Adaptively Secure Threshold ElGamal Decryption from DDH

Sourav Das[1], Ling Ren[2], and Ziling Yang[2]

[1] Category Labs
[2] University of Illinois at Urbana-Champaign
souravdas1547@gmail.com, {renling, zilingy2}@illinois.edu

Abstract. Threshold decryption schemes allow a group of decryptors, each holding a private key share, to jointly decrypt ciphertexts. Over the years, numerous threshold decryption schemes have been proposed for applications such as secure data storage, internet auctions, and voting, and recently as a tool to protect against miner-extractable value attacks in blockchain. Despite the importance and popularity of threshold decryption, many natural and practical threshold decryption schemes have only been proven secure against static adversaries.

In this paper, we present two threshold decryption schemes that withstand malicious adaptive corruption. Our first scheme is based on the standard ElGamal encryption scheme and is secure against chosen plaintext attack (CPA). Our second scheme, based on the chosen ciphertext attack (CCA) secure Shoup-Gennaro encryption scheme, is also CCA secure. Both of our schemes have non-interactive decryption protocols and comparable efficiency to their static secure counterparts. Building on the technique introduced by Das and Ren (CRYPTO 2024), our threshold ElGamal decryption scheme relies on the hardness of Decisional Diffie-Hellman and the random oracle model.

Keywords: Threshold cryptography · adaptive security · chosen plaintext security · chosen ciphertext security · public key encryption.

1 Introduction

Threshold public-key decryption schemes [8, 9] distribute the private decryption key among a group of decryptors. To decrypt a ciphertext, each decryptor publishes a partial decryption share computed from their private key. Any threshold number of correct partial decryption shares can recover the plaintext, while fewer shares reveal no information about the plaintext.

One simple and popular threshold decryption scheme is the threshold ElGamal decryption scheme, first introduced by Desmedt [9] and later refined by Cramer, Gennaro, and Schoenmakers [6], which adapts the classic (non-threshold) ElGamal encryption scheme [12]. The ElGamal threshold decryption scheme preserves the advantages of ElGamal encryption, such as being simple and relying only on the Decisional Diffie-Hellman (DDH) assumption. Also

inherited from its non-threshold counterpart, threshold ElGamal encryption is only secure against chosen plaintext attacks (CPA), where the attacker can only obtain encryptions of plaintexts of its choosing.

A stronger security notion is chosen ciphertext security (CCA), where the attacker can additionally obtain decryptions of ciphertexts of its choosing. One of the first threshold CCA secure decryption schemes is Canetti and Goldwasser [4], which modified the Cramer-Shoup CCA (non-threshold) encryption. Their scheme requires decryptors to hold a large amount of shared randomness and to interact during decryption for robustness. Shoup and Gennaro [28] point out that CCA-secure threshold decryption schemes require partial decryptions to be publicly verifiable and constructed the first non-interactive CCA-secure threshold decryption scheme. Their scheme is based on ElGamal and DDH in the random oracle model (ROM), and uses a non-interactive zero-knowledge proof to achieve public verifiability of partial decryptions. Boneh et al. [1] proposed the first non-interactive CCA-secure threshold decryption scheme that does not rely on the ROM; it uses bilinear pairing for public verifiability. Other CCA-secure threshold decryption schemes include [29, 11, 22].

All of the above schemes assume a static adversary. A static adversary is one that decides the set of decryptors to corrupt before the execution of the protocol. A more powerful and more realistic adversary is an adaptive one that can adapt its corruption strategy according to its view during the protocol execution and can corrupt any party at any point of the execution. Many existing schemes that achieve adaptive security and CCA security require interaction during the decryption process or/and an offline phase [3, 14, 15, 16]. Only a few threshold decryption schemes exist that are non-interactive and CCA-secure in the adaptive corruption setting. Libert and Yung [19, 20] proposed such schemes using bilinear pairings, and Devevey et al. [10] introduced a lattice-based construction.

Our Contribution. In this paper, we propose two non-interactive threshold decryption schemes secure against adaptive corruptions. Both schemes rely on the DDH assumption in the Random Oracle Model.

1. A CPA-secure threshold decryption scheme compatible with the non-threshold ElGamal, i.e., the encryption algorithm is identical to classic ElGamal.
2. A CCA-secure threshold decryption scheme based on the Shoup-Gennaro, which is a natural CCA version of threshold ElGamal.

These are the first non-interactive threshold decryption schemes secure against adaptive corruptions based on the standard DDH assumption in the ROM.

Due to space constraints, we present the CCA scheme and analysis in the main body, and defer the CPA scheme and analysis to the appendix.

1.1 Technical Overview

We provide an overview of our CPA-secure scheme based on the standard ElGamal decryption scheme.

Threshold CPA secure decryption scheme. We first describe the standard ElGamal encryption [12]. The ElGamal encryption of a message m is $\mathsf{ctxt} =$

$(c, u) = (m \cdot \mathsf{pk}^r, g^r)$ where r is secret randomness chosen by the encryptor and pk is the public key in the form of $\mathsf{pk} = g^x$. To recover the message, a decryptor with the decryption key x, computes the message as $m = c/u^x$.

In the static secure threshold ElGamal decryption scheme [9, 6], the i-th decryptor holds a Shamir secret share of the decryption key x. Let x_i be the secret share of decryptor i. To decrypt the ciphertext ctxt, each decryptor i publishes its decryption share u^{x_i}. A combiner recovers the plaintext from a threshold number of valid decryption shares by interpolating in the exponent. In the threshold ElGamal decryption scheme, we publish a per party public key $\mathsf{pk}_i = g^{x_i}$ to check the validity of partial decryptions.

The static security of threshold ElGamal decryption reduces breaking IND-TCPA to the DDH assumption. The reduction embeds g^α into $\mathsf{pk} = g^\alpha$ and must provide consistent pk_i. For corrupt parties, it can pick secret keys directly while for honest parties, it solves for their pk_i through interpolation in the exponent. This works in the static setting but fails adaptively: once a new party is corrupted, the reduction must reveal sk_i consistent with pk_i, which requires solving discrete log. Thus, standard threshold ElGamal cannot be proved adaptively secure.

The recent work of Das and Ren [7] introduced a proof technique for an adaptively secure threshold BLS signature scheme. We adapt this to threshold decryption in our paper. Observe that the challenge of adaptive security lies in providing correct Shamir secret shares of α when only g^α is known after embedding the DDH challenge into the public key. Further observe that if the aggregate secret key, α is known, then the simulation becomes trivial.

To embed g^α into the public key and provide a consistent view, we follow Das and Ren's approach and modify the standard ElGamal public key to $\mathsf{pk} = g^x h^y$. The additional group element h in our public key allows the reduction adversary to set $h = g^\alpha$ and locally sample the decryption key (x, y). In our scheme, we use $y = 0$ so that the encryption protocol remains unchanged from the standard ElGamal scheme. However, in the security proof against adaptive corruptions, the reduction adversary, when interacting with the adaptive adversary, will run an instance of our protocol with a rigged public key with $y \neq 0$.

The final part of our protocol is how the reduction adversary simulates consistent decryption shares under the rigged public key. The original threshold ElGamal's partial decryption protocol will produce a ciphertext $m' = m \cdot (g^{rx}h^{ry})/g^{rx} = m \cdot h^{ry}$ where r is the randomness chosen in the encryption protocol. Notice that m' is not the underlying plaintext. Therefore, the reduction adversary cannot simulate a correct view with the original decryption protocol. Thus, we use a programmed random oracle to recover the plaintext. Our modified partial decryption protocol will output a partial decryption of $\hat{m}_i = g^{x_i r} \cdot \mathsf{H}(\mathsf{ctxt})^{y_i}$ instead of the original $g^{x_i r}$. Then, by programming $\mathsf{H}(\mathsf{ctxt}) = h^r$, we successfully recover the correct plaintext. In our modified scheme, the combiner computes $m \cdot \mathsf{pk}^r/\hat{m} = m \cdot (g^{rx}h^{ry})/g^{xr}h^{yr} = m$, where $\hat{m}$ is the aggregated partial decryption. Notice that in the real-world execution

of our scheme, when $y = 0$, the partial decryptions also correctly recover the underlying plaintext m.

Threshold CCA secure decryption scheme. Our threshold CCA secure decryption scheme is built upon Shoup-Gennaro's $\mathsf{TDH2}$ [‡] [28] scheme, which builds upon the basic threshold ElGamal scheme and achieves CCA security by including extra elements that allow the reduction simulator to simulate decryption queries. The classical Elgamal encryption is of the form $(u, c) = (g^r, g^r) \cdot m$. The extra elements take the form of $\bar{u} = \bar{g}^r$ where $\bar{g}$ is a generator and a NIZK proof of equality of discrete logarithm of u and $\bar{u}$. Shoup-Gennaro's scheme allows straight-line extraction of (with knowledge of a trapdoor) sufficient information for answering the adversary's decryption queries.

To encrypt a message m, the encryptor computes

$$u := g^r, c := m \cdot \mathsf{pk}^r, w := g^s, \bar{u} := \bar{g}^r, \bar{w} := \bar{g}^s, e := \mathsf{H}(c, u, w, \bar{u}, \bar{w}), f := s + re$$

The ciphertext is $\mathsf{ctxt} := (c, u, \bar{u}, e, f)$. Here u, c are consistent with threshold ElGamal while $\bar{u}$ allows the simulator to answer decryption queries during the security proof and e, f are part of the zero-knowledge proof of equality of discrete log of u and $\bar{u}$.

During decryption, each party verifies the zero-knowledge proof of knowledge by checking:

$$e = \mathsf{H}(c, u, w, \bar{u}, \bar{w}), \text{ where } w := g^f / u^e, \bar{w} := \bar{g}^f / \bar{u}^e$$

For each party i, if the check fails, the party outputs $\perp$; otherwise, it outputs $\hat{m}_i := u^{x_i}$ where x_i is party i's secret key as in the standard ElGamal scheme. A combiner recovers the plaintext in the same manner as the standard ElGamal scheme as described above. Since the zero-knowledge proof is publicly verifiable, the combiner first verifies that the zero-knowledge proof is valid before computing the plaintext.

