly referred to as a model with static corruptions only.
- **Adaptive Corruptions.** If we set $q_0^{\mathsf{cor}} = q_1^{\mathsf{cor}} = Q_0^{\mathsf{cor}} = Q_1^{\mathsf{cor}} = \infty$, and the classes $\mathcal{F}_0^{\mathsf{cor}}, \mathcal{F}_1^{\mathsf{cor}}$ contain only the function that always outputs 1 (class $\mathcal{F}_{\mathsf{accept}}$), the model is equivalent to a model with standard adaptive corruptions. To see this, note that a reduction can evaluate the function and then make a normal adaptive corruption query for each i for which the function outputs 1.
- **One-Shot Adaptive Corruptions.** The framework also covers the recent notion of security under one-shot adaptive corruptions [52]. Here, the adversary can make standard adaptive corruptions before seeing the challenge ciphertext. Then, it can make one final batch of corruptions in one shot. This can be modeled by setting $q_0^{\mathsf{cor}} = Q_0^{\mathsf{cor}} = \infty$ and $q_1^{\mathsf{cor}} = \infty, Q_1^{\mathsf{cor}} = 1$. Again, if $\mathcal{F}_0^{\mathsf{cor}} = \mathcal{F}_1^{\mathsf{cor}} = \mathcal{F}_{\mathsf{accept}}$, this is exactly the notion (in terms of corruptions) considered in [52]. If not, in particular if $\mathcal{F}_0^{\mathsf{cor}}$ or $\mathcal{F}_1^{\mathsf{cor}}$ may depend on the secret key share sk_i, then we obtain a variant of one-shot adaptive corruptions that are internally motivated.
- **CPA Security.** If we set $q_0^{\mathsf{PDec}} = q_1^{\mathsf{PDec}} = q^{\mathsf{chal}} = 0$, $Q_0^{\mathsf{PDec}} = Q_1^{\mathsf{PDec}} = Q^{\mathsf{chal}} = 0$, we get CPA security.
- **CCA Security.** If we now change $q_0^{\mathsf{PDec}} = \infty$ and $Q_0^{\mathsf{PDec}} = \infty$ and keep the rest, we get CCA1 security. If we additionally set $q_1^{\mathsf{PDec}} = \infty$ and $Q_1^{\mathsf{PDec}} = \infty$, we get CCA2 security. If we set $q^{\mathsf{chal}} = \infty, Q^{\mathsf{chal}} = \infty$, we get what is called high-threshold CCA security in [9]. Again, this is standard CCA security if the motivation function classes only contain functions that ignore the secret inputs d_i and sk_i. Otherwise, we get a notion of CCA security with internally motivated partial decryptions.

Monotonicity Facts. We continue by relating different instantiations of our framework, e.g., what happens when we start with a function class and further restrict it. Intuitively, the more functions we have in our function classes, the stronger the notion (as we give the adversary more choices). This can be made formal by a trivial reduction that just forwards the adversary's queries. We summarize such observations in the following facts, where the relations $\subseteq, \leq$ should be understood to apply to each component.

Fact 3.7 If $\tilde{\mathcal{F}} \subseteq \mathcal{F}$ and TE is $(\mathcal{F}, q, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure, then TE is $(\tilde{\mathcal{F}}, q, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure.

Fact 3.8 If $\tilde{Q} \leq Q$ and TE is $(\mathcal{F}, q, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure, then TE is $(\mathcal{F}, q, \tilde{Q})$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure.

Fact 3.9 If $\tilde{q} \leq q$ and TE is $(\mathcal{F}, q, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure, then TE is $(\mathcal{F}, \tilde{q}, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure.

Example 3.10 The three facts can easily be combined by applying them in sequence. This works as long as the preconditions *all* hold: consider $\tilde{\mathcal{F}} \subseteq \mathcal{F}$ and

$$Q = (\infty, 2, 2, n, 1) \qquad \tilde{Q} = (\infty, 1, 2, n/2, 0)$$
$$q = (\infty, n, n, \infty, n) \qquad \tilde{q} = (\infty, 1, n, \infty, 0).$$

Then, $(\mathcal{F}, q, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ security implies $(\tilde{\mathcal{F}}, \tilde{q}, \tilde{Q})$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ security. On the other hand, if, for instance, $\tilde{q} \leq q$ but $\tilde{\mathcal{F}} \supseteq \mathcal{F}$, the two notions seem to be orthogonal: enlarging $\mathcal{F}$ strengthens, shrinking q weakens, so neither notion implies the other trivially.

Useful Implications. We introduce two additional useful implications. First, observe that security under a partial decryption oracle on *non-challenge* ciphertexts without internal motivation already implies security under a partial decryption oracle with internal motivation that does not use the secret key shares.

Fact 3.11 If TE is $(\mathcal{F}, q, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure with $\mathcal{F}_0^{\mathsf{PDec}} \supseteq \mathcal{F}_{\mathsf{accept}}$ and $\mathcal{F}_1^{\mathsf{PDec}} \supseteq \mathcal{F}_{\mathsf{accept}}$, then TE is $(\tilde{\mathcal{F}}, q, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure for $\widetilde{\mathcal{F}_0^{\mathsf{PDec}}} = \widetilde{\mathcal{F}_1^{\mathsf{PDec}}} = \mathcal{F}_{\mathsf{no\text{-}sk}}$, where $\tilde{\mathcal{F}}$ is otherwise the same as $\mathcal{F}$.

The fact follows because a reduction receiving an $\mathcal{O}^{\mathsf{PDec}}$ query with a motivation function $\varphi \in \mathcal{F}_{\mathsf{no\text{-}sk}}$ can simply query the $\mathcal{O}^{\mathsf{PDec}}$ oracle of its challenger with the constant motivation function $\tilde{\varphi} = 1$ to learn the partial decryption(s) and use them afterwards to compute φ locally, returning the partial decryption to the adversary if the output of φ is 1. While executing φ, the reduction passes random oracle queries between φ and its challenger. Note that the same does not hold for $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, since a query to the $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ oracle of the challenger with $\tilde{\varphi} = 1$ will count towards $\mathsf{Cor}_{\mathsf{pd}}$, even if φ turns out to return 0 for the resulting partial decryption.

Next, we observe that $\mathcal{F}_1^{\mathsf{cor}} = \mathcal{F}_*$ implies $\mathcal{F}^{\mathsf{chal}} = \mathcal{F}_*$ if $q_1^{\mathsf{cor}} = Q_1^{\mathsf{cor}} = \infty$.[15]

Fact 3.12 If TE is $(\mathcal{F}, q, Q)$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure with $q_1^{\mathsf{cor}} = Q_1^{\mathsf{cor}} = \infty$ and $\mathcal{F}_1^{\mathsf{cor}} = \mathcal{F}_*$ then TE is $(\tilde{\mathcal{F}}, \tilde{q}, \tilde{Q})$-$\mathsf{IND\text{-}IMC\text{-}CCA}$ secure for $\widetilde{q^{\mathsf{chal}}} = \widetilde{Q^{\mathsf{chal}}} = \infty$ and $\widetilde{\mathcal{F}^{\mathsf{chal}}} = \mathcal{F}_*$, where $(\tilde{\mathcal{F}}, \tilde{q}, \tilde{Q})$ is otherwise the same as $(\mathcal{F}, q, Q)$.

Whenever a reduction receives an $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query for motivation function φ and set S, the reduction constructs $\psi(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i) = \varphi(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i, \mathsf{PDec}(\mathsf{sk}_i, c^*), c^*)$, hardcoding the challenge ciphertext c^*. It forwards (S, ψ) to $\mathcal{O}_1^{\mathsf{cor}}$ of its challenger, and uses the resulting secret key shares $(\mathsf{sk}_i)_{i \in S'}$ to compute partial decryptions on behalf of S' and answer the adversary's $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query.[16] In the reduction's simulation, S' will be joined with $\mathsf{Cor}_{\mathsf{pd}}$, and the challenger joins S' with $\mathsf{Cor}_{\mathsf{sk}}$. However, since the game only requires $|\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}| < t$, this difference is not significant.

4 Attacks

The general security model for internally motivated corruptions, which we have presented in Section 3, is strong. In this section, we show that without any restrictions on the number of queries or the allowed motivation functions, achieving the notion is not possible. We do so by showing generic attacks. We also show an attack against a more specific class of schemes with a silent setup. For our attacks, we assume that the message space contains at least three messages.

4.1 Attack with Many Partial Decryption Queries

We now present our first generic attack. It shows that our new notion is not possible in full generality, assuming the classes $\mathcal{F}_d^{\mathsf{PDec}}$ are expressive enough and the adversary can make many queries. Let $\mathsf{TE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{PDec}, \mathsf{Combine})$ be a threshold encryption scheme. We assume that w is an upper bound on the bitlength of secret key shares sk_i. This is polynomial in λ. For the attack, we consider the following setting:

- **No corruption queries.** $\mathcal{F}_0^{\mathsf{cor}} = \mathcal{F}_1^{\mathsf{cor}} = \emptyset$, $q_0^{\mathsf{cor}} = q_1^{\mathsf{cor}} = 0$, $Q_0^{\mathsf{cor}} = Q_1^{\mathsf{cor}} = 0$. Also, no static corruptions.
- **No queries on the challenge ciphertext.** $\mathcal{F}^{\mathsf{chal}} = \emptyset$, $q^{\mathsf{chal}} = Q^{\mathsf{chal}} = 0$.
- **Bit-Extraction Complete Classes.** Define

$$\varphi^{(j)}(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i, d_i, c) = j\text{-th bit of } \mathsf{sk}_i \quad \text{for all } j \in [w].$$

  Let $\mathcal{W}_0 := \{j \in [w] \mid \varphi^{(j)} \in \mathcal{F}_0^{\mathsf{PDec}}\}$ and $\mathcal{W}_1 := \{j \in [w] \setminus \mathcal{W}_0 \mid \varphi^{(j)} \in \mathcal{F}_1^{\mathsf{PDec}}\}$. We assume that $\mathcal{W}_0 \cup \mathcal{W}_1 = [w]$, i.e., every $\varphi^{(j)}$ is in $\mathcal{F}_0^{\mathsf{PDec}} \cup \mathcal{F}_1^{\mathsf{PDec}}$.

[15] Note that even a tighter formulation with respect to query bounds is possible here, but this rather loose formulation is sufficient for our needs.

[16] Observe that the reduction has to hardcode randomness for PDec into φ and then use the same randomness to derive the partial decryption(s) it returns to its adversary.

- **Many Partial Decryption Queries.** $q_0^{\mathsf{PDec}} = t \cdot |\mathcal{W}_0|$, $Q_0^{\mathsf{PDec}} = |\mathcal{W}_0|$, $q_1^{\mathsf{PDec}} = t \cdot |\mathcal{W}_1|$, $Q_1^{\mathsf{PDec}} = |\mathcal{W}_1|$.

Intuitively, we can use the predicates $\varphi^{(j)}$ to learn the bits of secret key shares bit by bit. For one secret key share, we then need w queries. Once we have learned all bits of at least t secret key shares, we can decrypt. Also, note that we can query all t secret key shares in a batch query, which explains the query parameters above. We now specify the adversary $\mathcal{A}$ in the game in Figure 3:

1. Output an empty set of static corruptions $\mathsf{Cor}_{\mathsf{sk}} = \emptyset$.
2. Get the public parameters and the public key pk.
3. Fix $S = \{1, \ldots, t\}$, i.e., the first t parties.
4. Pick an arbitrary message $\tilde{m}$ and generate a ciphertext $\tilde{c} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, \tilde{m})$.
5. For every $j \in \mathcal{W}_0$:
   (a) Query $\mathcal{O}_0^{\mathsf{PDec}}(S, \varphi^{(j)}, \tilde{c})$.
   (b) As a result of the query, learn S'. Note that $S' = \{i \in \{1, \ldots, t\} \mid \mathsf{sk}_{i,j} = 1\}$, where we denote the j-th bit of sk_i by $\mathsf{sk}_{i,j}$.
   (c) Thus, set
   $$\mathsf{sk}_{i,j} := \begin{cases} 0, & \text{if } i \notin S' \\ 1, & \text{if } i \in S' \end{cases} \quad \text{for all } i \in \{1, \ldots, t\}.$$
6. Pick two arbitrary messages $m_0 \neq m_1$ (both not equal to $\tilde{m}$) from the message space. Submit them to the game and get challenge ciphertext c^* in return.
7. Repeat Step 5, now for every $j \in \mathcal{W}_1$ and with oracle $\mathcal{O}_1^{\mathsf{PDec}}$.
8. Note that by our assumption, all bits of sk_i are now defined for all $i \in \{1, \ldots, t\}$.
9. Compute $m^* = \mathsf{Combine}(\mathsf{pk}, S, (\mathsf{PDec}(\mathsf{sk}_i, c^*))_{i \in S}, c^*)$. Return 1 if and only if $m^* = m_1$.

Note that this attack succeeds with probability 1, assuming $\tilde{c} \neq c^*$, and this assumption follows from perfect[17] correctness (cf. Definition 3.2), as if they were equal, then the ciphertext would decrypt to both $\tilde{m}$ and $m_b \neq \tilde{m}$.

Remark 4.1 (Trading Query Size for Static Corruptions). Note that if we modify the adversary to make $c \leq t - 1$ static corruptions initially, then the set S does not need to include those parties. This changes parameters of the attack to $q_0^{\mathsf{PDec}} = (t - c) \cdot |\mathcal{W}_0|$, $Q_0^{\mathsf{PDec}} = |\mathcal{W}_0|$, $q_1^{\mathsf{PDec}} = (t - c) \cdot |\mathcal{W}_1|$, $Q_1^{\mathsf{PDec}} = |\mathcal{W}_1|$. Intuitively, the total saving in the query budget is cw.

4.2 Attack with Decrypting Motivation Function

Our second attack makes $t - 1$ corruptions, and then *just one* partial decryption query where the motivation function internally decrypts and checks equality with one of the target messages. The query hardcodes the challenge ciphertext, but is submitted to the non-challenge partial decryption oracle $\mathcal{O}^{\mathsf{PDec}}$. Note that this attack is orthogonal to the previous attack, as a different motivation function is used, and only one partial decryption query is needed. As before, let $\mathsf{TE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{PDec}, \mathsf{Combine})$ be a threshold encryption scheme. We consider the following setting:

- **Static corruptions.** $\mathcal{F}_0^{\mathsf{cor}} = \mathcal{F}_1^{\mathsf{cor}} = \emptyset$, $q_0^{\mathsf{cor}} = q_1^{\mathsf{cor}} = 0$, $Q_0^{\mathsf{cor}} = Q_1^{\mathsf{cor}} = 0$. But we make $t - 1$ static corruptions.
- **No queries on the challenge ciphertext.** $\mathcal{F}^{\mathsf{chal}} = \emptyset$, $q^{\mathsf{chal}} = Q^{\mathsf{chal}} = 0$.

[17] With statistical correctness, the attack would still have overwhelming success probability.

- **Decrypt-and-Combine Classes.** For any fixed $D = (d_j)_{j=1}^{t-1}$, c^*, and m_0, consider

$$\varphi_{D,c^*,m_0}(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i, d_i, c) = \begin{cases} \big[m_0 = \mathsf{Combine}\big(\mathsf{pk}, [t], (d_j)_{j\in[t-1]} \cup (\mathsf{PDec}(\mathsf{sk}_i, c^*)), c^*\big)\big] & \text{if } i = t, \\ 0 & \text{else,} \end{cases}$$

  where D, c^*, m_0 and the randomness for PDec are hardcoded. We assume that each such φ_{D,c^*,m_0} is in $\mathcal{F}_1^{\mathsf{PDec}}$, whereas $\mathcal{F}_0^{\mathsf{PDec}} = \emptyset$ is sufficient.

- **One Partial Decryption Query.** $q_0^{\mathsf{PDec}} = 0$, $Q_0^{\mathsf{PDec}} = 0$, $q_1^{\mathsf{PDec}} = 1$, $Q_1^{\mathsf{PDec}} = 1$.

The intuition of the attack is to get $t-1$ decryption shares via the static corruptions, and then compute the final decryption share and combine and compare *within* the motivation function. The adversary $\mathcal{A}$ in the game in Figure 3 is as follows:

1. Output the set of static corruptions $\mathsf{Cor}_{\mathsf{sk}} = \{1, \ldots, t-1\}$.
2. Get public parameters, public key pk, as well as the secret key shares $\mathsf{sk}_j, j \in \mathsf{Cor}_{\mathsf{sk}}$.
3. Pick two arbitrary messages $m_0 \neq m_1$ from the message space. Submit them to the game and get the challenge ciphertext c^* in return.
4. For each $j \in \{1, \ldots, t-1\}$, compute $d_j \xleftarrow{\$} \mathsf{PDec}(\mathsf{sk}_j, c^*)$. Set $D := (d_j)_{j=1}^{t-1}$.
5. Pick an arbitrary message $\tilde{m} \notin \{m_0, m_1\}$ and generate a ciphertext $\tilde{c} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, \tilde{m})$.
6. Make a query $\mathcal{O}_1^{\mathsf{PDec}}(\{t\}, \varphi_{D,c^*,m_0}, \tilde{c})$.
7. If this query returns an empty result, output $b' := 1$. Else, output $b' := 0$.

As in the previous attack, we can argue that $\tilde{c} \neq c^*$ and so the query is valid. Further, note that if $b = 0$, then m_0 is encrypted in c^*, and so φ_{D,c^*,m_0} outputs 1, meaning that the result is non-empty. If $b = 1$, then m_1 is encrypted and φ_{D,c^*,m_0} outputs 0, and the result is empty. Thus, $b' = b$ holds in any case, meaning the attack succeeds with probability 1.

4.3 Attack on Committee-based Schemes à la HSW25

We will now show an attack (in our stronger security model!) that applies to a specific class of schemes, namely, committee-based schemes like the silent-setup scheme by Hall-Andersen, Simkin, and Wagner [52]. In their scheme, the encrypting party selects a random committee of parties, and then uses a naive scheme to encrypt the message towards those: secret share the message (with relative threshold c) and encrypt each share to one committee member via (anonymous) public key encryption. They show that this yields a scheme CCA1 secure under *one-shot adaptive corruptions* [52], assuming the committee size is $\Theta(\lambda)$, where the privacy threshold is $(\eta - \epsilon)n$, and the reconstruction threshold is $(\eta + \epsilon)n$, for constants η, ϵ. The intuition is that due to the anonymity of the public key encryption scheme, the adversary will not manage to corrupt sufficiently many committee members. However, note that in the internally motivated corruption setting, the motivation function can check if a party is in the committee for the challenge ciphertext, and only then corrupt it. This is the intuition of our attack.

Remark 4.2 (Relation to [52]). We note that the attack we present here does not invalidate the results in [52], as we work in a much stronger security model. Also, the scheme for which we formally present the attack here is a simplified variant of the scheme in [52], e.g., without relying on static dictionaries and non-interactive key exchange. The attack applies equally to the concrete scheme in [52], but it also applies to related constructions that rely on sampling a committee and then keeping it hidden.

Remark 4.3 (Template Description). We write the scheme below as a template, without specifying every detail. Also, we note that it is a scheme with a *silent setup*, i.e., each party generates its key material independently.

Construction 4.4 (Committee-based Scheme Template). *Let* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public key encryption scheme. Let* $M \in \mathbb{N}, \eta \in (0, 1)$ *be parameters, where we assume that* $M \ll t, n$.

Parameters. *The parameters include at least the parameters* $\mathsf{pp} \xleftarrow{\$} \mathsf{PKE.Setup}(1^\lambda)$ *of* PKE.

Keys. *Each party* $i \in [n]$ *sets up a key pair* $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$*, where* pk_i *is published, and* sk_i *is kept secret.*

Encryption. *To encrypt a message* m*, sample a random committee* $\mathcal{C} \subseteq [n]$ *of size* M *and secret share* m *with an* ηM*-of-*M *secret sharing. Assign a share* m_i *to each* $i \in \mathcal{C}$*. Compute* $c_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_i)$*, and include all* $c_i, i \in \mathcal{C}$ *in the final ciphertext* c.

Decryption. *To partially decrypt, each party* $i \in [n]$ *scans through* c *and finds the first* PKE *ciphertext for which decryption does not return* $\perp$*. The partial decryption is then the resulting* m_i*. Enough of these* m_i *can be combined to recover* m *via the reconstruction of the* ηM*-of-*M *sharing.*

Note that for the scheme to work correctly, one needs to be able to figure out from the partial decryption whether a party is in the committee or not. More precisely, one needs that decrypting a ciphertext $c_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, m_i)$ with a wrong key sk_j for $j \neq i$ yields $\perp$. This is explicitly achieved in [52] by adding random oracle-based commitments to the ciphertexts, but we just assume this property holds directly for PKE to keep the description simple. While this property is needed for correctness, it will also be useful for our attack. We consider the following setting:

- **No corruptions.** $\mathcal{F}_0^{\mathsf{cor}} = \mathcal{F}_1^{\mathsf{cor}} = \emptyset$, $q_0^{\mathsf{cor}} = q_1^{\mathsf{cor}} = 0$, $Q_0^{\mathsf{cor}} = Q_1^{\mathsf{cor}} = 0$. We also do not need any static corruptions.
- **No queries on non-challenge ciphertexts.** $q_0^{\mathsf{PDec}} = 0$, $Q_0^{\mathsf{PDec}} = 0$, $q_1^{\mathsf{PDec}} = 0$, $Q_1^{\mathsf{PDec}} = 0$, $\mathcal{F}_0^{\mathsf{PDec}} = \mathcal{F}_1^{\mathsf{PDec}} = \emptyset$.
- **One-shot query on the challenge ciphertext.** $q^{\mathsf{chal}} = n$, $Q^{\mathsf{chal}} = 1$, and $\mathcal{F}^{\mathsf{chal}}$ needs to contain the following function:
$$\varphi(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i, d_i, c^*) = \begin{cases} 1 & \text{if } d_i \neq \perp \\ 0 & \text{if } d_i = \perp \end{cases}.$$
Observe that the function is fixed independently of any public parameters or variables defined during the game. In that sense, this is a *selective* attack with respect to the choice of the motivation function.

Here is how the attack works: get the challenge ciphertext, make a query $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S, \varphi)$ with $S = [n]$ and φ as above. Then, the oracle returns all decryption shares that are not $\perp$, which is exactly the M shares of the committee members $i \in \mathcal{C}$. Then, reconstruct the message via the reconstruction of the ηM-of-M sharing and win the game. Note that because $M \ll t$, the adversary does not learn too many partial decryptions on the challenge ciphertext.

Remark 4.5 (Attack via Corruptions). We have presented the attack by using a query to the partial decryption oracle for the challenge ciphertext. One could equally run the attack by making a corruption query instead. Namely, the motivation function would have c^* hardcoded, would internally compute the partial decryption for it, and then run φ from above. The attack would then no longer be selective, as c^* changes the concrete function that is used.

5 Constructions with Internally Motivated Partial Decryptions

In this section, we show our first class of positive results. We start with a technical overview.

5.1 Overview

We consider the setting in which the adversary can make internally motivated partial decryption queries on the challenge ciphertext c^*. We will first build a simple construction that illustrates the core challenges and ideas. In the final step, we generalize the construction to improve the statistical part of the security bound. Throughout the overview, we will ignore corruptions, i.e., assume there are no (not even static) corruptions. In our final scheme, we will tolerate static corruptions.

Starting Point: Plain Threshold ElGamal. Our starting point is the threshold ElGamal encryption scheme [38, 32]. Let us recall this scheme first, working over a cyclic group $\mathbb{G}$ with generator g of order p. The public key is $\mathsf{pk} = g^x$ for secret key $x \in \mathbb{Z}_p$. Let $(x_i)_{i\in[n]}$ be a t-of-n sharing of x. Each party i gets $\mathsf{sk}_i = x_i$. To encrypt a message $m \in \mathbb{G}$ with randomness $r \in \mathbb{Z}_p$, one computes the ciphertext $c = (c_1, c_2) = (g^r, \mathsf{pk}^r \cdot m)$. To decrypt, each party releases their partial decryption $d_i = c_1^{x_i}$. Via Lagrange interpolation in the exponent, any t of these d_i can be combined to recompute the decryption key $Z = c_1^x = \mathsf{pk}^r$. From c_2 and this decryption key, m can be computed. It is easy to see that this scheme is CPA secure under the Diffie-Hellman assumption. Further, one can extend the scheme to support partial decryption queries on non-challenge ciphertexts by adding an appropriate (efficient) NIZK to the ciphertext. We will ignore this for the rest of this overview.[18]

Step One: High-Threshold CCA Security. Before considering internally motivated partial decryption queries, we first want to establish security against up to $t-1$ *ordinary* partial decryption queries on the challenge ciphertext.[19] Our idea here, following earlier works [9], is to make the partial decryptions look fully random individually using masking.[20] We implement this using a zero-sharing in combination with the Diffie-Hellman assumption. More precisely, we change the partial decryption of the i-th party to

$$d_i \leftarrow c_1^{x_i} \cdot \mathsf{H}_{\mathsf{PDec}}(c)^{y_i},$$

where $\mathsf{H}_{\mathsf{PDec}}$ is a random oracle and $(y_i)_{i\in[n]}$ is a random sharing of 0 that has been distributed during key generation, i.e., the i-th party now gets $\mathsf{sk}_i = (x_i, y_i)$. As we will show, for any set of at most $t-1$ parties, the corresponding masking terms $\mathsf{H}_{\mathsf{PDec}}(c)^{y_i}$ are jointly computationally indistinguishable from random under the Diffie-Hellman assumption. Consequently, as long as fewer than t partial decryptions are revealed, the adversary learns no non-trivial additional information about the encrypted message.

Proving Internally Motivated Partial Decryptions – Failed Attempt. It is instructive to examine why the natural proof strategy fails for the current scheme once internally motivated partial decryption queries are allowed. Suppose that, as described above, we first replace the at most $t-1$ partial decryptions of the challenge ciphertext revealed to the adversary by uniformly random values, relying on the Diffie-Hellman assumption. Note that the game rules out that there are t or more such revealed partial decryptions, since otherwise the adversary could reconstruct the full decryption and trivially win. Now, the standard ElGamal proof would replace the decryption key $Z = \mathsf{pk}^r$ with a random group element, again using the Diffie-Hellman assumption, and then argue that Z perfectly masks m as a one-time-pad.

However, note that this argument breaks down in the presence of internally motivated partial decryption. Namely, while only up to $t-1$ partial decryptions d_i are revealed to the adversary, evaluating the motivation function requires the game to potentially use *all* d_i. In particular, the game must be able to compute every d_i explicitly from the corresponding share x_i, and therefore must effectively know the full secret key x, that is, the discrete logarithm of pk. At a more intuitive level, the obstruction is that the outputs of the motivation functions, which are revealed to the adversary, may leak non-trivial information about x and Z.

Interpolation to the rescue ... almost. Fortunately, there is a way to (almost) save the proof. To understand the idea, note that the partial decryptions are on a degree $< t$ polynomial in the exponent. Namely, letting w be the discrete logarithm of $\mathsf{H}_{\mathsf{PDec}}(c)$, they have the form

$$d_i = g^{ra(i)+wz(i)} \quad \text{with } \deg a < t,\ a(0) = x;\ \deg z < t,\ z(0) = 0.$$

That is, the exponents are on the degree $< t$ polynomial $f(X) = ra(X) + wz(X)$. Moreover, we have $Z = c_1^{*x} = g^{ra(0)+wz(0)} = g^{ra(0)}$, which means that we have another evaluation of $f(X) = ra(X) + wz(X)$ in the exponent. Thus, once the $t-1$ first partial decryptions d_i on the challenge ciphertext are sampled

[18] In Sections 5.3 and 5.4 we assume for simplicity a simulation-extractable NIZK, but we explain how to relax to simulation-soundness in Section 5.5.

[19] Note that this is *high-threshold CCA security* in the terminology of [9].

[20] We note that the concept of masking individual shares has been used in other contexts as well [36, 54], although for a different purpose.

at random, we can compute all remaining ones via interpolation in the exponent, given those $t-1$ random ones and $Z = c_1^{*x} = g^{ra(0)+wz(0)} = g^{f(0)}$. Importantly, this can be done *without knowing* x. Once we do this, we can indeed switch Z from pk^r to random.

Unfortunately, this does not complete the proof. The standard proof template would now argue that the random value Z acts as a one-time-pad that hides m. Here, however, Z is used not only to form the challenge ciphertext but also to derive some of the partial decryptions through interpolation. These derived partial decryptions are therefore correlated with Z, and the usual one-time-pad argument fails.

Hashing saves the day. At first glance, the failure of the one-time-pad argument appears to leave us stuck. There is, however, a way forward. The key observation is that the adversary never receives in full any of the partial decryptions d_i that are derived from Z; otherwise, it would exhaust its partial decryption budget and lose. Instead, for each such d_i, the adversary learns *only a single bit*: the output of the corresponding motivation function. This means that if Z has, for instance, 2λ bits of entropy, and the adversary only makes λ many queries (learning at most λ bits about Z), Z should still have a sufficient remaining entropy. To make use of this intuition, we need to leverage a sufficiently good entropy extractor. In our case, we just use a random oracle $\mathsf{H}_{\mathsf{mask}}$ and redefine our ciphertext to be

$$c = (c_1, c_2) = (g^r, \mathsf{H}_{\mathsf{mask}}(Z) \oplus m).$$

Then, once Z is switched to random, we should be able to argue that even after all queries, it still has sufficient entropy, so that the adversary never queries $\mathsf{H}_{\mathsf{mask}}(Z)$. This implies that $\mathsf{H}_{\mathsf{mask}}(Z)$ now acts as a one-time-pad.

Statistical Core Lemma. Turning this intuition into a formal proof is non-trivial. In particular, the adaptive nature of the queries makes precisely counting the remaining entropy of Z challenging. To overcome this challenge, we first model the statistical core argument (after switching Z to random) as a separate game. For this game, we then show an unconditional lower bound against unbounded adversaries. The game is fully specified and analyzed in Section 5.2, and we only explain a simplified version here. At the beginning of the game, a random polynomial f of degree $< t$ is sampled, where we may think of $f(0)$ as the discrete logarithm of Z. The goal of the unbounded adversary is to guess $f(0)$. To do so, the adversary is allowed to specify (multiple) functions ρ. Whenever it specifies a function ρ, it learns all $(i, f(i))$ for $i \in [n]$ with $\rho(i, f(i)) = 1$. Of course, we assume that at most $t-1$ points are learned in total. To analyze the game, let $\mathcal{W}$ denote the set of polynomials f for which the adversary wins. Our goal is to bound the size of $\mathcal{W}$. To do so, we will show the existence of an encoding function

$$\mathsf{Enc}\colon \mathcal{W} \to \binom{[n]}{t-1} \times \mathbb{Z}_p^{t-1}.$$

Crucially, we can use the correctness of the adversary's final guess to argue that this encoding is *injective*. Thus, the size of $\mathcal{W}$ is bounded by the size of the encoding's range. As the total set of polynomials that can be chosen has size p^t, the success probability is at most $\binom{n}{t-1}p^{t-1}/p^t = \binom{n}{t-1}/p$. In our actual core lemma, we let the adversary specify multiple functions, and it can win at any time by submitting the solution $f(0)$ to a dedicated oracle. The encoding needs to be adapted accordingly, by also including the index of the query that made the adversary win. This adds another (polynomial) term to the actual bound, but we ignore this in this overview.

Increasing Entropy. The bound we have so far is the sum of advantages against the Diffie-Hellman assumption and a statistical term of the form $\binom{n}{t-1}/p$. Notably, the exponential loss $\binom{n}{t-1}$ is just in a statistical term and not related to any computational assumption. If n and t are small, this is sufficient. However, if $\binom{n}{t-1}$ is large, a bigger $p = 2^{\mathsf{poly}(\lambda)}$ must be chosen, e.g., $p \geq 2^{n+\lambda}$, which means we would have to use non-standard groups. To avoid this, we want to increase the entropy of Z. This can be done via the use of parallel repetition in two ways:

1. We use multiple independent ciphertext components $g^{r_1}, \ldots, g^{r_u}$, and Z becomes $(\mathsf{pk}^{r_j})_{j\in[u]}$. We can show that the bound improves to $\binom{n}{t-1}/p^u$.
2. We use multiple independent keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_v$, and redefine Z to be $(\mathsf{pk}_k^r)_{k\in[v]}$. The bound becomes $\binom{n}{t-1}/p^v$.

In fact, we can even combine both solutions and get the bound $\binom{n}{t-1}/p^{uv}$. We can then pick v, u depending on n, t and p. This allows us to compensate for the binomial coefficient while sticking to standard groups (with $p \approx 2^{2\lambda}$), and while having a flexible tradeoff between ciphertext and key size. In particular, we get instantiations where both ciphertext and public key size are sublinear (e.g., $O(\sqrt{n})$) in n. This is the construction we present and analyze in Section 5.3.

Constant Ciphertexts and Keys via Pairings. So far, we need to pay with a larger ciphertext and/or public key if we have a large n and $t \approx n/2$. As we discuss in Section 7, the overhead is fairly small for any reasonable parameters (n, t). Still, we view it as an interesting question, both theoretically and practically, to improve this to a constant overhead. Indeed, we show how to do so via pairings. Instead of having many public keys, we use a trick inspired by Boneh-Franklin IBE [10]: for a pairing e and a public key $\mathsf{pk} = g^x$, define the ciphertext as

$$(g^r, \mathsf{H}_{\mathsf{mask}}(k_1, \ldots, k_\ell) \oplus m), \quad \text{where } k_j = e(\mathsf{pk}, \mathsf{H}_{\mathsf{id}}(j))^r, \ j \in [\ell],$$

where ℓ is chosen such that $\binom{n}{t-1}/p^\ell \leq 2^{-\lambda}$. The proof now uses the Bilinear Diffie-Hellman assumption and our statistical core lemma from above. We refer to Section 5.4 for details.

Motivation Functions with Oracle Access. Our discussion so far has ignored the fact that motivation functions may also make queries to random oracles. This somewhat complicates the analysis. For example, when $t-1$ parties are already corrupted, a motivation function call for another party has enough information to query $g^{f(0)}$. The main observation for why security is preserved is that such a call must always return 0 on a winning execution, since otherwise the adversary would exceed its budget[21]. Therefore, *no* information is leaked from the motivation function to the adversary, and it is as if the query was never made. In the other case, where the budget is $\leq t-2$, we can think of the motivation function as an additional corrupted party, increasing the budget virtually to $\leq t-2+1 = t-1$, which is still insufficient to query $g^{f(0)}$, by our previous arguments. We make this intuition formal by adding suitable adaptations to the final steps of the game and the core lemma.

5.2 Statistical Core Lemma

In this section, we introduce our core statistical lemma. Informally, the lemma states that a random polynomial f over $\mathbb{Z}_p$ of degree $< t$ remains unpredictable at $f(0)$ if the adversary observes fewer than t evaluations at indices among the set $[n]$. This remains true even if the indices where evaluations are revealed are selected by a predicate that depends on the polynomial's value at that point itself. The setting is further generalized as follows: First, the adversary can initially specify up to $t-1$ points (other than 0) of the polynomial.[22] Second, we consider *vectors of ℓ polynomials* $\boldsymbol{f} = (f_j)_{j\in[\ell]}$, where each f_j is a random degree $< t$ polynomial consistent with the values specified by the adversary.[23] Concretely, we consider the following rough structure for the game:

1. The adversary outputs a corrupted set $\mathsf{Cor} \subseteq [n]$, value vectors $z_i \in \mathbb{Z}_p^\ell$ for all $i \in \mathsf{Cor}$, and a predicate ρ.
2. A random ℓ-valued polynomial $\boldsymbol{f} \in \mathbb{Z}_p[X]^\ell$ of degree $< t$ is sampled, which is consistent on Cor with the z_i.
3. The adversary gets all evaluation points $(i, \boldsymbol{f}(i))$ for which $\rho(i, \boldsymbol{f}(i)) = 1$,
4. and has to guess $\boldsymbol{f}(0)$ by querying an oracle. The adversary wins if it guesses correctly while it learned fewer than t evaluations (including the z_i's).

We also allow the adversary to pick multiple predicates ρ_k in an *adaptive* way, one after another, seeing the evaluations of the functions before. We only evaluate motivation functions when the budget (i.e., the number of evaluations learned) is below $t-1$. This is sufficient for our purposes, as any new evaluation

[21] The budget is the size of the union of corrupted parties and parties who revealed their decryption share on the challenge ciphertext.

[22] We will use this to support static corruptions.

[23] We will use this to increase the entropy by a factor of ℓ.

that it would learn when the budget is already at $t-1$ would make the adversary lose immediately. Additionally, we allow the ρ_k to query the oracle directly. This means that the adversary can also win by submitting a ρ_k that queries $\boldsymbol{f}(0)$. Looking forward, this is what enables the motivation functions to query random oracles in our construction.

$\mathsf{GameStat}_{\mathcal{A}}(n,t,p,\ell,Q)$

$(\mathsf{Cor},(\boldsymbol{z}_i)_{i\in\mathsf{Cor}},\mathsf{st}) \xleftarrow{\$} \mathcal{A}(n,t,p,\ell,Q)$ // $\forall i\in\mathsf{Cor}, z_i\in\mathbb{Z}_p^\ell$

assert $\mathsf{Cor}\subseteq[n]\wedge|\mathsf{Cor}|<t$

$\boldsymbol{f}\xleftarrow{\$}\{\boldsymbol{F}=(F_1,\ldots,F_\ell)\in\mathbb{Z}_p[X]^\ell_{\deg<t}\colon \forall i\in\mathsf{Cor}\colon \boldsymbol{F}(i)=\boldsymbol{z}_i\}$

$S^*:=\emptyset; S^{(0)}:=\emptyset$

for $k=1,\ldots,Q$:

  $(\rho_k,\mathsf{st})\xleftarrow{\$}\mathcal{A}^{\mathcal{O}^{\mathsf{check}}}(\mathsf{st},S^{(k-1)},(\boldsymbol{f}(i))_{i\in S^{(k-1)}})$

  if $|\mathsf{Cor}\cup S^*|\geq t-1$ **then**

    $S^{(k)}\leftarrow\emptyset$

  else

    $S^{(k)}\leftarrow\{i\in[n]\colon \rho_k^{\mathcal{O}^{\mathsf{check}}}(i,\boldsymbol{f}(i))=1\}$

  $S^*\leftarrow S^*\cup S^{(k)}$

$\mathcal{A}^{\mathcal{O}^{\mathsf{check}}}(\mathsf{st},S^{(Q)},(\boldsymbol{f}(i))_{i\in S^{(Q)}})$

return 0

Oracle $\mathcal{O}^{\mathsf{check}}(\boldsymbol{v})$

if $\boldsymbol{v}=\boldsymbol{f}(0)$ **then**

  if $|\mathsf{Cor}\cup S^*|<t$ **then**

    $\mathcal{A}$ wins, i.e., game outputs 1.

  else return 1

return 0

Figure 4: The game in the Statistical Core Lemma (Lemma 5.1) for an adversary $\mathcal{A}$.

Lemma 5.1 (Statistical Core Lemma). *For all adversaries $\mathcal{A}$, all $\ell,Q\in\mathbb{N}$, and all $0<t\leq n<p\in\mathbb{N}$, where p is prime, it holds that*

$$\Pr[\mathsf{GameStat}_{\mathcal{A}}(n,t,p,\ell,Q)=1]\leq(n+1)\cdot q_{\mathsf{check}}\cdot\frac{\binom{n}{t-1}}{p^\ell},$$

where GameStat *is defined in Figure 4, the randomness is taken over the choice of $\boldsymbol{f}$ and the random coins of $\mathcal{A}$, and q_{check} bounds the total number of queries made to $\mathcal{O}^{\mathsf{check}}$ by $\mathcal{A}$ directly and through ρ_k invocations.*

Proof of Lemma 5.1. Fix any $\mathcal{A}$, ℓ, Q, q_{check}, t,n,p meeting the requirements of the lemma. We prove the lemma by analyzing two mutually exclusive, but collectively exhaustive, winning conditions: WinSelf denotes the event that the adversary wins via an $\mathcal{O}^{\mathsf{check}}$ query made by the adversary itself, and not a motivation function. Conversely, WinMot denotes the event that the adversary wins via an $\mathcal{O}^{\mathsf{check}}$ query issued by a motivation function. We will next analyze $\Pr[\mathsf{WinSelf}]$ and $\Pr[\mathsf{WinMot}]$ separately and then apply the law of total probability. We start by bounding WinSelf.

Claim 5.2 $\Pr[\mathsf{WinSelf}]\leq q_{\mathsf{check}}\cdot\frac{\binom{n}{t-1}}{p^\ell}$.

Proof of Claim 5.2. Let us first fix the random coins coins given to the adversary, and assume that the entire coins are input only to the initial call to the adversary, and all subsequent calls are deterministic already. This is without loss of generality, as an adversary can always store the coins within st. We next show that

$$\Pr[\mathsf{WinSelf}\mid\mathsf{coins}]\leq q_{\mathsf{check}}\cdot\frac{\binom{n}{t-1}}{p^\ell}.$$

The bound on the claim then follows from the law of total probability.

First, observe that once coins are fixed, the values Cor and $(\boldsymbol{z}_i)_{i\in\mathsf{Cor}}$ are fixed, and the only remaining randomness is in the choice of the polynomial vector $\boldsymbol{f}\in\mathbb{Z}_p[X]^\ell_{\deg<t}$, i.e., once $\boldsymbol{f}$ is fixed as well, the output of the game is uniquely determined. Denote $c:=|\mathsf{Cor}|$. We can partition the randomness space into the set of winning vectors of polynomials

$$\mathcal{W}=\left\{\boldsymbol{f}\in\mathbb{Z}_p[X]^\ell_{\deg<t}\mid\forall i\in\mathsf{Cor}\colon\boldsymbol{f}(i)=\boldsymbol{z}_i\wedge\mathsf{WinSelf}_{\mathcal{A}}(n,t,p,\ell,Q;\mathsf{coins},\boldsymbol{f})=1\right\}$$

and the set of losing vectors of polynomials

$$\mathcal{L} = \left\{ \boldsymbol{f} \in \mathbb{Z}_p[X]^{\ell}_{\deg<t} \mid \forall i \in \mathsf{Cor}\colon \boldsymbol{f}(i) = \boldsymbol{z}_i \wedge \mathsf{WinSelf}_{\mathcal{A}}(n,t,p,\ell,Q;\mathsf{coins},\boldsymbol{f}) = 0 \right\}.$$

As a result, given that the vectors of polynomials are chosen uniformly at random, we get that

$$\Pr[\mathsf{WinSelf}_{\mathcal{A}}(n,t,p,\ell,Q) = 1 \mid \mathsf{coins}] = \frac{|\mathcal{W}|}{|\mathcal{W}| + |\mathcal{L}|}.$$

We will now bound the size of the numerator and compute the size of the denominator.

Claim 5.3 $|\mathcal{W}| \leq \binom{n}{t-1} \cdot p^{\ell\cdot(t-1-c)} \cdot q_{\mathsf{check}}$.

Proof of Claim 5.3. We will prove this by giving an injective encoding of $\mathcal{W}$ and then bounding the size of its co-domain. For each $\boldsymbol{f} \in \mathcal{W}$ and each $k \in [Q]$, let $S^{(k)}_{\boldsymbol{f}}$ be the values of $S^{(k)}$, i.e., the set of evaluation points selected in round k, when the game is run for $\mathcal{A}$ on $\boldsymbol{f}$ and coins, and let $S^*_{\boldsymbol{f}}$ be the set S^*, i.e., the set of all evaluation points the adversary has seen when the winning query is made.

Now, we define the following encoding Enc for all $\boldsymbol{f} \in \mathcal{W}$:[24]

$$\mathsf{Enc}\colon \mathcal{W} \to \binom{[n]}{t-1} \times (\mathbb{Z}_p^{\ell})^{t-1-c} \times [q_{\mathsf{check}}], \quad \mathsf{Enc}(\boldsymbol{f}) = \left(P_{\boldsymbol{f}}, (\boldsymbol{v}_i)_{i\in P_{\boldsymbol{f}}\setminus \mathsf{Cor}}, \mathsf{idx}_{\boldsymbol{f}}\right),$$

where $P_{\boldsymbol{f}}$ is a canonical completion of $S^*_{\boldsymbol{f}} \cup \mathsf{Cor}$ of size $t-1$, i.e. $P_{\boldsymbol{f}}$ is the lexicographically smallest set such that $P_{\boldsymbol{f}} \subseteq [n]$, $S^*_{\boldsymbol{f}} \cup \mathsf{Cor} \subseteq P_{\boldsymbol{f}}$ and $|P_{\boldsymbol{f}}| = t-1$, $\boldsymbol{v}_i = \boldsymbol{f}(i) \in \mathbb{Z}_p^{\ell}$ for all $i \in P_{\boldsymbol{f}} \setminus \mathsf{Cor}$, and $\mathsf{idx}_{\boldsymbol{f}} \in [q_{\mathsf{check}}]$ denotes the index of the $\mathcal{O}^{\mathsf{check}}$ query that triggers $\mathsf{WinSelf}$, counting among only queries made by $\mathcal{A}$ directly.[25] Clearly, Enc is well-defined, i.e., for all $\boldsymbol{f} \in \mathcal{W}$ there exists such a $P_{\boldsymbol{f}}$, $(\boldsymbol{v}_i)_{i\in P_{\boldsymbol{f}}}$, and $\mathsf{idx}_{\boldsymbol{f}}$.

The crux is to argue that Enc is *injective*, i.e., if $\mathsf{Enc}(\boldsymbol{f}) = \mathsf{Enc}(\boldsymbol{f}')$ for $\boldsymbol{f}, \boldsymbol{f}' \in \mathcal{W}$, then it holds that $\boldsymbol{f} = \boldsymbol{f}'$: Let $P_{\boldsymbol{f}} = P_{\boldsymbol{f}'} =: P$ and $\mathsf{idx}_{\boldsymbol{f}} = \mathsf{idx}_{\boldsymbol{f}'} =: \mathsf{idx}$. Then, by the definition of the encoding, it holds that $\boldsymbol{f}(i) = \boldsymbol{f}'(i)$ for all $i \in P$. To see this, observe that $\boldsymbol{f}(i) = \boldsymbol{f}'(i) = \boldsymbol{z}_i$ for $i \in \mathsf{Cor}$ and $\boldsymbol{f}(i) = \boldsymbol{f}'(i) = \boldsymbol{v}_i$ for all $i \in P \setminus \mathsf{Cor}$. We denote $\boldsymbol{q}_i := \boldsymbol{f}(i) = \boldsymbol{f}'(i)$ for all $i \in P$. Let $ctr_{\boldsymbol{f}}$ and $ctr_{\boldsymbol{f}'}$ be counters, accumulating the number of queries made by $\mathcal{A}$ directly to $\mathcal{O}^{\mathsf{check}}$, when running with $\boldsymbol{f}$ and $\boldsymbol{f}'$, respectively. Denote with $ctr^{(k)}_{\boldsymbol{f}}$ and $ctr^{(k)}_{\boldsymbol{f}'}$ their respective state at the end of round k.

We show by induction over $k = 0, 1, \ldots, Q'$ that:

(i) the adversary's state st_k after round k is the same on $\boldsymbol{f}$ and on $\boldsymbol{f}'$,

(ii) $S^{(k)}_{\boldsymbol{f}} = S^{(k)}_{\boldsymbol{f}'} \subseteq P$,

(iii) $(\boldsymbol{f}(i))_{i\in S^{(k)}_{\boldsymbol{f}}} = (\boldsymbol{f}'(i))_{i\in S^{(k)}_{\boldsymbol{f}'}}$.

(iv) $ctr^{(k)}_{\boldsymbol{f}} = ctr^{(k)}_{\boldsymbol{f}'}$

Here, Q' is the iteration where the idx'th query is issued by $\mathcal{A}$ directly to $\mathcal{O}^{\mathsf{check}}$. If the query is issued by the adversary invocation after the loop, then $Q' = Q$. Our induction stops prior to the idx'th direct query to $\mathcal{O}^{\mathsf{check}}$.

Base case ($k = 0$): We have $\mathsf{st}_0 = \mathsf{st}$ on both runs by definition and because we fixed the random coins, and $S^{(0)} = \emptyset$ and $ctr^{(0)}_{\boldsymbol{f}} = ctr^{(0)}_{\boldsymbol{f}'} = 0$.

Inductive step: assume (i)–(iv) hold up to round $k-1$. Then the round-k adversary call receives identical inputs $(\mathsf{st}_{k-1}, S^{(k-1)}_{\boldsymbol{f}}, (\boldsymbol{f}(i))_{i\in S^{(k-1)}_{\boldsymbol{f}}})$ on $\boldsymbol{f}$ and on $\boldsymbol{f}'$. Whenever the adversary directly queries $\mathcal{O}^{\mathsf{check}}$ prior to the idx'th query, the oracle returns 0 by the definition of the encoding,[26] and both $ctr_{\boldsymbol{f}}$ and $ctr_{\boldsymbol{f}'}$ are incremented. Since either counter only counts queries made directly by $\mathcal{A}$, and thus is not affected by

[24] Intuitively, this is the smallest possible encoding of winning functions that allows us to simulate the run of an adversary given coins, conditioned on the fact that it triggers $\mathsf{WinSelf}$.

[25] Since q_{check} bounds the total number of $\mathcal{O}^{\mathsf{check}}$ queries, it also bounds the number of queries made by $\mathcal{A}$ directly.

[26] Here, we use the fact that the budget $|\mathsf{Cor} \cup S^*|$ is non-decreasing throughout the game. Since $\boldsymbol{f} \in \mathcal{W}$ it must hold that the idx'th query triggers $\mathsf{WinSelf}$, and thus the budget is $< t$ throughout the entire run. Any earlier query on $\boldsymbol{f}(0)$ would have already triggered $\mathsf{WinSelf}$, which contradicts the fact that idx is the winning query. The same applies to $\boldsymbol{f}'$.

potential queries made by motivation functions, this implies (iv). Since everything is deterministic, it produces the same (ρ_k, st_k), giving (i).

For (ii), we analyze two cases. If $|\mathsf{Cor} \cup \bigcup_{j=1}^{k-1} S^{(j)}| \geq t-1$, then it trivially holds that $S_{\boldsymbol{f}}^{(k)} = S_{\boldsymbol{f}'}^{(k)} = \emptyset$, since no motivation functions are run. Otherwise, the fact that $\boldsymbol{f} \in \mathcal{W}$ gives $S_{\boldsymbol{f}}^{(k)} \subseteq S_{\boldsymbol{f}}^{*} \subseteq P_{\boldsymbol{f}} = P$, and we get that[27]

$$S_{\boldsymbol{f}}^{(k)} = \{i \in P \colon \rho_k^{\mathcal{O}^{\mathsf{check}}}(i, \boldsymbol{f}(i)) = 1\} = \{i \in P \colon \rho_k^{\mathcal{O}^{\mathsf{check}}}(i, \boldsymbol{q}_i) = 1\} = \{i \in P \colon \rho_k^{\mathcal{O}^{\mathsf{check}}}(i, \boldsymbol{f}'(i)) = 1\} = S_{\boldsymbol{f}'}^{(k)},$$

where we use the fact that due to $\boldsymbol{f}, \boldsymbol{f}' \in \mathcal{W}$, the $\mathcal{O}^{\mathsf{check}}$ oracle always returns 0 to the motivation functions because motivation functions are only executed at budget $\leq t-2$. Thus, if $\mathcal{O}^{\mathsf{check}} = 1$ is ever triggered by a motivation function, this would terminate $\mathsf{GameStat}$ without causing $\mathsf{WinSelf}$. The last equality uses the symmetric inclusion $S_{\boldsymbol{f}'}^{(k)} \subseteq P$ (from $\boldsymbol{f}' \in \mathcal{W}$). Then (iii) follows from $S_{\boldsymbol{f}}^{(k)} \subseteq P$ and $\boldsymbol{f}|_P = \boldsymbol{f}'|_P$.

By the above induction, the input $\boldsymbol{v}$ in the idx'th query to $\mathcal{O}^{\mathsf{check}}$ is the same for $\boldsymbol{f}$ and $\boldsymbol{f}'$, i.e., $\boldsymbol{v}_{\boldsymbol{f}} = \boldsymbol{v}_{\boldsymbol{f}'} := \boldsymbol{v}$. Since both runs are winning, it must hold that

$$\boldsymbol{f}(0) = \boldsymbol{v}_{\boldsymbol{f}} = \boldsymbol{v}_{\boldsymbol{f}'} = \boldsymbol{f}'(0).$$

Now, since for all $j \in [\ell]$ the component polynomials f_j and f'_j are of degree at most $t-1$ and agree on t points (namely on $P \cup \{0\}$), they must be the same, i.e., $f_j = f'_j$ for all $j \in [\ell]$, which proves that Enc is injective.

Bounding the size of the co-domain of Enc, there are at most $\binom{n}{t-1} \cdot p^{\ell(t-1-c)} \cdot q_{\mathsf{check}}$ possible encodings, and thus we get the claim, by injectivity. ∎

Claim 5.4 We have $|\mathcal{W}| + |\mathcal{L}| = p^{\ell \cdot (t-c)}$.

Proof of Claim 5.4. $|\mathcal{W}| + |\mathcal{L}|$ is equal to the number of ℓ-tuples of degree $< t$ polynomials $\boldsymbol{f}$, where $\boldsymbol{f}(i) = \boldsymbol{z}_i$ for all $i \in \mathsf{Cor}$, which is $p^{\ell \cdot (t-c)}$. ∎

Concluding for Claim 5.2. We get that

$$\Pr[\mathsf{WinSelf}_{\mathcal{A}}(n, t, p, \ell, Q; \mathsf{coins}) = 1 \mid \mathsf{coins}] = \frac{|\mathcal{W}|}{|\mathcal{W}| + |\mathcal{L}|} \leq \frac{\binom{n}{t-1} \cdot p^{\ell \cdot (t-1-c)} \cdot q_{\mathsf{check}}}{p^{\ell \cdot (t-c)}} = q_{\mathsf{check}} \cdot \frac{\binom{n}{t-1}}{p^{\ell}}.$$

∎

Next, we continue with the second part, i.e., bounding $\Pr[\mathsf{WinMot}]$.

Claim 5.5 For all adversaries $\mathcal{A}$ there exists a reduction $\mathcal{B}$ such that $\Pr[\mathsf{WinMot}_{\mathcal{A}}] \leq n \cdot \Pr[\mathsf{WinSelf}_{\mathcal{B}}]$.

Proof of Claim 5.5. We prove the claim by reduction to the game induced by the $\mathsf{WinSelf}$ winning condition. Observe that whenever WinMot happens, the winning query must be made by a motivation function that is evaluated on some party. For all $u \in [n]$, let us define WinMot_u as the event that WinMot happens through a motivation function evaluated for party u. Note that the WinMot_u are mutually exclusive and collectively cover WinMot. Hence, we get that

$$\Pr[\mathsf{WinMot}_{\mathcal{A}}] = \sum_{u \in [n]} \Pr[\mathsf{WinMot}_{u,\mathcal{A}}].$$

Next, we will show that for all fixed $u \in [n]$ there exists a reduction $\mathcal{B}_u$ such that

$$\Pr[\mathsf{WinMot}_{u,\mathcal{A}}] \leq \Pr[\mathsf{WinSelf}_{\mathcal{B}_u}].$$

The reduction $\mathcal{B}_u$ initially receives the parameters (n, t, p, ℓ, Q) and forwards them to $\mathcal{A}$, receiving $(\mathsf{Cor}, (\boldsymbol{z}_i)_{i \in \mathsf{Cor}})$. If $u \in \mathsf{Cor}$, $\mathcal{B}_u$ sets $\mathsf{Cor}' \leftarrow \mathsf{Cor}$.

[27] The actual definition of $S_{\boldsymbol{f}}^{(k)}$ whenever $|\mathsf{Cor} \cup \bigcup_{j=1}^{k-1} S^{(j)}| < t-1$ is $S_{\boldsymbol{f}}^{(k)} = \{i \in [n] \colon \rho_k^{\mathcal{O}^{\mathsf{check}}}(i, \boldsymbol{f}(i)) = 1\}$, but as we know that $S_{\boldsymbol{f}}^{(k)} \subseteq P$, we know that $S_{\boldsymbol{f}}^{(k)} = \{i \in P \colon \rho_k^{\mathcal{O}^{\mathsf{check}}}(i, \boldsymbol{f}(i)) = 1\}$. The same holds for $S_{\boldsymbol{f}'}^{(k)}$.

If, however, the party u is not yet corrupted by $\mathcal{A}$, $\mathcal{B}_u$ sets $\mathsf{Cor}' \leftarrow \mathsf{Cor} \cup \{u\}$, and samples $z_u \xleftarrow{\$} \mathbb{Z}_p^\ell$, i.e., it additionally corrupts party u. If this causes $|\mathsf{Cor}'| \geq t$, then $\mathcal{B}_u$ aborts. Note that in this case, it must hold that $|\mathsf{Cor}| \geq t-1$, and thus $\mathsf{GameStat}$ would not execute any motivation functions, meaning WinMot_u is vacuous. Sampling $\boldsymbol{z}_u \xleftarrow{\$} \mathbb{Z}_p^\ell$ is consistent, as the $\mathcal{A}$ would expect the $\mathsf{GameStat}$ challenger to sample $\boldsymbol{f}(u) \xleftarrow{\$} \mathbb{Z}_p^\ell$ if $u \notin \mathsf{Cor}$ as part of sampling a random vector of polynomials $\boldsymbol{f}$ with degree $< t$ consistent with $(\boldsymbol{z}_i)_{i\in\mathsf{Cor}}$. It then forwards $(\mathsf{Cor}', (\boldsymbol{z}_i)_{i\in\mathsf{Cor}'})$ to its challenger.

From this point, $\mathcal{B}_u$ continues to forward messages between $\mathcal{A}$ and its challenger with one exception. In each round k where $|\mathsf{Cor} \cup S^*_{\mathcal{A}}| \leq t-2$, after $\mathcal{A}$ outputs ρ_k, $\mathcal{B}_u$ simulates execution of ρ_k for party u first.[28] In particular, it runs $\rho_k^{\mathcal{O}^{\mathsf{check}}}(u, \boldsymbol{z}_u)$, forwarding $\mathcal{O}^{\mathsf{check}}$ queries to its challenger. $\mathcal{B}_u$ saves the outputs of $\rho_k^{\mathcal{O}^{\mathsf{check}}}(u, \boldsymbol{z}_u)$ in a bit b_k. If b_k is 1, i.e., the simulated motivation function returned 1, $\mathcal{B}_u$ pre-sets $S^{(k)} \leftarrow \{u\}$. Otherwise, $\mathcal{B}_u$ initializes $S^{(k)} \leftarrow \emptyset$ normally. Then, $\mathcal{B}_u$ constructs $\rho'^{\mathcal{O}^{\mathsf{check}}}_k$, which is identical to $\rho_k^{\mathcal{O}^{\mathsf{check}}}$, except that on input $(u, \boldsymbol{z}_u)$, $\rho'^{\mathcal{O}^{\mathsf{check}}}_k$ immediately outputs b_k. Note that, as a result, $\rho'^{\mathcal{O}^{\mathsf{check}}}_k(u, \boldsymbol{z}_u)$ does not make any queries to $\mathcal{O}^{\mathsf{check}}$. Finally, $\mathcal{B}_u$ forwards ρ'_k to its challenger.

Analysis. First, we argue that $\mathcal{B}_u$'s simulation of the experiment induced by WinMot_u is perfect for sufficiently long to capture every occurrence of WinMot_u. To see this, couple a run of $\mathcal{A}$ within $\mathsf{GameStat}$ with the corresponding run of $\mathcal{A}$ within $\mathcal{B}_u$ by coupling $\boldsymbol{z}_u$, as sampled by $\mathcal{B}_u$ with $\boldsymbol{f}(u)$ and sharing all remaining random choices. This coupling preserves the distribution of $\mathsf{GameStat}$, because if $|\mathsf{Cor} \cup \{u\}| \leq t-1$, the value $\boldsymbol{f}(u)$ in the real game is uniform over $\mathbb{Z}_p^\ell$ and independent of $(\boldsymbol{z}_i)_{i\in\mathsf{Cor}}$, and $\mathcal{B}_u$ samples $\boldsymbol{z}_u$ in the same manner. If $|\mathsf{Cor} \cup \{u\}| \geq t$, then WinMot_u cannot happen anyway due to budget monotonicity.

Under this coupling, the two runs proceed in an identical manner, with only one potential exception: While $u \notin \mathsf{Cor} \cup S^*_{\mathcal{A}}$, the budget tracked by $\mathcal{B}_u$'s challenger, i.e., $|\mathsf{Cor}' \cup S^*_{\mathcal{B}_u}|$ exceeds $\mathcal{A}$'s budget by exactly one:

$$|\mathsf{Cor} \cup S^*_{\mathcal{A}}| + 1 = |\mathsf{Cor}' \cup S^*_{\mathcal{B}_u}|.$$

Thus, in a round where $|\mathsf{Cor} \cup S^*_{\mathcal{A}}| = t-2$, the challenger of $\mathcal{B}_u$ does not execute ρ'_k, whereas the real game would evaluate ρ_k for all parties. In such a round, however, the evaluation for party u itself is still simulated accurately, since $\mathcal{B}_u$ runs $\rho_k(u, \boldsymbol{z}_u)$ locally and forwards any $\mathcal{O}^{\mathsf{check}}$ queries. If $\rho_k(u, \boldsymbol{z}_u)$ outputs 1, then u is added to $S^*_{\mathcal{A}}$, and from then on, both budgets and runs agree again. Hence, the two runs can only diverge if, while $|\mathsf{Cor} \cup S^*_{\mathcal{A}}| = t-2$, $\rho_k(i, \boldsymbol{f}(i))$ outputs 1 for some $i \neq u$. We denote the first such round as the *divergent round*.

The key observation is that $\mathsf{WinMot}_{u,\mathcal{A}}$ can never occur after the divergent round. This is because in the divergent round, the budget $|\mathsf{Cor} \cup S^*_{\mathcal{A}}|$ rises from $t-2$ to at least $t-1$. At this point, $\mathsf{GameStat}$ no longer executes motivation functions, which means $\mathsf{WinMot}_{u,\mathcal{A}}$ can no longer happen. Note that in the divergent round itself, $\mathcal{B}_u$ performs evaluation at u locally with the correct input, before any divergence becomes visible to $\mathcal{A}$.

Let us now consider the case that $\mathsf{WinMot}_{u,\mathcal{A}}$ occurs, which means that the winning $\mathcal{O}^{\mathsf{check}}$ query is issued during an evaluation of some ρ_k at $(u, \boldsymbol{f}(u))$ while $|\mathsf{Cor} \cup S^*_{\mathcal{A}}| \leq t-2$. By the coupling argument above, $\mathcal{B}_u$ simulates the evaluation locally and issues the same query *directly* to the $\mathcal{O}^{\mathsf{check}}$ oracle of its challenger. At this point, $\mathcal{B}_u$'s budget satisfies

$$|\mathsf{Cor}' \cup S^*_{\mathcal{B}_u}| \leq |\mathsf{Cor} \cup S^*_{\mathcal{A}}| + 1 \leq t-1,$$

which means that $\mathcal{B}_u$ triggers $\mathsf{WinSelf}$ on the same query.

Clearly, $\mathcal{B}_u$ makes at most the same number of total direct and indirect queries as $\mathcal{A}$, i.e., q_{check} also bounds the number of queries made by $\mathcal{B}_u$ and we can bound

$$\Pr[\mathsf{WinMot}_{u,\mathcal{A}}] \leq \Pr[\mathsf{WinSelf}_{\mathcal{B}_u}]$$

for all $u \in [n]$, yielding the bound in the claim. ∎

Finishing the proof of Lemma 5.1. Finally, applying Claims 5.2 and 5.5 alongside the law of total

[28] Once $|\mathsf{Cor} \cup S^*_{\mathcal{A}}| \geq t-1$, WinMot_u is vacuous because motivation functions are no longer executed.

probability, we get that

$$\begin{aligned}\Pr[\mathsf{GameStat}_{\mathcal{A}}(n,t,p,\ell,Q)=1] &= \Pr[\mathsf{WinSelf}_{\mathcal{A}}] + \Pr[\mathsf{WinMot}_{\mathcal{A}}] \\ &\leq \Pr[\mathsf{WinSelf}_{\mathcal{A}}] + n \cdot \Pr[\mathsf{WinSelf}_{\mathcal{B}}] \\ &\leq (n+1) \cdot q_{\mathsf{check}} \cdot \frac{\binom{n}{t-1}}{p^{\ell}}.\end{aligned}$$

■

5.3 Construction

$\mathsf{Setup}(1^\lambda)$

$E \coloneqq (\mathbb{G}, g, p) \xleftarrow{\$} \mathcal{G}(1^\lambda)$
$\mathsf{pp}_{\mathsf{PS}} \xleftarrow{\$} \mathsf{PS.Setup}(1^\lambda)$
return $\mathsf{pp} \coloneqq (E, \mathsf{pp}_{\mathsf{PS}})$

$\mathsf{KGen}(1^\lambda, n, t)$

for $k \in [v]$ **do**
  $x_k \xleftarrow{\$} \mathbb{Z}_p$
  $\mathsf{pk}_k \leftarrow g^{x_k}$
  $(x_{i,k})_{i\in[n]} \xleftarrow{\$} \mathsf{Share}(x_k, n, t)$
$(y_i)_{i\in[n]} \xleftarrow{\$} \mathsf{Share}(0, n, t)$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \ldots, \mathsf{pk}_v)$
$\forall i \in [n] \colon \mathsf{sk}_i \leftarrow (x_{i,1}, \ldots, x_{i,v}, y_i)$
return $(\mathsf{pk}, (\mathsf{sk}_i)_{i\in[n]})$

$\mathsf{Enc}(\mathsf{pk}, m)$

$(r_1, \ldots, r_u) \xleftarrow{\$} \mathbb{Z}_p^u$
$c_1 \coloneqq (c_{1,1}, \ldots, c_{1,u}) \leftarrow (g^{r_1}, \ldots, g^{r_u})$
$c_2 \leftarrow \mathsf{H}_{\mathsf{mask}}\big((\mathsf{pk}_k^{r_j})_{j\in[u],k\in[v]}\big) \oplus m$
$\pi \xleftarrow{\$} \mathsf{PS.Prove}((E, c_1, c_2), (r_1, \ldots, r_u))$
return $c \coloneqq (c_1, c_2, \pi)$

$\mathsf{PDec}(\mathsf{sk}_i, c)$

if $\mathsf{PS.Verify}((E, c_1, c_2), \pi) = 0$ **then**
  return $\perp$
for $(j,k) \in [u] \times [v]$ **do**
  $d_{i,j,k} \leftarrow c_{1,j}^{x_{i,k}} \cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c)^{y_i}$
return $d_i \coloneqq (d_{i,j,k})_{j\in[u],k\in[v]}$

$\mathsf{Combine}(\mathsf{pk}, S, (d_i)_{i\in S}, c)$

$$m \leftarrow c_2 \oplus \mathsf{H}_{\mathsf{mask}}\left(\left(\prod\nolimits_{i\in S} d_{i,j,k}^{\lambda_{i,S}^{(0)}}\right)_{j\in[u],k\in[v]}\right)$$

return m

Figure 5: The construction of Π_{ZEG}, parameterized with the ciphertext randomness multiplicity u and key multiplicity v. Recall that both can be functions in λ, n, and t.

Construction 1 (ElGamal with Zero-Sharing Π_{ZEG})**.** Let μ denote the message length and let $\mathsf{H}_{\mathsf{PDec}} \colon \{0,1\}^* \to \mathbb{G}$ and $\mathsf{H}_{\mathsf{mask}} \colon \{0,1\}^* \to \{0,1\}^\mu$ be hash functions, modeled as random oracles. The scheme is parameterized with $u, v \in \mathbb{N}$. Both can be functions in λ, n, and t. Let PS be a proof of knowledge system for the relation

$$\mathsf{R} = \left\{ (\chi, \omega) \colon \begin{array}{c} \chi = ((\mathbb{G}, g, p), c_1, c_2) \wedge \omega = (\omega_1, \ldots, \omega_u) \wedge \\ c_1 = (c_{1,1}, \ldots, c_{1,u}) \wedge \\ \forall j \in [u] \colon c_{1,j} = g^{\omega_j} \end{array} \right\}.$$

The algorithms of Π_{ZEG} can be found in Figure 5.

Correctness holds because, given an authorized set S, for all $(j,k) \in [u] \times [v]$ and honestly generated keys, ciphertexts, and partial decryptions, we get that

$$\begin{aligned}\prod_{i\in S} d_{i,j,k}^{\lambda_{i,S}^{(0)}} &= \prod_{i\in S} c_{1,j}^{x_{i,k}\lambda_{i,S}^{(0)}} \cdot \prod_{i\in S} \mathsf{H}_{\mathsf{PDec}}(j,k,c)^{y_i\lambda_{i,S}^{(0)}} = \prod_{i\in S} g^{r_j x_{i,k}\lambda_{i,S}^{(0)}} \cdot \prod_{i\in S} \mathsf{H}_{\mathsf{PDec}}(j,k,c)^{y_i\lambda_{i,S}^{(0)}} \\ &= g^{r_j \sum_{i\in S} x_{i,k}\lambda_{i,S}^{(0)}} \cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c)^{\sum_{i\in S} y_i\lambda_{i,S}^{(0)}} = g^{r_j x_k} \cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c)^0 = \mathsf{pk}_k^{r_j}.\end{aligned}$$

Remark 5.6 (Proof System Instantiation). We can instantiate the proof system PS with Schnorr proofs of knowledge of exponent, which are straight-line simulation-extractable in the algebraic group model

(AGM) [47] and random oracle model (see, e.g., [12] for a proof). Alternatively, one can instead apply the Fischlin transform [45] to the underlying Σ-protocol to avoid using the AGM at the cost of a larger proof size. The same applies to the construction Π_{Pair} introduced in Section 5.4. Additionally, we show in Section 5.5 how one can avoid simulation-extractability entirely at the cost of a small increase in ciphertext size.

Theorem 5.7 Π_{ZEG} *(Construction 1) is* IND-IMC-CCA*-secure in the random oracle model for* $q = Q = (0, 0, \infty, \infty, \infty)$*, i.e., static corruptions and high-threshold access to the partial decryption oracle, and* $\mathcal{F}_0^{\mathsf{PDec}} = \mathcal{F}_1^{\mathsf{PDec}} = \mathcal{F}^{\mathsf{chal}} = \mathcal{F}_{\mathsf{no\text{-}sk}}$*, i.e., motivation functions do not receive secret key shares as input, if* $\log(p^{u \cdot v}) \geq n + \lambda$ *and* $\log(p) \geq \lambda$*,* DDH *is hard for* $\mathcal{G}$*, and* PS *is simulation-extractable. In more detail, for all PPT adversaries* $\mathcal{A}$*, there exist PPT reductions* $\mathcal{B}$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}IMC\text{-}CCA}}_{\mathcal{A},\Pi_{\mathsf{ZEG}}}(1^\lambda, n, t, \mathcal{F}, q, Q) \leq 2 \cdot \mathsf{Adv}^{\mathsf{SimExt}}_{\mathcal{B},(\mathsf{PS},\mathsf{Sim})}(1^\lambda) + tuv \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda) \\ + \frac{q_{\mathsf{H}_{\mathsf{PDec}}}}{p^u} + \frac{uv}{p} + \frac{q_{\mathsf{H}_{\mathsf{mask}}}}{p^{uv}} + (n+1) q_{\mathsf{H}_{\mathsf{mask}}} \frac{\binom{n}{t-1}}{p^{uv}},\end{aligned}$$

where $q_{\mathsf{H}_{\mathsf{mask}}}$ *and* $q_{\mathsf{H}_{\mathsf{PDec}}}$ *bound the total number of queries made by* $\mathcal{A}$ *to the random oracles (directly or through motivation functions), respectively, and the running time of* $\mathcal{B}$ *is at most that of* $\mathcal{A}$ *plus an additive overhead polynomial in* λ *and* n*.*

Corollary 5.8 *Theorem 5.7 implies the following instantiations:*

- $u = v = 1$*: The group must be large, i.e.,* $\log(p) \geq n + \lambda$*, thus all parameters scale as* $O(n)$*.*
- $u \cdot v = 1 + n/\lambda$*: We can use standard groups with* $\log(p) \approx \lambda$*, ciphertext size scales as* $O(u)$ *and public key and share sizes scale as* $O(v)$*. In particular, this implies*
  - $u = 1 + n/\lambda, v = 1$*:* $O(n)$ *ciphertext, but* $O(1)$ *public key and share size.*
  - $u = 1, v = 1 + n/\lambda$*:* $O(1)$ *ciphertext, but* $O(n)$ *public key and share size.*
  - $v = u = \sqrt{1 + n/\lambda}$*: Ciphertext, public key, and share sizes are sublinear as* $O(\sqrt{n})$*.*

We present a concrete analysis of the tradeoff between public key size and ciphertext overhead in Section 7.