Similar to threshold ElGamal, the CCA security proof of Shoup-Gennaro relies on embedding the DDH tuple of $(g^\alpha, g^\beta, g^\zeta)$ into the scheme. The reduction adversary in the Shoup-Gennaro scheme faces the same difficulty as in the standard threshold ElGamal when interacting with an adaptive adversary. Namely, the reduction adversary must provide consistent secret keys upon corruption after committing to each secret key via the per-party public key while the aggregate secret is unknown to the reduction adversary.

We again follow the approach of Das and Ren [7] to embed the DDH challenge elsewhere so that the aggregate secret key is known to the reduction adversary. As in our CPA secure scheme, we include an additional group element, h, in our public key to help embed the DDH challenge, i.e., $h = g^\alpha$ where g^α is part of the DDH challenge. However, if we modify the public key to $\mathsf{pk} := g^x h^y$ with $y = 0$ and rig the public key in our security proof such that $y \neq 0$ as in our

[‡] We modified $\mathsf{TDH2}$ to rely on the DDH assumption. The original work relies on the Computational Diffie-Hellman assumption.

CPA secure scheme, the reduction adversary immediately runs into a problem when simulating decryption queries. As in our CPA secure scheme, the simulated decryption shares are inconsistent with the underlying plaintext with a rigged public key. Recall that under the rigged public key, the decryption protocol will produce $m' := m \cdot g^{xr} h^{yr} / g^{xr} := m \cdot h^{rv}$ where r is the chosen randomness in the ciphertext. Thus, the simulation is clearly distinguishable from the real execution of the protocol. The reduction adversary must program $\mathsf{H}(\mathsf{ctxt}) = h^r$ to produce consistent decryption shares of ctxt. The CCA adversary, compared to a CPA adversary, can additionally query ciphertexts to obtain all decryption shares of the queried ciphertext. During decryption queries, the reduction adversary runs into two challenges: 1) the adversary knows r, so $\mathsf{H}(\mathsf{ctxt}) = h^r$ is distinguishable from true randomness; 2) r is unknown to the reduction adversary.

To remedy the first problem, we introduce a third group element, v^z, in the public key, i.e., $\mathsf{pk} = g^x h^y v^z$ where $z = 0$. The inclusion of a third group element allows us to introduce a third programmed random oracle in the partial decryption share to mask h^r with additional randomness. Then, the partial decryption takes the form of $\hat{m}_i := u^{x_i} \mathsf{H}_2(\mathsf{ctxt})^{y_i} \mathsf{H}_3(\mathsf{ctxt})^{z_i}$. We rig z such that $z = 1$ in the security proof.

The simulator programs $\mathsf{H}_2(\mathsf{ctxt}), \mathsf{H}_3(\mathsf{ctxt})$ as follows:

$$\mathsf{H}_2(\mathsf{ctxt}) := g^b \quad \text{and} \quad \mathsf{H}_3(\mathsf{ctxt}) := \frac{h^{ry} v^r}{g^{by}} \quad \text{where } b \leftarrow\$ \mathbb{Z}_p$$

$\mathsf{H}_2(\mathsf{ctxt}) = g^b$ is clearly random. Intuitively, the output of $\mathsf{H}_3(\mathsf{ctxt})$ is indistinguishable from true randomness because both b and y are unknown to the adversary. By assuming the hardness of DDH, we argue that $\frac{h^{ry} v^r}{g^{by}}$ is indistinguishable from $\frac{h^{ry} v^r}{g^{\eta}}$ where η is sampled randomly. Next, we show that the recovered plaintext is consistent:

$$\hat{m} = u^x \cdot \mathsf{H}_2(\mathsf{ctxt})^y \cdot \mathsf{H}_3(\mathsf{ctxt})^z = u^x \cdot g^{by} \cdot \frac{h^{ry} v^r}{g^{by}} = u^x h^{ry} v^{rz}$$

The remaining challenge is computing $\mathsf{H}_3(\mathsf{ctxt})$ when r is unknown to the reduction adversary. More specifically, the reduction adversary needs to compute $(h^y v)^r$ in order to program H_3. Recall that the adversary in its decryption query provides both $u := g^r$ and $\bar{u} := \bar{g}^r$ where $\bar{g}$ is a random generator. Borrowing Shoup-Gennaro's proof technique of straightline (without rewinding) extracting pk^r from $\bar{u}$ to simulate decryption queries, we extract $(h^y v)^r$. Our simulator picks $\bar{g} := (h^y v)^k, k \leftarrow\$ \mathbb{Z}_p$. Then $(h^y v)^r := \bar{u}^{1/k}$.

Notice that in the real-world execution of our scheme, when $y = z = 0$, the partial decryptions also correctly recover the underlying plaintext m.

2 Preliminaries

Notations. For any integer a, we use $[a]$ to denote the ordered set $\{1, 2, \ldots, a\}$. For any set S, we use $s \leftarrow\$ S$ to indicate that s is sampled uniformly randomly

from S. We use $|S|$ to denote the size of set S. Throughout the paper, we will use "$\leftarrow$" for probabilistic assignment and ":=" for deterministic assignment. We use λ to denote the security parameter and $\mathsf{negl}(\lambda)$ to denote functions negligible in λ. We use the terms *party* (resp. *parties*) and *decryptor* (resp. *decryptors*) interchangeably.

2.1 Model

We consider a set of n parties denoted by $\{1, 2, \ldots, n\}$. We consider a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ who can adaptively corrupt up to $t < n$ out of the n parties. The adversary can adaptively decide to corrupt a party at any time during the execution of the protocol. Corrupted parties are controlled by the adversary and can deviate arbitrarily from the protocol specification. Note that with $t \geq n/2$, i.e., with a dishonest majority, it is impossible to achieve both secrecy and guaranteed output delivery [17]. We focus on secrecy over guaranteed output delivery for the dishonest majority case.

2.2 Shamir Secret Sharing, NIZK protocols, and Assumptions

Shamir secret sharing. The Shamir secret sharing [26] embeds the secret s in the constant term of a polynomial $p(x) = s + a_1x + a_2x^2 + \cdots + a_dx^d$, where other coefficients $a_1, \cdots, a_d$ are chosen uniformly randomly from a field $\mathbb{Z}_p$. The i-th share of the secret is $p(i)$, i.e., the polynomial evaluated at i. Given $d + 1$ distinct shares, one can efficiently reconstruct the polynomial and the secret s using Lagrange interpolation. Also, s is information-theoretically hidden from an adversary that knows d or fewer shares.

NIZK. Throughout our paper, we will use non-interactive zero-knowledge proofs of knowledge (NIZK) via the Fiat-Shamir heuristic [13]. A zero-knowledge proof of knowledge protocol allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that he knows a secret witness that satisfies some property without revealing to $\mathcal{V}$ any additional information about the witness. Starting from an interactive Σ-protocol with three rounds: commit, challenge, response, Fiat–Shamir makes it non-interactive by replacing the challenge round with a hash of prior messages. We require the NIZK protocols to satisfy the standard *completeness*, *soundness*, and *zero-knowledge*. We adapt the definition of NIZK from [2, §20.3].

Definition 1 (NIZK). *Let $\mathcal{R}$ be a binary relation and $(x, y) \in \mathcal{R}$ be a witness statement pair. A NIZK protocol for $\mathcal{R}$ is a pair of polynomial-time algorithms:*

1. $\mathsf{SigmaProve}(x, y) \rightarrow \pi$*: $\mathcal{P}$ generates a proof for $(x, y) \in \mathcal{R}$.*
2. $\mathsf{SigmaVer}(y, \pi) \rightarrow 0/1$*: $\mathcal{V}$ to accept or reject π.*

A NIZK protocol satisfies the following properties:

- Completeness. *If $(x, y) \in \mathcal{R}$ then,* $\mathsf{SigmaVer}$*(y,* $\mathsf{SigmaProve}$*(x,y)) = 1.*

- Soundness. *An adversary can only cause* $\mathcal{V}$ *to output* 1 *on* $(x, y) \notin \mathcal{R}$ *with negligible probability.*
- Zero-Knowledge. *There exists an efficient probabilistic simulator* $\mathcal{S}$ *that takes as input a statement* $y \in \mathcal{R}$ *and outputs* π *that is an accepting proof of* y *and has the same distribution as an honestly computed proof for statement* y.

We also use the following computational assumption defined for groups generated by the GGen algorithm. More precisely, let GGen be a group generation algorithm that on input 1^λ outputs the description of a prime order group $\mathbb{G}$. The description includes the prime order p, a generator $g \in \mathbb{G}$, and a description of the group operation.

Assumption 1 (DDH) *We say that the decisional Diffie-Hellman (DDH) assumption holds relative to* GGen*, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \left| \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^{xy}) = 1 \;\middle|\; \begin{array}{r}(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda),\\ x, y \leftarrow\!\!\$\ \mathbb{Z}_p\end{array}\right] \right.$$
$$\left. - \Pr\left[\mathcal{A}(\mathbb{G}, p, g, g^x, g^y, g^z) = 1 \;\middle|\; \begin{array}{r}(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda),\\ x, y, z \leftarrow\!\!\$\ \mathbb{Z}_p\end{array}\right] \right| = \varepsilon_{\mathsf{ddh}}.$$

2.3 Threshold Decryption

Definition 2 (Public Key Threshold Decryption scheme). *Let* λ *be the security parameter and* $t, n \in \mathbb{N}$ *with* $t < n$*. A non-interactive threshold decryption scheme* TD *for a finite message space* $\mathcal{M}$ *is a tuple of polynomial time algorithms* $\mathsf{TD} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{PDec}, \mathsf{PVer}, \mathsf{Comb})$ *defined as follows:*

1. $\mathsf{Setup}(1^\lambda, n, t) \to \mathsf{pp}$: *the setup algorithm takes as input a security parameter and outputs public parameters* pp *(which are given implicitly as input to all other algorithms).*
2. $\mathsf{KGen}(\mathsf{pp}) \to \mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]}$: *the key generation algorithm outputs a public key* pk*, a vector of per party public keys* $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$*, and a vector of secret key shares* $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}$*. The* j*-th decryptor receives* $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{sk}_j)$.
3. $\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ctxt}$: *the encryption algorithm takes as input a public key* pk *and a message* $m \in \mathcal{M}$*. It outputs a ciphertext* ctxt.
4. $\mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt}) \to \hat{m}_i, \pi_i$: *the partial decryption algorithm takes as input a secret key share* sk_i *and a ciphertext* ctxt *and outputs a decryption share* $\hat{m}_i$ *for* c*, and a proof* π_i *for* $\hat{m}$.
5. $\mathsf{PVer}(\mathsf{pk}_i, \mathsf{ctxt}, \hat{m}_i, \pi_i) \to 0/1$*: On input the per party public key* pk_i *the ciphertext* ctxt*, and a decryption share* $\hat{m}_i$*, and a proof for* $\hat{m}_i$*,* π_i*. It outputs* 0 *(reject) or* 1 *(accept).*
6. $\mathsf{Comb}(\{pk_i\}_{i\in[n]}, \mathsf{ctxt}, S, \{\hat{m}_i, \pi_i\}_{i\in S}) \to m$: *the combine algorithm takes as input a set of* n *per party public keys* $\{pk_i\}_{i\in[n]}$*, a ciphertext* ctxt*, a set* S *that is a subset of* $\{1, \ldots, n\}$ *and* $|S| > t$*, and a set of decryption share of* ctxt *and proof pairs,* $\{\hat{m}_i, \pi_i\}_{i\in S}$ *and outputs a message* m.