Proof of Theorem 5.7. We prove IND-IMC-CCA security through a sequence of game hops. Let Game_0 be the game $\mathsf{IND\text{-}IMC\text{-}CCA}^{\Pi_{\mathsf{ZEG}}}(1^\lambda, n, t, \mathcal{F}, q, Q)$ instantiated with $\mathcal{F} = (\emptyset, \emptyset, \mathcal{F}_{\mathsf{no\text{-}sk}}, \mathcal{F}_{\mathsf{no\text{-}sk}}, \mathcal{F}_{\mathsf{no\text{-}sk}})$, $q = Q = (0, 0, \infty, \infty, \infty)$. Henceforth, we omit the inputs $n, t, \mathcal{F}$, q, and Q to security games to improve readability.

Game_0 (Real game): We recall Game_0 for completeness, and to fix some notation:

1. The adversary outputs a set of static corruptions $\mathsf{Cor}_{\mathsf{sk}}$. W.l.o.g., we assume that $\mathsf{Cor}_{\mathsf{sk}} \subseteq [n]$ and $|\mathsf{Cor}_{\mathsf{sk}}| \leq t - 1$.
2. The game generates the group description E and the proof system parameters $\mathsf{pp}_{\mathsf{PS}}$, as well as the public key $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_v) = (g^{x_1}, \ldots, g^{x_v})$, Shamir sharings $(x_{i,k})_{i \in [n]}$ of all x_k for $k \in [v]$, and the zero-sharing $(y_i)_{i \in [n]}$, respectively. It gives to the adversary $(E, \mathsf{pp}_{\mathsf{PS}}, \mathsf{pk})$, and $\mathsf{sk}_i = ((x_{i,k})_{k \in [v]}, y_i)$ for all $i \in \mathsf{Cor}_{\mathsf{sk}}$. Note that there will be no other corruptions during the game.
3. The adversary can now query a partial decryption oracle $\mathcal{O}^{\mathsf{PDec}}$, which on input S, φ, c runs the partial decryption (including checking the proof contained in c) and then evaluates φ for each $i \in S$, simulating the random oracles, and returns the partial decryption of the parties for which φ returned 1. Note that φ does not take the secret key share as input (because $\mathcal{F}_0^{\mathsf{PDec}} = \mathcal{F}_1^{\mathsf{PDec}} = \mathcal{F}_{\mathsf{no\text{-}sk}}$). Also note that if the proof contained in c does not verify, then the oracle can be simulated without knowledge of any secret key share by returning $\perp$.
4. The adversary outputs two messages m_0, m_1, and the game encrypts m_b under pk, for a random bit b. Recall that this challenge ciphertext c^* consists of

$$c_1^* = (g^{r_1}, \ldots, g^{r_u}),\ c_2^* = \mathsf{H}_{\mathsf{mask}}\big((\mathsf{pk}_k^{r_j})_{j \in [u], k \in [v]}\big) \oplus m_b,\ \pi^*,$$

where π^* is a proof, attesting knowledge of $(r_1, \ldots, r_u)$.

5. The adversary gets c^*, and it also gets access to a partial decryption oracle $\mathcal{O}^{\mathsf{PDec}}$ as above.

6. Further, it gets access to a partial decryption oracle for the challenge ciphertext, which works in the same way, except that for each query (S, φ), the set of parties S' for which $\varphi(\mathsf{pp}, \mathsf{pk}, i, d_i, c^*) = 1$ is unioned with the initially empty set $\mathsf{Cor}_{\mathsf{pd}}$.

7. Finally, the adversary outputs a guess b' and wins if $b = b'$, and $|\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}| < t$, i.e., not too many partial decryptions were learned on the challenge ciphertext.

8. Throughout execution, the adversary and motivation functions have access to the random oracles $\mathsf{H}_{\mathsf{mask}}$ and $\mathsf{H}_{\mathsf{PDec}}$. We henceforth call queries made by the adversary "direct" queries, and queries made through a motivation function "indirect" queries.

In one of our later proof steps, we will need to replace all $g^{r_j x_k}$ in the challenge ciphertext with random. Our reduction in that step will need to simulate the $\mathcal{O}^{\mathsf{PDec}}$ oracle without knowing all secret key shares, and thus we will first modify $\mathcal{O}^{\mathsf{PDec}}$ to extract the encryption randomness and use it to simulate partial decryptions, and thus be independent of the $(x_{i,1}, \ldots, x_{i,v})$ part of the key shares (Game_1 and Game_2). Further, the same reduction will need to simulate π^* without knowing the encryption randomness of the challenge ciphertext, hence we replace π^* with a simulated proof beforehand (Game_1).

Game_1 (Simulate and Extract NIZKs): In Game_1 we modify how the challenger instantiates the NIZK-PoK system PS. In particular, we make the following modifications, where Sim is the simulation-extractability simulator of PS:

- During Setup, PS is set up as $(\mathsf{pp}_{\mathsf{PS}}, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{Sim.Setup}(1^\lambda)$.
- When generating the challenge ciphertext c^*, instead of computing π^* as $\mathsf{PS.Prove}(\chi, (r_j)_{j\in[u]})$, where $\chi = (E, c_1^*, c_2^*)$, the simulator is used to simulate a proof $(\pi^*, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{Sim.Prove}(\chi, \mathsf{st}_{\mathsf{Sim}})$.
- In $\mathcal{O}_d^{\mathsf{PDec}}(S, \varphi, (c_1, c_2, \pi))$ for $d \in \{0, 1\}$, after verifying the proof π when computing partial decryptions d_i for all $i \in S$, the simulator is used to extract a witness from π as $(\omega, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{Sim.Ext}(\chi, \pi, \mathsf{st}_{\mathsf{Sim}})$, where $\chi = (E, c_1, c_2)$. The challenger aborts if the proof verifies but $(\chi, \omega) \notin \mathsf{R}$, causing the game to output 0. Otherwise, the challenger continues execution of $\mathcal{O}_d^{\mathsf{PDec}}$ normally.

Claim 5.9 For all PPT adversaries $\mathcal{A}$ there exists a PPT reduction $\mathcal{B}$ such that

$$\left|\Pr\left[\mathsf{Game}_{1,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{0,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq 2 \cdot \mathsf{Adv}^{\mathsf{SimExt}}_{\mathcal{B},(\mathsf{PS}_{\mathsf{R}},\mathsf{Sim})}(1^\lambda).$$

Proof of Claim 5.9. The claim follows a straightforward reduction to simulation-extractability of PS. The reduction $\mathcal{B}$ embeds the $\mathsf{pp}_{\mathsf{PS}}$ it receives from its challenger. To compute π^*, it queries the $\mathcal{O}^{\mathsf{Prove}}(\chi, \omega)$ oracle of its challenger using $\chi = (E, c_1^*, c_2^*)$ and $\omega = (r_j)_{j\in[u]}$. In $\mathcal{O}^{\mathsf{PDec}}(S, \varphi, (c_1, c_2, \pi))$, the reduction queries the $\mathcal{O}^{\mathsf{Verify}}(\chi, \pi)$ oracle of its challenger using $\chi = (E, c_1, c_2)$ to verify the proof. If the oracle outputs $\perp$, $\mathcal{B}$ aborts and outputs $b' = 0$ to its challenger. Finally, $\mathcal{B}$ outputs $b' = 1$ whenever the internal adversary $\mathcal{A}$ wins the simulated game, and $b' = 0$ otherwise.

Analysis. Denote with b the internal bit of the SimExt challenger, and recall that if $b = 0$, the SimExt challenger is using the real proof system, and if $b = 1$, the challenger is using the simulator to simulate proofs and extract. Hence, $\mathcal{B}$ simulates Game_b perfectly to $\mathcal{A}$ and we get that

$$\left|\Pr\left[\mathsf{Game}_{1,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{0,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq 2 \cdot \mathsf{Adv}^{\mathsf{SimExt}}_{\mathcal{B},(\mathsf{PS}_{\mathsf{R}},\mathsf{Sim})}(1^\lambda).$$

■

Game_2 (Simulate Non-Challenge Decryptions): In Game_2, we modify how partial decryptions d_i are computed within $\mathcal{O}^{\mathsf{PDec}}$. In particular, we make use of the extracted witness $\omega = (\omega_1, \ldots, \omega_u)$ to compute

$$d_{i,j,k} \leftarrow (g^{x_{i,k}})^{\omega_j} \cdot \mathsf{H}_{\mathsf{PDec}}(j, k, c)^{y_i}$$

for all $i \in [n]$, $j \in [u]$, $k \in [v]$. Note that at this point, $\mathcal{O}^{\mathsf{PDec}}$ no longer uses the secret shares of $(x_1, \ldots, x_v)$, it just needs the exponentiated versions $g^{x_{i,k}}$.

Claim 5.10 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\big[\mathsf{Game}_{2,\mathcal{A}}(1^\lambda) = 1\big] = \Pr\big[\mathsf{Game}_{1,\mathcal{A}}(1^\lambda) = 1\big].$$

Proof of Claim 5.10. Observe that in Game_1, $\mathcal{O}^{\mathsf{PDec}}$ already aborts unless $(\chi, \omega) \in \mathsf{R}$, where $\chi = (E, c_1, c_2)$. In particular, this means that $c_1 = (g^{\omega_1}, \ldots, g^{\omega_u})$. In this case, it holds that

$$c_{1,j}^{x_{i,k}} \cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c)^{y_i} = (g^{\omega_j})^{x_{i,k}} \cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c)^{y_i} = (g^{x_{i,k}})^{\omega_j} \cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c)^{y_i}.$$

■

Our next game hops will modify the $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ oracle such that for all $j \in [u]$ and $k \in [v]$, the discrete logarithms of the partial decryptions on the challenge ciphertext of all parties $i \in [n]$ are points on a random degree $< t$ polynomial $f_{j,k}$ chosen by the challenger to be consistent with $f_{j,k}(0) = r_j x_k$ and the discrete logarithms of the real partial decryptions $f_{j,k}(i) = r_j x_{i,k} + w_{j,k,c^*} y_i$ for all corrupted parties $i \in \mathsf{Cor}_{\mathsf{sk}}$. Here, we assume that $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*) = g^{w_{j,k,c^*}}$, i.e., w_{j,k,c^*} is the discrete logarithm of $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*)$.

Game_3 (No Premature $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*)$ Queries): In Game_3, we add an additional abort condition that can trigger after the challenger has computed c^*: If the adversary has queried $\mathsf{H}_{\mathsf{PDec}}$ on input (j,k,c^*) for any $(j,k) \in [u] \times [v]$ (either directly or indirectly) before it receives c^* from the challenger, Game_3 aborts and terminates with output 0.

Claim 5.11 For all PPT adversaries $\mathcal{A}$, it holds that

$$\big|\Pr\big[\mathsf{Game}_{3,\mathcal{A}}(1^\lambda) = 1\big] - \Pr\big[\mathsf{Game}_{2,\mathcal{A}}(1^\lambda) = 1\big]\big| \leq \frac{q_{\mathsf{H}_{\mathsf{PDec}}}}{p^u},$$

where p is the order of $\mathbb{G}$ and $q_{\mathsf{H}_{\mathsf{PDec}}}$ bounds the total number of direct and indirect queries made to $\mathsf{H}_{\mathsf{PDec}}$.

Proof of Claim 5.11. Observe that $c^* = (c_1^*, c_2^*, \pi^*)$, where $c_1^* = (g^{r_j})_{j\in[u]}$ for a random $(r_j)_{j\in[u]} \xleftarrow{\$} \mathbb{Z}_p$ that is sampled when the challenger generates the challenge ciphertext. Hence, by a union bound, the probability that making at most $q_{\mathsf{H}_{\mathsf{PDec}}}$ queries to $\mathsf{H}_{\mathsf{PDec}}$ triggers the additional abort condition is at most $q_{\mathsf{H}_{\mathsf{PDec}}}/p^u$. If the abort condition is not triggered, Game_2 and Game_3 are identical. ■

Before we can conclude this part of the proof, we need to eliminate one last failure condition that occurs if $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*) = g^0$. The intuition behind this is that, in this case, the $x_{i,k}$-dependent part of partial decryptions is no longer masked, since $(c_{1,j}^*)^{x_{i,k}} \cdot (g^0)^{y_i} = (c_{1,j}^*)^{x_{i,k}}$.

Game_4 (Abort if $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*) = g^0$): In Game_4, the challenger aborts if $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*) = g^0$ for any $(j,k) \in [u] \times [v]$.

Claim 5.12 For all adversaries $\mathcal{A}$ it holds that

$$\big|\Pr\big[\mathsf{Game}_{4,\mathcal{A}}(1^\lambda) = 1\big] - \Pr\big[\mathsf{Game}_{3,\mathcal{A}}(1^\lambda) = 1\big]\big| \leq \frac{u \cdot v}{p}.$$

Proof of Claim 5.12. Since $\mathsf{H}_{\mathsf{PDec}}$ is a random oracle with co-domain $\mathbb{G}$, any given input will map to g^0 with probability $1/p$, and the claim follows from a union bound. ■

Game_5 (Independent Zero-Sharing): In this game hop, we modify how $d_{i,j,k}$ is computed in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. First, note that for all $i \in [n]$, $j \in [u]$ and $k \in [v]$, computation of the partial decryption can be viewed as

$$d_{i,j,k} \leftarrow (c_{1,j}^*)^{x_{i,k}} \cdot h_{i,j,k},$$

where $h_{i,j,k} = \mathsf{H}_{\mathsf{PDec}}(j,k,c^*)^{y_i}$.

We now modify how $h_{i,j,k}$ is computed. In particular, when $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ for the first time the challenger now samples an independent zero-sharing for each $(j,k) \in [u] \times [v]$ that is consistent with y_i for $i \in \mathsf{Cor}_{\mathsf{sk}}$ and uses it to compute $d_{i,j,k}$ for all $i \in [n]$. More formally, let P be a canonical completion of the set $\mathsf{Cor}_{\mathsf{sk}}$ such that $\mathsf{Cor}_{\mathsf{sk}} \subseteq P \subseteq [n]$ and $|P| = t - 1$. For all $(j,k) \in [u] \times [v]$ the challenger generates

$$\widetilde{y_{i,j,k}} \leftarrow \begin{cases} y_i, & \text{if } i \in \mathsf{Cor_{sk}} \\ \xleftarrow{\$} \mathbb{Z}_p, & \text{if } i \in P \setminus \mathsf{Cor_{sk}} \\ \sum_{p \in P} \lambda^{(i)}_{p,P\cup\{0\}} \cdot \widetilde{y_{p,j,k}}, & \text{if } i \in [n] \setminus P \end{cases}.$$

Then, the challenger computes

$$h_{i,j,k} \leftarrow \mathsf{H_{PDec}}(j,k,c^*)^{\widetilde{y_{i,j,k}}},$$

and uses it to compute $d_{i,j,k}$. The same zero-sharings, and thus partial decryptions, are used when simulating subsequent queries to $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$.

Claim 5.13 For all PPT adversaries $\mathcal{A}$ there exists a PPT reduction $\mathcal{B}$ such that

$$\left|\Pr\left[\mathsf{Game}_{5,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{4,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \mathsf{Adv}^{(t-1,u\cdot v)\text{-mDDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda).$$

Proof of Claim 5.13. Let $\mathcal{A}$ be a PPT distinguisher between Game_4 and Game_5 and let $\mathcal{B}$ be a PPT reduction against the $(t-1, u\cdot v)$-mDDH challenger. $\mathcal{B}$ initially receives the $(t-1, u\cdot v)$-mDDH tuple, which we denote by labeling the second component as $(j,k) \in [u] \times [v]$ as follows

$$\left((A_p)_{p\in[t-1]}, (B_{j,k})_{(j,k)\in[u]\times[v]}, (T_{p,j,k})_{p\in[t-1],(j,k)\in[u]\times[v]}\right),$$

where $A_p = g^{\alpha_p}$ for all $p \in [t-1]$, $B_{j,k} = g^{\beta_{j,k}}$ for all $(j,k) \in [u]\times[v]$, and either $T_{p,j,k} = g^{\alpha_p\beta_{j,k}}$ or $T_{p,j,k}$ is a random group element for all $p \in [t-1]$, $(j,k) \in [u]\times[v]$.

$\mathcal{B}$ now simulates either game to $\mathcal{A}$. After $\mathcal{A}$ outputs a set $\mathsf{Cor_{sk}} \subseteq [n]$ such that $|\mathsf{Cor_{sk}}| < t$, $\mathcal{B}$ computes the canonical set P that completes $\mathsf{Cor_{sk}}$ such that $\mathsf{Cor_{sk}} \subseteq P \subseteq [n]$ and $|P| = t-1$. Denote $Q \coloneqq P \setminus \mathsf{Cor_{sk}}$, and let $q\colon Q \to [t-1]$ be any injective mapping.[29] $\mathcal{B}$ continues to simulate such that $g^{y_i} = A_{q(i)}$ for all $i \in Q$, and $\mathsf{H_{PDec}}(j,k,c^*) = B_{j,k}$ for all $(j,k) \in [u]\times[v]$. More formally, $\mathcal{B}$ carries out the following simulation:

Simulating Key Generation. $\mathcal{B}$ simulates shares of $(x_k)_{k\in[v]}$ according to either game, setting $\mathsf{pk} = (g^{x_k})_{k\in[v]}$. For all $i \in \mathsf{Cor_{sk}}$, $\mathcal{B}$ samples $y_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mathsf{sk}_i \leftarrow ((x_{i,k})_{k\in[v]}, y_i)$. It outputs $(\mathsf{pk}, (\mathsf{sk}_i)_{i\in\mathsf{Cor_{sk}}})$ to $\mathcal{A}$. Observe that $\mathcal{B}$ can consistently simulate g^{y_i} for all $i \in [n]$ as follows:

$$g^{y_i} \leftarrow \prod_{p\in\mathsf{Cor_{sk}}} g^{y_p \lambda^{(i)}_{p,P\cup\{0\}}} \cdot \prod_{p\in Q} A_{q(p)}^{\lambda^{(i)}_{p,P\cup\{0\}}}.$$

Simulating Random Oracles. For $\mathsf{H_{mask}}$, $\mathcal{B}$ outputs a random element from $\{0,1\}^\mu$ consistently. Whenever $\mathcal{A}$ directly or indirectly queries $\mathsf{H_{PDec}}$ on input (j,k,c), $\mathcal{B}$ samples $w_{j,k,c} \xleftarrow{\$} \mathbb{Z}_p$ and outputs $g^{w_{j,k,c}}$ consistently.

Simulating $\mathcal{O}_0^{\mathsf{PDec}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$. On input $(S, \varphi, c = (c_1, c_2, \pi))$, when computing the partial decryptions $(d_{i,j,k})_{(j,k)\in[u]\times[v]}$ for all $i \in S$, $\mathcal{B}$ simulates

$$d_{i,j,k} \leftarrow (g^{x_{i,k}})^{\omega_j} \cdot (g^{y_i})^{w_{j,k,c}},$$

where $\omega = (\omega_j)_{j\in[u]}$ denotes the witness extracted from the proof π. This simulation is possible since $\mathcal{B}$ can perfectly simulate g^{y_i} for all $i \in [n]$, as outlined in key generation.

Simulating the Challenge Ciphertext. On input (m_0, m_1) of equal length, $\mathcal{B}$ samples a random bit b and encrypts m_b as mandated by either game, yielding the challenge ciphertext $c^* = (c^*_{1,1}, \ldots, c^*_{1,u}, c^*_2, \pi^*)$. Now, if $\mathcal{A}$ has already queried $\mathsf{H_{PDec}}$ on (j,k,c^*) for any $(j,k) \in [u]\times[v]$, $\mathcal{B}$ aborts the simulation, as mandated by either game, and outputs 0 to its challenger. Otherwise, $\mathcal{B}$ programs $\mathsf{H_{PDec}}(j,k,c^*) \leftarrow B_{j,k}$ for all $(j,k) \in [u]\times[v]$. If $B_{j,k} = g^0$ for any $(j,k) \in [u]\times[v]$, then, as mandated by the condition introduced in the previous game hop, $\mathcal{B}$ aborts and outputs 0 to its challenger.

Simulating $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, $\mathcal{B}$ embeds the $(t-1, u\cdot v)$-mDDH challenge into the partial decryptions. In particular, it computes $h_{i,j,k}$ for all $i \in [n]$ and $(j,k) \in [u]\times[v]$ as follows

$$h_{i,j,k} \leftarrow \begin{cases} \mathsf{H_{PDec}}(j,k,c^*)^{y_i} \text{ for all } i \in \mathsf{Cor_{sk}} \\ T_{q(i),j,k} \text{ for all } i \in Q \\ \prod_{p\in P} h_{p,j,k}^{\lambda^{(i)}_{p,P\cup\{0\}}} \text{ for all remaining } i \in [n] \setminus P. \end{cases}$$

[29] Such a mapping always exists because $|Q| \leq |P| = t-1$.

$\mathcal{B}$ then uses the $h_{i,j,k}$ values to compute partial decryptions and continues the simulation according to either game. Finally, once $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs 1 if $b = b'$ and 0 otherwise. If any assertion fails that is present in both games, $\mathcal{B}$ aborts and outputs 0 to its challenger.

Analysis. Observe that if the $T_{p,j,k}$ elements from the $(t-1, u \cdot v)$-mDDH challenge are *real*, i.e., if $T_{p,j,k} = g^{\alpha_p \beta_{j,k}}$, then $\mathcal{B}$ perfectly simulates Game_4, since in this case, for $i \in Q$ it holds that

$$h_{i,j,k} = T_{q(i),j,k} = (g^{\beta_{j,k}})^{\alpha_{q(i)}} = \mathsf{H}_{\mathsf{PDec}}(j,k,c^*)^{y_i}.$$

If, however, the $T_{p,j,k}$ are random, then $\mathcal{B}$ perfectly simulates Game_5. As a result, if the challenge tuple is real, then $\mathcal{B}$ outputs 1 with probability $\Pr\big[\mathsf{Game}_{4,\mathcal{A}}(1^\lambda) = 1\big]$, and if it is random, $\mathcal{B}$ outputs 1 with probability $\Pr\big[\mathsf{Game}_{5,\mathcal{A}}(1^\lambda) = 1\big]$, which concludes the proof of Claim 5.13. ∎

Game_6 (Interpolation in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$): In this game, we modify how $d_{i,j,k}$ is computed in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. Let P be a canonical completion of $\mathsf{Cor}_{\mathsf{sk}}$ such that $\mathsf{Cor}_{\mathsf{sk}} \subseteq P \subseteq [n]$ and $|P| = t-1$. Let $Q := P \setminus \mathsf{Cor}_{\mathsf{sk}}$. When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ for the first time, the challenger computes for all $(j,k) \in [u] \times [v]$

$$d_{i,j,k} \leftarrow \begin{cases} (c^*_{1,j})^{x_{i,k}} \cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c^*)^{y_i} & \text{for all } i \in \mathsf{Cor}_{\mathsf{sk}} \\ \xleftarrow{\$} \mathbb{G} & \text{for all } i \in Q \\ ((c^*_{1,j})^{x_k})^{\lambda^{(i)}_{0,P\cup\{0\}}} \cdot \prod_{p\in P} d_{p,j,k}^{\lambda^{(i)}_{p,P\cup\{0\}}} & \text{for all remaining } i \in [n] \setminus P \end{cases}.$$

Note that, as a result, for all $i \in \mathsf{Cor}_{\mathsf{sk}}$, partial decryptions are computed as before, while for all remaining $i \in [n] \setminus \mathsf{Cor}_{\mathsf{sk}}$, partial decryptions are now computed in a different, but identically distributed manner.

Observe that the above is equivalent to sampling a random $< t$ degree polynomial $f_{j,k}$ for all $(j,k) \in [u] \times [v]$ that is consistent with $f_{j,k}(0) = r_j x_k$ and the discrete logarithms of real partial decryptions $f_{j,k}(i) = r_j x_{i,k} + w_{j,k,c^*} y_i$ for all $i \in \mathsf{Cor}_{\mathsf{sk}}$. Partial decryptions can be expressed as $d_{i,j,k} = g^{f_{j,k}(i)}$.

Claim 5.14 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\big[\mathsf{Game}_{6,\mathcal{A}}(1^\lambda) = 1\big] = \Pr\big[\mathsf{Game}_{5,\mathcal{A}}(1^\lambda) = 1\big].$$

Proof of Claim 5.14. Fix any $(j,k) \in [u] \times [v]$, and start with the view of $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ as it is in Game_5. Define the degree $< t$ polynomial a_k induced by the sharing over x_k (i.e., $a_k(i) = x_{i,k}$ for all $i \in [n]$). Further, let $\tilde{z}_{j,k}$ be the degree $< t$ polynomial induced by the random independent zero-sharing $(\widetilde{y_{i,j,k}})_{i\in[n]}$ consistent with y_i for $i \in \mathsf{Cor}_{\mathsf{sk}}$. Let w_{j,k,c^*} denote the discrete logarithm of $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*)$ with respect to g, i.e., $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*) = g^{w_{j,k,c^*}}$. Then, in Game_5, we can express partial decryptions as

$$d_{i,j,k} = g^{r_j a_k(i) + w_{j,k,c^*} \tilde{z}_{j,k}(i)}.$$

Let $f_{j,k}(i) = r_j a_k(i) + w_{j,k,c^*} \tilde{z}_{j,k}(i)$ denote the discrete logarithm of partial decryptions in Game_5. Now, since both a_k and $\tilde{z}_{j,k}$ are polynomials of degree $< t$, and $\tilde{z}_{j,k}(0) = 0$, we can rewrite $f_{j,k}(i)$ as follows:

$$\begin{aligned} f_{j,k}(i) = {} & \lambda^{(i)}_{0,P\cup\{0\}} r_j a_k(0) \\ & + \sum_{p\in\mathsf{Cor}_{\mathsf{sk}}} \lambda^{(i)}_{p,P\cup\{0\}} \left(r_j a_k(p) + w_{j,k,c^*} \tilde{z}_{j,k}(p)\right) \\ & + \sum_{q\in Q} \lambda^{(i)}_{q,P\cup\{0\}} \left(r_j a_k(q) + w_{j,k,c^*} \tilde{z}_{j,k}(q)\right). \end{aligned}$$

Observe that the zero-sharing $\tilde{z}_{j,k}$ can be sampled by first fixing $\tilde{z}_{j,k}(p) = y_p$ for all $p \in \mathsf{Cor}_{\mathsf{sk}}$, then sampling $\tilde{z}_{j,k}(q) \xleftarrow{\$} \mathbb{Z}_p$ for all $q \in Q$ and interpolating all remaining positions in $[n] \setminus P$ with $(0,0)$, as to form a consistent zero-sharing. For all $q \in Q$, sampling $\tilde{z}_{j,k}(q)$ at random, and then using it to compute $(r_j a_k(q) + w_{j,k,c^*} \tilde{z}_{j,k}(q))$ is equivalent to directly replacing $(r_j a_k(q) + w_{j,k,c^*} \tilde{z}_{j,k}(q))$ with a uniformly

random value $f_{q,j,k} \xleftarrow{\$} \mathbb{Z}_p$.[30] Hence, we can write

$$\begin{aligned} f_{j,k}(i) &= \lambda^{(i)}_{0,P\cup\{0\}} r_j a_k(0) \\ &\quad + \sum_{p\in\mathsf{Cor_{sk}}} \lambda^{(i)}_{p,P\cup\{0\}} \left(r_j a_k(p) + w_{j,k,c^*} \tilde{z}_{j,k}(p)\right) \\ &\quad + \sum_{q\in Q} \lambda^{(i)}_{q,P\cup\{0\}} f_{q,j,k} \\ &= \lambda^{(i)}_{0,P\cup\{0\}} r_j x_k \\ &\quad + \sum_{p\in\mathsf{Cor_{sk}}} \lambda^{(i)}_{p,P\cup\{0\}} \left(r_j x_{p,k} + w_{j,k,c^*} y_p\right) \\ &\quad + \sum_{q\in Q} \lambda^{(i)}_{q,P\cup\{0\}} f_{q,j,k}. \end{aligned}$$

Note that the above is already equivalent to Game_6:

- We have $f_{j,k}(0) = r_j x_k$.
- For $i \in \mathsf{Cor_{sk}}$, we have $f_{j,k}(i) = r_j x_{i,k} + w_{j,k,c^*} y_i$.
- For $i \in Q$, we have $f_{j,k}(i) = f_{i,j,k}$ for random $f_{i,j,k} \xleftarrow{\$} \mathbb{Z}_p$.
- For all remaining $i \in [n] \setminus P$, $f_{j,k}(i)$ is interpolated from the values at $P \cup \{0\}$ noted above.

■

In the final steps of the proof, we consistently switch the elements $(c^*_{1,j})^{x_k}$ to random. This will affect both the input to $\mathsf{H_{mask}}$ in the challenge ciphertext, and the partial decryptions in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, which are now interpolated from a random $f_{j,k}(0)$. As a result, the polynomials $(f_{j,k})_{(j,k)\in[u]\times[v]}$ are sampled in the same manner as the vector of polynomials $\boldsymbol{f}$ in $\mathsf{GameStat}$. By reduction to $\mathsf{GameStat}$ and applying the core lemma (Lemma 5.1), we can show that the adversary cannot submit $(g^{f_{j,k}(0)})_{(j,k)\in[u]\times[v]}$ to $\mathsf{H_{mask}}$ (i.e., compute $g^{\boldsymbol{f}(0)}$) either directly or indirectly, except with probability $(n+1)q_{\mathsf{H_{mask}}}\binom{n}{t-1}/p^{u\cdot v}$. We add an abort condition for this, and thus, in every winning execution, m_b is masked by a random oracle evaluation on an input not queried by the adversary, thereby perfectly hiding m_b behind a one-time-pad.

Game_7 (Replace $\mathsf{pk}_k^{r_j}$ with $(c^*_{1,j})^{x_k}$ in the Challenge Ciphertext): In Game_7, we modify how the challenge ciphertext is computed. Instead of computing $c^*_2 \leftarrow \mathsf{H_{mask}}\big((\mathsf{pk}_k^{r_j})_{j\in[u],k\in[v]}\big) \oplus m_b$, we compute

$$c^*_2 = \mathsf{H_{mask}}\big(((c^*_{1,j})^{x_k})_{j\in[u],k\in[v]}\big) \oplus m_b.$$

Claim 5.15 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\big[\mathsf{Game}_{7,\mathcal{A}}(1^\lambda) = 1\big] = \Pr\big[\mathsf{Game}_{6,\mathcal{A}}(1^\lambda) = 1\big].$$

Since $\mathsf{pk}_k = g^{x_k}$ and $c^*_{1,j} = g^{r_j}$, both games are identical.

Game_8 (Replace all Occurrences of $(c^{x_k}_{1,j})_{j\in[u],k\in[v]}$ with Random): In this game, for all $(j,k) \in [u]\times[v]$, we sample a random $z_{j,k} \xleftarrow{\$} \mathbb{Z}_p$, compute $Z_{j,k} \leftarrow g^{z_{j,k}}$, and replace all occurrences of $c^{x_k}_{1,j}$ with $Z_{j,k}$. That is, in the challenge ciphertext, we compute

$$c^*_2 \leftarrow \mathsf{H_{mask}}\big((Z_{j,k})_{j\in[u],k\in[v]}\big) \oplus m_b,$$

and in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, we set $f_{j,k}(0) = z_{j,k}$. More formally, for all $i \in P$, we compute $d_{i,j,k}$ as before, and for all $i \in [n] \setminus P$, we compute

$$d_{i,j,k} \leftarrow Z_{j,k}^{\lambda^{(i)}_{0,P\cup\{0\}}} \cdot \prod_{p\in P} d_{p,j,k}^{\lambda^{(i)}_{p,P\cup\{0\}}}.$$

We highlight that both occurrences in the challenge ciphertext and in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ are replaced *consistently* by the same random $Z_{j,k}$.

[30] This holds because $w_{j,k,c^*} \in \mathbb{Z}_p^*$ due to the abort condition introduced in Game_4.

Claim 5.16 For all PPT adversaries $\mathcal{A}$ there exists a PPT reduction $\mathcal{B}$ such that

$$\left|\Pr\left[\mathsf{Game}_{8,\mathcal{A}}(1^\lambda)=1\right]-\Pr\left[\mathsf{Game}_{7,\mathcal{A}}(1^\lambda)=1\right]\right| \leq \mathsf{Adv}^{(u,v)\text{-mDDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda).$$

Proof of Claim 5.16. Let $\mathcal{A}$ be a PPT distinguisher. We construct a reduction $\mathcal{B}$ playing against the (u,v)-mDDH challenger. $\mathcal{B}$ initially receives the tuple

$$\left((A_j)_{j\in[u]},(B_k)_{k\in[v]},(T_{j,k})_{j\in[u],k\in[v]}\right),$$

where $A_j = g^{\alpha_j}$, $B_k = g^{\beta_k}$, and either $T_{j,k} = g^{\alpha_j\beta_k}$ for all $j\in[u]$, $k\in[v]$, or all $T_{j,k}$ are random $\mathbb{G}$ elements. $\mathcal{B}$ simulates either game to $\mathcal{A}$ by embedding the challenge tuple such that $A_j = c^*_{1,j}$, and $B_k = \mathsf{pk}_k$.

Simulating Key Generation. To simulate key generation, $\mathcal{B}$ sets $\mathsf{pk}_k \leftarrow B_k$ for all $k\in[v]$. To simulate key shares on behalf of corrupted parties, $\mathcal{B}$ samples $(x_{i,k})_{k\in[v]} \xleftarrow{\$} \mathbb{Z}_p^v$ for all $i\in\mathsf{Cor}_{\mathsf{sk}}$. Then, $\mathcal{B}$ samples the zero-sharing $(y_i)_{i\in[n]}$, sets $\mathsf{sk}_i \leftarrow ((x_{i,k})_{k\in[v]}, y_i)$ for all $i\in\mathsf{Cor}_{\mathsf{sk}}$, and forwards $(\mathsf{pk},(\mathsf{sk}_i)_{i\in\mathsf{Cor}_{\mathsf{sk}}})$ to $\mathcal{A}$. Note that $\mathcal{B}$ can simulate $g^{x_{i,k}}$ for all $k\in[v]$ and all $i\in[n]$ by simply sampling random $x_{i,k}$ for an additional $t-1-|\mathsf{Cor}_{\mathsf{sk}}|$ parties and interpolating in the exponent together with pk_k.

Simulating $\mathcal{O}^{\mathsf{PDec}}$. $\mathcal{B}$ can simulate $\mathcal{O}^{\mathsf{PDec}}$ as specified by either game. This is possible, since partial decryptions in $\mathcal{O}^{\mathsf{PDec}}$ no longer depend on the x-components of sk_i. In particular, on input (S,φ,c), $\mathcal{B}$ computes

$$d_{i,j,k} \leftarrow (g^{x_{i,k}})^{\omega_j}\cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c)^{y_i}$$

for all $i\in S$, $j\in[u]$, and $k\in[v]$, where $\omega=(\omega_1,\ldots,\omega_u)$ is the encryption randomness extracted from c, and continues to execute motivation functions using the partial decryptions.

Simulating the Challenge Ciphertext. $\mathcal{B}$ embeds the (u,v)-mDDH challenge into c^* by setting

$$c^*_1 \leftarrow (A_1,\ldots,A_u) \text{ and } c^*_2 \leftarrow \mathsf{H}_{\mathsf{mask}}\left((T_{j,k})_{j\in[u],k\in[v]}\right)\oplus m_b,$$

and computes π^* using the simulator.

Simulating $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. On input (S,φ), $\mathcal{B}$ computes the canonical completion P of $\mathsf{Cor}_{\mathsf{sk}}$ such that $\mathsf{Cor}_{\mathsf{sk}}\subseteq P\subseteq[n]$ and $|P|=t-1$, and sets $Q\leftarrow P\setminus\mathsf{Cor}_{\mathsf{sk}}$. Then, for $i\in\mathsf{Cor}_{\mathsf{sk}}$, $\mathcal{B}$ computes all $d_{i,j,k}$ normally using sk_i and samples $d_{i,j,k}\xleftarrow{\$}\mathbb{G}$ for all $i\in Q$ as mandated by either game. For all remaining $i\in S\setminus P$, $\mathcal{B}$ embeds the (u,v)-mDDH challenge into the interpolation at position 0, computing

$$d_{i,j,k} \leftarrow T_{j,k}^{\lambda^{(i)}_{0,P\cup\{0\}}}\cdot\prod_{p\in P} d_{p,j,k}^{\lambda^{(i)}_{p,P\cup\{0\}}}.$$

$\mathcal{B}$ continues the simulation with the above partial decryptions as mandated by either game.

Finally, $\mathcal{B}$ outputs 1 if $\mathcal{A}$ wins the simulated game and 0 otherwise.

Analysis. Observe that $\mathcal{B}$ simulates either game such that $r_j=\alpha_j$, $x_k=\beta_k$, and $(c^*_{1,j})^{x_k}=T_{j,k}$. Hence, if the challenge elements $T_{j,k}$ are real, $\mathcal{B}$ simulates such that $(c^*_{1,j})^{x_k}=g^{\alpha_j\beta_k}=(g^{r_j})^{x_k}$, which perfectly corresponds to Game_7. If, however, the values $T_{j,k}$ are random, then $\mathcal{B}$ perfectly simulates Game_8. As a result, $\mathcal{B}$ outputs 1 when given the real $T_{j,k}$ with probability $\Pr\left[\mathsf{Game}_{7,\mathcal{A}}(1^\lambda)=1\right]$, and outputs 1 when given random $T_{j,k}$ with probability $\Pr\left[\mathsf{Game}_{8,\mathcal{A}}(1^\lambda)=1\right]$, which concludes the proof of Claim 5.16. ∎

It may be tempting to think that we are done at this point. In fact, the message m_b is masked by $\mathsf{H}_{\mathsf{mask}}((Z_{j,k})_{(j,k)\in[u]\times[v]})$ in the challenge ciphertext for random $(Z_{j,k})_{(j,k)\in[u]\times[v]}$. However, this is not quite true yet, because the partial decryptions in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ are still interpolated from the $Z_{j,k}$. We need an additional argument to show that the information obtained by the adversary via $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ is insufficient to compute $(Z_{j,k})_{(j,k)\in[u]\times[v]}$ and thus undo the mask on m_b. We do so by applying Lemma 5.1 in a reduction to GameStat.

First, though, we make the following intuitive argument to simplify the analysis.

Game_9 ($\mathcal{A}$ cannot compute $(Z_{j,k})_{j\in[u],k\in[v]}$ before making an $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query): In Game_9, we modify the behavior of $\mathsf{H}_{\mathsf{mask}}$ queries made by the adversary on $(Z_{j,k})_{j\in[u],k\in[v]}$ prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query. In particular, prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, the challenger answers $\mathsf{H}_{\mathsf{mask}}$ queries made directly or indirectly on the target value $(Z_{j,k})_{j\in[u],k\in[v]}$ by consistently outputting a random element from $\{0,1\}^\mu$ instead of the actual value of $\mathsf{H}_{\mathsf{mask}}((Z_{j,k})_{j\in[u],k\in[v]})$.

Claim 5.17 For all PPT adversaries $\mathcal{A}$ that make at most $q_{\mathsf{H}_{\mathsf{mask}}}$ total direct and indirect queries to $\mathsf{H}_{\mathsf{mask}}$, it holds that

$$\left|\Pr\left[\mathsf{Game}_{9,\mathcal{A}}(1^\lambda)=1\right]-\Pr\left[\mathsf{Game}_{8,\mathcal{A}}(1^\lambda)=1\right]\right| \leq \frac{q_{\mathsf{H}_{\mathsf{mask}}}}{p^{u\cdot v}}.$$

Proof of Claim 5.17. Both games are identical, unless the adversary directly or indirectly queries $\mathsf{H}_{\mathsf{mask}}$ on $(Z_{j,k})_{j\in[u],k\in[v]}$ prior to querying $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. At this point $\mathcal{A}$'s view is entirely independent of $(Z_{j,k})_{j\in[u],k\in[v]}$, except for c_2^*, which only depends on the random oracle *output* $\mathsf{H}_{\mathsf{mask}}((Z_{j,k})_{j\in[u],k\in[v]})$. Thus, the claim follows from a union bound over the $q_{\mathsf{H}_{\mathsf{mask}}}$ guessing opportunities. ∎

Note that the core lemma does not execute the functions submitted by the adversary if the adversary's budget is $|\mathsf{Cor}\cup S^*|\geq t-1$. In order to apply the core lemma, we must first argue that such motivation functions are useless to an adversary.

Game_{10} (Never evaluate useless motivation functions): Within $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S,\varphi)$, if $|\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}}|\geq t-1$, the challenger only executes motivation functions for parties in $S\cap(\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}})$. For the remaining parties in $S\setminus(\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}})$, the challenger sets the output of the motivation function to 0 *without* evaluating it.

Claim 5.18 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\mathsf{Game}_{10,\mathcal{A}}(1^\lambda)=1\right]\geq\Pr\left[\mathsf{Game}_{9,\mathcal{A}}(1^\lambda)=1\right].$$

Proof of Claim 5.18. Clearly Game_{10} and Game_9 are identical until $|\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}}|$ reaches $t-1$ and the first subsequent $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S,\varphi)$ query is made by $\mathcal{A}$ where a motivation function hits $\varphi(\mathsf{pp},\mathsf{pk},i,d_i,c^*)=1$ for some $i\in S\setminus(\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}})$. Denote that i as i^*. Note, however, that immediately after, the oracle in Game_9 would add i^* to $\mathsf{Cor}_{\mathsf{pd}}$. Since i^* was neither in $\mathsf{Cor}_{\mathsf{sk}}$ nor $\mathsf{Cor}_{\mathsf{pd}}$ previously, their union increases by at least one, and we have $|\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}}|\geq t$, meaning the adversary can no longer win in Game_9. As a result, a run is either identical in both games or loses in Game_9, yielding the claim. ∎

The purpose of the next game is to bound the probability that $\mathcal{A}$ ever queries $\mathsf{H}_{\mathsf{mask}}((Z_{j,k})_{j\in[u],k\in[v]})$, either directly, or through a motivation function, by a reduction to $\mathsf{GameStat}$. Here, queries prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query can be disregarded due to the change introduced in Game_9.

Game_{11} ($\mathcal{A}$ cannot query $(Z_{j,k})_{j\in[u],k\in[v]}$): In Game_{11}, the challenger aborts if the adversary queries $(Z_{j,k})_{j\in[u],k\in[v]}$ to the $\mathsf{H}_{\mathsf{mask}}$ oracle while $|\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}}|<t$, *either directly or through a motivation function* (in either $\mathcal{O}_1^{\mathsf{PDec}}$ or $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$). This abort only triggers if such a query happens starting from the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query.

Claim 5.19 For all PPT adversaries $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathsf{Game}_{11,\mathcal{A}}(1^\lambda)=1\right]-\Pr\left[\mathsf{Game}_{10,\mathcal{A}}(1^\lambda)=1\right]\right|\leq(n+1)\cdot q_{\mathsf{H}_{\mathsf{mask}}}\cdot\frac{\binom{n}{t-1}}{p^{u\cdot v}},$$

where $q_{\mathsf{H}_{\mathsf{mask}}}$ bounds the total number of direct and indirect queries made by $\mathcal{A}$ to $\mathsf{H}_{\mathsf{mask}}$.

Proof of Claim 5.19. Let $\mathcal{A}$ be a PPT adversary against either game and denote with Bad the event that $\mathcal{A}$ triggers the additional abort condition. Since both games are identical unless Bad happens, we get that

$$\left|\Pr\left[\mathsf{Game}_{11,\mathcal{A}}(1^\lambda)=1\right]-\Pr\left[\mathsf{Game}_{10,\mathcal{A}}(1^\lambda)=1\right]\right|\leq\Pr[\mathsf{Bad}].$$

We next bound $\Pr[\mathsf{Bad}]$ by a reduction $\mathcal{B}$ to $\mathsf{GameStat}(n,t,p,\ell:=u\cdot v,Q)$, making use of Lemma 5.1, where Q bounds the number of $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ queries made by $\mathcal{A}$. Note that Lemma 5.1 does not bound the runtime of the $\mathsf{GameStat}$ adversary, hence $\mathcal{B}$ can be unbounded. The reduction $\mathcal{B}$ perfectly simulates either game up to the generation of the challenge ciphertext: For key generation, $\mathcal{B}$ simulates as mandated by either game. In particular, this means that $\mathcal{B}$ knows all secret key shares as well as x_k for all $k\in[v]$.

Simulating Random Oracles. $\mathcal{B}$ pre-generates exhaustive tables for the random oracles $\mathsf{H}_{\mathsf{mask}}$ and $\mathsf{H}_{\mathsf{PDec}}$, and answers queries by consulting the table. For $\mathsf{H}_{\mathsf{PDec}}$, $\mathcal{B}$ will simulate consistently from the table throughout, while we will later adjust how $\mathcal{B}$ simulates $\mathsf{H}_{\mathsf{mask}}$ starting from $\mathcal{A}$'s first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, according to the modifications introduced in Game_9.

Simulating the Challenge Ciphertext. To simulate the challenge ciphertext c^* for messages m_0, m_1, $\mathcal{B}$ samples random $(r_j)_{j\in[u]} \xleftarrow{\$} \mathbb{Z}_p^u$ and sets $c_1^* \leftarrow (g^{r_1}, \ldots, g^{r_u})$. Instead of sampling $(Z_{j,k})_{j\in[u],k\in[v]}$ and setting $c_2^* \leftarrow \mathsf{H}_{\mathsf{mask}}((Z_{j,k})_{j\in[u],k\in[v]}) \oplus m_b$, $\mathcal{B}$ just samples a random $H \xleftarrow{\$} \{0,1\}^\mu$ and sets

$$c_2^* \leftarrow H \oplus m_b$$

for a random bit b. Note that as such, $\mathcal{B}$ essentially programs $\mathsf{H}_{\mathsf{mask}}((Z_{j,k})_{j\in[u],k\in[v]}) \coloneqq H$ without knowing $(Z_{j,k})_{j\in[u],k\in[v]}$ yet. Also, consistent with the changes introduced in Game_9, $\mathcal{B}$ consistently responds to $\mathsf{H}_{\mathsf{mask}}((Z_{j,k})_{j\in[u],k\in[v]})$ queries with the value from the table, instead of H prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query. We write $\mathsf{H}_{\mathsf{PDec}}(j,k,c^*) = g^{w_{j,k,c^*}}$ for all $j \in [u], k \in [v]$, where $w_{j,k,c^*} \in \mathbb{Z}_p$. Clearly, $\mathcal{B}$ knows all w_{j,k,c^*} as it is unbounded.

Simulating $\mathcal{O}^{\mathsf{PDec}}$. $\mathcal{B}$ can simulate $\mathcal{O}^{\mathsf{PDec}}$ exactly as mandated by either game. In particular, $\mathcal{B}$ executes the motivation functions and simulates the random oracles to them.

Simulating $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S,\varphi)$, $\mathcal{B}$ embeds this into a query to its challenger as follows. It computes

$$z_{i,j,k} \leftarrow r_j x_{i,k} + w_{j,k,c^*} y_i$$

for all $i \in \mathsf{Cor}_{\mathsf{sk}}$ and all $j \in [u]$, $k \in [v]$, and sets $z_i \coloneqq (z_{i,j,k})_{j\in[u],k\in[v]}$ for all $i \in \mathsf{Cor}_{\mathsf{sk}}$.

Only upon the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query... If this is the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, $\mathcal{B}$ forwards $(\mathsf{Cor}_{\mathsf{sk}}, (z_i)_{i\in\mathsf{Cor}_{\mathsf{sk}}})$ to its challenger, causing the challenger to initialize $\mathsf{Cor} \leftarrow \mathsf{Cor}_{\mathsf{sk}}$ and to sample the vector of polynomials $\boldsymbol{f}$ that is consistent with $(z_i)_{i\in\mathsf{Cor}_{\mathsf{sk}}}$. At this point, $\mathcal{B}$ has access to the $\mathcal{O}^{\mathsf{check}}$ oracle of its challenger, and now $\mathcal{B}$ adjusts how it simulates the random oracle $\mathsf{H}_{\mathsf{mask}}$: Whenever $\mathcal{A}$ or a motivation function makes a query to $\mathsf{H}_{\mathsf{mask}}$ on input $(V_{j,k})_{j\in[u],k\in[v]}$, $\mathcal{B}$ first extracts the discrete logarithms $v_{j,k} \leftarrow \mathsf{DLog}_g(V_{j,k})$ for all $j \in [u], k \in [v]$, assembles $\boldsymbol{v} \leftarrow (v_{j,k})_{j\in[u],k\in[v]}$, and queries $\mathcal{O}^{\mathsf{check}}(\boldsymbol{v})$. If $\boldsymbol{v} = \boldsymbol{f}(0)$, or equivalently, $(V_{j,k})_{j\in[u],k\in[v]} = g^{\boldsymbol{f}(0)}$, and the current budget is $|\mathsf{Cor} \cup S^*| < t$, this query makes $\mathcal{B}$ win immediately, ending the simulation. If, instead, $|\mathsf{Cor} \cup S^*| \geq t$, the oracle returns 1 without causing $\mathcal{B}$ to win. In this case, the remainder of the run is vacuous because, given the non-decreasing nature of the budget, the event Bad can no longer occur. Finally, if the queried value is not the target, the response to the $\mathsf{H}_{\mathsf{mask}}$ query is taken from the pre-computed table.

On every $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query... $\mathcal{B}$ then computes

$$d_{i,j,k} \leftarrow g^{z_{i,j,k}} \quad \left[\equiv (c_{1,j}^*)^{x_{i,k}} \cdot \mathsf{H}_{\mathsf{PDec}}(j,k,c^*)^{y_i}\right]$$

for all $i \in \mathsf{Cor}_{\mathsf{sk}}$ and generates the circuit ρ such that

$$\rho(i, v_i \coloneqq (v_{i,j,k})_{j\in[u],k\in[v]}) = \begin{cases} \varphi(\mathsf{pp}, \mathsf{pk}, i, d_i \coloneqq (g^{v_{i,j,k}})_{j\in[u],k\in[v]}, c^*), & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases}.$$

Additionally, $\mathcal{B}$ hardcodes the random oracle tables for $\mathsf{H}_{\mathsf{mask}}$ and $\mathsf{H}_{\mathsf{PDec}}$ into ρ. Whenever the internal φ makes a random oracle query, ρ answers from the table, except for $\mathsf{H}_{\mathsf{mask}}$, where ρ first extracts the discrete logarithm and forwards it to $\mathcal{O}^{\mathsf{check}}$ as outlined above. This means that $\mathcal{B}$ wins whenever a motivation function calls $\mathsf{H}_{\mathsf{mask}}(g^{\boldsymbol{f}(0)})$ within $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$.

Edge case if $|\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}| \geq t-1$. If $|\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}| \geq t-1$ and $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S,\varphi)$, $\mathcal{B}$ no longer needs to run motivation functions for all $S \setminus (\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}})$ due to the change introduced in Game_{10}. However, $\mathcal{B}$ still needs to simulate motivation functions for $\bar{S} = S \cap (\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}})$.[31] For $\bar{S}$, $\mathcal{B}$ cannot rely on simply forwarding ρ to its challenger, because $\mathsf{GameStat}$ no longer executes motivation functions while $|\mathsf{Cor} \cup S^*| \geq t-1$. Hence, $\mathcal{B}$ instead simulates the motivation functions for $\bar{S}$ locally. For all $i \in \bar{S}$, $\mathcal{B}$ distinguishes the following two cases. If $i \in S \cap \mathsf{Cor}_{\mathsf{sk}}$, $\mathcal{B}$ uses the partial decryption $d_{i,j,k} = g^{z_{i,j,k}}$ it computed from i's secret key share directly and then runs $\varphi(\mathsf{pp}, \mathsf{pk}, i, (d_{i,j,k})_{j\in[u],k\in[v]}, c^*)$. If, instead, $i \in S \cap \mathsf{Cor}_{\mathsf{pd}}$, it must be the case that a prior $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query already output $(d_{i,j,k})_{j\in[u],k\in[v]}$, and $\mathcal{B}$ uses that partial decryption to simulate the motivation function.[32] Whenever the motivation function makes a query to $\mathsf{H}_{\mathsf{mask}}$ in either case, $\mathcal{B}$ first extracts the discrete logarithm and forwards the result to $\mathcal{O}^{\mathsf{check}}$ (as a direct query), i.e., $\mathcal{B}$ also wins if motivation functions query $\mathsf{H}_{\mathsf{mask}}$ on $g^{\boldsymbol{f}(0)}$ for any $i \in \bar{S}$. Denote as $\bar{S}'$ the set of parties where the above execution of the motivation function outputs 1.

[31] Intuitively, this is because even if these motivation functions output 1, this would not affect the adversary's budget, as $\bar{S} \subseteq \mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}$ already.

[32] Crucially, we are using the fact that PDec is deterministic in Π_{ZEG}.

Then, after dealing with the potential edge case, $\mathcal{B}$ forwards ρ to its challenger.[33] As a response, $\mathcal{B}$ receives the set S' as well as the tuple $(v_i := \boldsymbol{f}(i))_{i\in S'}$, where $\boldsymbol{f}$ is the vector of polynomials sampled by the $\mathsf{GameStat}$ challenger that is now consistent with the discrete logarithms of partial decryptions on the challenge ciphertext. $\mathcal{B}$ parses $(v_{i,j,k})_{j\in[u],k\in[v]} \leftarrow \boldsymbol{v}_i$ and sets $d_{i,j,k} \leftarrow g^{v_{i,j,k}}$ for all $i \in S'$ and $j \in [u], k \in [v]$. Finally, $\mathcal{B}$ assembles $d_i := (d_{i,j,k})_{j\in[u],k\in[v]}$ and returns $(S' \cup \bar{S}', (d_i)_{i\in S'\cup\bar{S}'})$ to $\mathcal{A}$ as the output of $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. At the end of this query, the challenger updates $S^* \leftarrow S^* \cup S'$ and $\mathcal{B}$ updates the initially empty $\mathsf{Cor_{pd}} \leftarrow \mathsf{Cor_{pd}} \cup S'$. For subsequent $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ queries, $\mathcal{B}$ simulates in an identical manner. Note that, with the above simulation, the set $\mathsf{Cor_{sk}}$ of $\mathcal{B}$'s simulation is always the same as the set Cor of the $\mathsf{GameStat}$ challenger, and $\mathsf{Cor_{pd}}$ of $\mathcal{B}$'s simulation mirrors the set S^* of the $\mathsf{GameStat}$ challenger.

Analysis. Consider the case that the event Bad happens. In this case, $\mathcal{B}$'s simulation of either game to $\mathcal{A}$ is perfect until the winning $\mathsf{H_{mask}}$ happens, either directly or within a motivation function. In particular, $\mathcal{B}$ perfectly simulates for $(Z_{j,k})_{j\in[u],k\in[v]} = g^{\boldsymbol{f}(0)}$ and $\mathsf{H_{mask}}((Z_{j,k})_{j\in[u],k\in[v]}) = H$, where $\boldsymbol{f}$ is the $u \cdot v$-tuple of $< t$ degree polynomials sampled by the $\mathsf{GameStat}$ challenger that is consistent with the discrete logarithms of $d_{i,j,k}$ for all $i \in \mathsf{Cor_{sk}}$.[34] Considering the polynomial view of the discrete logarithms of partial decryptions introduced in Game_6, $\mathcal{B}$ simulates perfectly for $(f_{j,k})_{j\in[u],k\in[v]} = \boldsymbol{f}$. As a result, if Bad happens, the input of the winning query is equal to $(Z_{j,k})_{j\in[u],k\in[v]} = g^{\boldsymbol{f}(0)}$ and $\mathcal{B}$ queries $\mathcal{O}^{\mathsf{check}}(\boldsymbol{f}(0))$ to its challenger. Also, Bad implies that $|\mathsf{Cor_{sk}} \cup \mathsf{Cor_{pd}}| < t$, and thus the condition $|\mathsf{Cor} \cup S^*| < t$ passes in $\mathsf{GameStat}$ and $\mathcal{B}$ makes at most $q_{\mathsf{H_{mask}}}$ (direct and indirect) queries to $\mathcal{O}^{\mathsf{check}}$. This allows us to bound

$$\Pr[\mathsf{Bad}] \leq \Pr[\mathsf{GameStat}_{\mathcal{B}}(n, t, p, u \cdot v, Q) = 1].$$

The claim then follows directly from Lemma 5.1. ∎

Final Bound on Game_{11}: Observe that in any winning execution of Game_{11}, the adversary cannot query $\mathsf{H_{mask}}$ on $(Z_{j,k})_{j\in[u],k\in[v]}$, neither directly, nor via any submitted motivation function that is ever evaluated starting from the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query. If the adversary should issue such a $\mathsf{H_{mask}}$ query prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, then Game_{11} consistently returns a random value instead of the actual $\mathsf{H_{mask}}((Z_{j,k})_{j\in[u],k\in[v]})$ value used in the challenge ciphertext. Thus $c_2^* = \mathsf{H_{mask}}((Z_{j,k})_{j\in[u],k\in[v]}) \oplus m_b$ perfectly hides m_b behind a one-time-pad. As this is the only element dependent on b in the adversary's view, it holds that

$$\Pr\big[\mathsf{Game}_{11,\mathcal{A}}(1^\lambda) = 1\big] \leq \frac{1}{2}.$$

Putting it all together: When applying Claims 5.9-5.19 alongside the above bound on the advantage in $\mathsf{Game}_{11,\mathcal{A}}$, we get that for all PPT adversaries $\mathcal{A}$ there exist PPT reductions $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ such that

$$\begin{aligned}\Pr\big[\mathsf{Game}_{0,\mathcal{A}}(1^\lambda) = 1\big] \leq \frac{1}{2} + 2\mathsf{Adv}^{\mathsf{SimExt}}_{\mathcal{B}_1,(\mathsf{PS},\mathsf{Sim})}(1^\lambda) + \mathsf{Adv}^{(t-1,uv)\text{-mDDH}}_{\mathcal{B}_2,\mathcal{G}}(1^\lambda) + \mathsf{Adv}^{(u,v)\text{-mDDH}}_{\mathcal{B}_3,\mathcal{G}}(1^\lambda) \\ + \frac{q_{\mathsf{H_{PDec}}}}{p^u} + \frac{uv}{p} + \frac{q_{\mathsf{H_{mask}}}}{p^{uv}} + (n+1) \cdot q_{\mathsf{H_{mask}}} \cdot \frac{\binom{n}{t-1}}{p^{uv}}.\end{aligned}$$

Whenever $\log(p^{uv}) \geq n + \lambda$, it holds that

$$\frac{\binom{n}{t-1}}{p^{uv}} \leq \frac{2^n}{2^{n+\lambda}} \leq 2^{-\lambda},$$

and thus the advantage is negligible, assuming $\log(p) \geq \lambda$, $u, v, q_{\mathsf{H_{PDec}}}, q_{\mathsf{H_{mask}}} \in \mathsf{poly}(\lambda)$, simulation-extractability of PS, and that the DDH assumption holds for $\mathcal{G}$. ∎

5.4 Constant Parameter Sizes from Pairings

Construction 2 (Pairing-Based Variant Π_{Pair})**.** Let μ denote the message length and let $\mathsf{H_{mask}} \colon \{0,1\}^* \to \{0,1\}^\mu$, $\mathsf{H_{PDec}} \colon \{0,1\}^* \to \mathbb{G}_1$, and $\mathsf{H_{id}} \colon \{0,1\}^* \to \mathbb{G}_2$ be hash functions modeled as random oracles. The

[33] Note that, if $|\mathsf{Cor_{sk}} \cup \mathsf{Cor_{pd}}| \geq t - 1$, i.e., if the edge case above was hit, the challenger will not run ρ and this is a no-op, returning an empty set back to $\mathcal{B}$.

[34] Notably, the simulation of $\mathsf{H_{mask}}$ is consistent whenever Bad happens, because $\mathcal{B}$'s simulation via the pre-computed table on input $(Z_{j,k})_{j\in[u],k\in[v]} = g^{\boldsymbol{f}(0)}$ prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query is consistent with the modification introduced in Game_9. For $\mathsf{H_{mask}}((Z_{j,k})_{j\in[u],k\in[v]})$ queries *after* the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, $\mathcal{B}$ is supposed to return H. Here, $\mathcal{B}$ always queries $\mathcal{O}^{\mathsf{check}}$ on its discrete logarithm. Since Bad requires $|\mathsf{Cor_{sk}} \cup \mathsf{Cor_{pd}}| < t$ to trigger, and the budget is non-decreasing, the same must hold at all earlier points of the run, and the first query triggering Bad makes $\mathcal{B}$ win.

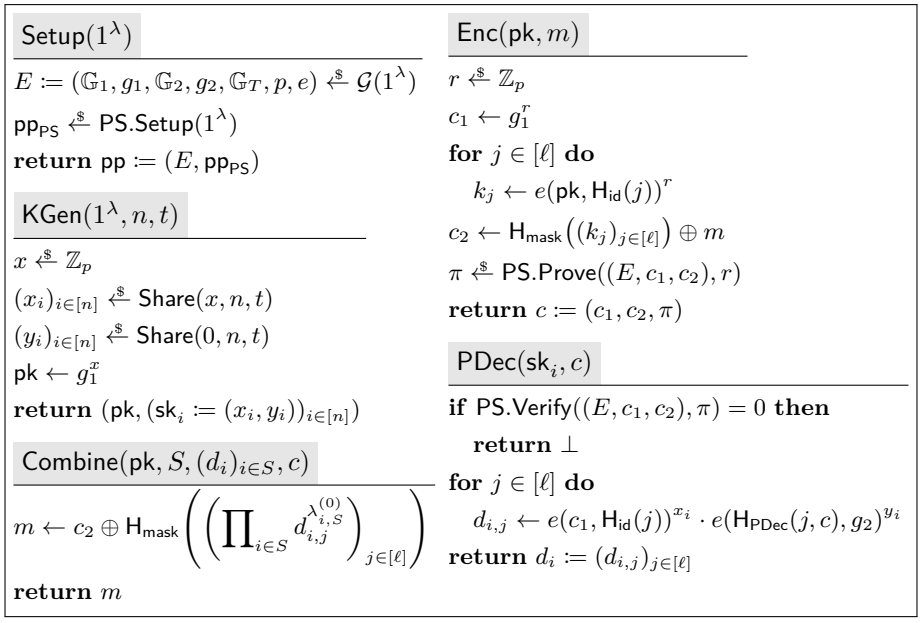


Figure 6: The construction of Π_{Pair}, parameterized with $\ell \in \mathbb{N}$. Recall that ℓ can be a function in λ, n, and t.

scheme is parameterized with $\ell \in \mathbb{N}$, which can be a function of λ, n, and t. Let PS be a proof of knowledge system for the relation

$$\mathsf{R} = \left\{ (\chi, \omega) \colon \begin{array}{c} \chi = ((\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e), c_1, c_2) \ \wedge \\ c_1 = g_1^\omega \end{array} \right\}.$$

The algorithms of Π_{Pair} can be found in Figure 6.

To verify correctness, consider that for any $j \in [\ell]$ and $S \subseteq [n]$ with $|S| \geq t$ it holds that

$$\begin{aligned} \prod_{i\in S} d_{i,j}^{\lambda_{i,S}^{(0)}} &= \prod_{i\in S} e(c_1, \mathsf{H}_{\mathsf{id}}(j))^{\lambda_{i,S}^{(0)} x_i} \cdot \prod_{i\in S} e(\mathsf{H}_{\mathsf{PDec}}(j, c), g_2)^{\lambda_{i,S}^{(0)} y_i} \\ &= e(g_1^r, \mathsf{H}_{\mathsf{id}}(j))^{\sum_{i\in S} \lambda_{i,S}^{(0)} x_i} \cdot e(\mathsf{H}_{\mathsf{PDec}}(j, c), g_2)^{\sum_{i\in S} \lambda_{i,S}^{(0)} y_i} \\ &= e(g_1, \mathsf{H}_{\mathsf{id}}(j))^{xr} \cdot e(\mathsf{H}_{\mathsf{PDec}}(j, c), g_2)^0 = e(\mathsf{pk}, \mathsf{H}_{\mathsf{id}}(j))^r = k_j. \end{aligned}$$

Again, the proof system PS can be instantiated with Schnorr proofs (see Remark 5.6). Alternatively, one can instantiate it with DLEQ proofs, avoiding simulation-extractability at the cost of increasing ciphertext size by an additional $\mathbb{G}_1$ element (see Section 5.5).

Theorem 5.20 Π_{Pair} *(Construction 2) is* IND-IMC-CCA*-secure in the random oracle model for p, ℓ such that $\log(p^\ell) \geq n + \lambda$ and $\log(p) \geq \lambda$, $q = Q = (0, 0, \infty, \infty, \infty)$, i.e., static corruptions and high-threshold access to the partial decryption oracle, and $\mathcal{F}_0^{\mathsf{PDec}} = \mathcal{F}_1^{\mathsf{PDec}} = \mathcal{F}^{\mathsf{chal}} = \mathcal{F}_{\mathsf{no\text{-}sk}}$, i.e., motivation functions receive partial decryptions but no secret key shares as input, if* XDH *and* BDDH *are hard for $\mathcal{G}$ and* PS *is simulation-extractable. In more detail, for all PPT adversaries $\mathcal{A}$ there exist PPT reductions $\mathcal{B}$ such that*

$$\begin{aligned} \mathsf{Adv}^{\mathsf{IND\text{-}IMC\text{-}CCA}}_{\mathcal{A},\Pi_{\mathsf{Pair}}}(1^\lambda, n, t, \mathcal{F}, q, Q) \leq\ & 2 \cdot \mathsf{Adv}^{\mathsf{SimExt}}_{\mathcal{B},(\mathsf{PS}_{\mathsf{R}},\mathsf{Sim})}(1^\lambda) + \ell(t-1) \cdot \mathsf{Adv}^{\mathsf{XDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda) + \ell \cdot \mathsf{Adv}^{\mathsf{BDDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda) \\ & + \frac{q_{\mathsf{H}_{\mathsf{PDec}}}}{p} + \frac{\ell}{p} + \frac{q_{\mathsf{H}_{\mathsf{mask}}}}{p^\ell} + (n+1) \cdot q_{\mathsf{H}_{\mathsf{mask}}} \cdot \frac{\binom{n}{t-1}}{p^\ell}, \end{aligned}$$

where $q_{\mathsf{H}_{\mathsf{mask}}}$ and $q_{\mathsf{H}_{\mathsf{PDec}}}$ bound the number of queries made by $\mathcal{A}$ to the random oracles (directly or through motivation functions), respectively, and the running time of $\mathcal{B}$ is at most that of $\mathcal{A}$ plus an additive overhead polynomial in λ and n.

Proof of Theorem 5.20. The proof has a very similar structure to that of Theorem 5.7 and thus we only highlight important parts. Let Game_0 be the game $\mathsf{IND\text{-}IMC\text{-}CCA}^{\Pi_{\mathsf{Pair}}}(1^\lambda, n, t, \mathcal{F}, q, Q)$ instantiated with the parameters from Theorem 5.20. Henceforth, we omit the inputs $n, t, \mathcal{F}, q$, and Q to security games to improve readability. Game_1 to Game_3 are largely the same as previously: The challenger simulates the proof for the challenge ciphertext, extracts the witness from proofs given to the $\mathcal{O}_0^{\mathsf{PDec}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$ oracles, and aborts if the adversary directly or indirectly queries $\mathsf{H}_{\mathsf{PDec}}(j, c^*)$ for any $j \in [\ell]$ prior to seeing c^*.

$\underline{\mathsf{Game}_3}$: We start by summarizing how Game_3 is defined in this proof.

1. The adversary outputs a set of static corruptions $\mathsf{Cor}_{\mathsf{sk}}$. W.l.o.g., we assume that $\mathsf{Cor}_{\mathsf{sk}} \subseteq [n]$ and $|\mathsf{Cor}_{\mathsf{sk}}| < t$.

2. The challenger generates the pairing ensemble $E \xleftarrow{\$} \mathcal{G}(1^\lambda)$ and initializes the proof system via the simulator $(\mathsf{pp}_{\mathsf{PS}}, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{Sim.Setup}(1^\lambda)$. The challenger generates the public key $\mathsf{pk} := g_1^x$ and the secret key shares $\mathsf{sk}_i := (x_i, y_i)$, where x_i are shares of x and y_i are zero-shares. It forwards to the adversary $(E, \mathsf{pp}_{\mathsf{PS}}, \mathsf{pk})$, and sk_i for all $i \in \mathsf{Cor}_{\mathsf{sk}}$. Note that there will be no additional corruptions during the game.

3. The adversary can now query a partial decryption oracle $\mathcal{O}_0^{\mathsf{PDec}}$, which on input $(S, \varphi, c := (c_1, c_2, \pi))$, first verifies the proof π and then uses the simulator to extract $(\omega, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{Sim.Ext}(\chi, \pi, \mathsf{st}_{\mathsf{Sim}})$, where $\chi = (E, c_1, c_2)$. The challenger then computes partial decryptions for parties $i \in S$ using the witness as
$$d_{i,j} \leftarrow e(g_1^{x_i}, \mathsf{H}_{\mathsf{id}}(j))^\omega \cdot e(\mathsf{H}_{\mathsf{PDec}}(j, c), g_2)^{y_i} \text{ for all } j \in [\ell].$$
Then, the challenger assembles $d_i \leftarrow (d_{i,j})_{j \in [\ell]}$ and uses the partial decryptions to run the motivation functions φ. It outputs $(S', (d_i)_{i \in S'})$ to the adversary, where S' is the subset of S, where $\varphi(\mathsf{pp}, \mathsf{pk}, i, d_i, c) = 1$.

4. The adversary outputs two messages m_0, m_1 of length μ, and the challenger encrypts m_b under pk for a random bit b. The ciphertext consists of
$$c_1^* = g_1^r, \quad c_2^* = \mathsf{H}_{\mathsf{mask}}((k_j)_{j \in [\ell]}) \oplus m_b, \quad \pi^*,$$
where $k_j = e(\mathsf{pk}, \mathsf{H}_{\mathsf{id}}(j))^r$, and π^* is simulated as $(\pi^*, \mathsf{st}_{\mathsf{Sim}}) \xleftarrow{\$} \mathsf{Sim.Prove}((E, c_1^*, c_2^*), \mathsf{st}_{\mathsf{Sim}})$. The challenger inspects the trace of queries made by the adversary to $\mathsf{H}_{\mathsf{PDec}}$. If it contains a query on (j, c^*) for any $j \in [\ell]$, the challenger aborts. Otherwise, the challenger forwards c^* to the adversary.

5. The adversary can now make $\mathcal{O}_1^{\mathsf{PDec}}$ and $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ queries. $\mathcal{O}_1^{\mathsf{PDec}}$ is simulated identically to $\mathcal{O}_0^{\mathsf{PDec}}$, except for the additional $c \neq c^*$ assertion. In $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, the challenger computes partial decryptions according to protocol, i.e.,
$$d_{i,j} \leftarrow e(c_1^*, \mathsf{H}_{\mathsf{id}}(j))^{x_i} \cdot e(\mathsf{H}_{\mathsf{PDec}}(j, c^*), g_2)^{y_i}.$$
The challenger executes the motivation function using these partial decryptions on the challenge ciphertext and updates the initially empty set $\mathsf{Cor}_{\mathsf{pd}} \leftarrow \mathsf{Cor}_{\mathsf{pd}} \cup S'$, where S' is the set of parties, for which the motivation function, given the partial decryption on the challenge ciphertext, outputs 1. The challenger forwards S' alongside their partial decryptions to the adversary.