Game $\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}(1^\lambda, n, t)$:

1: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n, t,)$ // public parameters
2: $\mathcal{C} := \varnothing$
3: $\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\} \leftarrow \mathsf{KGen}(\mathsf{pp})$
4: $\mathsf{inp} := \mathsf{pp}, \mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}$
5: $m_0, m_1 \leftarrow \mathcal{A}^{\textsc{Corr},\textsc{Enc},\textsc{Dec}}(\mathsf{inp})$
6: $\mu \leftarrow\$ \{0,1\}$
7: $\mathsf{ctxt}_\mu \leftarrow \mathsf{Enc}(\mathsf{pk}, m_\mu)$
8: $\mu_{\mathcal{A}} \leftarrow \mathcal{A}^{\textsc{Corr},\textsc{Enc},\textsc{Dec}}(\mathsf{ctxt}_\mu)$
9: **if** $\mu = \mu_{\mathcal{A}}$ **then**
10: **return** 1
11: **else**
12: **return** 0

Oracle $\textsc{Corr}(i)$:
13: **if** $|\mathcal{C}| \geq t$ **then**
14: **return** $\perp$
15: $\mathcal{C} := \mathcal{C} \cup \{i\}$
16: **return** sk_i

Oracle $\textsc{Enc}(m)$:
17: $\mathsf{ctxt} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$
18: $\hat{m}_i \leftarrow \mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt})$ for all $i \in [n]$
19: **return** $(\mathsf{ctxt}, \{\hat{m}_i, \pi_i\}_{i\in[n]})$

Oracle $\textsc{Dec}(\mathsf{ctxt}, i)$:
20: **assert** $\mathsf{ctxt} \neq \mathsf{ctxt}_\mu$ // $\mathcal{A}$ cannot invoke $\textsc{Dec}$ on ctxt_μ
21: $\hat{m}_i \leftarrow \mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt})$
22: **return** $\hat{m}_i$

Fig. 1: The chosen ciphertext game $\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ for a non-interactive threshold decryption $\mathsf{TD} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{PDec}, \mathsf{Comb})$ with an adaptive adversary $\mathcal{A}$.

We require that a threshold decryption scheme satisfy correctness, robustness, and indistinguishability. We formally define correctness and robustness in Section A of the appendix.

There are two notions of indistinguishability, IND-TCPA and IND-TCCA. In the $\mathsf{IND\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ game, an adversary has access to an encryption oracle $\textsc{Enc}$, which provides encryptions and their partial decryptions of all parties to chosen messages. In the $\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ game, the adversary additionally has access to a decryption oracle $\textsc{Dec}$, which provides decryption shares of the adversary's choice of ciphertext (except for the challenge). The adversary then submits two plaintexts and receives the encryption of one at random. Since the adversary is adaptive, it selects up to t parties at any time to see their secret keys. The encryption scheme is secure if the adversary cannot determine, with probability better than random, which plaintext is encrypted. We give the IND-TCCA definition below and defer IND-TCPA to Definition 4 of the appendix.

Definition 3 (Indistinguishability under Chosen-Ciphertext Attack). *Consider the* $\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ *game in Figure 1. We say that a threshold decryption scheme* TD *is IND-TCCA secure if, for all security parameters* λ, $n, t \in \mathbb{N}$ *with* $t < n$, *PPT adversaries* $\mathcal{A}$, *the following guessing advantage is negligible, i.e.,*

$$\varepsilon_{\text{IND-TCCA}} := \left| \Pr[\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}(1^\lambda, n, t) \Rightarrow 1] - \frac{1}{2} \right| = \mathsf{negl}(\lambda) \tag{1}$$

2.4 Helper Lemmas

We first restate two lemmas from Naor-Reingold [23, Lemma 4.4] and [7, Lemma 4]. We refer the readers to those papers for the proofs.

Lemma 1 (Lemma 4.4 in [23]). *For any security parameter λ, let $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$. For all $q \leq \mathsf{poly}(\lambda)$, assuming hardness of decisional Diffie-Hellman (DDH) assumption in $\mathbb{G}$, the following two distributions are indistinguishable.*

$$\mathcal{D}_0 := g, g^\alpha, \{(g^{\beta_i}, g^{\gamma_i})\}_{i\in[q_\mathsf{H}]} \text{ for } \alpha \leftarrow\$ \mathbb{Z}_p \text{ and } \forall i \in [q]\ (\beta_i, \gamma_i) \leftarrow\$ \mathbb{Z}_p^2 \quad (2)$$

$$\mathcal{D}_1 := g, g^\alpha, \{(g^{\beta_i}, g^{\alpha\cdot\beta_i})\}_{i\in[q]} \text{ for } \alpha \leftarrow\$ \mathbb{Z}_p \text{ and } \forall i \in [q]\ \beta_i \leftarrow\$ \mathbb{Z}_p \quad (3)$$

More precisely, if an adversary $\mathcal{A}$ can distinguish between a sample from $\mathcal{D}_0$ and $\mathcal{D}_1$ with probability ε, then $\mathcal{A}$ can break the DDH assumption with probability at least $\varepsilon - 1/p$. This implies $\varepsilon \leq \varepsilon_{\mathsf{ddh}} + 1/p$.

Lemma 2 (Lemma 4 in [7]). *Let (X_0, X_1) and (Y_0, Y_1) be two tuples of discrete random variables where X_0 is independent of Y_0, and X_1 is independent of Y_1. For every function $f(X_\theta, Y_\theta)$ for either $\theta \in \{0, 1\}$, if $\Pr[X_0 = x] = \Pr[X_1 = x]$ holds for all $x \in X_0 \cup X_1$ and $\Pr[Y_0 = y] = \Pr[Y_1 = y]$ holds for all $y \in Y_0 \cup Y_1$, then $\Pr[f(X_0, Y_0) = z] = \Pr[f(X_1, Y_1) = z]$ holds for all z in the range of f.*

3 Design of the IND-TCCA Decryption Scheme

In this section, we describe our threshold decryption scheme based on the Shoup-Gennaro scheme, which is a variant of the ElGamal encryption scheme that acheives non-malleability, thus CCA security, through additional ciphertext elements. Our scheme is summarized in Figure 6. We also provide a concrete instantiation of an NIZK protocol for our scheme in Figure 7 of the appendix. Let $\mathcal{M}$ denote the message space.

$\underline{\mathsf{Setup}(1^\lambda, n, t)}$: Let $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$. Let $h, v, \bar{g} \in \mathbb{G}$ be additional independent uniformly random generators of $\mathbb{G}$. Let $\mathsf{H}_1 : \{0,1\}^\ell \times \mathbb{G}^4 \to \mathbb{Z}_p$, $\mathsf{H}_2, \mathsf{H}_3 : \{0,1\}^\ell \times \mathbb{G}^2 \times \mathbb{Z}_p^2 \to \mathbb{G}$ and $\mathsf{H}_\mathsf{FS} : \{0,1\}^* \to \mathbb{G}$ be four distinct hash functions modeled as random oracles. The public parameters of our scheme are then $(\mathbb{G}, g, h, v, \bar{g}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_\mathsf{FS})$.

We assume that all the algorithms below implicitly take the public parameters as input.

$\underline{\mathsf{KGen}(\mathsf{pp})}$: The KGen algorithm samples three uniformly random polynomials $x(\cdot), y(\cdot), z(\cdot)$ of degree t with the constraint that $y(0) := z(0) := 0$. The decryption key of party i is then $\mathsf{sk}_i := (x(i), y(i), z(i))$. Let $\mathsf{pk} := g^{x(0)}$ be the public key, $\mathsf{pk}_i := g^{x(i)} h^{y(i)} v^{z(i)}$ for all $i \in [n]$ be the per-party public key.

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$: Our encryption algorithm is identical to the Shoup-Gennaro encryption scheme [27]. First, sample $r, s \leftarrow\$ \mathbb{Z}_p$, and then compute the following: $c := \mathsf{pk}^r \cdot m$, $u := g^r$, $w := g^s$, $\bar{u} := \bar{g}^r$, $\bar{w} := \bar{g}^s$, $e := \mathsf{H}_1(c, u, w, \bar{u}, \bar{w})$, and $f := s + er$. The ciphertext is $\mathsf{ctxt} := (c, u, \bar{u}, e, f)$

$\underline{\mathsf{PDec}(\mathsf{sk}_i, \mathrm{ctxt} = (c, u, \bar{u}, e, f))}$: On input the partial decryption key sk_i and the ciphertext tuple $\mathrm{ctxt} := (c, u, \bar{u}, e, f)$, check

$$e = \mathsf{H}_1(c, u, w, \bar{u}, \bar{w}), \text{ where } w := g^f / u^e, \bar{w} := \bar{g}^f / \bar{u}^e \quad (4)$$

Output $\perp$ if the check fails. Otherwise, compute partial decryption $\hat{m}_i := u^{x(i)} \cdot \mathsf{H}_2(\mathsf{ctxt})^{y(i)} \cdot \mathsf{H}_3(\mathsf{ctxt})^{z(i)}$ and a NIZK proof π_i that $\hat{m}_i$ is correctly computed from the ciphertext. π_i is computed using Σ-protocol described in Figure 4.

$\underline{\mathsf{PVer}(\mathsf{pk}_i, \text{ctxt} := (c, u, \bar{u}, e, f), \hat{m}_i, \pi_i)}$: On input the threshold public key pk_i the ciphertext ctxt $:= (c, u, \bar{u}, e, f)$, and partial decryption share tuple $(\hat{m}_i, \pi_i)$, validate the partial decryption share by running the Σ-protocol SigmaVer protocol. Output 1 if and only if SigmaVer outputs 1.

$\underline{\mathsf{Comb}(\{\mathsf{pk}_i\}_{i\in[n]}, \text{ctxt} := (c, u, \bar{u}, e, f), S, \{(\hat{m}_i, \pi_i)\}_{i\in S})}$: The input is a set of per party public keys of all parties, the ciphertext ctxt $:= (c, u, \bar{u}, e, f)$, a set of parties S, a set of partial decryption tuples that correspond to the parties in S, $(\hat{m}_i, \pi_i)$ for each $i \in S$. The combiner first validates the ciphertext by running the check in equation 4 and outputs $\perp$ if equation 4 evaluates to 0. Then the combiner uses PVer to validate each non-$\perp$ decryption share in S. Let T be the set of indices of parties with valid partial decryptions. It then computes the plaintext m as:

$$m := c \cdot \hat{m}^{-1}, \quad \text{where} \quad \hat{m} := \prod_{i\in T} \hat{m}_i^{L_{i,T}} \tag{5}$$

where $L_{i,T}$ is the i-th Lagrange coefficient with respect to the set T.

We provide an analysis of the IND-TCCA security of our scheme in the next section. We include analysis of the correctness and robustness property of our scheme in Section E of the appendix.

4 Analysis of IND-TCCA security

We will prove IND-TCCA of our threshold decryption scheme using a sequence of attack games. Game $\mathbf{G}_0$ is the $\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ game in which $\mathcal{A}$ makes queries to the decryption and encryption oracles and tries to guess the challenge bit μ. Game $\mathbf{G}_6$ is a game where $\mathcal{A}$ receives a ciphertext ctxt that is independent of the challenge bit μ. Therefore, the probability of $\mathcal{A}$ winning game $\mathbf{G}_6$ is exactly 1/2. For any adversary $\mathcal{A}$ in the k-th game $\mathbf{G}_k$, we will use G_k to denote the event that the game outputs 1, i.e., $\mathcal{A}$ wins the game $\mathbf{G}_k$.

$\underline{\text{GAME } \mathbf{G}_0}$: This is the security game $\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ for our threshold decryption scheme, where the game follows the honest protocol. Here, the game provides $\mathcal{A}$ with access to all random oracles via the standard lazy simulation technique. We assume that $\mathcal{A}$ always corrupts exactly t parties.

$\underline{\text{GAME } \mathbf{G}_1}$: This game is identical to game $\mathbf{G}_0$, except we sample $\alpha_h, \alpha_v, a, k \leftarrow\$ \mathbb{Z}_p$. $h = g^{\alpha_h}$, $v = g^{\alpha_v}$, $\bar{g} = g^{ak}$. Clearly, $\Pr[G_0] = \Pr[G_1]$.

$\underline{\text{GAME } \mathbf{G}_2}$: This game is identical to game $\mathbf{G}_1$, except that we sample $b_j, \eta_j \leftarrow\$ \mathbb{Z}_p$, and program the random oracles for the jth decryption query on ctxt_j to be $\mathsf{H}_2(\mathsf{ctxt}_j) := g^{b_j}$ and $\mathsf{H}_3(\mathsf{ctxt}_j) := (g^{-\eta_j}(g^{b_j})^{\alpha_v})^{1/\alpha_h}\bar{u}^{1/k}$. More precisely, whenever $\mathcal{A}$ queries the random oracles H_2 or H_3 with some ctxt_j for the first time, we program H_2 and H_3 as described above.

Similarly, for each encryption query on message m_j, we honestly compute the encryption of m_j as per the protocol description. Then, to compute the partial decryptions of the ciphertext of m_j, we program the random oracles H_2 and H_3 according to the description above.

Looking ahead, this game and the following game are preparing for game $\mathbf{G}_5$ where we "rig" the secret keys such that y, z are non-zero. We program $\mathsf{H}_2, \mathsf{H}_3$ in such a way that in Comb, $\hat{m} := \mathsf{pk}^r$ and the decrypted plaintext is consistent with the honest protocol despite dishonestly chosen secret keys.

Since b_j, η_j are independently random, $\mathsf{H}_2, \mathsf{H}_3$ are indistinguishable from uniformly random functions. Thus $\Pr[\mathbf{G}_1] = \Pr[\mathbf{G}_2]$.

GAME $\mathbf{G}_3$: This game is identical to game $\mathbf{G}_2$, except we correlate H_2 and H_3 by setting $\eta_j := a \cdot b_j$. We show the indistinguishability between game $\mathbf{G}_2$ and game $\mathbf{G}_3$ assuming the hardness of DDH. More formally, we prove the following:

Lemma 3. *Let $\varepsilon_{\mathsf{ddh}}$ be the advantage of breaking DDH in $\mathbb{G}$ as defined in as defined in Assumption 1, then $|\Pr[G_2] - \Pr[G_3]| \leq \varepsilon_{\mathsf{ddh}} + 1/p$.*

Proof. We define a distinguisher $\mathcal{D}$ for distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ defined in lemma 1. Let $\{g, A, \{B_i, C_i\}_{i \in q_{\mathsf{H}}}\}$ be the input to $\mathcal{D}$. $\mathcal{D}$ interacts with $\mathcal{A}$ by setting $\bar{g} := A^k, k \leftarrow\$ \mathbb{Z}_p$, for the jth query, either encryption or decryption query, program $\mathsf{H}_2(\mathsf{ctxt}_j) := B_j, \mathsf{H}_3(\mathsf{ctxt}_j) := (C_j^{-1} {B_j}^{\alpha_v})^{1/\alpha_h} (\bar{u}_j)^{1/k}$ where ctxt_j is queried by $\mathcal{A}$ or computed by the simulator. Lastly, $\mathcal{D}$ outputs 1 if $\mathcal{A}$ wins the attack game.

If $\mathcal{D}$ receives $\mathcal{D}_0$, $\mathcal{A}$'s view is identical to that of game $\mathbf{G}_2$ and $\Pr[\mathcal{D} \Rightarrow 1 | \mathcal{D}_0] = \Pr[\mathbf{G}_2]$; if $\mathcal{D}$ receives $\mathcal{D}_1$, $\mathcal{A}$'s view is identical to that of game $\mathbf{G}_3$ and $\Pr[\mathcal{D} \Rightarrow 1 | \mathcal{D}_1] = \Pr[\mathbf{G}_3]$. Thus combined with Lemma 1, $|\Pr[G_2] - \Pr[G_3]| \leq \varepsilon_{\mathsf{ddh}} + 1/p$. □

GAME $\mathbf{G}_4$: This game is identical to game $\mathbf{G}_3$, except that we replace the two real instances of NIZK proofs with simulated ones. The first is the NIZK used to prove correctness of partial decryptions described in Figure 7. Looking ahead, we simulate the NIZK proofs in our decryption queries so that the NIZK proofs do not leak information about the secret keys. This step is preparing for game $\mathbf{G}_5$. The second is the additional elements in the Shoup-Gennaro construction that can be seen as an equality of discrete logarithm NIZK proof with the modification that we also hash the ciphertext. Looking ahead, we switch to simulated proofs in this game to later argue in game $\mathbf{G}_6$ that the proofs do not reveal any information about the discrete log of u and $\bar{u}$.

Lemma 4. *Let q_d be the upper bound on the number of combined encryption and decryption queries, q_{FS} be the number of H_{FS} queries $\mathcal{A}$ makes, and q_{H} be the number of H_1 queries $\mathcal{A}$ makes, then,*

$$|\Pr[G_3] - \Pr[G_4]| \leq \varepsilon_{\text{nizk-fail}} \tag{6}$$

where $\varepsilon_{\text{nizk-fail}} = \varepsilon_{\mathsf{ZK}} \cdot q_d \cdot n + q_{\mathsf{H}}/p^2$.

Let $\varepsilon_{\text{nizk-fail}}$ bound the advantage in distinguishing simulated proofs from real proofs for either the first or second NIZK instance. Since $\mathcal{A}$ makes at most q_d decryption queries, and we need to simulate at most n NIZK proofs per decryption

query, a simple union bound shows that the probability that simulated proofs for the first NIZK instance can be distinguished from real proofs is at most $\varepsilon_{\mathsf{ZK}} \cdot q_d \cdot n$. The distinguishing advantage of the second instance of NIZK is $1/p^2$ by an argument similar to our analysis of the zero-knowledge property of our NIZK scheme in Lemma 15 of the appendix. By a similar argument, the probability that simulated proofs for the second NIZK instance can be distinguished from real proofs is at most q_{H}/p^2. Then, we obtain $\varepsilon_{\text{nizk-fail}} = \varepsilon_{\mathsf{ZK}} \cdot q_d \cdot n + q_{\mathsf{H}}/p^2$.

GAME $\mathbf{G}_5$: This game is identical to game $\mathbf{G}_4$ except we *rig* the secret keys by sampling $y \leftarrow\$ \mathbb{Z}_p$ and $z := 1$. Furthermore, we compute $a := \alpha_h y + \alpha_v$, where $\bar{g} := g^{ak}$. To help us bound $\Pr[G_4] - \Pr[G_5]$, we define event Neq: during any decryption query, $\mathcal{A}$ queries the decryption oracle with a ciphertext, $\mathsf{ctxt} := (c, u, \bar{u}, e, f)$, such that the following two conditions hold:

1. equation 4 returns 1.
2. $u = g^r, \bar{u} = \bar{g}^{r'}$, $r \neq r'$.

To show that events G_4 and G_5 are close, we first show that event Neq occurs with negligible probability, and then show that $\Pr[G_4|\neg\mathsf{Neq}] = \Pr[G_5|\neg\mathsf{Neq}]$.