6. Finally, the adversary outputs a guess b' and wins if $b' = b$ and $|\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}| < t$, i.e., not too many partial decryptions were learned on the challenge ciphertext.

7. Throughout execution, the adversary and motivation functions have access to the random oracles $\mathsf{H}_{\mathsf{mask}}$, $\mathsf{H}_{\mathsf{PDec}}$, and H_{id}. We henceforth call queries made by the adversary "direct" queries, and queries made through a motivation function "indirect" queries.

Claim 5.21 For all PPT adversaries $\mathcal{A}$, there exists a PPT reduction $\mathcal{B}$ such that
$$\left|\Pr\left[\mathsf{Game}_{3,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{0,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq 2\mathsf{Adv}^{\mathsf{SimExt}}_{\mathcal{B},(\mathsf{PS}_{\mathsf{R}},\mathsf{Sim})}(1^\lambda) + \frac{q_{\mathsf{H}_{\mathsf{PDec}}}}{p},$$
where $q_{\mathsf{H}_{\mathsf{PDec}}} \in \mathsf{poly}(\lambda)$ bounds the number of queries made by $\mathcal{A}$ to $\mathsf{H}_{\mathsf{PDec}}$, directly or indirectly via motivation functions.

The proof is identical to the proofs of Claims 5.9-5.11.

Game_4 (Abort if $\mathsf{H}_{\mathsf{PDec}}(j, c^*) = g_1^0$): In Game_4, the challenger aborts after generating the challenge ciphertext if $\mathsf{H}_{\mathsf{PDec}}(j, c^*) = g_1^0$ for any $j \in [\ell]$.

Claim 5.22 For all adversaries $\mathcal{A}$ it holds that

$$|\Pr[\mathsf{Game}_{4,\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Game}_{3,\mathcal{A}}(1^\lambda) = 1]| \leq \frac{\ell}{p}.$$

The claim follows from a union bound, as, in the random oracle model, $\mathsf{H}_{\mathsf{PDec}}(j, c^*) = g_1^0$ holds with probability $1/p$ for any given j.

Game_5 (Independent Zero-Sharings): In this game hop, we replace how $d_{i,j}$ is computed in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. In particular, we replace the zero-shares y_i with independent zero-sharings for every $j \in [\ell]$. More formally, when the adversary queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, the challenger samples an independent random zero-sharing for each $j \in [\ell]$ that is consistent with y_i for $i \in \mathsf{Cor}_{\mathsf{sk}}$: Let P be a canonical completion of the set $\mathsf{Cor}_{\mathsf{sk}}$ such that $\mathsf{Cor}_{\mathsf{sk}} \subseteq P$ and $|P| = t - 1$. For all $j \in [\ell]$, the challenger generates

$$\widetilde{y_{i,j}} \leftarrow \begin{cases} y_i, & \text{if } i \in \mathsf{Cor}_{\mathsf{sk}} \\ \xleftarrow{\$} \mathbb{Z}_p, & \text{if } i \in P \setminus \mathsf{Cor}_{\mathsf{sk}} \\ \sum_{u \in P} \lambda^{(i)}_{u, P \cup \{0\}} \cdot \widetilde{y_{u,j}}, & \text{if } i \in [n] \setminus P \end{cases}.$$

The challenger then computes

$$d_{i,j} \leftarrow e(c_1^*, \mathsf{H}_{\mathsf{id}}(j))^{x_i} \cdot e(h_{i,j}, g_2),$$

where $h_{i,j} \coloneqq \mathsf{H}_{\mathsf{PDec}}(j, c^*)^{\widetilde{y_{i,j}}}$.

Claim 5.23 For all PPT adversaries $\mathcal{A}$, there exists a PPT reduction $\mathcal{B}$ such that

$$\left|\Pr[\mathsf{Game}_{5,\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Game}_{4,\mathcal{A}}(1^\lambda) = 1]\right| \leq \mathsf{Adv}^{(t-1,\ell)\text{-}\mathsf{mXDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda).$$

Proof of Claim 5.23. Let $\mathcal{A}$ be a PPT distinguisher between Game_5 and Game_4 and let $\mathcal{B}$ be a PPT reduction playing against the $(t-1,\ell)$-mXDH challenger. $\mathcal{B}$ initially receives the $(t-1,\ell)$-mXDH challenge tuple

$$\left((A_u)_{u \in [t-1]}, (B_j)_{j \in [\ell]}, (T_{u,j})_{u \in [t-1], j \in [\ell]}\right),$$

where $A_u = g_1^{\alpha_u}$ for all $u \in [t-1]$, $B_j = g_1^{\beta_j}$ for all $j \in [\ell]$ and either $T_{u,j} = g_1^{\alpha_u \beta_j}$ or $T_{u,j}$ is a random $\mathbb{G}_1$ element for all $u \in [t-1]$ and $j \in [\ell]$.

$\mathcal{B}$ now simulates either game to $\mathcal{A}$. After $\mathcal{A}$ outputs a set $\mathsf{Cor}_{\mathsf{sk}} \subseteq [n]$ such that $|\mathsf{Cor}_{\mathsf{sk}}| < t$, $\mathcal{B}$ computes the canonical set P that completes $\mathsf{Cor}_{\mathsf{sk}}$ such that $\mathsf{Cor}_{\mathsf{sk}} \subseteq P \subseteq [n]$ and $|P| = t-1$. Denote $Q \coloneqq P \setminus \mathsf{Cor}_{\mathsf{sk}}$, and define any injective mapping $q \colon Q \to [t-1]$, which always exists since $|Q| \leq |P| = t-1$. $\mathcal{B}$ continues to simulate such that $g_1^{y_i} = A_{q(i)}$ for all $i \in Q$, and $\mathsf{H}_{\mathsf{PDec}}(j, c^*) = B_j$ for all $j \in [\ell]$. More formally, $\mathcal{B}$ carries out the following simulation.

Simulating Key Generation. $\mathcal{B}$ simulates $(x_i)_{i \in [n]}$ according to either game as a sharing of a random $x \xleftarrow{\$} \mathbb{Z}_p$, setting $\mathsf{pk} \leftarrow g_1^x$. Then, $\mathcal{B}$ samples a random $y_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mathsf{sk}_i \leftarrow (x_i, y_i)$ for all $i \in \mathsf{Cor}_{\mathsf{sk}}$. It outputs $(\mathsf{pk}, (\mathsf{sk}_i)_{i \in \mathsf{Cor}_{\mathsf{sk}}})$ to $\mathcal{A}$. Observe that, given y_i for all $i \in \mathsf{Cor}_{\mathsf{sk}}$ and $g_1^{y_i} \leftarrow A_{q(i)}$ for all $i \in Q$, $\mathcal{B}$ can simulate $g_1^{y_i}$ for all $i \in [n] \setminus P$ as a consistent zero-sharing as follows:

$$g_1^{y_i} \leftarrow \prod_{k \in \mathsf{Cor}_{\mathsf{sk}}} g_1^{y_k \lambda^{(i)}_{k, P \cup \{0\}}} \cdot \prod_{k \in Q} A_{q(k)}^{\lambda^{(i)}_{k, P \cup \{0\}}}.$$

Simulating the random oracles. For H_{id} and $\mathsf{H}_{\mathsf{mask}}$, $\mathcal{B}$ outputs a random element from $\mathbb{G}_2$ and $\{0,1\}^\mu$ consistently. Whenever $\mathcal{A}$ directly or indirectly queries $\mathsf{H}_{\mathsf{PDec}}$, on input (j, c), $\mathcal{B}$ samples $w_{j,c} \xleftarrow{\$} \mathbb{Z}_p$ and outputs $g_1^{w_{j,c}}$ consistently.

Simulating $\mathcal{O}_0^{\mathsf{PDec}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$. On input (S, φ, c), when computing the partial decryptions $d_{i,j}$ for all $i \in S$ and $j \in [\ell]$, $\mathcal{B}$ simulates

$$d_{i,j} \leftarrow e(g_1^{x_i}, \mathsf{H}_{\mathsf{id}}(j))^\omega \cdot e(g_1^{y_i}, g_2)^{w_{j,c}},$$

which $\mathcal{B}$ can do using the simulated $g_1^{y_i}$ as outlined in key generation. Note that the simulation is perfect due to the bilinearity of e.

Simulating the challenge ciphertext. On input (m_0, m_1) with $|m_0| = |m_1| = \mu$, $\mathcal{B}$ samples a random bit b and encrypts m_b as mandated by either game, yielding the ciphertext $c^* = (c_1^*, c_2^*, \pi^*)$. We denote the randomness for $c_1^* = g_1^r$ as r. Now, if $\mathcal{A}$ has already queried $\mathsf{H}_{\mathsf{PDec}}$ on input (j, c^*) for any $j \in [\ell]$, $\mathcal{B}$ aborts, as mandated by either game, and outputs 0. Otherwise, $\mathcal{B}$ programs $\mathsf{H}_{\mathsf{PDec}}(j, c^*) \leftarrow B_j$ for all $j \in [\ell]$. If $B_j = g_1^0$ for any $j \in [\ell]$, $\mathcal{B}$ aborts, as mandated by the condition introduced in Game_4, and outputs 0 to its challenger.

Simulating $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, $\mathcal{B}$ embeds the $(t-1,\ell)$-mXDH challenge into the partial decryptions. In particular, it computes $h_{i,j}$ for all $i \in [n]$ and $j \in [\ell]$ as follows

$$h_{i,j} \leftarrow \begin{cases} \mathsf{H}_{\mathsf{PDec}}(j, c^*)^{y_i}, \text{for } i \in \mathsf{Cor}_{\mathsf{sk}}, \\ T_{q(i),j}, \text{for } i \in Q, \\ \prod_{v \in P} h_{v,j}^{\lambda^{(i)}_{v,P\cup\{0\}}} \text{for } i \in [n] \setminus P. \end{cases}$$

$\mathcal{B}$ then uses each $h_{i,j}$ to compute $d_{i,j}$ and continues the simulation according to either game.

Finally, once $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs 1 if $b = b'$ and 0 otherwise. If any assertion fails that is present in both games, $\mathcal{B}$ immediately aborts and outputs 0.

Analysis. Observe that, if the $T_{u,j}$ from the $(t-1,\ell)$-mXDH challenge are *real*, i.e., if $T_{u,j} = g_1^{\alpha_u \beta_j}$, then $\mathcal{B}$ perfectly simulates Game_4, since in this case, for $i \in Q$, it holds that

$$h_{i,j} = T_{q(i),j} = (g_1^{\beta_j})^{\alpha_{q(i)}} = \mathsf{H}_{\mathsf{PDec}}(j, c^*)^{y_i}.$$

If, however the $T_{u,j}$ are random, then $\mathcal{B}$ perfectly simulates Game_5. As a result, if the challenge tuple $\mathcal{B}$ receives is *real*, then $\mathcal{B}$ outputs 1 with probability $\Pr\big[\mathsf{Game}_{4,\mathcal{A}}(1^\lambda) = 1\big]$, and if it is random, it outputs 1 with probability $\Pr\big[\mathsf{Game}_{5,\mathcal{A}}(1^\lambda) = 1\big]$, which concludes the proof of Claim 5.23. ∎

Game_6 (Interpolation in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$): In this game, we change again how $d_{i,j}$ is computed in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. More formally, denote with P the canonical completion of $\mathsf{Cor}_{\mathsf{sk}}$ such that $\mathsf{Cor}_{\mathsf{sk}} \subseteq P \subseteq [n]$ and $|P| = t-1$. Denote $Q := P \setminus \mathsf{Cor}_{\mathsf{sk}}$. When $\mathcal{A}$ makes the first query to $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, for all $i \in Q$, the challenger samples $(d_{i,j})_{j\in[\ell]} \xleftarrow{\$} \mathbb{G}_T^\ell$. For all $i \in \mathsf{Cor}_{\mathsf{sk}}$, compute $(d_{i,j})_{j\in[\ell]}$ as in Game_5, and for all remaining $i \in [n] \setminus P$ and all $j \in [\ell]$, interpolate

$$d_{i,j} \leftarrow k_j^{\lambda^{(i)}_{0,P\cup\{0\}}} \cdot \prod_{u \in P} d_{u,j}^{\lambda^{(i)}_{u,P\cup\{0\}}},$$

where $k_j = e(g_1, \mathsf{H}_{\mathsf{id}}(j))^{rx}$ are the inputs to $\mathsf{H}_{\mathsf{mask}}$ in the challenge ciphertext. In subsequent $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ queries, partial decryptions are set consistently.

Observe that the above is equivalent to sampling a random $< t$ degree polynomial f_j for all $j \in [\ell]$ that is consistent with $f_j(0) = w_j r x$ and the discrete logarithms of real partial decryptions $f_j(i) = w_j r x_i + w_{j,c^*} y_i$ for all $i \in \mathsf{Cor}_{\mathsf{sk}}$. Here, $\mathsf{H}_{\mathsf{id}}(j) = g_2^{w_j}$ and $\mathsf{H}_{\mathsf{PDec}}(j, c^*) = g_1^{w_{j,c^*}}$. Partial decryptions can then be expressed as $d_{i,j} = e(g_1, g_2)^{f_j(i)}$.

Claim 5.24 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\big[\mathsf{Game}_{6,\mathcal{A}}(1^\lambda) = 1\big] = \Pr\big[\mathsf{Game}_{5,\mathcal{A}}(1^\lambda) = 1\big].$$

Again, the proof is analogous to that of Claim 5.14.

Game_7 (Replace k_j with $e(c_1^*, \mathsf{H}_{\mathsf{id}}(j))^x$ in the challenge ciphertext): In Game_7, the challenger computes

$$k_j \leftarrow e(c_1^*, \mathsf{H}_{\mathsf{id}}(j))^x$$

in the challenge ciphertext for all $j \in [\ell]$ instead of $k_j \leftarrow e(\mathsf{pk}, \mathsf{H}_{\mathsf{id}}(j))^r$.

Claim 5.25 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\big[\mathsf{Game}_{7,\mathcal{A}}(1^\lambda) = 1\big] = \Pr\big[\mathsf{Game}_{6,\mathcal{A}}(1^\lambda) = 1\big].$$

Since $\mathsf{pk} = g_1^x$ and $c_1^* = g_1^r$, we have that

$$e(c_1^*, \mathsf{H}_{\mathsf{id}}(j))^x = e(g_1, \mathsf{H}_{\mathsf{id}}(j))^{xr} = e(\mathsf{pk}, \mathsf{H}_{\mathsf{id}}(j))^r.$$

Game_8 (Replace all occurrences of k_j with Random): In this game, we consistently replace all occurrences of k_j with a random $Z_j \xleftarrow{\$} \mathbb{G}_T$ for all $j \in [\ell]$. That is, the challenger samples $(Z_j)_{j\in[\ell]} \xleftarrow{\$} \mathbb{G}_T^\ell$ and now computes $c_2^* \leftarrow \mathsf{H}_{\mathsf{mask}}((Z_j)_{j\in[\ell]}) \oplus m_b$ and, in $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, computes $d_{i,j} \leftarrow Z_j^{\lambda^{(i)}_{0,P\cup\{0\}}} \cdot \prod_{u\in P} d_{u,j}^{\lambda^{(i)}_{u,P\cup\{0\}}}$ for all $i \in [n] \setminus P$ and $j \in [\ell]$.

Claim 5.26 For all PPT adversaries $\mathcal{A}$ there exists a PPT reduction $\mathcal{B}$ such that

$$\left|\Pr\left[\mathsf{Game}_{8,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{7,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \ell \cdot \mathsf{Adv}^{\mathsf{BDDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda).$$

Proof of Claim 5.26. We prove the claim using a hybrid argument. Let $\mathcal{H}_0$ be identical to Game_7 and, for all $u \in [\ell]$, let $\mathcal{H}_u$ be the hybrid that replaces k_j with a random Z_j for all $j \leq u$. Clearly, it holds that $\mathcal{H}_\ell$ is identical to Game_8.

Let $\mathcal{A}$ be any PPT distinguisher. We construct a reduction $\mathcal{B}$ that plays against the BDDH challenger and initially receives a BDDH tuple

$$\left(A \coloneqq g_1^\alpha, B_1 \coloneqq g_1^\beta, B_2 \coloneqq g_2^\beta, C \coloneqq g_2^\gamma, T\right),$$

where either $T = e(g_1, g_2)^{\alpha\beta\gamma}$ or T is a random element from $\mathbb{G}_T$.

$\mathcal{B}$ initially picks a random $u \in [\ell]$ and can simulate either $\mathcal{H}_u$ or $\mathcal{H}_{u-1}$ to $\mathcal{A}$ by embedding the BDDH challenge into k_u and simulating such that $\mathsf{pk} \coloneqq A$, $c_1^* \coloneqq B_1$, programming the random oracle such that $\mathsf{H}_{\mathsf{id}}(u) \coloneqq C$, and setting $k_u \coloneqq T$. $\mathcal{B}$ can simulate static corruptions by sampling random $x_i \xleftarrow{\$} \mathbb{Z}_p$ for all $i \in \mathsf{Cor}_{\mathsf{sk}}$ and compute $g_1^{x_i}$ for all $i \in [n] \setminus \mathsf{Cor}_{\mathsf{sk}}$ using interpolation.

$\mathcal{B}$ simulates the random oracles $\mathsf{H}_{\mathsf{mask}}$ and $\mathsf{H}_{\mathsf{PDec}}$ lazily. To simulate H_{id}, $\mathcal{B}$ embeds C into $\mathsf{H}_{\mathsf{id}}(u)$. For all $j > u$, $\mathcal{B}$ samples $w_j \xleftarrow{\$} \mathbb{Z}_p$ and embeds $\mathsf{H}_{\mathsf{id}}(j) \coloneqq g_2^{w_j}$. Otherwise, $\mathcal{B}$ samples H_{id} in the standard lazy manner. $\mathcal{B}$ can simulate $\mathcal{O}^{\mathsf{PDec}}$ queries as mandated by either hybrid, because it can compute $g_1^{x_i}$ for all $i \in [n]$. To simulate the challenge ciphertext, $\mathcal{B}$ sets $c_1^* \leftarrow B_1$, samples $k_j \coloneqq Z_j \xleftarrow{\$} \mathbb{G}_T$ for all $j < u$ and computes $k_j \leftarrow e(A, B_2)^{w_j}$ for all $j > u$. It then computes c_2^* as

$$c_2^* \leftarrow \mathsf{H}_{\mathsf{mask}}\left((Z_j)_{j\in[u-1]} \mid (T) \mid (k_j)_{j\in[u+1,\ell]}\right) \oplus m_b,$$

where $\mid$ denotes concatenation of vectors.

$\mathcal{B}$ continues the simulation until $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, where it uses the k_j as above when computing $d_{i,j}$ for all $i \in [n], j \in [\ell]$. Finally, $\mathcal{B}$ outputs 1 whenever $\mathcal{A}$ wins the simulated game and 0 otherwise.

Analysis. Observe that $\mathcal{B}$ simulates either hybrid such that $\alpha = x$, $\beta = r$, $\gamma = w_u$ (i.e., the dlog of $\mathsf{H}_{\mathsf{id}}(u)$). If the challenge T is real, i.e., $T = e(g_1, g_2)^{\alpha\beta\gamma}$, then $\mathcal{B}$ simulates such that $k_u = T \equiv e(g_1, \mathsf{H}_{\mathsf{id}}(u))^{xr}$, which corresponds to $\mathcal{H}_{u-1}$. If, however, T is a uniform random element from $\mathbb{G}_T$, then $\mathcal{B}$ perfectly simulates $\mathcal{H}_u$. Since $\mathcal{B}$ outputs 1 whenever $\mathcal{A}$ wins the simulated game, we get that for all $u \in [\ell]$

$$\left|\Pr\left[\mathcal{H}_{u,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{H}_{u-1,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{BDDH}}_{\mathcal{B}_u,\mathcal{G}}(1^\lambda),$$

where $\mathcal{B}_u$ denotes the reduction above for a given u. A routine calculation gives us

$$\left|\Pr\left[\mathcal{H}_{\ell,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{H}_{0,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \ell \cdot \mathsf{Adv}^{\mathsf{BDDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda),$$

which concludes the proof of Claim 5.26. ∎

Game_9 ($\mathcal{A}$ cannot compute $(Z_j)_{j\in[\ell]}$ before making an $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query): In Game_9, we modify the behavior of $\mathsf{H}_{\mathsf{mask}}$ queries made by the adversary on $(Z_j)_{j\in[\ell]}$ prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query. In particular, prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, the challenger answers $\mathsf{H}_{\mathsf{mask}}$ queries made directly or indirectly on the target value $(Z_j)_{j\in[\ell]}$ by consistently outputting a random element from $\{0,1\}^\mu$ instead of the actual value of $\mathsf{H}_{\mathsf{mask}}((Z_j)_{j\in[\ell]})$.

Claim 5.27 For all PPT adversaries $\mathcal{A}$ that make at most $q_{\mathsf{H}_{\mathsf{mask}}}$ total direct and indirect queries to $\mathsf{H}_{\mathsf{mask}}$, it holds that

$$\left|\Pr\left[\mathsf{Game}_{9,\mathcal{A}}(1^\lambda)=1\right]-\Pr\left[\mathsf{Game}_{8,\mathcal{A}}(1^\lambda)=1\right]\right| \leq \frac{q_{\mathsf{H}_{\mathsf{mask}}}}{p^\ell}.$$

Proof of Claim 5.27. Both games are identical, unless the adversary directly or indirectly queries $\mathsf{H}_{\mathsf{mask}}$ on $(Z_j)_{j\in[\ell]}$ prior to querying $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. At this point $\mathcal{A}$'s view is entirely independent of $(Z_j)_{j\in[\ell]}$, except for c_2^*, which only depends on the random oracle *output* $\mathsf{H}_{\mathsf{mask}}((Z_j)_{j\in[\ell]})$. Thus, the claim follows from a union bound over the $q_{\mathsf{H}_{\mathsf{mask}}}$ guessing opportunities. ∎

Note that the core lemma does not execute the functions submitted by the adversary if the adversary's budget is $|\mathsf{Cor}\cup S^*|\geq t-1$. In order to apply the core lemma, we must first argue that such motivation functions are useless to an adversary.

Game_{10} (Never evaluate useless motivation functions): Within $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S,\varphi)$, if $|\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}}|\geq t-1$, the challenger only executes motivation functions for parties in $S\cap(\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}})$. For the remaining parties in $S\setminus(\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}})$ the challenger sets the output of the motivation function to 0 *without* evaluating it.

Claim 5.28 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\mathsf{Game}_{10,\mathcal{A}}(1^\lambda)=1\right]\geq\Pr\left[\mathsf{Game}_{9,\mathcal{A}}(1^\lambda)=1\right].$$

Proof of Claim 5.28. Clearly Game_{10} and Game_9 are identical until $|\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}}|$ reaches $t-1$ and the first subsequent $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S,\varphi)$ query is made by $\mathcal{A}$ where a motivation function hits $\varphi(\mathsf{pp},\mathsf{pk},i,d_i,c^*)=1$ for some $i\in S\setminus(\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}})$. Denote that i as i^*. Note, however, that immediately after, the oracle in Game_9 would add i^* to $\mathsf{Cor}_{\mathsf{pd}}$. Since i^* was neither in $\mathsf{Cor}_{\mathsf{sk}}$ nor $\mathsf{Cor}_{\mathsf{pd}}$ previously, their union increases by at least one, and we have $|\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}}|\geq t$, meaning the adversary can no longer win in Game_9. As a result, a run is either identical in both games or loses in Game_9, yielding the claim. ∎

The purpose of the next game is to bound the probability that $\mathcal{A}$ ever queries $\mathsf{H}_{\mathsf{mask}}((Z_j)_{j\in[\ell]})$, either directly, or through a motivation function, by a reduction to $\mathsf{GameStat}$. Here, queries prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query can be disregarded due to the change introduced in Game_9.

Game_{11} ($\mathcal{A}$ cannot query $(Z_j)_{j\in[\ell]}$): In Game_{11}, the challenger aborts if the adversary queries $(Z_j)_{j\in[\ell]}$ to the $\mathsf{H}_{\mathsf{mask}}$ oracle while $|\mathsf{Cor}_{\mathsf{sk}}\cup\mathsf{Cor}_{\mathsf{pd}}|<t$, *either directly or through a motivation function* (in either $\mathcal{O}_1^{\mathsf{PDec}}$ or $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$). This abort only triggers if such a query happens starting from the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query.

Claim 5.29 For all PPT adversaries $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathsf{Game}_{11,\mathcal{A}}(1^\lambda)=1\right]-\Pr\left[\mathsf{Game}_{10,\mathcal{A}}(1^\lambda)=1\right]\right|\leq(n+1)\cdot q_{\mathsf{H}_{\mathsf{mask}}}\cdot\frac{\binom{n}{t-1}}{p^\ell},$$

where $q_{\mathsf{H}_{\mathsf{mask}}}$ bounds the total number of queries made to $\mathsf{H}_{\mathsf{mask}}$, either by $\mathcal{A}$ directly, or indirectly through motivation functions.

Proof of Claim 5.29. The proof follows a very similar structure to that of Claim 5.19. Let $\mathcal{A}$ be a PPT adversary against either game and denote with Bad the event that $\mathcal{A}$ triggers the additional abort condition. Since both games are identical unless Bad happens, we get that

$$\left|\Pr\left[\mathsf{Game}_{11,\mathcal{A}}(1^\lambda)=1\right]-\Pr\left[\mathsf{Game}_{10,\mathcal{A}}(1^\lambda)=1\right]\right|\leq\Pr[\mathsf{Bad}].$$

We next bound $\Pr[\mathsf{Bad}]$ by a reduction $\mathcal{B}$ to $\mathsf{GameStat}(n,t,p,\ell,Q)$, making use of Lemma 5.1, where Q bounds the number of $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ queries made by $\mathcal{A}$. Note that Lemma 5.1 does not bound the runtime of the $\mathsf{GameStat}$ adversary, hence $\mathcal{B}$ can be unbounded. The reduction $\mathcal{B}$ perfectly simulates either game up to the generation of the challenge ciphertext: For key generation, $\mathcal{B}$ simulates as mandated by either game. In particular, this means that $\mathcal{B}$ knows x as well as all key shares $\mathsf{sk}_i=(x_i,y_i)$.

Simulating Random Oracles. $\mathcal{B}$ pre-generates exhaustive tables for all random oracles $\mathsf{H}_{\mathsf{mask}}$, $\mathsf{H}_{\mathsf{PDec}}$ and H_{id}, and answers queries by consulting the table. We write $\mathsf{H}_{\mathsf{id}}(j)=g_2^{w_j}$ for all $j\in[\ell]$. For

$\mathsf{H}_{\mathsf{PDec}}$ and H_{id}, $\mathcal{B}$ will do so consistently throughout, while we will later adjust how $\mathcal{B}$ simulates $\mathsf{H}_{\mathsf{mask}}$ starting from $\mathcal{A}$'s first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query.

Simulating the Challenge Ciphertext. To simulate the challenge ciphertext c^* for messages m_0, m_1, $\mathcal{B}$ samples a random $r \xleftarrow{\$} \mathbb{Z}_p$ and sets $c_1^* \leftarrow g_1^r$. Instead of sampling $(Z_j)_{j\in[\ell]}$ and setting $c_2^* \leftarrow \mathsf{H}_{\mathsf{mask}}((Z_j)_{j\in[\ell]}) \oplus m_b$, $\mathcal{B}$ just samples a random $H \xleftarrow{\$} \{0,1\}^\mu$ and sets

$$c_2^* \leftarrow H \oplus m_b$$

for a random bit b. Note that as such, $\mathcal{B}$ essentially programs $\mathsf{H}_{\mathsf{mask}}((Z_j)_{j\in[\ell]}) := H$ without knowing $(Z_j)_{j\in[\ell]}$ yet. We write $\mathsf{H}_{\mathsf{PDec}}(j, c^*) = g_1^{w_{j,c^*}}$ for all $j \in [\ell]$. Clearly, $\mathcal{B}$ knows all w_{j,c^*} as it is unbounded.

Simulating $\mathcal{O}^{\mathsf{PDec}}$. $\mathcal{B}$ can simulate $\mathcal{O}^{\mathsf{PDec}}$ exactly as mandated by either game. In particular, $\mathcal{B}$ executes the motivation functions and simulates the random oracles to them.

Simulating $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S, \varphi)$, $\mathcal{B}$ embeds this into a query to its challenger as follows. It computes

$$z_{i,j} \leftarrow w_j r x_i + w_{j,c^*} y_i$$

for all $i \in \mathsf{Cor}_{\mathsf{sk}}$ and all $j \in [\ell]$, and sets $z_i := (z_{i,j})_{j\in[\ell]}$ for all $i \in \mathsf{Cor}_{\mathsf{sk}}$.

Only upon the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query... If this is the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, $\mathcal{B}$ forwards $(\mathsf{Cor}_{\mathsf{sk}}, (z_i)_{i\in\mathsf{Cor}_{\mathsf{sk}}})$ to its challenger, causing the challenger to initialize $\mathsf{Cor} \leftarrow \mathsf{Cor}_{\mathsf{sk}}$ and to sample the vector of polynomials $\boldsymbol{f}$ that is consistent with $(z_i)_{i\in\mathsf{Cor}_{\mathsf{sk}}}$. At this point, $\mathcal{B}$ has access to the $\mathcal{O}^{\mathsf{check}}$ oracle of its challenger, and now $\mathcal{B}$ adjusts how it simulates the random oracle $\mathsf{H}_{\mathsf{mask}}$: Whenever $\mathcal{A}$ or a motivation function makes a query to $\mathsf{H}_{\mathsf{mask}}$ on input $(V_j)_{j\in[\ell]}$, $\mathcal{B}$ first extracts the discrete logarithm $v_j \leftarrow \mathsf{DLog}_{e(g_1,g_2)}(V_j)$ for all $j \in [\ell]$, assembles $\boldsymbol{v} \leftarrow (v_j)_{j\in[\ell]}$, and queries $\mathcal{O}^{\mathsf{check}}(\boldsymbol{v})$. If $\boldsymbol{v} = \boldsymbol{f}(0)$, or equivalently, $(V_j)_{j\in[\ell]} = e(g_1, g_2)^{\boldsymbol{f}(0)}$, and the current budget is $|\mathsf{Cor} \cup S^*| < t$, this query makes $\mathcal{B}$ win immediately, ending the simulation. If, instead, $|\mathsf{Cor} \cup S^*| \geq t$, the oracle returns 1 without causing $\mathcal{B}$ to win. In this case, the remainder of the run is vacuous because, given the non-decreasing nature of the budget, the event Bad can no longer occur. Finally, if the queried value is not the target, the response to the $\mathsf{H}_{\mathsf{mask}}$ query is taken from the pre-computed table.

On every $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query... $\mathcal{B}$ then computes

$$d_{i,j} \leftarrow e(g_1, g_2)^{z_{i,j}} \quad [\equiv e(c_1^*, \mathsf{H}_{\mathsf{id}}(j))^{x_i} \cdot e(\mathsf{H}_{\mathsf{PDec}}(j, c^*), g_2)^{y_i}]$$

for all $i \in \mathsf{Cor}_{\mathsf{sk}}$ and generates the circuit ρ such that

$$\rho(i, v_i := (v_{i,j})_{j\in[\ell]}) = \begin{cases} \varphi(\mathsf{pp}, \mathsf{pk}, i, d_i := (e(g_1, g_2)^{v_{i,j}})_{j\in[\ell]}, c^*), & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases}.$$

Additionally, $\mathcal{B}$ hardcodes the random oracle tables for $\mathsf{H}_{\mathsf{mask}}$, H_{id} and $\mathsf{H}_{\mathsf{PDec}}$ into ρ. Whenever the internal φ makes a random oracle query, ρ answers from the table, except for $\mathsf{H}_{\mathsf{mask}}$, where ρ first extracts the discrete logarithm and forwards it to $\mathcal{O}^{\mathsf{check}}$ as outlined above. This means that $\mathcal{B}$ wins whenever a motivation function calls $\mathsf{H}_{\mathsf{mask}}(g_1^{\boldsymbol{f}(0)})$ within $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$.

Edge case if $|\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}| \geq t - 1$. If $|\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}| \geq t - 1$ and $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec\text{-}chal}}(S, \varphi)$, $\mathcal{B}$ no longer needs to run motivation functions for all $S \setminus (\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}})$ due to the change introduced in Game_{10}. However, $\mathcal{B}$ still needs to simulate motivation functions for $\bar{S} = S \cap (\mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}})$.[35] For $\bar{S}$, $\mathcal{B}$ cannot rely on simply forwarding ρ to its challenger, because $\mathsf{GameStat}$ no longer executes motivation functions while $|\mathsf{Cor} \cup S^*| \geq t - 1$. Hence, $\mathcal{B}$ instead simulates the motivation functions for $\bar{S}$ locally. For all $i \in \bar{S}$, $\mathcal{B}$ distinguishes the following two cases. If $i \in S \cap \mathsf{Cor}_{\mathsf{sk}}$, $\mathcal{B}$ uses the partial decryption $d_{i,j} = e(g_1, g_2)^{z_{i,j}}$ it computed from i's secret key share directly and then runs $\varphi(\mathsf{pp}, \mathsf{pk}, i, (d_{i,j})_{j\in[\ell]}, c^*)$. If, instead, $i \in S \cap \mathsf{Cor}_{\mathsf{pd}}$, it must be the case that a prior $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query already output $(d_{i,j})_{j\in[\ell]}$, and $\mathcal{B}$ uses that partial decryption to simulate the motivation function.[36] Whenever the motivation function makes a query to $\mathsf{H}_{\mathsf{mask}}$ in either case, $\mathcal{B}$ first extracts the discrete logarithm and forwards the result to $\mathcal{O}^{\mathsf{check}}$ (as a direct query), i.e., $\mathcal{B}$ also wins if motivation functions query $\mathsf{H}_{\mathsf{mask}}$ on $e(g_1, g_2)^{\boldsymbol{f}(0)}$ for any $i \in \bar{S}$. Denote as $\bar{S}'$ the set of parties where the above execution of the motivation function outputs 1.

[35] Intuitively, this is because even if these motivation functions output 1, this would not affect the adversary's budget, as $\bar{S} \subseteq \mathsf{Cor}_{\mathsf{sk}} \cup \mathsf{Cor}_{\mathsf{pd}}$ already.

[36] Crucially, we are using the fact that PDec is deterministic in Π_{Pair}.

Then, after dealing with the potential edge case, $\mathcal{B}$ forwards ρ to its challenger.[37] As a response, $\mathcal{B}$ receives the set S' as well as the tuple $(v_i := \boldsymbol{f}(i))_{i\in S'}$, where $\boldsymbol{f}$ is the vector of polynomials sampled by the $\mathsf{GameStat}$ challenger that is now consistent with the discrete logarithms of partial decryptions on the challenge ciphertext. $\mathcal{B}$ parses $(v_{i,j})_{j\in[\ell]} \leftarrow \boldsymbol{v}_i$ and sets $d_{i,j} \leftarrow e(g_1,g_2)^{v_{i,j}}$ for all $i \in S'$ and $j \in [\ell]$. Finally, $\mathcal{B}$ assembles $d_i := (d_{i,j})_{j\in[\ell]}$ and returns $(S' \cup \bar{S}', (d_i)_{i\in S'\cup\bar{S}'})$ to $\mathcal{A}$ as the output of $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$. At the end of this query, the challenger updates $S^* \leftarrow S^* \cup S'$ and $\mathcal{B}$ updates the initially empty $\mathsf{Cor_{pd}} \leftarrow \mathsf{Cor_{pd}} \cup S'$. For subsequent $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ queries, $\mathcal{B}$ simulates in an identical manner. Note that, with the above simulation, the set $\mathsf{Cor_{sk}}$ of $\mathcal{B}$'s simulation is always the same as the set Cor of the $\mathsf{GameStat}$ challenger, and $\mathsf{Cor_{pd}}$ of $\mathcal{B}$'s simulation mirrors the set S^* of the $\mathsf{GameStat}$ challenger.