We first bound $\Pr[\mathsf{Neq}]$ in the following lemma:

Lemma 5. $\Pr[\mathsf{Neq}] \leq \frac{q_d}{p}$ *where* q_d *is the number of decryption and encryption queries.*

Proof. First, we compute the probability that $\mathcal{A}$ queries exactly one decryption oracle and the input ciphertext satisfies the above two conditions, i.e., check 4 returns 1 but $r \neq r'$. We have the following:

$$g^f = wg^{re}, \qquad \bar{g}^f = \bar{w}\bar{g}^{r'e} \tag{7}$$

where $w = g^s$ and $\bar{w} = \bar{g}^{s'}$ for some $s, s' \in \mathbb{Z}_p$. Equation (7) implies that $(s - s') + e(r - r') = 0$. Since $r - r' \neq 0$, there is only one $e := -\frac{s-s'}{r-r'}$. Since e is the output of random oracle H_1, and perfectly binds and is independent of r, r', s, s' ($u, w, \bar{u}, \bar{w}$ are inputs to H_1 therefore e is decided after $\mathcal{A}$ chooses r, r', s, s'), the probability of such an e occurring is $1/p$.

Next, we bound $\Pr[\mathsf{Neq}]$. Event Neq occurs if $\mathcal{A}$ queries at least one accepted invalid ciphertext. By union bound, we have $\Pr[\mathsf{Neq}] \leq \frac{q_d}{p}$. □

Lemma 6. $|\Pr[G_4] - \Pr[G_5]| \leq q_d/p$

Proof. First, we show that $\Pr[G_4|\neg\mathsf{Neq}] = \Pr[G_5|\neg\mathsf{Neq}]$. To prove this, we rely on the H-coefficient technique and show that for all feasible transcripts τ, $\Pr[T = \tau|\mathbf{G}_4, \neg\mathsf{Neq}] = \Pr[T = \tau|\mathbf{G}_5, \neg\mathsf{Neq}]$. In other words, we show that the probability of obtaining any feasible transcript is identical in games $\mathbf{G}_5$ and $\mathbf{G}_6$, conditioned on the event Neq.

The transcript includes all group elements, i.e., $g, h, v, \bar{g}$, the public key pk, corruption set $\mathcal{C}$, corrupted secret keys $\{x_i, y_i, z_i\}_{i\in[\mathcal{C}]}$, all parties' per-party public keys pk_i, all random oracle queries, q_d ciphertexts and their corresponding

partial decryption shares, and $\mathcal{A}$'s challenge plaintexts and their corresponding encryption.

To argue that $\Pr[T = \tau|\mathbf{G}_4, \neg\mathsf{Neq}] = \Pr[T = \tau|\mathbf{G}_5, \neg\mathsf{Neq}]$, we invoke lemma 2. We define two distributions, Ω and Ψ. We will show that there exists a deterministic function f such that $f(\Omega, \Psi) = T$ for both games $\mathbf{G}_4$ and $\mathbf{G}_5$, provided that event Neq does not occur. By lemma 2, if $\Pr[\Omega = \omega|\mathbf{G}_4] = \Pr[\Omega = \omega|\mathbf{G}_5]$, $\Pr[\Psi = \psi|\mathbf{G}_4] = \Pr[\Psi = \psi|\mathbf{G}_5]$, and Ω is independent from Ψ, then $\Pr[T = \tau|\mathbf{G}_4, \neg\mathsf{Neq}] = \Pr[T = \tau|\mathbf{G}_5, \neg\mathsf{Neq}]$. We condition on $\neg\mathsf{Neq}$ because there does not exist a deterministic function f such that $f(\Omega, \Psi) = T$ in game $\mathbf{G}_5$.

We define

$$\Omega := \{\alpha_h, \alpha_v, a, k, \mathsf{pk}, \{x_i, y_i, z_i\}_{i\in\mathcal{C}}\}$$
$$\Psi := \{\mathsf{H}_1', \mathsf{H}_{\mathsf{FS}}', \{b_j\}_{j\in q_{\mathsf{H}}}, \{r_j, e_j, f_j\}_{j\in q_d}, r, e, f\}$$

Here H_1' denotes the additional outputs of H_1 not used in the computation of encryptions and $\mathsf{H}_{\mathsf{FS}}'$ denotes the set of all outputs of H_{FS} queries. $\{r_j, e_j, f_j\}_{j\in q_d}$ denote the set of randomness required for up to q_d number of encryption queries. r, e, f are the randomness required to encrypt $\mathcal{A}$'s challenge m_μ. Recall that we simulate the NIZK of discrete logarithm equality in the encryption protocol by sampling e, f randomly. In both games, we have $h := g^{\alpha_h}, v := g^{\alpha_v}, \bar{g} := g^{ak}$. In game $\mathbf{G}_4$, $\mathsf{pk} := g^x$ while in game $\mathbf{G}_5$, $\mathsf{pk} := g^{x+\alpha_h y+\alpha_v}$. Recall in both games $\mathbf{G}_5$ and $\mathbf{G}_6$, $\alpha_h, \alpha_v, k, \mathsf{H}_1', \mathsf{H}_{\mathsf{FS}}', \{b_j\}_{j\in q_{\mathsf{H}}}, r, s$ are sampled randomly and independently. Clearly, Ω and Ψ are independently distributed.

Next, we show that there is a deterministic function f for both games $\mathbf{G}_4$ and $\mathbf{G}_5$ for any fixed $\mathcal{A}$ queries such that $f(\Omega, \Psi) := T$.

- challenge response. Clearly, in both games, the challenge ciphertext is the output of a deterministic function of $\mathcal{A}$'s query and the values $r, s, \mathsf{H}_1, \mathsf{pk}, a, k$.
- random oracle queries. In both games, sets H_1' and $\mathsf{H}_{\mathsf{FS}}'$ determines all outputs of random oracles H_1, H_{FS} respectively. $\{b_j\}_{j\in q_{\mathsf{H}}}$ contain all responses to H_2 queries. Responses to H_3 are computed from $\{b_j\}_{j\in q_{\mathsf{H}}}$, a, α_v, α_h, and k.
- decryption query. Let $\mathsf{ctxt} := (c, u, \bar{u}, e, f)$ be the jth decryption query. Clearly, for both games, for $i \in \mathcal{C}$, $\hat{m}_i$ is computed deterministically from $x_i, y_i, z_i, a, b, \alpha_v, \alpha_h$.
  Next, we show that the combined $\hat{m}$ is identical in both games. In game $\mathbf{G}_4$, $\hat{m} = u^x = \mathsf{pk}^r$.
  In game $\mathbf{G}_5$, if event Neq does not occur, i.e., $u := g^r$ and $\bar{u} := \bar{g}^r$, we have $\bar{u}^{1/k} := g^{ar}$. Then we have:

$$\begin{aligned}\hat{m} &:= u^x g^{by} (g^{-ab} g^{b\alpha_v})^{1/\alpha_h} \bar{u}^{1/k} \\ &:= u^x g^{by} g^{-by} g^{ar} \quad \text{by } y := \frac{a - \alpha_v}{\alpha_h} \\ &:= \mathsf{pk}^r\end{aligned}$$

- encryption query. Let m_j be the jth encryption query. In both games, the encryption of m_j is computed according to $r_j, e_j, f_j, \mathsf{pk}$ where $\mathsf{Enc}(m_j) := (c, u, \bar{u}, e, f) := (\mathsf{pk}^{r_j} \cdot m_j, g^{r_j}, \bar{g}^{r_j}, e_j, f_j)$. The decryption shares are computed identically to the decryption query on $\mathsf{Enc}(m_j)$

Clearly $\Pr[\Psi = \psi|\mathbf{G}_4] = \Pr[\Psi = \psi|\mathbf{G}_5]$ for all feasible ψ since Ψ is randomly sampled from the same distribution in both games.

Lastly, we show that $\Pr[\Omega = \omega|\mathbf{G}_4] = \Pr[\Omega = \omega|\mathbf{G}_5]$ for all feasible ω. Without loss of generality, let $\omega := \{\alpha_h^*, \alpha_v^*, a^*, k^*, \mathsf{pk}^*, \{x_i^*, y_i^*, z_i^*\}_{i\in\mathcal{C}}\}$. In game $\mathbf{G}_4$, there is one such x such that $\mathsf{pk}^* = g^x$. Since t evaluation points and the constant term uniquely defines a degree t polynomial and polynomials x, y, z are randomly and independently chosen, $\Pr[\{x_i, y_i, z_i\}_{i\in\mathcal{C}} = \{x_i^*, y_i^*, z_i^*\}_{i\in\mathcal{C}}|x_0 = x, y_0 = 0, z_0 = 0] = 1/p^{3t}$. Then $\Pr[\Omega = \omega|\mathbf{G}_4] := \frac{1}{p-1} \cdot \frac{1}{p-1} \cdot \frac{1}{p} \cdot \frac{1}{p} \cdot \frac{1}{p} \cdot \frac{1}{p^{3t}}$.

In game $\mathbf{G}_5$, given $a^*, \alpha_h^*, \alpha_v^*$, there is one such y such that $a^* = \alpha_h^* y + \alpha_v^*$. Then given pk^*, y, there is one such x such that $\mathsf{pk}^* := g^{x'+\alpha_h^* y+\alpha_v^*}$. By the same argument as in game $\mathbf{G}_4$, $\Pr[\{x_i, y_i, z_i\}_{i\in\mathcal{C}} = \{x_i^*, y_i^*, z_i^*\}_{i\in\mathcal{C}}|x_0 = x, y_0 = y, z_0 = 1] = 1/p^{3t}$. Then $\Pr[\Omega = \omega|\mathbf{G}_5] := \frac{1}{p-1} \cdot \frac{1}{p-1} \cdot \frac{1}{p} \cdot \frac{1}{p} \cdot \frac{1}{p} \cdot \frac{1}{p^{3t}}$. Therefore $\Pr[\Omega = \omega|\mathbf{G}_4] = \Pr[\Omega = \omega|\mathbf{G}_5]$.

Then we have $|\Pr[G_4] - \Pr[G_5]| = |\Pr[G_4|\mathsf{Neq}] - \Pr[G_5|\mathsf{Neq}]| \cdot \Pr[\mathsf{Neq}] \leq \Pr[\mathsf{Neq}]$. Then from lemma 5, $|\Pr[G_4] - \Pr[G_5]| \leq q_d/p$. □

GAME $\mathbf{G}_6$: In this game, when $\mathcal{A}$ challenges m_0, m_1, instead of encrypting $\mathcal{A}$'s challenge, we respond with an encryption that does not correlate with it. This helps bound $|\Pr[G_0] - 1/2|$. This game is identical to game $\mathbf{G}_5$, except that we sample random group elements $\{g^\alpha, g^\beta, g^\zeta\} \leftarrow\$\ \mathbb{G}^3$. Set $v = g^\alpha$. Compute $g^a := g^{\alpha_h y} g^\alpha$, $\bar{g} := (g^a)^k$. To answer encryption query on m_μ, compute $u := g^\beta$, $\bar{u} := (g^\beta)^{\alpha_h y k} (g^\zeta)^k$, $c := (g^\beta)^x (g^\beta)^{\alpha_h y} g^\zeta \cdot m_\mu$. For the jth decryption query, program $\mathsf{H}_3(\mathsf{ctxt}) := ((g^a)^{-b}(g^\alpha)^b)^{1/\alpha_h} \bar{u}^{1/k}$.

Lemma 7. *Let $\varepsilon_{\mathsf{ddh}}$ be the advantage of breaking* DDH *in $\mathbb{G}$ as defined in as defined in Assumption 1, then $|\Pr[G_5] - \Pr[G_6]| \leq \varepsilon_{\mathsf{ddh}} + q_d/p$.*

Proof. $\mathcal{A}$ distinguishes game $\mathbf{G}_5$ from game $\mathbf{G}_6$ either by making a decryption query with a ciphertext that satisfies the two conditions laid out in game $\mathbf{G}_5$, i.e., inducing event Neq or, by breaking the DDH assumption in $\mathbb{G}$, since $\mathcal{A}$ cannot query the decryption query on ctxt_μ. We have

$$\begin{aligned}|\Pr[G_5] - \Pr[G_6]| &= |\Pr[G_5|\neg\mathsf{Neq}] - \Pr[G_6|\neg\mathsf{Neq}]| \cdot \Pr[\neg\mathsf{Neq}] \\ &\quad + |\Pr[G_5|\mathsf{Neq}] - \Pr[G_6|\mathsf{Neq}]| \cdot \Pr[\mathsf{Neq}] \\ &\leq |\Pr[G_5|\neg\mathsf{Neq}] - \Pr[G_6|\neg\mathsf{Neq}]| + \Pr[\mathsf{Neq}]\end{aligned}$$

Next, we show that $|\Pr[G_5|\neg\mathsf{Neq}] - \Pr[G_6|\neg\mathsf{Neq}]| \leq \varepsilon_{\mathsf{ddh}}$ We will prove this by building an adversary against the DDH assumption, $\mathcal{A}_{\mathsf{ddh}}$, from $\mathcal{A}$. $\mathcal{A}_{\mathsf{ddh}}$ receives a tuple $(g^\alpha, g^\beta, g^\zeta)$ and interacts with $\mathcal{A}$ as described. If $\mathcal{A}_{\mathsf{ddh}}$'s input is a DDH tuple, then $\mathcal{A}$ plays against game $\mathbf{G}_5$. If $\mathcal{A}_{\mathsf{ddh}}$'s input is a random tuple, then $\mathcal{A}$ plays against game $\mathbf{G}_6$. We have the following:

$$\Pr[\mathcal{A}_{\mathsf{ddh}}(g^\alpha, g^\beta, g^\zeta) = 1|\zeta = \alpha\beta] = \Pr[G_5] \tag{8}$$

$$\Pr[\mathcal{A}_{\mathsf{ddh}}(g^\alpha, g^\beta, g^\zeta) = 1|\zeta \leftarrow\$\ \mathbb{Z}_p] = \Pr[G_6] \tag{9}$$

Hence, combined with Lemma 5, $\mathcal{A}_{\mathsf{ddh}}$'s advantage is $|\Pr[G_5] - \Pr[G_6]| \leq \varepsilon_{\mathsf{ddh}} + q_d/p$. □

In the last part of our analysis, we will prove that $\mathcal{A}$ wins in game $\mathbf{G}_6$ with probability $1/2$.

Lemma 8. $\Pr[G_6] = 1/2$.

Proof. The only part of $\mathcal{A}$'s view that depends on the hidden bit μ is c. In game $\mathbf{G}_6$, $\mathcal{A}$ receives $c = (g^\beta)^x (g^\beta)^{\alpha_h y} g^\zeta \cdot m_\mu$ from the game. Since $\beta, \zeta \leftarrow\$ \mathbb{Z}_p$, c is uniformly randomly distributed in $\mathbb{G}$ for both m_0 and m_1, and is independent of μ. Thus:

$$\Pr[\mathsf{ctxt}|\mu = 0] = \Pr[\mathsf{ctxt}|\mu = 1] = \frac{1}{p^2}$$

Thus, we have

$$\Pr[\mathcal{A} = 1|\mu = 0] = \Pr[\mathcal{A} = 1|\mu = 1] = \Pr[G_6] = 1/2. \qquad \square$$

We now prove the following main theorem.

Theorem 2 (Adaptively IND-TCCA Secure Threshold Decryption). *Let λ be the security parameter, and let $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$ be a group of prime order p. For any $n \in \mathbb{N}$ and $t < n$, assuming the hardness of DDH in $\mathbb{G}$ and the random oracle model, any PPT adversary making at most q_d decryption queries, q_H random oracle queries to H, and q_{FS} random oracle queries to H_{FS} has guessing advantage in the $\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ game (Figure 1) for our scheme in Figure 3 of at most $\varepsilon_{\text{IND-TCCA}}$, where*

$$\varepsilon_{\text{IND-TCCA}} \leq 2\varepsilon_{\mathsf{ddh}} + \frac{q_{\mathsf{FS}} \cdot q_d \cdot n}{p^2} + \frac{q_\mathsf{H}}{p^2} + \frac{1}{p} + 2\frac{q_d}{p} \tag{10}$$

where $\varepsilon_{\mathsf{ddh}}$ is the advantage an adversary in breaking DDH in $\mathbb{G}$.

Proof. From the above series of games and lemma 14, we get

$$|\Pr[G_0] - \frac{1}{2}| \leq 2\varepsilon_{\mathsf{ddh}} + \frac{q_{\mathsf{FS}} \cdot q_d \cdot n}{p^2} + \frac{q_\mathsf{H}}{p^2} + \frac{1}{p} + 2\frac{q_d}{p} \leq \mathsf{negl}(\lambda) \tag{11}$$

which is $\mathcal{A}$'s guessing advantage against our threshold encryption scheme. $\square$

5 Implementation and Evaluation

5.1 Evaluation Setup

We implemented the standard threshold ElGamal, Shoup-Gennaro, and the two adaptive threshold ElGamal schemes proposed in this work. Our implementation is in Python and is publicly available at `https://github.com/Zilingy2/adaptive_elgamal/`. We use the NIST P-256 elliptic curve and the *pycryptodome* [18] library. All random oracles are instantiated using the SHA-256 hash function.

All benchmarks were conducted for a threshold parameter of $t = 64$. Running times were averaged over 5 runs and reported in milliseconds. All experiments were conducted with a single thread on a machine equipped with an Apple M4 chip with 16GB RAM.

Scheme	Partial Dec. (ms)	Comb. (ms)	Σ-Prove (ms)	Σ-Ver (ms)	Partial Dec. Size (bytes)	Σ proof Size (bytes)
Basic ElGamal	0.8	101.4	0.5	1.1	224	160
Adaptive CPA	1.8	140.6	1.1	1.8	256	192
Shoup–Gennaro	1.9	106.5	0.5	1.2	224	160
Adaptive CCA	3.9	181.6	1.7	2.4	288	224

Table 1: Performance comparison for threshold parameter $t = 64$. All schemes are instantiated over the NIST P-256 elliptic curve.

5.2 Evaluation Results

The standard threshold Elgamal serves as a baseline for our adaptive CPA scheme while the Shoup-Gennaro scheme serves as a baseline for our adaptive CCA scheme. We report our results in Table 1.

Partial decryption time. The partial decryption times of our adaptive CPA and CCA schemes are 2.3× and 2.0× of their corresponding baselines, since our Σ-protocol's prover computation is more involved.

Combine time. We measured the optimal case combination time where all decryption shares verify correctly. The combination times of our adaptive CPA and CCA schemes are 1.4× and 1.7× of their corresponding baselines. This is due to a more involved Σ-protocol verifier computation.

Partial decryption size. The partial decryption size depends on the underlying elliptic curve group we use. For the NIST P-256 elliptic curve, our adaptive CPA scheme partial decryption is 32 bytes longer than that of the baseline due to the extra field element in the Σ-proof. The partial decryption of our adaptive CCA scheme is 64 bytes longer than that of the baseline due to the two extra field elements in the Σ-proof.

6 Conclusion

In this paper, we presented two new adaptively secure non-interactive threshold decryption schemes. The first one is based on the standard ElGamal decryption scheme and achieves CPA security. The second one is based on the Shoup-Genarro threshold decryption scheme and achieves CCA security. Our scheme makes minimal changes to their static counterparts, maintains the scheme's simplicity, and is compatible with its encryption. Both schemes are proven adaptively secure assuming the hardness of DDH and the random oracle model.

Acknowledgments. This work is funded in part by the National Science Foundation award #2240976.