Analysis. Consider the case that the event Bad happens. In this case, $\mathcal{B}$'s simulation of either game to $\mathcal{A}$ is perfect until the winning $\mathsf{H_{mask}}$ happens, either directly or within a motivation function. In particular, $\mathcal{B}$ perfectly simulates for $(Z_j)_{j\in[\ell]} = e(g_1,g_2)^{\boldsymbol{f}(0)}$ and $\mathsf{H_{mask}}((Z_j)_{j\in[\ell]}) = H$, where $\boldsymbol{f}$ is the ℓ-tuple of $< t$ degree polynomials sampled by the $\mathsf{GameStat}$ challenger that is consistent with the discrete logarithms of $d_{i,j}$ for all $i \in \mathsf{Cor_{sk}}$.[38] Considering the polynomial view of the discrete logarithms of partial decryptions introduced in Game_6, $\mathcal{B}$ simulates perfectly for $(f_j)_{j\in[\ell]} = \boldsymbol{f}$. As a result, if Bad happens, the input of the winning query is equal to $(Z_j)_{j\in[\ell]} = e(g_1,g_2)^{\boldsymbol{f}(0)}$ and $\mathcal{B}$ queries $\mathcal{O}^{\mathsf{check}}(\boldsymbol{f}(0))$ to its challenger. Also, Bad implies that $|\mathsf{Cor_{sk}} \cup \mathsf{Cor_{pd}}| < t$, and thus the condition $|\mathsf{Cor} \cup S^*| < t$ passes in $\mathsf{GameStat}$ and $\mathcal{B}$ makes at most $q_{\mathsf{H_{mask}}}$ (direct and indirect) queries to $\mathcal{O}^{\mathsf{check}}$. This allows us to bound

$$\Pr[\mathsf{Bad}] \le \Pr[\mathsf{GameStat}_{\mathcal{B}}(n,t,p,\ell,Q) = 1].$$

The claim then follows directly from Lemma 5.1. ■

Final Bound on Game_{11}: Observe that in any winning execution of Game_{11}, the adversary cannot query $\mathsf{H_{mask}}$ on $(Z_j)_{j\in[\ell]}$, neither directly, nor via any submitted motivation function that is ever evaluated starting from the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query. If the adversary should issue such a $\mathsf{H_{mask}}$ query prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, then Game_{11} consistently returns a random value instead of the actual $\mathsf{H_{mask}}((Z_j)_{j\in[\ell]})$ value used in the challenge ciphertext. Thus $c_2^* = \mathsf{H_{mask}}((Z_j)_{j\in[\ell]}) \oplus m_b$ perfectly hides m_b behind a one-time-pad. As this is the only element dependent on b in the adversary's view, it holds that

$$\Pr\big[\mathsf{Game}_{11,\mathcal{A}}(1^\lambda) = 1\big] \le \frac{1}{2}.$$

Putting it all together: When applying Claims 5.21-5.29 alongside the above bound on the advantage in Game_{11}, we get that for all PPT adversaries $\mathcal{A}$ there exist PPT reductions $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ such that

$$\begin{aligned}\Pr\big[\mathsf{Game}_{0,\mathcal{A}}(1^\lambda) = 1\big] \le \frac{1}{2} + 2\cdot \mathsf{Adv}^{\mathsf{SimExt}}_{\mathcal{B}_1,(\mathsf{PS_R},\mathsf{Sim})}(1^\lambda) + \mathsf{Adv}^{(t-1,\ell)\text{-mXDH}}_{\mathcal{B}_2,\mathcal{G}}(1^\lambda) + \ell\cdot \mathsf{Adv}^{\mathsf{BDDH}}_{\mathcal{B}_3,\mathcal{G}}(1^\lambda)\\ + \frac{q_{\mathsf{H_{PDec}}}}{p} + \frac{\ell}{p} + \frac{q_{\mathsf{H_{mask}}}}{p^\ell} + (n+1)\frac{q_{\mathsf{H_{mask}}}\binom{n}{t-1}}{p^\ell}.\end{aligned}$$

Whenever $\log(p^\ell) \ge n + \lambda$, it holds that

$$\frac{\binom{n}{t-1}}{p^\ell} \le \frac{2^n}{2^{n+\lambda}} \le 2^{-\lambda},$$

and thus the advantage is negligible assuming $\log(p) \ge \lambda$, $\ell, q_{\mathsf{H_{mask}}}, q_{\mathsf{H_{PDec}}} \in \mathsf{poly}(\lambda)$, simulation-extractability of PS, and that the XDH and BDDH assumptions hold for $\mathcal{G}$.

■

[37] Note that, if $|\mathsf{Cor_{sk}} \cup \mathsf{Cor_{pd}}| \ge t-1$, i.e., if the edge case above was hit, the challenger will not run ρ and this is a no-op, returning an empty set back to $\mathcal{B}$.

[38] Notably, the simulation of $\mathsf{H_{mask}}$ is consistent whenever Bad happens, because $\mathcal{B}$'s simulation via the pre-computed table on input $(Z_j)_{j\in[\ell]} = e(g_1,g_2)^{\boldsymbol{f}(0)}$ prior to the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query is consistent with the modification introduced in Game_9. For $\mathsf{H_{mask}}((Z_j)_{j\in[\ell]})$ queries *after* the first $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$ query, $\mathcal{B}$ is supposed to return H. Here, $\mathcal{B}$ always queries $\mathcal{O}^{\mathsf{check}}$ on its discrete logarithm. Since Bad requires $|\mathsf{Cor_{sk}} \cup \mathsf{Cor_{pd}}| < t$ to trigger, and the budget is non-decreasing, the same must hold at all earlier points of the run, and the first query triggering Bad makes $\mathcal{B}$ win.

5.5 Avoiding Simulation-Extractability

A limitation of the Π_{ZEG} and Π_{Pair} constructions is their reliance on simulation-extractable NIZKs with straight-line extraction. Typically, one would instantiate these with Schnorr-style proofs of knowledge, which can be proven to be simulation-extractable in the AGM. To avoid reliance on the AGM, one could instead apply the Fischlin transform [45] to the underlying protocol, at the cost of a larger proof size.

We next briefly outline[39] how one can adjust the construction and proof slightly to avoid simulation-extractability entirely in Π_{ZEG} and Π_{Pair}, and instead use only simulation-soundness (Definition 2.4). Simulation-soundness can be easily and efficiently achieved for the DLEQ relation we are interested in via Chaum-Pedersen proofs [28, 46, 43] in the random oracle model, without relying on the AGM.

Simulating $\mathcal{O}^{\mathsf{PDec}}$ without Simulation-Extractability. In our proofs for both Π_{ZEG} and Π_{Pair}, we use simulation-extractability to simulate (non-challenge) partial decryptions in $\mathcal{O}^{\mathsf{PDec}}$ without knowledge of honest parties' secret key shares sk_i. Henceforth, let us first consider the pairing-based scheme Π_{Pair}. In the proof of Π_{Pair}, a reduction simulating $\mathcal{O}^{\mathsf{PDec}}$ without knowledge of x or x_i for honest parties i can extract the encryption randomness r from the ciphertext and compute pk^r, which is equivalent to c_1^x. This allows the reduction to simulate all partial decryptions via interpolation (cf. Claim 5.25). One can adapt an approach originating in the CCA secure TDH1 scheme of Shoup and Gennaro [67] that avoids an analogous extraction problem by relying only on the soundness of a DLEQ proof. Namely, one can compute c_1^x without explicit extraction, and then continue as before.

In more detail, we let the encryptor derive a new random group element linked to their ciphertext via a random oracle $\mathsf{H}_{\mathsf{lnk}}\colon \{0,1\}^* \to \mathbb{G}_1$ as

$$Y \leftarrow \mathsf{H}_{\mathsf{lnk}}(c_1, c_2),$$

and then compute $V = Y^r$, where r is the encryption randomness (i.e., $c_1 = g_1^r$). The element V will be included in the ciphertext. Observe that, as a result (g_1, Y, c_1, V) is a Diffie-Helman tuple such that

$$\mathsf{DLog}_{g_1}(c_1) = \mathsf{DLog}_Y(V) \quad [= r].$$

Now, we require the encryptor to prove the above statement via a simulation-sound proof system (via DLEQ), attaching a proof π such that $c = (c_1, c_2, V, \pi)$.

As a result, we can simulate partial decryptions without needing to extract the witness r as follows. The reduction programs $\mathsf{H}_{\mathsf{lnk}}(c_1, c_2)$ on every input by consistently sampling random $\vartheta_{c_1,c_2} \xleftarrow{\$} \mathbb{Z}_p$ and programming $\mathsf{H}_{\mathsf{lnk}}(c_1, c_2) \coloneqq \mathsf{pk} \cdot g_1^{\vartheta_{c_1,c_2}}$.[40] Now, whenever the adversary submits a ciphertext $c = (c_1, c_2, V, \pi)$ with a valid proof π to $\mathcal{O}^{\mathsf{PDec}}$ (and $c \neq c^*$), the reduction can compute

$$X \leftarrow V \cdot c_1^{-\vartheta_{c_1,c_2}}.$$

Because of (simulation-)soundness of the proof system, it holds that

$$X = V \cdot c_1^{-\vartheta_{c_1,c_2}} = Y^r \cdot g_1^{-r\vartheta_{c_1,c_2}} = \mathsf{pk}^r \cdot g_1^{r\vartheta_{c_1,c_2}} \cdot g_1^{-r\vartheta_{c_1,c_2}} = c_1^x,$$

where $r = \mathsf{DLog}_{g_1}(c_1) = \mathsf{DLog}_Y(V)$, and $\mathsf{pk} = g_1^x$. Now, given c_1^x, the reduction can simulate all partial decryptions via interpolation.

It remains to argue that the additional ciphertext-element V and the proof π do not leak sensitive information from the challenge ciphertext. For this, one can first apply zero-knowledge to replace π with a simulated proof, and then apply XDH to replace V with a random group element in the challenge ciphertext. Then, one can use the strategy outlined above to simulate partial decryptions.

Applying it to Π_{ZEG}. Recall that, to simulate $\mathcal{O}^{\mathsf{PDec}}$ in Π_{Pair}, it is sufficient to compute c_1^x, as it allows to compute partial decryptions for the different $j \in [\ell]$ by pairing with $\mathsf{H}_{\mathsf{id}}(j)$. In Π_{ZEG}, however, the encryptor samples a random r_j for all $j \in [u]$, and there are public keys pk_k for all $k \in [v]$. As a result, when trying to simulate $\mathcal{O}^{\mathsf{PDec}}$, the reduction needs to compute $c_{1,j}^{x_k}$ for *all pairs* $j \in [u]$, $k \in [v]$, where $c_{1,j} = g^{r_j}$. Naively translating our strategy from above, we would require the encryptor to include $V_{j,k} = \mathsf{H}_{\mathsf{lnk}}(j, k, c_1, c_2)^{r_j}$ for all $(j, k) \in [u] \times [v]$, such that we can embed $\mathsf{H}_{\mathsf{lnk}}(j, k, c_1, c_2) \coloneqq \mathsf{pk}_k \cdot g^{\vartheta_{j,k,c_1,c_2}}$

[39] Note that this is not the technical focus of our work, and therefore we only sketch this modification.

[40] Note that all reductions simulate $\mathsf{H}_{\mathsf{lnk}}$ both to the adversary and motivation functions.

for random ϑ_{j,k,c_1,c_2} and compute $c_{1,j}^{x_k} \leftarrow V_{j,k} \cdot c_{1,j}^{-\vartheta_{j,k,c_1,c_2}}$. Unfortunately, this would add $u \cdot v$ group elements to the ciphertext, which defeats the purpose of our tradeoff, as it increases ciphertext overhead back to $O(n)$.

To keep ciphertext overhead $O(u)$, we make use of the observation that the only step in the proof of Π_{ZEG}, where the reduction does not know *all* secret keys $(x_{i,k})_{i\in[n],k\in[v]}$ is Claim 5.16, i.e., in all other steps, the reduction can compute all partial decryptions in $\mathcal{O}^{\mathsf{PDec}}$ honestly using the secret key shares directly. In Claim 5.16 of our current proof, a reduction to (u,v)-mDDH replaces all $(c_{1,j}^*)^{x_k}$ with random. To do so, it embeds part of the (u,v)-mDDH challenge into the public keys $(\mathsf{pk}_k)_{k\in[v]}$, hence it needs to simulate without knowing any x_k. Now, instead of applying the (u,v)-mDDH assumption directly, one can instead perform a more low-level analysis and execute a hybrid argument iterating over the different public key positions $q \in [v]$. I.e., hybrid $\mathcal{H}_q$ replaces $(c_{1,j}^*)^{x_k}$ with random for all $(j,k) \in [u] \times [v]$ where $k \leq q$. To bound $|\Pr[\mathcal{H}_{q,\mathcal{A}}(1^\lambda)=1] - \Pr[\mathcal{H}_{q-1,\mathcal{A}}(1^\lambda)=1]|$, we can use a reduction to $(u,1)$-mDDH that only embeds the challenge into pk_q, but knows $x_k \in \mathbb{Z}_p$ for all other $k \neq q$. The crucial observation is now that all these reductions between hybrids only need to simulate without knowing *a single key* x_q. Consequently, within a single reduction $\mathcal{B}_q$, we only ever need to use the trick above to compute $c_{1,j}^{x_k}$ for a fixed $k = q$. This allows us to reduce the group elements in the ciphertext: We make the encryptor attach $V_j = \mathsf{H}_{\mathsf{Ink}}(j,c_1,c_2)^{r_j}$ for all $j \in [u]$. Then, we can apply the soundness argument directly when bounding the hop between $\mathcal{H}_q$ and $\mathcal{H}_{q-1}$, embed $\mathsf{H}_{\mathsf{Ink}}(j,c_1,c_2) \coloneqq \mathsf{pk}_q \cdot g^{\vartheta_{j,c_1,c_2}}$ and compute $c_{1,j}^{x_q}$ for all $j \in [u]$ as $V_j \cdot c_{1,j}^{-\vartheta_{j,c_1,c_2}}$. For all remaining $k \in [v]$ with $k \neq q$, the reduction knows x_k (and $(x_{i,k})_{i\in[n]}$), and can thus compute $d_{i,j,k}$ honestly.

Efficiency Impact. We evaluate the efficiency of the Π_{ZEG} and Π_{Pair} constructions (i.e., the variants *with* simulation-extractability) in Section 7, where the proof system is instantiated with Schnorr-style proofs of knowledge of the encryption randomness. Note that, when instantiating the DLEQ proof system above with Chaum-Pedersen [28], the proof π has exactly the same size as the Schnorr proofs. Hence, avoiding simulation-extractability only costs the extra element(s) V in ciphertext size. This amounts to u group elements for Π_{ZEG} and one $\mathbb{G}_1$ element for Π_{Pair}.

6 Construction with Adaptive Internally Motivated Corruptions

In the previous section, we constructed schemes that are secure in the context of internally motivated partial decryptions. Now, we want to focus on *internally motivated corruptions*. That is, we want to construct a scheme that is secure against corruptions that depend on secret key shares sk_i.

6.1 Overview

Before we formally present a construction, we give an intuition for it and its analysis. While we will achieve CCA security for the formal construction below by adding appropriate NIZKs, we omit them from the description here and just focus on CPA security under internally motivated corruptions.

Recap. To understand the main intuition, it is worth recalling the intuition of the schemes in Section 5. The strategy there was to switch to a hybrid in which the challenge ciphertext is associated with a random polynomial f of degree $< t$, where each partial decryption corresponds to one evaluation $f(i)$. As winning the game was only possible by querying the random oracle on $f(0)$, the task was to show that $f(0)$ has sufficient remaining entropy. The main technical challenge was that partial decryption queries can "consume" entropy by learning information about *every* $f(i)$.

Main Intuition and Starting Point. If we now instead consider internally motivated corruptions, then each corruption query consumes entropy of the joint secret key sk by learning non-trivial information about *every* sk_i. Thus, if we want to follow a similar strategy as before, we need to ensure that sk has enough remaining entropy, and that sk plays the role of $f(0)$. A fundamental difference is that in the previous setting, there was *no* information about $f(0)$ (except other evaluations), whereas now we have the public key revealing information about $\mathsf{sk} = f(0)$. For instance, if we just have a standard key relation $\mathsf{pk} = g^{\mathsf{sk}}$, then information-theoretically $f(0)$ is fully determined, and we cannot hope to prove a

statistical core lemma similar to the one in Section 5. This is why our starting point is to make the key $\mathsf{pk} = g^x h^y$, $\mathsf{sk} = (x, y)$, for two independent generators $g, h \in \mathbb{G}$. In this way, there is still entropy in sk given pk.

Scheme Overview. We may note that this key relation resembles the one in the famous Cramer-Shoup encryption scheme [33, 34]. Indeed, a simplified (and thresholdized) CPA variant of Cramer-Shoup will be our candidate:[41] Let μ denote the message length and let $\mathsf{H}_{\mathsf{mask}}$ be a random oracle. Let $h \in \mathbb{G}$ be a public parameter. We define the keys as

$$\mathsf{pk} = g^x h^y, \quad \mathsf{sk} = (x, y), \quad \mathsf{sk}_i = (x_i, y_i) \quad \text{for } i \in [n],$$

where $(x_i)_{i\in[n]}$ and $(y_i)_{i\in[n]}$ are t-of-n sharings of x, y, respectively. To encrypt a message m, one constructs the ciphertext

$$(c_1, c_2, c_3) \leftarrow (g^r, h^r, \mathsf{H}_{\mathsf{mask}}(\mathsf{pk}^r) \oplus m).$$

To decrypt, each party i would reveal the partial decryption $d_i = c_1^{x_i} c_2^{y_i}$. Via interpolation in the exponent, sufficiently many of these can be combined into $c_1^x c_2^y = g^{rx} h^{ry} = \mathsf{pk}^r$. From that, parties can recover m.

Security Analysis. Let us now sketch the security analysis. Recall that our idea is to use the remaining entropy of our secret key (x, y) to argue that m is hidden. More precisely, as in the proofs in Section 5, we want to argue that the adversary will never query the relevant input to $\mathsf{H}_{\mathsf{mask}}$. Thus, we first need to make this input depend on (x, y). Concretely, in a first hybrid, we change the challenge ciphertext to

$$(c_1^*, c_2^*, c_3^*) \leftarrow (g^r, h^r, \mathsf{H}_{\mathsf{mask}}(\, c_1^{*x} c_2^{*y} \,) \oplus m).$$

Now, we are still not ready to apply a statistical argument, as an unbounded adversary can compute the highlighted term from pk and r. To solve this, we can apply the Diffie-Hellman assumption:

$$(c_1^*, c_2^*, c_3^*) \leftarrow (\, g^\beta, g^\gamma \,, \mathsf{H}_{\mathsf{mask}}(\underbrace{c_1^{*x} c_2^{*y}}_{=g^{\beta x+\gamma y}}) \oplus m), \text{ for random } \beta, \gamma.$$

Let α be the discrete logarithm of h, i.e., $h = g^\alpha$.

Statistical Core Lemma. Intuitively, at this point, the adversary's job is to compute (in the exponent) $\beta x + \gamma y$, given α, β, γ, as well as $x + \alpha y$ (the public key), and internally motivated queries on shares of (x, y). We can show that this is statistically hard. As in Section 5, we formulate this final statistical step as a core lemma. One thing to note is that the lemma even works if we let the adversary output α, β, γ, as long as the invertibility condition $\gamma \neq \alpha\beta$ holds.[42] We also note the difference to the core lemma from Section 5: here, the hidden polynomials are related to the secret key (x, y), whereas there they were related to the decryption key Z that was switched to random before.

Increasing Entropy. The statistical bound we get for the construction so far is $\binom{n}{t-1}/p$. As in Section 5, we aim to find a way to improve the bound by increasing the denominator. A natural idea is to increase entropy further using parallel repetition: we would make x and y vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^v$ which are shared componentwise. Then, we can write the public key as[43]

$$[1\ \alpha] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = [1\ \alpha] \begin{bmatrix} x^{(1)} & \dots & x^{(v)} \\ y^{(1)} & \dots & y^{(v)} \end{bmatrix} \in \mathbb{G}^v,$$

using implicit notation [41]. Generalizing our statistical core lemma, this yields a bound of the form $\binom{n}{t-1}/p^v$. In our formal construction below, we will also increase the ciphertext size so that we get a flexible tradeoff between ciphertext size and key size. Precisely, we introduce a parameter u, so that

[41] Interestingly, the main point of Cramer-Shoup is to achieve CCA security, whereas here we use the entropy in the secret key to deal with corruptions. The Cramer-Shoup cryptosystem, and more generally hash proof systems, also play a central role in the literature on leakage-resilient public key encryption [61, 51], but techniques differ.

[42] Intuitively, this condition just ensures that we did not by accident sample a valid Diffie-Hellman tuple.

[43] Recall that we view vectors as row vectors by default.

the denominator in our bound becomes p^{uv}, where the ciphertext now contains $\Theta(u^2)$ group elements, and the public keys contain $\Theta(v)$ many elements. The scheme is easily presented when using the matrix notation hinted at above. Namely, the secret key is a matrix $\boldsymbol{X} \xleftarrow{\$} \mathbb{Z}_p^{(1+u)\times v}$, which is shared in a componentwise fashion. The public key is $\mathsf{pk} = [\boldsymbol{a}]\boldsymbol{X}$ for $[\boldsymbol{a}] \leftarrow ([1] \mid [\boldsymbol{\alpha}]) \in \mathbb{G}^{1+u}$. A ciphertext has the form $(\boldsymbol{r}^\top[\boldsymbol{a}], \mathsf{H}_{\mathsf{mask}}(\boldsymbol{r}^\top \mathsf{pk}) \oplus m)$ for encryption randomness $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^u$.

6.2 Second Statistical Core Lemma

Before we formally present the scheme and its analysis, we first state and prove the statistical core lemma. We assume that the reader is familiar with the rough outline sketched in our overview. The core lemma that we prove here is more general, and in particular is written in matrix form so that it can be used for general parameters u, v as explained above. Assume that the parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}$ are given (by the adversary). Intuitively, $\boldsymbol{\alpha}$ defines the public parameters, whereas $\boldsymbol{\beta}, \boldsymbol{\Gamma}$ correspond to the encryption randomness of the scheme (after a Diffie-Hellman switch). The secret key is a matrix $\boldsymbol{X} = \boldsymbol{f}(0)$, where $\boldsymbol{f}$ is a matrix of random polynomials of degree $< t$. The adversary's job in the core lemma is to compute $(\boldsymbol{\beta}^\top \mid \boldsymbol{\Gamma})\boldsymbol{f}(0)$ (the input to $\mathsf{H}_{\mathsf{mask}}$ after the switch) and submit it to $\mathcal{O}^{\mathsf{check}}$ either directly or through a motivation function without overshooting its budget, given $(1 \mid \boldsymbol{\alpha})\boldsymbol{f}(0)$ (representing the public key), and adaptive internally motivated queries that reveal $\boldsymbol{f}(i)$ for some i. We also allow the adversary to pick the secret key shares of a set Cor of corrupted parties, denoted by $(\boldsymbol{X}_i)_{i\in\mathsf{Cor}}$, we give all "public key shares" $(1 \mid \boldsymbol{\alpha})\boldsymbol{f}(i)$ to the adversary,[44] and we adapt the invertibility condition from above to the matrix setting.

$\mathsf{GameStatCS}_{\mathcal{A}}(n, t, p, u, v, Q)$

$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \mathsf{Cor}, (\boldsymbol{X}_i)_{i\in\mathsf{Cor}}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(n, t, p, u, v, Q)$

$/\!/\ \alpha, \beta \in \mathbb{Z}_p^u,\ \Gamma \in \mathbb{Z}_p^{u\times u},\ \forall i \in \mathsf{Cor}\colon \boldsymbol{X}_i \in \mathbb{Z}_p^{(1+u)\times v}$

$\boldsymbol{a} \coloneqq (1 \mid \boldsymbol{\alpha}) \in \mathbb{Z}_p^{1+u}$

assert $\mathsf{Cor} \subseteq [n] \wedge |\mathsf{Cor}| < t$

assert $\det \begin{pmatrix} 1 & \boldsymbol{\alpha} \\ \boldsymbol{\beta}^\top & \boldsymbol{\Gamma} \end{pmatrix} \neq 0$ $/\!/$ invertibility condition

$\boldsymbol{f} \xleftarrow{\$} \{\boldsymbol{F} \in \mathbb{Z}_p[X]^{(1+u)\times v}_{\deg<t}\colon \forall i \in \mathsf{Cor}\colon \boldsymbol{F}(i) = \boldsymbol{X}_i\}$

$S^{(0)} \coloneqq \emptyset; S^* \coloneqq \emptyset$

$\boldsymbol{s} \coloneqq \boldsymbol{a}\boldsymbol{f}(0) \in \mathbb{Z}_p^v$

$\forall i \in [n]\colon \boldsymbol{s}_i \coloneqq \boldsymbol{a}\boldsymbol{f}(i) \in \mathbb{Z}_p^v$

for $k = 1, \ldots, Q$:

  $(\rho_k, \mathsf{st}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^{\mathsf{check}}}(\mathsf{st}, \boldsymbol{s}, (\boldsymbol{s}_i)_{i\in[n]}, S^{(k-1)}, (\boldsymbol{f}(i))_{i\in S^{(k-1)}})$

  if $|\mathsf{Cor} \cup S^*| \geq t - 1$ **then**

    $S^{(k)} \leftarrow \emptyset$

  else

    $S^{(k)} \leftarrow \{i \in [n]\colon \rho_k^{\mathcal{O}^{\mathsf{check}}}(i, \boldsymbol{f}(i)) = 1\}$

  $S^* \leftarrow S^* \cup S^{(k)}$

$\mathcal{A}^{\mathcal{O}^{\mathsf{check}}}(\mathsf{st}, \boldsymbol{s}, (\boldsymbol{s}_i)_{i\in[n]}, S^{(Q)}, (\boldsymbol{f}(i))_{i\in S^{(Q)}})$

return 0

Oracle $\mathcal{O}^{\mathsf{check}}(\hat{\boldsymbol{S}})$

$/\!/\ \hat{\boldsymbol{S}} \in \mathbb{Z}_p^{u\times v}$

if $\hat{\boldsymbol{S}} = (\boldsymbol{\beta}^\top \mid \boldsymbol{\Gamma})\boldsymbol{f}(0)$ **then**

  if $|\mathsf{Cor} \cup S^*| < t$ **then**

    $\mathcal{A}$ wins, i.e., game outputs 1.

  else return 1

return 0

Figure 7: The game in the Second Statistical Core Lemma (Lemma 6.1) for an adversary $\mathcal{A}$. The game outputs 0 if an assertion fails.

Lemma 6.1 (Second Statistical Core Lemma). *For all adversaries $\mathcal{A}$, all $u, v, Q \in \mathbb{N}$ and all $0 < t \leq$*

[44] They are used by the reduction to simulate partial decryptions.

$n < p \in \mathbb{N}$, *where* p *is prime, it holds that*

$$\Pr[\mathsf{GameStatCS}_{\mathcal{A}}(n,t,p,u,v,Q)=1] \le (n+1)\cdot q_{\mathsf{check}} \cdot \frac{\binom{n}{t-1}}{p^{u\cdot v}},$$

where $\mathsf{GameStatCS}_{\mathcal{A}}(n,t,p,u,v,Q)$ *is defined in Figure 7, the randomness is taken over the choice of* $\boldsymbol{f}$ *and the random coins of* $\mathcal{A}$*, and* q_{check} *bounds the total number of queries made to* $\mathcal{O}^{\mathsf{check}}$ *by* $\mathcal{A}$ *directly and through* ρ_k *invocations.*

Proof of Lemma 6.1. The proof mirrors the proof of Lemma 5.1, but with a different encoding and an additional argument involving the invertibility condition on $\boldsymbol{A} = \begin{pmatrix} 1 & \boldsymbol{\alpha} \\ \boldsymbol{\beta}^\top & \boldsymbol{\Gamma} \end{pmatrix}$. We refer the reader back to the proof of Lemma 5.1, and repeat only the pieces that differ, in particular, the part of the proof concerning $\mathsf{WinSelf}$. Recall that $\mathsf{WinSelf}$ denotes the event that the adversary wins $\mathsf{GameStatCS}$ via a *direct* query to $\mathcal{O}^{\mathsf{check}}$, while WinMot denotes the event that the adversary wins $\mathsf{GameStatCS}$ via an indirect query issued through a motivation function. The events are disjoint and collectively exhaustively cover winning $\mathsf{GameStatCS}$.

Our goal in this part of the proof is to show that

$$\Pr[\mathsf{WinSelf}] \le q_{\mathsf{check}} \cdot \frac{\binom{n}{t-1}}{p^{u\cdot v}}. \tag{3}$$

Fix $\mathcal{A}$ and random coins coins; as before, we assume coins are input only in the initial call to $\mathcal{A}$ and stashed in st thereafter. Once coins are fixed, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\Gamma}$, Cor, and $(\boldsymbol{X}_i)_{i\in\mathsf{Cor}}$ are determined, and the only remaining randomness lies in the function $\boldsymbol{f} \in \mathbb{Z}_p[X]^{(1+u)\times v}_{\deg<t}$. Denote $c := |\mathsf{Cor}|$. Partition the remaining randomness space into the set of functions that trigger $\mathsf{WinSelf}$

$$\mathcal{W} = \{\boldsymbol{f} \mid \forall i \in \mathsf{Cor}\colon \boldsymbol{f}(i) = \boldsymbol{X}_i \wedge \mathsf{WinSelf}_{\mathcal{A}}(n,t,p,u,v,Q;\mathsf{coins},\boldsymbol{f}) = 1\},$$

and the set of losing functions $\mathcal{L}$ analogously such that

$$\Pr[\mathsf{WinSelf}_{\mathcal{A}}(n,t,p,u,v,Q;\mathsf{coins})=1] = \frac{|\mathcal{W}|}{|\mathcal{W}|+|\mathcal{L}|}.$$

Claim 6.2 $|\mathcal{W}| \le \binom{n}{t-1} \cdot p^{(t-c-1)(1+u)\cdot v+v} \cdot q_{\mathsf{check}}$.

Proof of Claim 6.2. For $\boldsymbol{f} \in \mathcal{W}$ and each $k \in [Q]$, let $S^{(k)}_{\boldsymbol{f}}$ be the values of $S^{(k)}$, i.e., the set of evaluation points selected in round k, when the game is run for $\mathcal{A}$ on $\boldsymbol{f}$ and coins, and let $S^*_{\boldsymbol{f}}$ be the set S^*, i.e., the set of all evaluation points the adversary has seen when the winning query is made. Let $P_{\boldsymbol{f}} \in \binom{[n]}{t-1}$ be the lex-smallest completion of $\mathsf{Cor} \cup S^*_{\boldsymbol{f}}$. Let $\boldsymbol{s}_{\boldsymbol{f}} = (1 \mid \boldsymbol{\alpha})\boldsymbol{f}(0)$, and define the encoding

$$\mathsf{Enc}\colon \mathcal{W} \to \binom{[n]}{t-1} \times (\mathbb{Z}_p^{(1+u)\times v})^{t-1-c} \times \mathbb{Z}_p^v \times [q_{\mathsf{check}}], \quad \mathsf{Enc}(\boldsymbol{f}) = \left(P_{\boldsymbol{f}}, (\boldsymbol{V}_i)_{i\in P_{\boldsymbol{f}}\setminus\mathsf{Cor}}, \boldsymbol{s}_{\boldsymbol{f}}, \mathsf{idx}_{\boldsymbol{f}}\right),$$

where $\boldsymbol{V}_i = \boldsymbol{f}(i)$ and $\mathsf{idx}_{\boldsymbol{f}}$ is the index of the winning query, counted only among *direct* queries made by $\mathcal{A}$. Well-definedness is as in Lemma 5.1. To show injectivity, let us assume $\mathsf{Enc}(\boldsymbol{f}) = \mathsf{Enc}(\boldsymbol{g})$ for two winning functions $\boldsymbol{f}, \boldsymbol{g} \in \mathcal{W}$. Write $P := P_{\boldsymbol{f}} = P_{\boldsymbol{g}}$, $\boldsymbol{s} := \boldsymbol{s}_{\boldsymbol{f}} = \boldsymbol{s}_{\boldsymbol{g}}$, and $\mathsf{idx} := \mathsf{idx}_{\boldsymbol{f}} = \mathsf{idx}_{\boldsymbol{g}}$. As before, it holds that $\boldsymbol{f}|_P = \boldsymbol{g}|_P$ because both functions match the fixed $\boldsymbol{X}_i$ on $i \in \mathsf{Cor}$ by the sampling procedure and must equal on $P \setminus \mathsf{Cor}$ by the definition of the encoding.

The induction of Lemma 5.1 goes through with three modifications:

- The round-k adversary call receives the extra fixed argument $\boldsymbol{s}$, which is identical for both functions by the definition of the encoding.
- The round-k adversary call receives the extra fixed arguments $\boldsymbol{s}_i$ for all $i \in [n]$. Since $\boldsymbol{f}|_P = \boldsymbol{g}|_P$ and $\boldsymbol{s} = \boldsymbol{s}_{\boldsymbol{f}} = \boldsymbol{s}_{\boldsymbol{g}}$, we have for all $i \in [n]$ by interpolation

  $$\boldsymbol{s}_{i,\boldsymbol{f}} = \boldsymbol{a}\boldsymbol{f}(i) = \lambda^{(i)}_{0,P\cup\{0\}}\boldsymbol{s} + \sum_{j\in\mathsf{Cor}} \lambda^{(i)}_{j,P\cup\{0\}}\boldsymbol{a}\boldsymbol{X}_j + \sum_{j\in P\setminus\mathsf{Cor}} \lambda^{(i)}_{j,P\cup\{0\}}\boldsymbol{a}\boldsymbol{V}_j = \boldsymbol{s}_{i,\boldsymbol{g}},$$

  where $(\boldsymbol{V}_j)_{j\in P\setminus\mathsf{Cor}}$ is from the matching encodings. As a result, we have $\boldsymbol{s}_i := \boldsymbol{s}_{i,\boldsymbol{f}} = \boldsymbol{s}_{i,\boldsymbol{g}}$ for all $i \in [n]$.

- Revealed shares are elements of $\mathbb{Z}_p^{(1+u)\times v}$ instead of $\mathbb{Z}_p^{\ell}$.

When the idx'th query is made, its input (denoted as $\hat{\boldsymbol{S}}$) is equal for both runs, so $\hat{\boldsymbol{S}}_{\boldsymbol{f}} = \hat{\boldsymbol{S}}_{\boldsymbol{g}}$ is the same for both functions. Since the runs of both functions win, we have that

$$(\boldsymbol{\beta}^\top \mid \boldsymbol{\Gamma})\boldsymbol{f}(0) = \hat{\boldsymbol{S}}_{\boldsymbol{f}} = \hat{\boldsymbol{S}}_{\boldsymbol{g}} = (\boldsymbol{\beta}^\top \mid \boldsymbol{\Gamma})\boldsymbol{g}(0).$$

Combining with $\boldsymbol{s} = (1 \mid \boldsymbol{\alpha})\boldsymbol{f}(0) = (1 \mid \boldsymbol{\alpha})\boldsymbol{g}(0)$, we get the equation system

$$\begin{pmatrix} 1 & \boldsymbol{\alpha} \\ \boldsymbol{\beta}^\top & \boldsymbol{\Gamma} \end{pmatrix} \cdot (\boldsymbol{f}(0) - \boldsymbol{g}(0)) = \boldsymbol{0}.$$

Since invertibility of $\boldsymbol{A} = \begin{pmatrix} 1 & \boldsymbol{\alpha} \\ \boldsymbol{\beta}^\top & \boldsymbol{\Gamma} \end{pmatrix}$ is asserted in the game,[45] the system permits only the trivial solution, and we get that $\boldsymbol{f}(0) = \boldsymbol{g}(0)$. Recall that $\boldsymbol{f}$ and $\boldsymbol{g}$ are sampled as matrices of polynomials of degree $< t$. We showed above that $\boldsymbol{f}|_{P\cup\{0\}} = \boldsymbol{g}|_{P\cup\{0\}}$, i.e., all component polynomials agree on t distinct points, which implies $\boldsymbol{f} = \boldsymbol{g}$ and therefore Enc is injective. The co-domain of Enc has size $\binom{n}{t-1} \cdot p^{(t-c-1)(1+u)\cdot v+v} \cdot q_{\mathsf{check}}$, which by injectivity bounds $|\mathcal{W}|$. ∎

Claim 6.3 $|\mathcal{W}| + |\mathcal{L}| = p^{(t-c)\cdot(1+u)\cdot v}$.