References

1. Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Cryptographers' Track at the RSA Conference. pp. 226–243. Springer (2006)
2. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.6 (2023)
3. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In: Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 98–116. Springer (1999)
4. Canetti, R., Goldwasser, S.: An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 90–106. Springer (1999)
5. Chen, S., Steinberger, J.: Tight security bounds for key-alternating ciphers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 327–350. Springer (2014)
6. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) Advances in Cryptology — EUROCRYPT '97. pp. 103–118. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
7. Das, S., Ren, L.: Adaptively secure bls threshold signatures from ddh and co-cdh. In: Annual International Cryptology Conference. pp. 251–284. Springer (2024)
8. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Advances in Cryptology—CRYPTO'87: Proceedings 7. pp. 120–127. Springer (1988)
9. Desmedt, Y.: Threshold cryptosystems. In: Advances in Cryptology—AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques Gold Coast, Queensland, Australia, December 13–16, 1992 Proceedings 3. pp. 1–14. Springer (1993)
10. Devevey, J., Libert, B., Nguyen, K., Peters, T., Yung, M.: Non-interactive cca2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. In: Garay, J.A. (ed.) Public-Key Cryptography – PKC 2021. pp. 659–690. Springer International Publishing, Cham (2021)
11. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Theory of Cryptography Conference. pp. 188–209. Springer (2005)
12. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory **31**(4), 469–472 (1985)
13. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)
14. Frankel, Y., MacKenzie, P., Yung, M.: Adaptively-secure distributed public-key systems. In: European Symposium on Algorithms. pp. 4–27. Springer (1999)
15. Frankel, Y., MacKenzie, P., Yung, M.: Adaptively-secure optimal-resilience proactive rsa. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 180–194. Springer (1999)
16. Jarecki, S., Lysyanskaya, A.: Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In: Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19. pp. 221–242. Springer (2000)
17. Katz, J., Lindell, Y.: Introduction to modern cryptography: principles and protocols. Chapman and hall/CRC (2007)

18. Legrand, S.: Pycryptodome. `https://github.com/Legrandin/pycryptodome` (2014)
19. Libert, B., Yung, M.: Non-interactive cca-secure threshold cryptosystems with adaptive security: New framework and constructions. In: Cramer, R. (ed.) Theory of Cryptography. pp. 75–93. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
20. Libert, B., Yung, M.: Adaptively secure non-interactive threshold cryptosystems. Theoretical Computer Science **478**, 76–100 (2013)
21. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. Cryptology ePrint Archive, Paper 2001/107 (2001), `https://eprint.iacr.org/2001/107`
22. Mac Kenzie, P.: An efficient two-party public key cryptosystem secure against adaptive chosen ciphertext attack. In: Desmedt, Y.G. (ed.) Public Key Cryptography — PKC 2003. pp. 47–61. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
23. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. Journal of the ACM (JACM) **51**(2), 231–262 (2004)
24. Patarin, J.: The "coefficients h" technique. In: International Workshop on Selected Areas in Cryptography. pp. 328–345. Springer (2008)
25. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference. pp. 129–140. Springer (1991)
26. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
27. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–16. Springer (1998)
28. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. J. Cryptol. **15**(2), 75–96 (Jan 2002)
29. Wee, H.: Threshold and revocation cryptosystems via extractable hash proofs. In: Paterson, K.G. (ed.) Advances in Cryptology – EUROCRYPT 2011. pp. 589–609. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

A Additional Definitions

In this section, we present definitions for IND-TCPA, correctness, and robustness of a threshold decryption scheme that could not be included in the main body of the paper due to space constraints.

First, we provide the definition for the IND-TCPA game. Our indistinguishability under threshold chosen-**plaintext** attack (IND-TCPA) in the presence of an adaptive adversary game is identical to the $\mathsf{IND\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ game defined in Figure 1, except the adversary does not have access to a decryption oracle Dec.

We have the following definition.

Definition 4 (Indistinguishability under chosen-plaintext attack). *Consider the* $\mathsf{IND\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ *game provided above. We say that a threshold decryption scheme* TD *is IND-TCPA secure if, for all security parameters* λ, $n, t \in \mathbb{N}$ *with*

Game $\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}(1^\lambda, n, t)$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]} \leftarrow \mathsf{KGen}(\mathsf{pp})$
Let $\mathcal{C} := \varnothing, \mathcal{H} := [n]$
$\mathsf{inp} := \mathsf{pp}, \mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}$
$\mathcal{S}', \mathsf{ctxt}, \{\hat{m}_i, \pi_i\}_{i\in\mathcal{S}} \leftarrow \mathcal{A}^{\textsc{Corr},\textsc{Enc},\textsc{Dec}}(\mathsf{inp})$
$T' := \{i \in \mathcal{S}' | \mathsf{PVer}(\mathsf{pk}_i, \mathsf{ctxt}, \hat{m}_i, \pi_i) = 1\}$
if $|T'| < t+1$ **then**
  return 0
$\mathcal{S} := [1, \ldots, t+1]$
$\mathcal{D} := \{\mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt})\}_{i\in\mathcal{S}}$
$m \leftarrow \mathsf{Comb}(\{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{ctxt}, \mathcal{S}, \mathcal{D})$
$m' \leftarrow \mathsf{Comb}(\{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{ctxt}, T', \{\hat{m}_i, \pi_i\}_{i\in T'})$
if $m' \neq m$ **then**
  return 1
else
  return 0

Oracle $\textsc{Corr}(i)$:

if $|\mathcal{C}| \geq t$ **then**
  return $\perp$
$\mathcal{C} := \mathcal{C} \cup \{i\}$
return sk_i

Oracle $\textsc{Enc}(m)$:

$\mathsf{ctxt} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$
$\hat{m}_i \leftarrow \mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt})$ for all $i \in [n]$
return $(\mathsf{ctxt}, \{\hat{m}_i, \pi_i\}_{i\in[n]})$

Oracle $\textsc{Dec}(\mathsf{ctxt}, i)$:

assert $\mathsf{ctxt} \neq \mathsf{ctxt}_\mu$ // $\mathcal{A}$ cannot invoke $\textsc{Dec}$ on ctxt_μ
$\hat{m}_i \leftarrow \mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt})$
return $\hat{m}_i$

Fig. 2: The robustness security game $\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ for a non-interactive threshold decryption $\mathsf{TD} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{PDec}, \mathsf{Comb})$ with an adaptive adversary $\mathcal{A}$.

$t < n$, PPT adversaries $\mathcal{A}$, the following guessing advantage is negligible, i.e.,

$$\varepsilon_{\text{IND-TCPA}} := \left| \Pr[\mathsf{IND\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}(1^\lambda, n, t) \Rightarrow 1] - \frac{1}{2} \right| = \mathsf{negl}(\lambda) \tag{12}$$

Intuitively, the correctness property guarantees that when all parties behave honestly, the decryption protocol correctly outputs the original message. More formally,

Definition 5 (Correctness). *Let $\mathcal{M}$ be the message space. For all $m \in \mathcal{M}$, for all obtainable keys, $\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\} \leftarrow \mathsf{KGen}(\mathsf{pp})$, for all obtainable ciphertext, $\mathsf{ctxt} \leftarrow \mathsf{Enc}(m, \mathsf{pk})$, and any set $S \subseteq [n], |S| \geq t+1$, we have the following:*

$$\Pr[\mathsf{Comb}(\{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{ctxt}, S, \{\mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt})\}_{i\in S}) \neq m] < \mathsf{negl}(\lambda) \tag{13}$$

Additionally, we require a threshold decryption scheme to satisfy the robustness property [1], either $\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ or $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$, against up to t actively malicious parties. We formally define the robustness property for our IND-CCA secure (resp. IND-CPA) protocol using the $\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ (resp. $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$) game. The $\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ game is defined in Figure 5. The $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ game is identical to the $\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ game except that the adversary does not have access to the decryption oracle $\textsc{Dec}$.

Intuitively, the $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ and $\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ games state that an adversary that can corrupt up to t parties adaptively cannot, with non-negligible probability, produce a ciphertext and a set of decryption shares of that ciphertext such

that all decryption shares are valid but do not combine to the actual plaintext that corresponds to the ciphertext.

More formally, we have the following two definitions:

Definition 6 (TCCA Robustness). *Consider the* $\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}$ *game in Figure 5. We say that a threshold decryption scheme* TD *is robust if, for all security parameters* λ, $n, t \in \mathbb{N}$ *with* $t < n$, *PPT adversaries* $\mathcal{A}$, *the following advantage is negligible, i.e.,*

$$\mathsf{Adv}^{\mathsf{RB\text{-}TCCA}}_{\mathcal{A},\mathsf{TD}}(\lambda) := \Pr[\mathsf{RB\text{-}TCCA}^{\mathcal{A}}_{\mathsf{TD}}(1^\lambda, n, t) \Rightarrow 1] = \mathsf{negl}(\lambda) \tag{14}$$

Definition 7 (TCPA Robustness). *Consider the* $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ *game provided above. We say that a threshold decryption scheme* TD *is robust if, for all security parameters* λ, $n, t \in \mathbb{N}$ *with* $t < n$, *PPT adversaries* $\mathcal{A}$, *the following advantage is negligible, i.e.,*

$$\mathsf{Adv}^{\mathsf{RB\text{-}TCPA}}_{\mathcal{A},\mathsf{TD}}(\lambda) := \Pr[\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}(1^\lambda, n, t) \Rightarrow 1] = \mathsf{negl}(\lambda) \tag{15}$$

B Design of the IND-TCPA Threshold Decryption Scheme

We now describe our non-interactive IND-TCPA secure threshold decryption scheme based on the ElGamal encryption scheme in Figure 3 in the appendix. Let $\mathcal{M}$ be the message space.

$\underline{\mathsf{Setup}(1^\lambda, n, t)}$: Let $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$. Let $h \in \mathbb{G}$ be an additional uniformly random independent generator of $\mathbb{G}$. Let $\mathsf{H} : \mathcal{M} \to \mathbb{G}$ and $\mathsf{H}_{\mathsf{FS}} : \{0,1\}^* \to \mathbb{G}$ be two distinct hash functions modeled as random oracles. The public parameters of our scheme are then $(\mathbb{G}, g, h, \mathsf{H}, \mathsf{H}_{\mathsf{FS}})$. We assume that all the algorithms below implicitly take the public parameters as input.

$\underline{\mathsf{KGen}(\mathsf{pp})}$: The KGen algorithm samples two uniformly random polynomials $x(\cdot), y(\cdot)$ of degree t with the constraint that $y(0) := 0$. The decryption key of party i is then $\mathsf{sk}_i := (x(i), y(i))$. Let $\mathsf{pk} := g^{x(0)}$ be the public key, and $\mathsf{pk}_i := g^{x(i)} h^{y(i)}$ for all $i \in [n]$ be the per-party public key.

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$: Our encryption algorithm is identical to the standard ElGamal encryption scheme. First, generate a random element $r \leftarrow\!\!\$\ \mathbb{Z}_p$, and then the encryption of message m is the tuple $\mathsf{ctxt} := (c, u)$ where $c := m \cdot \mathsf{pk}^r$ and $u := g^r$.