Proof of Claim 6.3. During sampling of $\boldsymbol{f}$, we sample $(1+u)\cdot v$ random polynomials of degree $< t$ with c evaluation constraints, which is equivalent to picking $t-c$ random additional points per polynomial. This gives p^{t-c} choices per polynomial. ∎

Concluding: Combining the two claims, we get that

$$\Pr[\mathsf{WinSelf}_{\mathcal{A}}(n,t,p,u,v,Q) = 1 \mid \mathsf{coins}] = \frac{|\mathcal{W}|}{|\mathcal{W}| + |\mathcal{L}|} \leq \frac{\binom{n}{t-1} \cdot p^{(t-c-1)(1+u)\cdot v+v} \cdot q_{\mathsf{check}}}{p^{(t-c)\cdot(1+u)\cdot v}} = q_{\mathsf{check}} \cdot \frac{\binom{n}{t-1}}{p^{u\cdot v}},$$

which implies Equation 3 by averaging over the choice of coins.

Note that the other part of the proof, i.e., bounding WinMot, mirrors the proof of Claim 5.5, and we get that

$$\Pr[\mathsf{WinMot}_{\mathcal{A}}] \leq n \cdot \Pr[\mathsf{WinSelf}_{\mathcal{B}}]$$

for an appropriate reduction $\mathcal{B}$ that is conceptually identical to that of Claim 5.5 with trivial modifications to adapt to the differences in the lemma. This concludes the proof of Lemma 6.1 by the law of total probability. ∎

6.3 Construction

Construction 6.4 (Variant of Cramer-Shoup Π_{CS}). *Let μ denote the message length. Let $\mathsf{H}_{\mathsf{mask}} \colon \{0,1\}^* \to \{0,1\}^\mu$ be a hash function, modeled as a random oracle. The scheme is parameterized with ciphertext randomness multiplicity u and key multiplicity v. Both can be functions in λ, n, and t. Let PS be a proof system for the relation*

$$\mathsf{R} = \left\{ (\chi, \boldsymbol{\omega}) \colon \begin{array}{c} \chi = ((\mathbb{G}, [1], p), [\boldsymbol{a}], c_1, c_2) \wedge \boldsymbol{\omega} \in \mathbb{Z}_p^u \\ c_1 = \boldsymbol{\omega}^\top [\boldsymbol{a}] \end{array} \right\}.$$

The algorithms of Π_{CS} are depicted in Figure 8.

Correctness follows because

$$\sum\nolimits_{i\in S} \lambda_{i,S}^{(0)} d_i = \sum\nolimits_{i\in S} \lambda_{i,S}^{(0)} c_1 \boldsymbol{X}_i = c_1 \sum\nolimits_{i\in S} \lambda_{i,S}^{(0)} \boldsymbol{X}_i = c_1 \boldsymbol{X} = \left[(\boldsymbol{r}^\top \boldsymbol{a})\boldsymbol{X}\right] = \left[\boldsymbol{r}^\top (\boldsymbol{a}\boldsymbol{X})\right] = \boldsymbol{r}^\top \mathsf{pk} = [\boldsymbol{Z}].$$

[45] Recall that $\boldsymbol{A}$ is fully determined by coins. If $\boldsymbol{A}$ is not invertible, then the assertion always fails and thus $|\mathcal{W}| = 0$, in which case the claim holds trivially.

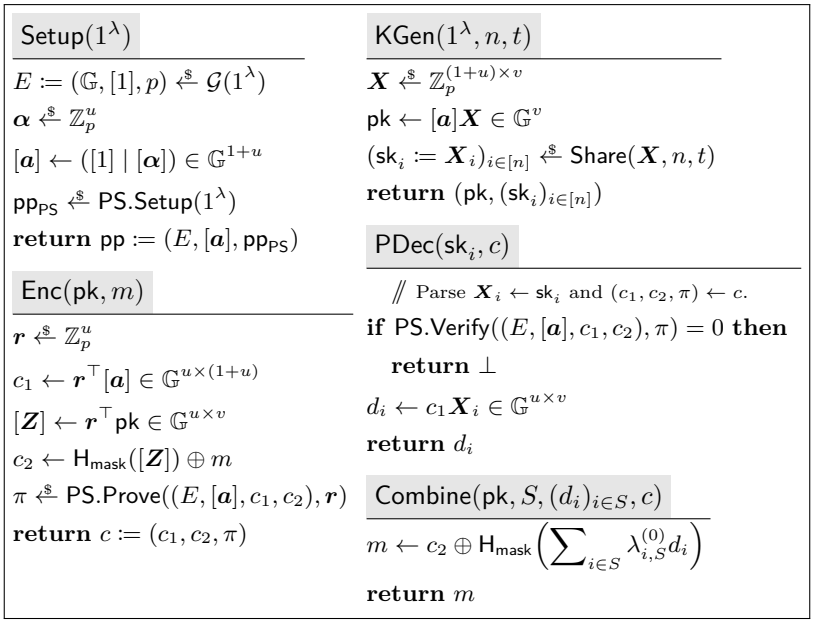


Figure 8: The construction of Π_{CS}, parameterized with the ciphertext randomness multiplicity u and key multiplicity v. Recall that both can be functions in λ, n, and t. Note that $\mathsf{Share}(\boldsymbol{X}, n, t)$ Shamir-shares the matrix $\boldsymbol{X}$ componentwise using independent polynomials.

Remark 6.5 We note that in Π_{CS}, the vector $[\boldsymbol{\alpha}] \in \mathbb{G}^u$ is sampled during Setup as part of the vector $[\boldsymbol{a}]$, which arguably contributes u group elements to the public key in the form of additional public parameters. This is undesirable, since we want the public key size to only scale with v. However, this can be solved by simply sampling $[\boldsymbol{\alpha}]$ via a random oracle. Nevertheless, we choose to make the sampling of $[\boldsymbol{\alpha}]$ an explicit part of Setup for ease of presentation.

Theorem 6.6 Π_{CS} *(Construction 6.4) is* IND-IMC-CCA*-secure for* $\log(p) \geq \lambda$, $\log(p^{u\cdot v}) \geq n + \lambda$, $q = Q = (\infty, \infty, \infty, \infty, \infty)$ *and for* $\mathcal{F} = (\mathcal{F}_*, \mathcal{F}_*, \mathcal{F}_{\mathsf{no\text{-}sk}}, \mathcal{F}_{\mathsf{no\text{-}sk}}, \mathcal{F}_*)$ *in the random oracle model under the* DDH *assumption if* PS *is zero-knowledge and simulation-sound. In more detail, for all PPT adversaries* $\mathcal{A}$*, there exist PPT reductions* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}IMC\text{-}CCA}}_{\mathcal{A},\Pi_{\mathsf{CS}}}(1^\lambda, n, t, \mathcal{F}, q, Q) \leq 2 \cdot \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{B},(\mathsf{PS},\mathsf{Sim})}(1^\lambda) + \mathsf{Adv}^{\mathsf{SimSnd}}_{\mathcal{B},(\mathsf{PS},\mathsf{Sim})}(1^\lambda) + u^2 \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda) + \frac{1}{p-1} + (n+1) \cdot q_{\mathsf{H}_{\mathsf{mask}}} \cdot \frac{\binom{n}{t-1}}{p^{u\cdot v}},$$

where $q_{\mathsf{H}_{\mathsf{mask}}} \in \mathsf{poly}(\lambda)$ *bounds the total number of direct and indirect queries* $\mathcal{A}$ *makes to* $\mathsf{H}_{\mathsf{mask}}$*, and the running time of* $\mathcal{B}$ *is at most that of* $\mathcal{A}$ *plus an additive overhead polynomial in* λ *and* n*.*

Note that the proof system PS can be instantiated with discrete-logarithm equality (DLEQ) proofs, which satisfy simulation-soundness in the ROM [46, 43].

Proof of Theorem 6.6. We prove Theorem 6.6 by first showing that Π_{CS} satisfies IND-IMC-CCA security for a more restricted set of motivation function classes $\mathcal{F} = (\mathcal{F}_*, \mathcal{F}_*, \mathcal{F}_{\mathsf{accept}}, \mathcal{F}_{\mathsf{accept}}, \emptyset)$ and $q = Q = (\infty, \infty, \infty, \infty, 0)$. In this setting, the adversary does not have access to $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$, and the $\mathcal{O}^{\mathsf{PDec}}$ oracle can only be accessed in a classical manner, i.e., without internally motivated partial decryptions. We prove Π_{CS} to be secure in this setting via a reduction to DDH and by applying the second core lemma (Lemma 6.1). Then, we can apply Fact 3.11 to elevate the motivation function class for the $\mathcal{O}^{\mathsf{PDec}}$ oracle to $\mathcal{F}_{\mathsf{no\text{-}sk}}$, leaving us with $\mathcal{F} = (\mathcal{F}_*, \mathcal{F}_*, \mathcal{F}_{\mathsf{no\text{-}sk}}, \mathcal{F}_{\mathsf{no\text{-}sk}}, \emptyset)$. Finally, we can apply Fact 3.12 to achieve the full result stated in the theorem, i.e., upgrading to $q^{\mathsf{chal}} = Q^{\mathsf{chal}} = \infty$ and $\mathcal{F}^{\mathsf{chal}} = \mathcal{F}_*$.

In the following, we show IND-IMC-CCA security for $\mathcal{F} = (\mathcal{F}_*, \mathcal{F}_*, \mathcal{F}_{\mathsf{accept}}, \mathcal{F}_{\mathsf{accept}}, \emptyset)$ and $q = Q = (\infty, \infty, \infty, \infty, 0)$. Our proof will carry out a series of game hops; we start by recalling the IND-IMC-CCA experiment instantiated with Π_{CS} in the above setting as Game_0.

Game_0 (Real game): We recall the game for completeness:

1. The adversary initially outputs a set of static corruptions $\mathsf{Cor}_{\mathsf{sk}}$. W.l.o.g., we assume that $\mathsf{Cor}_{\mathsf{sk}} \subseteq [n]$ and $|\mathsf{Cor}_{\mathsf{sk}}| < t$.

2. The challenger generates the group description E, the public parameters $[\boldsymbol{a}]$, and the proof system setup $\mathsf{pp}_{\mathsf{PS}}$. It runs key generation, sampling $\boldsymbol{X} \xleftarrow{\$} \mathbb{Z}_p^{(1+u)\times v}$ as well as the component-wise Shamir sharing $(\boldsymbol{X_i})_{i\in[n]}$ of $\boldsymbol{X}$. It forwards to the adversary the public parameters $\mathsf{pp} = (E, [\boldsymbol{a}], \mathsf{pp}_{\mathsf{PS}})$, the public key $\mathsf{pk} = [\boldsymbol{a}]\boldsymbol{X}$ as well as $\mathsf{sk}_i = \boldsymbol{X}_i$ for all $i \in \mathsf{Cor}_{\mathsf{sk}}$.

3. The adversary can now query the partial decryption oracle $\mathcal{O}_0^{\mathsf{PDec}}$ and the corruption oracle $\mathcal{O}_0^{\mathsf{cor}}$. Note that, since $\mathcal{F}_0^{\mathsf{PDec}} = \mathcal{F}_{\mathsf{accept}}$, $\mathcal{O}_0^{\mathsf{PDec}}$ behaves like a standard partial decryption oracle. On input $(S, c := (c_1, c_2, \pi))$, it computes the partial decryptions on c on behalf of parties in S (which includes checking the proof π) and outputs them to the adversary. Note that, if the proof does not verify, all partial decryptions are set to $\perp$. When the adversary makes an $\mathcal{O}_0^{\mathsf{cor}}$ query on input (S, ψ), the challenger runs $\psi(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i)$ for all $i \in S$. Let S' denote the set of parties for which the function outputs 1. The challenger updates $\mathsf{Cor}_{\mathsf{sk}} \leftarrow \mathsf{Cor}_{\mathsf{sk}} \cup S'$ and forwards the keys $(\mathsf{sk}_i = \boldsymbol{X}_i)_{i\in S'}$ to the adversary.

4. The adversary outputs two equal-length messages m_0 and m_1, and the challenger encrypts m_b under pk for a random bit b. The resulting challenge ciphertext consists of

$$c_1^* \leftarrow \boldsymbol{r}^\top[\boldsymbol{a}], \quad c_2^* \leftarrow \mathsf{H}_{\mathsf{mask}}([\boldsymbol{Z}]) \oplus m_b, \quad \pi^*,$$

   where $\boldsymbol{r}$ is sampled uniformly from $\mathbb{Z}_p^u$, and $[\boldsymbol{Z}] \leftarrow \boldsymbol{r}^\top\mathsf{pk}$, and π^* is a proof generated using the witness $\boldsymbol{r}$ tagged to (c_1^*, c_2^*).

5. The adversary gets c^* and can access $\mathcal{O}_1^{\mathsf{cor}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$, which behave as above, except that in $\mathcal{O}_1^{\mathsf{PDec}}$, the challenger first asserts that $c \neq c^*$. Note that the adversary does not get access to $\mathcal{O}^{\mathsf{PDec\text{-}chal}}$.

6. Finally, the adversary outputs a guess b' and wins if $b = b'$ and $|\mathsf{Cor}_{\mathsf{sk}}| < t$.

7. Throughout, the challenger simulates the $\mathsf{H}_{\mathsf{mask}}$ oracle to the adversary and the motivation functions submitted to $\mathcal{O}^{\mathsf{cor}}$. Henceforth, we call queries to $\mathsf{H}_{\mathsf{mask}}$ made by the adversary "direct", and queries made by motivation functions "indirect".

Game_1 (Simulate π^*): In Game_1, we use the simulator of PS to set up $\mathsf{pp}_{\mathsf{PS}}$ and simulate π^* for the challenge ciphertext without knowing $\boldsymbol{r}$.

Claim 6.7 For all PPT adversaries $\mathcal{A}$ there exists a PPT reduction $\mathcal{B}$ such that

$$\left|\Pr\left[\mathsf{Game}_{1,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{0,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq 2 \cdot \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{B},(\mathsf{PS},\mathsf{Sim})}(1^\lambda).$$

Proof of Claim 6.7. The proof of Claim 6.7 is almost identical to that of Claim 5.9, except that we only need the zero-knowledge property here, as PS is not a proof of knowledge and we do not extract any witnesses. ∎

Game_2 (Compute $\boldsymbol{Z}$ using secret key): Next, we modify how the challenger computes $[\boldsymbol{Z}]$, and thus c_2^*. Instead of computing $[\boldsymbol{Z}] \leftarrow \boldsymbol{r}^\top\mathsf{pk}$, the challenger computes

$$[\boldsymbol{Z}] \leftarrow c_1^*\boldsymbol{X},$$

where $\boldsymbol{X}$ is the secret key sampled initially during key generation. As in the previous game $[\boldsymbol{Z}]$ is then used to compute c_2^* as $c_2^* \leftarrow \mathsf{H}_{\mathsf{mask}}([\boldsymbol{Z}]) \oplus m_b$.

Claim 6.8 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\mathsf{Game}_{2,\mathcal{A}}(1^\lambda) = 1\right] = \Pr\left[\mathsf{Game}_{1,\mathcal{A}}(1^\lambda) = 1\right].$$

Proof of Claim 6.8. Both games are identical because

$$c_1^*\boldsymbol{X} = (\boldsymbol{r}^\top[\boldsymbol{a}])\boldsymbol{X} = \boldsymbol{r}^\top([\boldsymbol{a}]\boldsymbol{X}) = \boldsymbol{r}^\top\mathsf{pk}.$$

∎

Game_3 (Make c_1^* random): In Game_3, we modify how c_1^* is computed. Instead of sampling $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^u$ and computing $c_1^* \leftarrow \boldsymbol{r}^\top[\boldsymbol{a}] \in \mathbb{G}^{u\times(1+u)}$, the challenger samples random $\boldsymbol{\beta} \xleftarrow{\$} \mathbb{Z}_p^u$ as well as $\boldsymbol{\Gamma} \xleftarrow{\$} \mathbb{Z}_p^{u\times u}$ at the beginning of the game, and sets

$$c_1^* \leftarrow \left[(\boldsymbol{\beta}^\top \mid \boldsymbol{\Gamma})\right].$$

The challenger continues to use this c_1^* to compute $[\boldsymbol{Z}] \leftarrow c_1^*\boldsymbol{X}$ as in the previous game. Intuitively, switching from an element in the span of $[\boldsymbol{a}]$ to a random element corresponds to the Diffie-Hellman assumption. We show this formally in the next claim.

Claim 6.9 For all PPT adversaries $\mathcal{A}$ there exists a PPT reduction $\mathcal{B}$ such that

$$\left|\Pr\left[\mathsf{Game}_{3,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{2,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \mathsf{Adv}^{(u,u)\text{-mDDH}}_{\mathcal{B},\mathcal{G}}(1^\lambda).$$

Proof of Claim 6.9. Let $\mathcal{A}$ be a PPT distinguisher. We construct a reduction $\mathcal{B}$ playing against the (u,u)-mDDH challenger. $\mathcal{B}$ initially receives the tuple

$$\left([\boldsymbol{\alpha}] \in \mathbb{G}^u, [\boldsymbol{\beta}] \in \mathbb{G}^u, [\boldsymbol{\Gamma}] \in \mathbb{G}^{u\times u}\right),$$

where either $\boldsymbol{\Gamma} = \boldsymbol{\beta}^\top\boldsymbol{\alpha}$, or $\boldsymbol{\Gamma}$ is uniformly and independently sampled from $\mathbb{Z}_p^{u\times u}$.[46] $\mathcal{B}$ simulates either game to $\mathcal{A}$ by embedding $[\boldsymbol{\alpha}]$ into the public parameters as $[\boldsymbol{a}] := ([1] \mid [\boldsymbol{\alpha}])$ and embedding $[\boldsymbol{\beta}]$ into the randomness of the challenge ciphertext $[\boldsymbol{r}] := [\boldsymbol{\beta}]$.

Simulating Key generation. $\mathcal{B}$ simulates key generation entirely according to protocol, sampling $\boldsymbol{X} \xleftarrow{\$} \mathbb{Z}_p^{(1+u)\times v}$ alongside shares $(\mathsf{sk}_i := \boldsymbol{X}_i)_{i\in[n]}$ of $\boldsymbol{X}$, and $\mathcal{B}$ sets $\mathsf{pk} \leftarrow [\boldsymbol{a}]\boldsymbol{X}$, forwarding $(\mathsf{pp}, \mathsf{pk}, (\mathsf{sk}_i)_{i\in\mathsf{Cor}_{\mathsf{sk}}})$ to $\mathcal{A}$ for the set $\mathsf{Cor}_{\mathsf{sk}}$ of static corruptions specified by $\mathcal{A}$.

Simulating $\mathcal{O}_0^{\mathsf{cor}}$ and $\mathcal{O}_0^{\mathsf{PDec}}$. Since $\mathcal{B}$ knows all secret key shares, $\mathcal{B}$ can simulate both oracles according to either game (the games are identical with respect to both oracles).

Simulating the Challenge Ciphertext. When $\mathcal{A}$ outputs two messages m_0, m_1 of equal length, $\mathcal{B}$ embeds the (u,u)-mDDH challenge in the challenge ciphertext, setting $c_1^* \leftarrow ([\boldsymbol{\beta}^\top] \mid [\boldsymbol{\Gamma}])$. Then, $\mathcal{B}$ computes $[\boldsymbol{Z}] \leftarrow c_1^*\boldsymbol{X}$ and $c_2^* \leftarrow \mathsf{H}_{\mathsf{mask}}([\boldsymbol{Z}]) \oplus m_b$ for a random bit b alongside the simulated proof π^* as mandated by either game. $\mathcal{B}$ forwards $c^* = (c_1^*, c_2^*, \pi^*)$ to $\mathcal{A}$ and continues to simulate $\mathcal{O}_1^{\mathsf{cor}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$ as above, according to either game.

When $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ verifies all winning conditions and outputs 1, if $b' = b$. If any assertion fails, any abort condition is hit, or $b' \neq b$, $\mathcal{B}$ outputs 0 to its challenger.

Analysis. If the (u,u)-mDDH-tuple is *real*, then $c_1^* = ([\boldsymbol{\beta}^\top] \mid [\boldsymbol{\beta}^\top\boldsymbol{\alpha}]) = \boldsymbol{\beta}^\top[\boldsymbol{a}]$, and thus $\mathcal{B}$ perfectly simulates Game_2 to $\mathcal{A}$, outputting 1 with probability $\Pr\left[\mathsf{Game}_{2,\mathcal{A}}(1^\lambda) = 1\right]$. If, however, the (u,u)-mDDH-tuple is *random*, then $\mathcal{B}$ perfectly simulates Game_3 to $\mathcal{A}$, outputting 1 with probability $\Pr\left[\mathsf{Game}_{3,\mathcal{A}}(1^\lambda) = 1\right]$, which yields the inequality stated in the claim. ∎

Game_4 (Abort if proof is forged): In Game_4 we add an abort condition to the $\mathcal{O}_0^{\mathsf{PDec}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$ oracles. On input $(S, c := (c_1, c_2, \pi))$, the challenger aborts if the proof π verifies, but there does not exist an $\boldsymbol{r} \in \mathbb{Z}_p^u$ such that $c_1 = \boldsymbol{r}^\top[\boldsymbol{a}]$.

For intuition, note that this is equivalent to parsing $([\boldsymbol{r}]^\top \mid [\boldsymbol{H}]) \leftarrow c_1$, where $[\boldsymbol{r}] \in \mathbb{G}^u$ and $[\boldsymbol{H}] \in \mathbb{G}^{u\times u}$, and asserting that $[\boldsymbol{r}^\top]\boldsymbol{\alpha} = [\boldsymbol{H}]$. The above check is (only) efficient if the challenger knows $\boldsymbol{\alpha} \in \mathbb{Z}_p^u$. I.e., it is crucial that this game hop is placed *after* the previous step, which was proven by a reduction that embedded the (u,u)-mDDH challenge into $[\boldsymbol{\alpha}]$, and thus would not be able to efficiently check for the extra abort condition, making its simulation of the $\mathcal{O}^{\mathsf{PDec}}$ oracle flawed.

The fact that one can efficiently check for the abort condition with knowledge of $\boldsymbol{\alpha}$ allows our reduction to simulation-soundness (Claim 6.10) to be tight.[47]

Claim 6.10 For all PPT adversaries $\mathcal{A}$, there exists a PPT reduction $\mathcal{B}$ such that

$$\left|\Pr\left[\mathsf{Game}_{4,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{3,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{SimSnd}}_{\mathcal{B},(\mathsf{PS},\mathsf{Sim})}(1^\lambda).$$

[46] Note that we are implicitly transposing $\boldsymbol{\Gamma}$ with respect to the formulation of (u,u)-mDDH in order to simplify notation.

[47] Without this, our reduction would have to guess which query violates simulation-soundness, and thus would incur a loss of q_{PDec}, where q_{PDec} bounds the number of accumulated $\mathcal{O}_0^{\mathsf{PDec}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$ queries made by the adversary.

Proof of Claim 6.10. The claim follows a straightforward reduction to simulation-soundness of PS. The reduction $\mathcal{B}$ initially samples the public parameters, embedding the $\mathsf{pp}_{\mathsf{PS}}$ received from the challenger, and thus knows $\boldsymbol{\alpha} \in \mathbb{Z}_p^u$. It generates the simulated proof for the challenge ciphertext by forwarding $\chi^* = (E, ([1] \mid [\boldsymbol{\alpha}]), c_1^*, c_2^*)$ to its challenger, receiving π^*.

Whenever $\mathcal{A}$ makes an $\mathcal{O}^{\mathsf{PDec}}$ query for a ciphertext $c = (c_1, c_2, \pi)$, where $c \neq c^*$ and the check $\mathsf{PS.Verify}((E, [\boldsymbol{a}], c_1, c_2), \pi) = 1$ succeeds, $\mathcal{B}$ parses $([\boldsymbol{r}]^\top \mid [\boldsymbol{H}]) \leftarrow c_1$ and checks whether $[\boldsymbol{r}^\top]\boldsymbol{\alpha} = [\boldsymbol{H}]$. If this check fails, $\mathcal{B}$ assembles $\tilde{\chi} \leftarrow (E, ([1] \mid [\boldsymbol{\alpha}]), c_1, c_2)$ and $\tilde{\pi} \leftarrow \pi$ and outputs $(\tilde{\chi}, \tilde{\pi})$ to its challenger.

Note that, if the above happens in $\mathcal{O}_0^{\mathsf{PDec}}$, i.e., before the simulated proof on the challenge ciphertext was given to $\mathcal{B}$, then $\mathcal{B}$ first sends a dummy statement to the SimSnd challenger, and proceeds to output the forgery afterward.

Analysis. Note that, whenever the abort condition is hit, i.e., whenever $\mathcal{B}$ returns $(\tilde{\chi}, \tilde{\pi})$ to its challenger, then it must hold that $(\tilde{\chi}, \tilde{\pi}) \neq (\chi^*, \pi^*)$, because otherwise, $\mathcal{B}$ would have returned $\perp$ in $\mathcal{O}^{\mathsf{PDec}}$ before. Also, it clearly holds that $\mathsf{PS.Verify}(\tilde{\chi}, \tilde{\pi}) = 1$ and $\tilde{\chi} \notin \mathcal{L}_{\mathsf{R}}$, and thus $\mathcal{B}$ wins whenever $\mathcal{A}$ triggers the additional abort condition. The claim follows because both games are identical unless the abort condition is triggered. ∎

Game_5 (Compute partial decryption using $(\boldsymbol{a}\boldsymbol{X}_i)$): Next, we modify how partial decryptions are computed in $\mathcal{O}_0^{\mathsf{PDec}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$. On input $(S, (c_1, c_2, \pi))$, after verifying the proof (and $c \neq c^*$ in the case of $\mathcal{O}_1^{\mathsf{PDec}}$) and passing the previously introduced assertion, the challenger now parses $([\boldsymbol{r}^\top] \mid [\boldsymbol{H}]) \leftarrow c_1$, where $[\boldsymbol{r}] \in \mathbb{G}^u$ and $[\boldsymbol{H}] \in \mathbb{G}^{u \times u}$, and computes

$$d_i \leftarrow [\boldsymbol{r}^\top] \cdot (\boldsymbol{a}\boldsymbol{X}_i).$$

Claim 6.11 For all adversaries $\mathcal{A}$ it holds that $\Pr\big[\mathsf{Game}_{5,\mathcal{A}}(1^\lambda) = 1\big] = \Pr\big[\mathsf{Game}_{4,\mathcal{A}}(1^\lambda) = 1\big]$.

Proof of Claim 6.11. To see that both games are identical, observe that, after passing the assertion, it holds that $\boldsymbol{c_1} = \boldsymbol{r}^\top[\boldsymbol{a}]$ and thus $\boldsymbol{c}_1\boldsymbol{X}_i = [\boldsymbol{r}^\top\boldsymbol{a}]\boldsymbol{X}_i = [\boldsymbol{r}^\top](\boldsymbol{a}\boldsymbol{X}_i)$. ∎

Game_6 (Abort if matrix not invertible): Consider the vectors $\boldsymbol{\alpha} \in \mathbb{Z}_p^u$, $\boldsymbol{\beta} \in \mathbb{Z}_p^u$ and the matrix $\boldsymbol{\Gamma} \in \mathbb{Z}_p^{u \times u}$ from before. Let

$$\boldsymbol{A} \coloneqq \begin{pmatrix} 1 & \boldsymbol{\alpha} \\ \boldsymbol{\beta}^\top & \boldsymbol{\Gamma} \end{pmatrix} \in \mathbb{Z}_p^{(u+1)\times(u+1)}.$$

In Game_6, the challenger aborts at the beginning of the game if $\det(\boldsymbol{A}) = 0$.

Claim 6.12 For all adversaries $\mathcal{A}$ it holds that

$$\big|\Pr\big[\mathsf{Game}_{6,\mathcal{A}}(1^\lambda) = 1\big] - \Pr\big[\mathsf{Game}_{5,\mathcal{A}}(1^\lambda) = 1\big]\big| \leq \frac{1}{p-1}.$$

Proof. To show the claim, we prove that $\Pr[\det(\boldsymbol{A}) = 0] \leq 1/(p-1)$, where the probability is over the choice of $\boldsymbol{\alpha}, \boldsymbol{\beta}$, and $\boldsymbol{\Gamma}$. First, observe that $\det(\boldsymbol{A}) = \det(\boldsymbol{\Gamma} - \boldsymbol{\beta}^\top\boldsymbol{\alpha})$. Let $\boldsymbol{M} = \boldsymbol{\Gamma} - \boldsymbol{\beta}^\top\boldsymbol{\alpha} \in \mathbb{Z}_p^{u\times u}$. We will proceed to show that $\Pr[\det(\boldsymbol{M}) = 0] \leq 1/(p-1)$.

Fix any $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^u$, and observe that sampling $\boldsymbol{\Gamma} \in \mathbb{Z}_p^{u\times u}$ and then computing $\boldsymbol{M} \leftarrow \boldsymbol{\Gamma} - \boldsymbol{\beta}^\top\boldsymbol{\alpha}$ is identically distributed to directly sampling $\boldsymbol{M} \xleftarrow{\$} \mathbb{Z}_p^{u\times u}$. Now, $\det(\boldsymbol{M}) = 0$ is equivalent to the rows of $\boldsymbol{M}$ being linearly dependent. Denote with $\boldsymbol{m}_j$ the j-th row of $\boldsymbol{M}$ and suppose $\boldsymbol{M}$ is sampled row-wise. Denote the event $F_1 = \{\boldsymbol{m}_1 = \boldsymbol{0}\}$ as well as $F_j = \{\boldsymbol{m_j} \in \mathrm{span}(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{j-1})\}$ for all $j \in [2, u]$. Clearly, it holds that $\Pr[F_1] = 1/p^u$ and for any $j \in [2, u]$, there are at most p^{j-1} vectors in $\mathrm{span}(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{j-1})$. Hence, we get that for all $j \in [u]$

$$\Pr[F_j] \leq \frac{p^{j-1}}{p^u} = \frac{1}{p^{u-j+1}}.$$

By a union bound, we have

$$\Pr[\det(\boldsymbol{M}) = 0] \leq \sum_{j\in[u]} \Pr[F_j] \leq \sum_{j\in[u]} \frac{1}{p^{u-j+1}} = \sum_{i\in[u]} \frac{1}{p^i}.$$

Computing the geometric series, we get that

$$\sum_{i\in[u]} \frac{1}{p^i} = \frac{1-p^{-u}}{p-1} \leq \frac{1}{p-1}.$$

Putting it all together, we get that

$$\Pr[\det(\boldsymbol{A}) = 0] = \Pr\big[\det(\boldsymbol{M}) = 0 \colon \boldsymbol{M} \xleftarrow{\$} \mathbb{Z}_p^{u\times u}\big] \leq \frac{1}{p-1}.$$

■

Game_7 (Never evaluate useless motivation functions): Within $\mathcal{O}_0^{\mathsf{cor}}$ and $\mathcal{O}_1^{\mathsf{cor}}$, on input (S,ψ), if $|\mathsf{Cor}_{\mathsf{sk}}| \geq t-1$, the challenger only executes motivation functions for parties already in $S \cap \mathsf{Cor}_{\mathsf{sk}}$. For the remaining parties in $S \setminus \mathsf{Cor}_{\mathsf{sk}}$, the challenger sets the output of the motivation function to 0 without evaluating it.

Claim 6.13 For all adversaries $\mathcal{A}$ it holds that

$$\Pr\big[\mathsf{Game}_{7,\mathcal{A}}(1^\lambda) = 1\big] \geq \Pr\big[\mathsf{Game}_{6,\mathcal{A}}(1^\lambda) = 1\big].$$

Proof of Claim 6.13. Clearly Game_7 and Game_6 are identical until $|\mathsf{Cor}_{\mathsf{sk}}|$ reaches $t-1$ and the first subsequent $\mathcal{O}_0^{\mathsf{cor}}(S,\psi)$ or $\mathcal{O}_1^{\mathsf{cor}}(S,\psi)$ query is made by $\mathcal{A}$ where a motivation function hits $\psi(\mathsf{pp},\mathsf{pk},i,\mathsf{sk}_i) = 1$ for some $i \in S \setminus \mathsf{Cor}_{\mathsf{sk}}$. Denote that i as i^*. Note, however, that immediately after, the oracle in Game_6 would add i^* to $\mathsf{Cor}_{\mathsf{sk}}$. Since i^* was not in $\mathsf{Cor}_{\mathsf{sk}}$ previously, it increases by at least one, and we have $|\mathsf{Cor}_{\mathsf{sk}}| \geq t$, meaning the adversary can no longer win in Game_6. As a result, a run is either identical in both games or loses in Game_6, yielding the claim. ■

Game_8 (Abort if decryption key is queried): In Game_8, we add an additional abort condition. If the adversary ever queries $\mathsf{H}_{\mathsf{mask}}$ on input $[\boldsymbol{Z}]$ (either directly or within motivation functions), where $[\boldsymbol{Z}] = c_1^*\boldsymbol{X}$ while $|\mathsf{Cor}_{\mathsf{sk}}| < t$, then the challenger aborts and outputs 0. Here $[\boldsymbol{Z}]$ is the decryption key that is used to compute the challenge ciphertext (see Game_2).

Claim 6.14 For all PPT adversaries $\mathcal{A}$, it holds that

$$\big|\Pr\big[\mathsf{Game}_{8,\mathcal{A}}(1^\lambda) = 1\big] - \Pr\big[\mathsf{Game}_{7,\mathcal{A}}(1^\lambda) = 1\big]\big| \leq (n+1)\cdot q_{\mathsf{H}_{\mathsf{mask}}} \cdot \frac{\binom{n}{t-1}}{p^{u\cdot v}},$$

where $q_{\mathsf{H}_{\mathsf{mask}}}$ bounds the total number of direct and indirect $\mathsf{H}_{\mathsf{mask}}$ queries made by $\mathcal{A}$.

Proof of Claim 6.14. Let $\mathcal{A}$ be a PPT adversary against either game and denote with Bad the event that $\mathcal{A}$ triggers the additional abort condition. Since both games are identical unless Bad happens, we get that

$$\big|\Pr\big[\mathsf{Game}_{8,\mathcal{A}}(1^\lambda) = 1\big] - \Pr\big[\mathsf{Game}_{7,\mathcal{A}}(1^\lambda) = 1\big]\big| \leq \Pr[\mathsf{Bad}].$$

We next bound $\Pr[\mathsf{Bad}]$ by an unbounded reduction $\mathcal{B}$ to the statistical game $\mathsf{GameStatCS}(n,t,p,u,v,Q)$, making use of Lemma 6.1, where Q is a bound on the total number of $\mathcal{O}^{\mathsf{cor}}$ queries made by $\mathcal{A}$. The reduction $\mathcal{B}$ will make no more than $q_{\mathsf{H}_{\mathsf{mask}}}$ queries to the $\mathcal{O}^{\mathsf{check}}$ oracle of its challenger.

Simulating Setup and KGen. The reduction $\mathcal{B}$ initially runs $\mathcal{A}$ to obtain the set of corrupted parties $\mathsf{Cor}_{\mathsf{sk}}$. W.l.o.g., assume that $\mathsf{Cor}_{\mathsf{sk}} \subseteq [n]$ and $|\mathsf{Cor}_{\mathsf{sk}}| < t$. $\mathcal{B}$ samples the values $\boldsymbol{\alpha},\boldsymbol{\beta} \xleftarrow{\$} \mathbb{Z}_p^u$ and $\boldsymbol{\Gamma} \xleftarrow{\$} \mathbb{Z}_p^{u\times u}$. Here, $\mathcal{B}$ aborts if $\det\begin{pmatrix} 1 & \boldsymbol{\alpha} \\ \boldsymbol{\beta}^\top & \boldsymbol{\Gamma}\end{pmatrix} = 0$, as mandated by either game. Then, it computes the public parameters $\mathsf{pp} \leftarrow (E, [\boldsymbol{a}], \mathsf{pp}_{\mathsf{PS}})$, where $[\boldsymbol{a}] \leftarrow ([1] \mid [\boldsymbol{\alpha}])$. For all $i \in \mathsf{Cor}_{\mathsf{sk}}$, $\mathcal{B}$ samples $\mathsf{sk}_i := \boldsymbol{X}_i \xleftarrow{\$} \mathbb{Z}_p^{(1+u)\times v}$. Then, $\mathcal{B}$ forwards $(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\Gamma},\mathsf{Cor}_{\mathsf{sk}},(\boldsymbol{X}_i)_{i\in\mathsf{Cor}_{\mathsf{sk}}})$ to its challenger. As a response, $\mathcal{B}$ receives $\boldsymbol{s}$ alongside $(\boldsymbol{s}_i)_{i\in[n]}$, and continues to simulate such that $\mathsf{pk} := [\boldsymbol{s}]$ and $\mathsf{pk}_i := [\boldsymbol{s}_i]$ for all $i \in [n]$, i.e. $\boldsymbol{s} \equiv \boldsymbol{a}\boldsymbol{X}$ and $\boldsymbol{s}_i \equiv \boldsymbol{a}\boldsymbol{X}_i$. $\mathcal{B}$ forwards $(\mathsf{pp},\mathsf{pk},(\mathsf{sk}_i)_{i\in\mathsf{Cor}_{\mathsf{sk}}})$ to $\mathcal{A}$.

Simulating $\mathsf{H}_{\mathsf{mask}}$. $\mathcal{B}$ initially pre-generates a table for the entire $\mathsf{H}_{\mathsf{mask}}$ oracle. Whenever $\mathcal{A}$ makes a query to the $\mathsf{H}_{\mathsf{mask}}$ oracle on input $[\hat{\boldsymbol{S}}] \in \mathbb{G}^{u\times v}$, $\mathcal{B}$ extracts $\hat{\boldsymbol{S}} \leftarrow \mathsf{DLog}_{[1]}([\hat{\boldsymbol{S}}])$ and forwards $\hat{\boldsymbol{S}}$ to the $\mathcal{O}^{\mathsf{check}}$ oracle of its challenger. If $\mathcal{B}$ does not immediately win as a result of this query, $\mathcal{B}$ responds to the query by answering from the table.

Simulating $\mathcal{O}_0^{\mathsf{PDec}}$ and $\mathcal{O}_1^{\mathsf{PDec}}$. Whenever $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PDec}}$ on ciphertext (c_1, c_2, π), $\mathcal{B}$, after verifying all checks and assertions, can simulate all partial decryptions by parsing $([\boldsymbol{r}^\top] \mid [\boldsymbol{H}]) \leftarrow c_1$ and computing

$$d_i \leftarrow [\boldsymbol{r}^\top]\boldsymbol{s_i} \qquad \Big| \equiv [\boldsymbol{r}^\top]\boldsymbol{a}\boldsymbol{X}_i.$$

Simulating $\mathcal{O}_0^{\mathsf{cor}}$ and $\mathcal{O}_1^{\mathsf{cor}}$. Denote with $\boldsymbol{f}$ the matrix of polynomials sampled by the $\mathsf{GameStatCS}$ challenger. Whenever $\mathcal{A}$ queries either oracle on input (S, ψ), $\mathcal{B}$ assembles the circuit ρ by embedding ψ and wiring the $\boldsymbol{f}(i)$ input of ρ to the sk_i input of ψ. More formally, it sets

$$\rho(i, \boldsymbol{f}(i)) \coloneqq \begin{cases} \psi(\mathsf{pp}, \mathsf{pk}, i, \boldsymbol{f}(i)) & \text{for } i \in S \\ 0 & \text{for } i \notin S. \end{cases}.$$

Additionally, $\mathcal{B}$ hardcodes the random oracle table for $\mathsf{H}_{\mathsf{mask}}$ into the ρ wrapper. Whenever the underlying ψ makes a query to $\mathsf{H}_{\mathsf{mask}}$, ρ first extracts the discrete logarithm from the query and forwards it to the $\mathcal{O}^{\mathsf{check}}$ oracle. If $\mathcal{B}$ does not immediately win through this query, ρ responds to the $\mathsf{H}_{\mathsf{mask}}$ query by consulting the table.

Edge case if $|\mathsf{Cor}_{\mathsf{sk}}| \geq t-1$. If $|\mathsf{Cor}_{\mathsf{sk}}| \geq t-1$ and this is an $\mathcal{O}^{\mathsf{cor}}(S, \psi)$ query made by $\mathcal{A}$, $\mathcal{B}$ no longer needs to run motivation functions for all $S \setminus \mathsf{Cor}_{\mathsf{sk}}$ due to the change introduced in Game_7. However, $\mathcal{B}$ still needs to simulate motivation functions for $\bar{S} = S \cap \mathsf{Cor}_{\mathsf{sk}}$.[48] For $\bar{S}$, $\mathcal{B}$ cannot rely on simply forwarding ρ to its challenger, because $\mathsf{GameStatCS}$ no longer executes motivation functions while $|\mathsf{Cor} \cup S^*| \geq t-1$. Instead, $\mathcal{B}$ simulates the motivation functions for $\bar{S}$ locally by simply running $\psi(\mathsf{pp}, \mathsf{pk}, i, \mathsf{sk}_i)$ for all $i \in \bar{S}$, which it can do because $\mathcal{B}$ knows sk_i for all $i \in \bar{S} \subseteq \mathsf{Cor}_{\mathsf{sk}}$. Whenever such a motivation function makes a query to $\mathsf{H}_{\mathsf{mask}}$, $\mathcal{B}$ forwards it to $\mathcal{O}^{\mathsf{check}}$ (as a direct query) before answering from the table. Denote as $\bar{S}'$ the set of parties where the above execution of the motivation function outputs 1.

Then, after dealing with the potential edge case, $\mathcal{B}$ forwards ρ to its challenger and receives $(S', (\boldsymbol{f}(i))_{i \in S'})$. $\mathcal{B}$ sets $\mathsf{sk}_i \leftarrow \boldsymbol{f}(i)$ for all $i \in S'$, updates $\mathsf{Cor}_{\mathsf{sk}} \leftarrow \mathsf{Cor}_{\mathsf{sk}} \cup S'$ and passes $(S' \cup \bar{S}', (\mathsf{sk}_i)_{i \in S' \cup \bar{S}'})$ to $\mathcal{A}$.

Simulating the Challenge Ciphertext. To simulate the challenge ciphertext on input (m_0, m_1), $\mathcal{B}$ sets $c_1^* \leftarrow ([\beta^\top] \mid [\Gamma])$ as mandated by either game. Then, $\mathcal{B}$ samples a random $H^* \xleftarrow{\$} \{0,1\}^\mu$ and sets $c_2^* \leftarrow H^* \oplus m_b$ for a random bit b and simulates the proof π^* using the simulator of PS. By this, $\mathcal{B}$ essentially simulates $\mathsf{H}_{\mathsf{mask}}$ for input $[\boldsymbol{Z}] = [(\boldsymbol{\beta}^\top \mid \boldsymbol{\Gamma})\boldsymbol{X}]$ without knowing $[\boldsymbol{Z}]$.

Analysis. Consider the case that Bad happens. In this case, $\mathcal{B}$'s simulation of either game to $\mathcal{A}$ is perfect until the $\mathsf{H}_{\mathsf{mask}}$ query happens that triggers Bad,[49] either directly or within a motivation function. To see this, consider the polynomial $\boldsymbol{f}$ sampled by the $\mathsf{GameStatCS}$ challenger and observe that $\mathcal{B}$ simulates such that $\boldsymbol{X} = \boldsymbol{f}(0)$ and $\boldsymbol{X}_i = \boldsymbol{f}(i)$. Hence, it holds that $\mathcal{B}$'s simulation is consistent with $\mathsf{pk} = [\boldsymbol{s}] = [\boldsymbol{a}\boldsymbol{f}(0)] = [\boldsymbol{a}\boldsymbol{X}]$ and $\mathsf{sk}_i = \boldsymbol{f}(i) = \boldsymbol{X}_i$ for all $i \in [n]$. Furthermore, whenever $\mathcal{A}$ makes a direct or indirect query to $\mathsf{H}_{\mathsf{mask}}$, $\mathcal{B}$ forwards its discrete logarithm to $\mathcal{O}^{\mathsf{check}}$. This means that, whenever $\mathcal{A}$ directly or indirectly queries $[\boldsymbol{Z}] = c_1^*\boldsymbol{X} = [(\boldsymbol{\beta}^\top \mid \boldsymbol{\Gamma})\boldsymbol{f}(0)]$, then $\mathcal{B}$ directly issues an $\mathcal{O}^{\mathsf{check}}$ query on $\hat{\boldsymbol{S}} = \boldsymbol{Z}$ that satisfies $\hat{\boldsymbol{S}} = \boldsymbol{Z} = (\boldsymbol{\beta}^\top \mid \boldsymbol{\Gamma})\boldsymbol{f}(0)$. Whenever Bad happens through a direct query, it holds that $|\mathsf{Cor}_{\mathsf{sk}}| = |\mathsf{Cor} \cup S^*| < t$. Whenever Bad happens through an indirect query (i.e., by a motivation function), one of two cases happens: (a) it happens outside the edge case, i.e., $|\mathsf{Cor}_{\mathsf{sk}}| < t-1$, and thus $|\mathsf{Cor} \cup S^*| < t-1$ and the wrapped ρ submitted by $\mathcal{B}$ is executed and makes the same query, or (b) it happens inside the edge case ($|\mathsf{Cor}_{\mathsf{sk}}| = t-1$) for some party $i \in \mathsf{Cor}_{\mathsf{sk}}$, and thus $\mathcal{B}$ makes the same query directly while $|\mathsf{Cor}| = t-1$.[50] This means $\mathcal{B}$ wins $\mathsf{GameStatCS}$ whenever Bad happens, and $\mathcal{B}$ makes at most $q_{\mathsf{H}_{\mathsf{mask}}}$ queries to $\mathcal{O}^{\mathsf{check}}$, allowing us to bound

$$\Pr[\mathsf{Bad}] \leq \Pr[\mathsf{GameStatCS}_{\mathcal{B}}(n, t, p, u, v, Q) = 1] \leq (n+1) \cdot q_{\mathsf{H}_{\mathsf{mask}}} \cdot \frac{\binom{n}{t-1}}{p^{u \cdot v}}.$$

■

[48] Intuitively, this is because even if these motivation functions output 1, this would not affect the adversary's budget, as $\bar{S} \subseteq \mathsf{Cor}_{\mathsf{sk}}$ already.

[49] Note that $\mathcal{B}$'s simulation of $\mathsf{H}_{\mathsf{mask}}$ is perfect prior to Bad. The answers from the table are consistent except at input $[\boldsymbol{Z}]$. Any prior queries on $[\boldsymbol{Z}]$ cannot be at budget $< t$, because otherwise, they would trigger Bad themselves. Due to budget monotonicity, any prior query at budget $\geq t$ would prevent Bad from happening later. Such queries may be answered inconsistently, but the corresponding runs can no longer trigger Bad and are thus irrelevant for this analysis.

[50] Note here that Bad happening implies that the invertibility condition holds, since either game would abort before Bad can happen otherwise.

Table 2: Concrete parameter sizes of our constructions with respect to parameters u, v, and ℓ in group elements $\mathbb{G}$ and field elements $\mathbb{F}$. Note that the proof systems for the respective constructions are instantiated with compact Schnorr proofs here, which accounts for the field elements in the ciphertext.

Scheme	Public key	Secret key share	Ciphertext overhead	Partial decryption
Π_{ZEG}	$v \cdot \mathbb{G}$	$(v+1) \cdot \mathbb{F}$	$u \cdot \mathbb{G} + (u+1) \cdot \mathbb{F}$	$u \cdot v \cdot \mathbb{G}$
Π_{Pair}	$1 \cdot \mathbb{G}_1$	$2 \cdot \mathbb{F}$	$1 \cdot \mathbb{G}_1 + 2 \cdot \mathbb{F}$	$\ell \cdot \mathbb{G}_T$
Π_{CS}	$v \cdot \mathbb{G}$	$(1+u) \cdot v \cdot \mathbb{F}$	$(u^2+u) \cdot \mathbb{G} + (u+1) \cdot \mathbb{F}$	$u \cdot v \cdot \mathbb{G}$

Final Bound on Game_8: Observe that in any winning execution of Game_8, the adversary cannot query $\mathsf{H}_{\mathsf{mask}}$ on $c_1^* \boldsymbol{X}$, and therefore $c_2^* = \mathsf{H}_{\mathsf{mask}}(c_1^* \boldsymbol{X}) \oplus m_b$ perfectly hides m_b behind a one-time-pad. Since the adversary's view does not depend on b otherwise, it holds that for all adversaries $\mathcal{A}$

$$\Pr\left[\mathsf{Game}_{8,\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2}.$$

Putting it all together: When applying Claims 6.7-6.14 alongside the above bound on the advantage in Game_8, we get that for all PPT adversaries $\mathcal{A}$ there exist PPT reductions $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ such that

$$\begin{aligned}\Pr\left[\mathsf{Game}_{0,\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + 2 \cdot \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{B}_1,(\mathsf{PS},\mathsf{Sim})}(1^\lambda) + \mathsf{Adv}^{(u,u)\text{-mDDH}}_{\mathcal{B}_2,\mathcal{G}}(1^\lambda) + \mathsf{Adv}^{\mathsf{SimSnd}}_{\mathcal{B}_3,(\mathsf{PS},\mathsf{Sim})}(1^\lambda) \\ + \frac{1}{p-1} + (n+1) \cdot q_{\mathsf{H}_{\mathsf{mask}}} \cdot \frac{\binom{n}{t-1}}{p^{u \cdot v}}.\end{aligned}$$

Whenever $\log(p^{uv}) \geq n + \lambda$, it holds that

$$\frac{\binom{n}{t-1}}{p^{uv}} \leq \frac{2^n}{2^{n+\lambda}} \leq 2^{-\lambda},$$

and thus the advantage is negligible, assuming $\log(p) \geq \lambda$, $u, v, q_{\mathsf{H}_{\mathsf{mask}}} \in \mathsf{poly}(\lambda)$, zero-knowledge and simulation-soundness of PS, and that the DDH assumption holds for $\mathcal{G}$. The theorem then follows from applying Facts 3.11 and 3.12. ∎

7 Tradeoff Discussion

We now discuss the efficiency of our constructions. In particular, we provide more insight into how to choose the parameter $\ell = uv$ and how it affects the ciphertext-key size tradeoff.

Concrete Choices of ℓ. For realistic settings, we have $p \approx 2^{2\lambda}$ and $t = cn$ for some constant $c \in (0,1)$, which is typically $c = 1/2$ or $c = 2/3$. Say our goal is to guarantee that the term in the statistical core lemmas is bounded by $2^{-\lambda}$. Since $p \approx 2^{2\lambda}$, this turns into

$$\binom{n}{t-1} \leq 2^{\lambda(2\ell-1)}.$$

Thus, the minimal integer $\ell \geq 1$ satisfying our requirement is

$$\ell = \max\left\{1, \left\lceil \frac{\log_2 \binom{n}{t-1}}{2\lambda} + \frac{1}{2} \right\rceil\right\}. \tag{4}$$

Using the entropy estimate

$$\log_2 \binom{n}{cn-1} \approx nH_2(c), \qquad H_2(c) = -c\log_2 c - (1-c)\log_2(1-c),$$

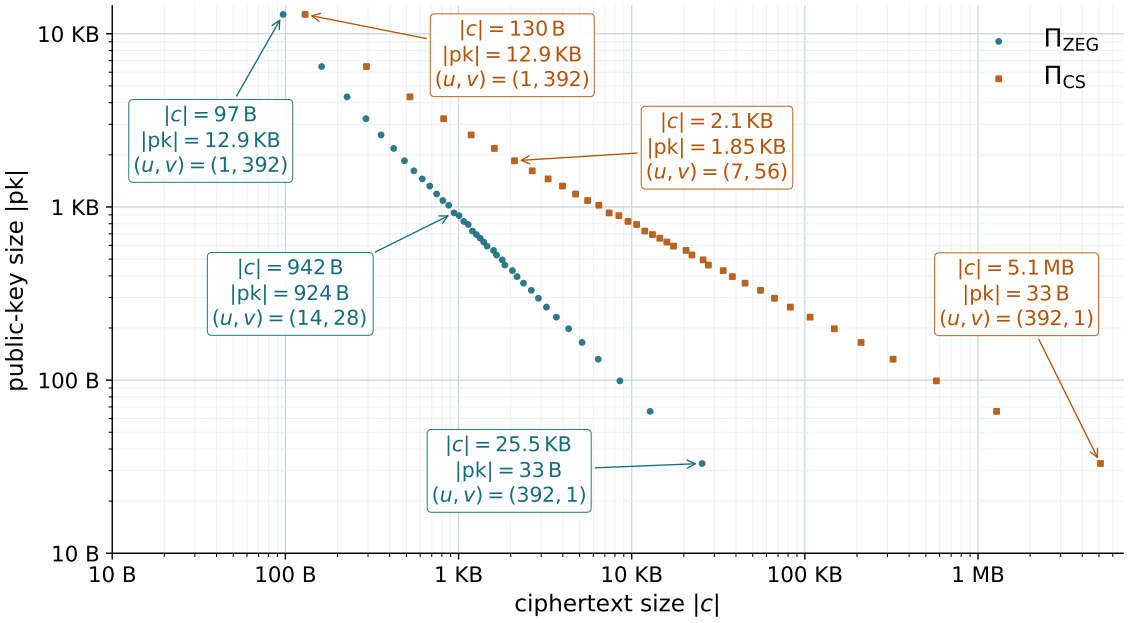


Figure 9: Ciphertext/public key size tradeoff for Π_{ZEG} and Π_{CS} at security level $\lambda = 128$, threshold $t = n/2$, $n = 100{,}000$, and $\ell = 392$. The plot shows the Pareto-optimal choices (u, v) satisfying $uv \geq \ell$, where ℓ satisfies Equation 4, using secp256k1 compressed group elements of size $|\mathbb{G}| = 33$ bytes and field elements of size 32 bytes. Callouts indicate the tradeoffs with the minimum and closest ciphertext overheads / public key sizes. Note that the partial decryption size for either scheme amounts to ≈ 12.9 KB in this setting.

we get the simpler approximation

$$\ell \approx \max\left\{1, \left\lceil \frac{nH_2(c)}{2\lambda} + \frac{1}{2} \right\rceil\right\}.$$

Here are some concrete estimates for $c = 1/2, \lambda = 128$:[51]

$$\begin{aligned} n = 10 &\implies \ell = 1, \\ n = 100 &\implies \ell = 1, \\ n = 1000 &\implies \ell = 5, \\ n = 10000 &\implies \ell = 40, \\ n = 100000 &\implies \ell = 392. \end{aligned}$$

The above shows that the factors hidden by the asymptotic requirement that ℓ is linear in n play in our favor here, in particular due to the 2λ in the denominator, making our constructions scale nicely for moderate n. In particular, the maximum n for which $\ell = 1$ at security level $\lambda = 128$ is $n = 131$, i.e., for $n \leq 131$ we do not need to amplify entropy and can set $u = v = 1$ in constructions Π_{ZEG} and Π_{CS}, and $\ell = 1$ in Π_{Pair}.

Concrete Parameters. We next compare concrete instantiations of our constructions. For this, we first list the parameter sizes with respect to u, v, and ℓ, including proof sizes, in Table 2. We can use this alongside Equation 4 to obtain concrete sizes, and explore tradeoffs between ciphertext size overhead and public key size of Π_{ZEG} and Π_{CS}. Consider Figure 9 for an example that illustrates the tradeoff and demonstrates concretely efficient results even for a relatively large $n = 100000$, with the only downside being relatively large partial decryptions of size ≈ 12.9 KB. This quickly becomes efficient for smaller n, though. For example, for $n = 10000$, partial decryptions are already only 1.32 KB.

[51] Note that the binary entropy $H_2(c := 1/2) = 1$.

Our pairing-based construction Π_{Pair} improves this. When instantiated on the `BLS12-381` curve, it admits $|\mathsf{pk}| = 48$ B, $|\mathsf{sk}_i| = 64$ B, and a ciphertext size overhead of 112 B, entirely independent of n and t. Partial decryptions scale as $\ell \cdot 576$ B.

Acknowledgments

This work was partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 and Horizon Europe research and innovation programs (grant CRYPTOLAYER-101044770) and by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts, through their joint sponsorship of the National Research Center for Applied Cybersecurity ATHENE.

References

[1] Agarwal, A., Das, S., Gilkalaye, B.P., Rindal, P., Shoup, V.: BTX: Simple and efficient batch threshold encryption. Cryptology ePrint Archive, Paper 2026/754 (2026), https://eprint.iacr.org/2026/754 (Cited on Page 8.)

[2] Agarwal, A., Fernando, R., Pinkas, B.: Efficiently-thresholdizable batched identity based encryption, with applications. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part III. LNCS, vol. 16002, pp. 69–100. Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01881-6_3 (Cited on Page 8.)

[3] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Berlin, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5_28 (Cited on Page 7.)

[4] Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 113–134. Springer, Berlin, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_6 (Cited on Page 7.)

[5] Bebel, J., Ojha, D.: Ferveo: Threshold decryption for mempool privacy in BFT networks. Cryptology ePrint Archive, Report 2022/898 (2022), https://eprint.iacr.org/2022/898 (Cited on Page 8.)

[6] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). https://doi.org/10.1145/168588.168596 (Cited on Page 10.)

[7] Benhamouda, F., Degwekar, A., Ishai, Y., Rabin, T.: On the local leakage resilience of linear secret sharing schemes. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 531–561. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_18 (Cited on Page 7.)

[8] Bhattacharyya, R., Bormet, J., Faust, S., Mukherjee, P., Othman, H.: CCA-secure traceable threshold (ID-based) encryption and application. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 2324–2338. ACM Press (Oct 2025). https://doi.org/10.1145/3719027.3744876 (Cited on Page 8.)

[9] Boneh, D., Bünz, B., Nayak, K., Rotem, L., Shoup, V.: Context-dependent threshold decryption and its applications. In: Hanaoka, G., Yang, B.Y. (eds.) ASIACRYPT 2025, Part VI. LNCS, vol. 16250, pp. 506–538. Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5119-4_16 (Cited on Page 4, 5, 7, 8, 12, 15, 20.)

[10] Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Berlin, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_13 (Cited on Page 22.)

[11] Boneh, D., Laufer, E., Tas, E.N.: Batch decryption without epochs and its application to encrypted mempools. Cryptology ePrint Archive, Report 2025/1254 (2025), https://eprint.iacr.org/2025/1254 (Cited on Page 8.)

[12] Boneh, D., Nema, R., Roy, A., Tas, E.N.: Efficient batch threshold encryption using partial fraction techniques. Cryptology ePrint Archive, Paper 2026/674 (2026), https://eprint.iacr.org/2026/674 (Cited on Page 8, 28.)

[13] Boneh, D., Partap, A., Rotem, L.: Accountability for misbehavior in threshold decryption via threshold traitor tracing. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part VII. LNCS, vol. 14926, pp. 317–351. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68394-7_11 (Cited on Page 8.)

[14] Boneh, D., Partap, A., Zhandry, M.: Catching many traitors in threshold traitor tracing: Lower bounds and constructions. In: ASIACRYPT 2026 (2026) (Cited on Page 8.)

[15] Bormet, J., Choudhuri, A.R., Faust, S., Garg, S., Othman, H., Policharla, G.V., Qu, Z., Wang, M.: BEAST-MEV: Batched threshold encryption with silent setup for MEV prevention. In: ACM CCS 2026 (2026) (Cited on Page 8.)

[16] Bormet, J., Faust, S., Othman, H.: Threshold traitor tracing revisited: Insider attacks and multi-traitor tracing. In: TCC 2026 (2026) (Cited on Page 8.)

[17] Bormet, J., Faust, S., Othman, H., Qu, Z.: BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention. In: Bauer, L., Pellegrino, G. (eds.) USENIX Security 2025. pp. 3457–3476. USENIX Association (Aug 2025), https://www.usenix.org/conference/usenixsecurity25/presentation/bormet (Cited on Page 3, 8.)

[18] Bormet, J., Hofmann, J., Othman, H.: Traceable threshold encryption without a trusted dealer. In: Hanaoka, G., Yang, B.Y. (eds.) ASIACRYPT 2025, Part VI. LNCS, vol. 16250, pp. 471–505. Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5119-4_15 (Cited on Page 8.)

[19] Bormet, J., Othman, H.: Threshold traitor tracing with partial-insider resilience. In: ASIACRYPT 2026 (2026) (Cited on Page 8.)

[20] Boudgoust, K., Simkin, M.: Scale, round, break: Simple leakage attacks on secret sharing schemes. Cryptology ePrint Archive, Paper 2026/833 (2026), https://eprint.iacr.org/2026/833 (Cited on Page 7.)

[21] Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: 51st FOCS. pp. 501–510. IEEE Computer Society Press (Oct 2010). https://doi.org/10.1109/FOCS.2010.55 (Cited on Page 7.)

[22] Brandão, L.T.A.N., Peralta, R.: NIST First Call for Multi-Party Threshold Schemes. NIST Internal Report 8214C, National Institute of Standards and Technology, Gaithersburg, MD (Jan 2026). https://doi.org/10.6028/NIST.IR.8214C, https://doi.org/10.6028/NIST.IR.8214C (Cited on Page 3.)

[23] Brzuska, C., Klooß, M., Woo, I.K.Y.: Threshold public-key encryption: Definitions, relations, and CPA-to-CCA transforms. In: Bai, S., Persichetti, E. (eds.) PKC 2026, Part IV. LNCS, vol. 16554, pp. 162–197. Springer, Cham (May 2026). https://doi.org/10.1007/978-3-032-26740-5_6 (Cited on Page 6, 7, 10.)

[24] Canard, S., Papon, N., Phan, D.H.: Public traceability in threshold decryption. CiC **2**(2), 21 (2025). https://doi.org/10.62056/akjb0lmol (Cited on Page 8.)

[25] Chakrabarti, A., Maitra, M., Mondal, A., Sehgal, K.: Silent threshold traitor tracing & enhancing mempool privacy. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 1769–1783. ACM Press (Oct 2025). https://doi.org/10.1145/3719027.3765099 (Cited on Page 8.)

[26] Chakraborty, S., Paul, G., Rangan, C.P.: Efficient compilers for after-the-fact leakage: From CPA to CCA-2 secure PKE to AKE. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 17, Part I. LNCS, vol. 10342, pp. 343–362. Springer, Cham (Jul 2017). https://doi.org/10.1007/978-3-319-60055-0_18 (Cited on Page 7, 8.)

[27] Chakraborty, S., Rangan, C.P.: Public key encryption resilient to post-challenge leakage and tampering attacks. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 23–43. Springer, Cham (Mar 2019). https://doi.org/10.1007/978-3-030-12612-4_2 (Cited on Page 7, 8.)

[28] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 89–105. Springer, Berlin, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_7 (Cited on Page 46, 47.)

[29] Choudhuri, A.R., Garg, S., Piet, J., Policharla, G.V.: Mempool privacy via batched threshold encryption: Attacks and defenses. In: Balzarotti, D., Xu, W. (eds.) USENIX Security 2024. USENIX Association (Aug 2024), https://www.usenix.org/conference/usenixsecurity24/presentation/choudhuri (Cited on Page 3, 8.)

[30] Choudhuri, A.R., Garg, S., Policharla, G.V., Wang, M.: Practical mempool privacy via one-time setup batched threshold encryption. In: Bauer, L., Pellegrino, G. (eds.) USENIX Security 2025. pp. 3477–3495. USENIX Association (Aug 2025), https://www.usenix.org/conference/usenixsecurity25/presentation/choudhuri (Cited on Page 8.)

[31] Con, R., Shutty, N., Tamo, I., Wootters, M.: Repairing reed-solomon codes over prime fields via exponential sums. In: IEEE International Symposium on Information Theory, ISIT 2023, Taipei, Taiwan, June 25-30, 2023. pp. 1330–1335. IEEE (2023). https://doi.org/10.1109/ISIT54713.2023.10206937, https://doi.org/10.1109/ISIT54713.2023.10206937 (Cited on Page 7.)

[32] Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 103–118. Springer, Berlin, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_9 (Cited on Page 20.)

[33] Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer, Berlin, Heidelberg (Aug 1998). https://doi.org/10.1007/BFb0055717 (Cited on Page 5, 48.)

[34] Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Berlin, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7_4 (Cited on Page 48.)

[35] Dao, Q., Grubbs, P.: Spartan and bulletproofs are simulation-extractable (for free!). In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 531–562. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30617-4_18 (Cited on Page 10.)

[36] Del Pino, R., Katsumata, S., Maller, M., Mouhartem, F., Prest, T., Saarinen, M.J.O.: Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 219–248. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58723-8_8 (Cited on Page 20.)

[37] Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 120–127. Springer, Berlin, Heidelberg (Aug 1988). https://doi.org/10.1007/3-540-48184-2_8 (Cited on Page 3.)

[38] Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_28 (Cited on Page 3, 20.)

[39] Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Berlin, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_22 (Cited on Page 7.)

[40] Dziembowski, S., Faust, S., Luhn, J.: Shutter network: Private transactions from threshold cryptography. Cryptology ePrint Archive, Report 2024/1981 (2024), https://eprint.iacr.org/2024/1981 (Cited on Page 8.)

[41] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Berlin, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_8 (Cited on Page 48.)

[42] ethereum.org contributors: Maximal extractable value (MEV). Ethereum developer documentation (Feb 2026), https://ethereum.org/developers/docs/mev, accessed July 14, 2026 (Cited on Page 3.)

[43] Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Berlin, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7_5 (Cited on Page 11, 46, 52.)

[44] Fernando, R., Policharla, G.V., Tonkikh, A., Xiang, Z.: TrX: Encrypted mempools in high performance BFT protocols. Cryptology ePrint Archive, Report 2025/2032 (2025), https://eprint.iacr.org/2025/2032 (Cited on Page 8.)

[45] Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Berlin, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_10 (Cited on Page 28, 46.)

[46] Fouque, P.A., Pointcheval, D.: Threshold cryptosystems secure against chosen-ciphertext attacks. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 351–368. Springer, Berlin, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_21 (Cited on Page 11, 46, 52.)

[47] Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_2 (Cited on Page 28.)

[48] Garg, S., Kolonelos, D., Policharla, G.V., Wang, M.: Threshold encryption with silent setup. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part VII. LNCS, vol. 14926, pp. 352–386. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68394-7_12 (Cited on Page 8.)

[49] Goldwasser, S., Kalai, Y.T., Park, S.: Adaptively secure coin-flipping, revisited. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015, Part II. LNCS, vol. 9135, pp. 663–674. Springer, Berlin, Heidelberg (Jul 2015). https://doi.org/10.1007/978-3-662-47666-6_53 (Cited on Page 8.)

[50] Gong, J., Waters, B., Wee, H., Wu, D.J.: Threshold batched identity-based encryption from pairings in the plain model. In: Daemen, J., Thomé, E. (eds.) EUROCRYPT 2026, Part V. LNCS, vol. 16545, pp. 548–578. Springer, Cham (May 2026). https://doi.org/10.1007/978-3-032-25330-9_19 (Cited on Page 8.)

[51] Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 107–124. Springer, Berlin, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_8 (Cited on Page 7, 8, 48.)

[52] Hall-Andersen, M., Simkin, M., Wagner, B.: Silent threshold encryption with one-shot adaptive security. Cryptology ePrint Archive, Report 2025/1384 (2025), https://eprint.iacr.org/2025/1384 (Cited on Page 3, 4, 6, 8, 15, 18, 19.)

[53] Kasser, D.: An improvement upon the bounds for the local leakage resilience of shamir's secret sharing scheme. In: Boyle, E., Mahmoody, M. (eds.) TCC 2024, Part IV. LNCS, vol. 15367, pp. 395–422. Springer, Cham (Dec 2024). https://doi.org/10.1007/978-3-031-78023-3_13 (Cited on Page 7.)

[54] Katsumata, S., Reichle, M., Takemure, K.: Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part VII. LNCS, vol. 14926, pp. 459–491. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68394-7_15 (Cited on Page 20.)

[55] Kavousi, A., Le, D.V., Jovanovic, P., Danezis, G.: Blindperm: Efficient mev mitigation with an encrypted mempool and permutation. In: 29th International Conference on Principles of Distributed Systems (OPODIS 2025). pp. 36–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2026) (Cited on Page 3.)

[56] Klein, O., Komargodski, I.: New bounds on the local leakage resilience of Shamir's secret sharing scheme. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 139–170. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_5 (Cited on Page 7.)

[57] Lewko, A.B., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 70–88. Springer, Berlin, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_6 (Cited on Page 7.)

[58] Lührs, F., Bezzenberger, L., Mosterts, F., Faust, S., Erwig, A.: The road towards an encrypted mempool on ethereum. Shutter Docs, https://docs.shutter.network/docs/shutter/research/the_road_towards_an_encrypted_mempool_on_ethereum, accessed July 14, 2026 (Cited on Page 3.)

[59] Maji, H.K., Nguyen, H.H., Paskin-Cherniavsky, A., Suad, T., Wang, M.: Leakage-resilience of the Shamir secret-sharing scheme against physical-bit leakages. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part II. LNCS, vol. 12697, pp. 344–374. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77886-6_12 (Cited on Page 7.)

[60] Maji, H.K., Nguyen, H.H., Paskin-Cherniavsky, A., Suad, T., Wang, M., Ye, X., Yu, A.: Leakage-resilient linear secret-sharing against arbitrary bounded-size leakage family. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 355–383. Springer, Cham (Nov 2022). https://doi.org/10.1007/978-3-031-22318-1_13 (Cited on Page 7.)

[61] Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Berlin, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03356-8_2 (Cited on Page 7, 48.)

[62] Nguyen, H.H.: Towards breaking the half-barrier of local leakage-resilient shamir's secret sharing. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part V. LNCS, vol. 14924, pp. 257–285. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68388-6_10 (Cited on Page 7.)

[63] Pedersen, T.P.: A threshold cryptosystem without a trusted party (extended abstract) (rump session). In: Davies, D.W. (ed.) EUROCRYPT'91. LNCS, vol. 547, pp. 522–526. Springer, Berlin, Heidelberg (Apr 1991). https://doi.org/10.1007/3-540-46416-6_47 (Cited on Page 3.)

[64] Policharla, G.V.: A simple batched threshold encryption scheme. Cryptology ePrint Archive, Paper 2026/760 (2026), https://eprint.iacr.org/2026/760 (Cited on Page 8.)

[65] Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS. pp. 543–553. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814628 (Cited on Page 11.)

[66] Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (Nov 1979). https://doi.org/10.1145/359168.359176 (Cited on Page 9.)

[67] Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT'98. LNCS, vol. 1403, pp. 1–16. Springer, Berlin, Heidelberg (May / Jun 1998). https://doi.org/10.1007/BFb0054113 (Cited on Page 46.)

[68] Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 755–784. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_25 (Cited on Page 10.)

[69] Wagner, B., Ma, J.: Hybrid encrypted mempools. Ethereum Research (Oct 2025), https://ethresear.ch/t/hybrid-encrypted-mempools/23360, accessed July 14, 2026 (Cited on Page 3.)

[70] Waters, B., Wu, D.J.: Silent threshold cryptography from pairings: Expressive policies in the plain model. In: Daemen, J., Thomé, E. (eds.) EUROCRYPT 2026, Part V. LNCS, vol. 16545, pp. 517–547. Springer, Cham (May 2026). https://doi.org/10.1007/978-3-032-25330-9_18 (Cited on Page 8.)

[71] Zhang, C., Yuen, T.H., Xiong, H., Chow, S.S.M., Yiu, S.M., He, Y.J.: Multi-key leakage-resilient threshold cryptography. In: Chen, K., Xie, Q., Qiu, W., Li, N., Tzeng, W.G. (eds.) ASIACCS 13. pp. 61–70. ACM Press (May 2013). https://doi.org/10.1145/2484313.2484321 (Cited on Page 8.)

[72] Zhang, Z., Chow, S.S.M., Cao, Z.: Post-challenge leakage in public-key encryption. Theoretical Computer Science **572**, 25–49 (2015). https://doi.org/10.1016/j.tcs.2015.01.010 (Cited on Page 7, 8.)