$\underline{\mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt} := (c, u))}$: On input the partial decryption key sk_i and the ciphertext tuple $\mathsf{ctxt} := (c, u)$, compute partial decryption $\hat{m}_i := u^{x(i)} \cdot \mathsf{H}(\mathsf{ctxt})^{y(i)}$ and a NIZK proof π_i that $\hat{m}_i$ is correctly computed from the ciphertext. π_i is computed using the NIZK protocol described in Figure 4. The multiplication by $\mathsf{H}(\mathsf{ctxt})^{y(i)}$ is the main difference compared to the static threshold ElGamal encryption.

$\underline{\mathsf{PVer}(\mathsf{pk}_i, \mathsf{ctxt} := (c, u), \hat{m}_i, \pi_i)}$: On input the per-party public key pk_i the ciphertext ctxt, and partial decryption share tuple $(\hat{m}_i, \pi_i)$, validate the partial

$\underline{\mathsf{Setup}(1^\lambda, n, t)}$:
1: $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$.
2: Let $h \in \mathbb{G}$ be a uniformly random generator of $\mathbb{G}$.
3: Let $\mathsf{H} : \mathcal{M} \to \mathbb{G}$ and $\mathsf{H_{FS}} : \{0,1\}^* \to \mathbb{Z}_p$ be two hash functions modeled as random oracle.
4: **return** $\mathsf{pp} := (\mathbb{G}, p, n, t, g, h, \mathsf{H}, \mathsf{H_{FS}})$.

// We assume all algorithms implicitly take pp as input. We use $\mathsf{H_{FS}}$ in SigmaProve and SigmaVer.

$\underline{\mathsf{KGen}(\mathsf{pp})}$:
5: Let $x(\cdot), y(\cdot) \leftarrow\$\ \mathbb{Z}_p[x]$ be two polynomials of degree t with $y(0) = 0$.
6: $\mathsf{pk} := g^{x(0)} h^{y(0)} = g^{x(0)}$
7: **for** each $i \in [n]$ **do**
8: $\quad \mathsf{sk}_i := (x(i), y(i))$
9: $\quad \mathsf{pk}_i := g^{x(i)} h^{y(i)}$
10: **return** $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{sk}_j)$ to party j for all $j \in [n]$

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$:
11: $r \leftarrow\$\ \mathbb{Z}_p$
12: $\mathsf{ctxt} := (c, u) = (m \cdot \mathsf{pk}^r, g^r)$
13: **return** ctxt

$\underline{\mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt} = (c, u))}$:
14: $\hat{m}_i := u^{x(i)} \cdot \mathsf{H}(\mathsf{ctxt})^{y(i)}$
15: $\pi_i := \mathsf{SigmaProve}(pk_i, u, \hat{m}_i, \mathsf{sk}_i)$
16: **return** $(\hat{m}_i, \pi_i)$

$\underline{\mathsf{PVer}(\mathsf{pk}_i, \mathsf{ctxt} = (c, u), \hat{m}_i, \pi_i)}$:
17: **return** $\mathsf{SigmaVer}(\mathsf{pk}_i, u, \hat{m}_i, \pi_i)$

$\underline{\mathsf{Comb}(\{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{ctxt}, S, \{(\hat{m}_i, \pi_i)\}_{i\in S})}$:
18: **parse** $(u, c) := \mathsf{ctxt}$
19: $T := \varnothing$
20: **for** each $i \in S$ **do**
21: $\quad$ **if** $\mathsf{PVer}(u, \mathsf{pk}_i, \hat{m}_i, \pi_i)$ **then**
22: $\quad\quad T \cup \{i\}$
23: **assert** $|T| \geq t + 1$
24: Let $L_{i,T}$ be the i-th Lagrange coefficients for T
25: $\hat{m} := \prod_{i\in T} \hat{m}_i^{L_{i,T}}$
26: **return** $m := c \cdot \hat{m}^{-1}$

Fig. 3: Adaptively IND-CPA secure threshold ElGamal decryption scheme with trusted key generation.

decryption share by running the NIZK protocol SigmaVer protocol. Output 1 if and only if SigmaVer outputs 1.

$\underline{\mathsf{Comb}(\{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{ctxt} := (c, u), S, \{(\hat{m}_i, \pi_i)\}_{i\in S})}$: The input is a set of per-party public keys of all parties, the ciphertext $\mathsf{ctxt} := (c, u)$, a set of parties S, a set of partial decryption tuples that correspond to the parties in S, $(\hat{m}_i, \pi_i)$. The aggregator first validates each decryption tuple using PVer. Let T be the set of indices of parties with valid partial decryptions. It then computes the plaintext m as:

$$m := c \cdot \hat{m}^{-1}, \quad \text{where} \quad \hat{m} := \prod_{i\in T} \hat{m}_i^{L_{i,T}} \tag{16}$$

where $L_{i,T}$ is the i-th Lagrange coefficient with respect to the set T.

C Analysis of IND-TCPA

We first define a helper lemma, that we adapt from Pedersen's VSS [25], to aid our IND-TCPA analysis.

Input: $g, h, \mathsf{pk}, H, u, \hat{m}$
Witness: $(x, y) \in \mathbb{Z}_p^2$
The prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that it knows $x, y \in \mathbb{Z}_p$ such that $\mathsf{pk} = g^x h^y$ and $\hat{m} = u^x H^y$.

// We assume that both algorithms implicitly take of $g, h, H, \hat{m}$ as input

$\underline{\mathsf{SigmaProve}(\mathsf{pk}, u, \hat{m}, x, y)}$:
1: Sample $\hat{x}, \hat{y} \leftarrow\!\!\$\ \mathbb{Z}_p$.
2: $\gamma := g^{\hat{x}} h^{\hat{y}}, \quad \psi := u^{\hat{x}} H^{\hat{y}}$
3: $e := \mathsf{H}_{\mathsf{FS}}(\gamma, \psi, \mathsf{pk}, g, h, u, \hat{m})$
4: $f_x := \hat{x} + x \cdot e, \quad f_y := \hat{y} + y \cdot e.$
5: **return** $\pi := (\gamma, \psi, f_x, f_y)$.

$\underline{\mathsf{SigmaVer}(\mathsf{pk}, u, \hat{m}, \pi = (\gamma, \psi, f_x, f_y))}$:
6: $e := \mathsf{H}_{\mathsf{FS}}(\gamma, \psi, \mathsf{pk}, g, h, u, \hat{m})$
7: **if** $g^{f_x} h^{f_y} = x \cdot \mathsf{pk}^e$ and $u^{f_x} H^{f_y} = \psi \cdot \hat{m}^e$ **then**
8: **return** 1
9: **return** 0

Fig. 4: NIZK-protocol for computing and verifying the correctness proof for partial decrytptions.

Lemma 9 (Adaptive Secrecy of Pedersen Secret Sharing). *Let* $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$ *and let* $h \in \mathbb{G}$ *be another uniformly random independent generator of* $\mathbb{G}$. *Let* $y \in \mathbb{Z}_p$ *be the secret to be shared and* $y(\cdot)$ *be a random polynomial of degree* t *such that* $y(0) = y$. *Let* $x \leftarrow\!\!\$\ \mathbb{Z}_p$ *and* $x(\cdot)$ *be a random polynomial of degree* t *such that* $x(0) = x$. *Let*

$$view_{\mathcal{C}} = \{g, h, g^x h^y, \{g^{x(i)} h^{y(i)}\}_{i \in [n]}, \{x(i), y(i)\}_{i \in \mathcal{C}}\} \tag{17}$$

For all subset $\mathcal{C}$ *of* $\{1, \ldots, n\}$ *of size at most* t *and all* $y^* \in \mathbb{Z}_p$*:*

$$\Pr[y = y^* | view_{\mathcal{C}}] = \Pr[y = y^*] \tag{18}$$

The view defined in lemma 9 is the view of an adversary in the Pedersen VSS scheme. One crucial step in our IND-TCPA proof is showing that an adversary cannot distinguish between an honest pk and a "rigged" pk with a non-zero y. This relies on the information-theoretic hiding property of Pedersen VSS. In other words, if an adversary cannot distinguish between an execution of Pedersen secret sharing for 0 and one for a non-zero value, then $\mathcal{A}$ cannot detect the rigged public key. In our IND-TCPA analysis in section 2.3, we rely on this lemma to show that the simulated game with a "rigged" pk is indistinguishable from the real execution of our protocol.

Proof. See Theorem 4.4 in [25]. There exists a bijection between our $\text{view}_{\mathcal{C}}$ defined in eq. (17) and $\text{view}_{\mathcal{C}}$ defined in [25].

We will prove IND-TCPA assuming the decisional Diffie-Hellman (DDH) assumption in $\mathbb{G}$ via a sequence of games. Game $\mathbf{G}_0$ is the $\mathsf{IND\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ game in

which $\mathcal{A}$ makes queries to the encryption oracle and tries to guess the challenge bit μ. In game $\mathbf{G}_5$, $\mathcal{A}$ receives a ciphertext ctxt that is independent of the challenge bit μ. Therefore, the probability that $\mathcal{A}$ wins game $\mathbf{G}_5$ is exactly $1/2$. We use a sequence of games to bound the difference between $\mathcal{A}$'s advantage in game $\mathbf{G}_0$ and game $\mathbf{G}_5$.

For any adversary $\mathcal{A}$ for the k-th game $\mathbf{G}_k$, we will use G_k to denote the event that the game outputs 1, i.e., $\mathcal{A}$ wins the game $\mathbf{G}_k$.

GAME $\mathbf{G}_0$: This is the security game $\mathsf{IND\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ for our threshold decryption scheme, where the game follows the honest protocol. The game provides $\mathcal{A}$ access to random oracles, H and H_{FS}, using the standard lazy simulation technique, i.e., the game uniformly and independently samples outputs of the random oracle on demand when $\mathcal{A}$ first queries the random oracle on a specific input.

We assume that $\mathcal{A}$ always corrupts exactly t parties. This assumption does not change $\mathcal{A}$'s winning advantage. This is without generality: we can build a wrapper adversary that invokes $\mathcal{A}$ and then additionally corrupt parties such that exactly t parties are corrupted.

GAME $\mathbf{G}_1$: This game is identical to $\mathbf{G}_0$, except that we sample $\alpha_h \leftarrow\$ \mathbb{Z}_p^*$ and set $h := g^{\alpha_h}$. Clearly, $\Pr[G_0] = \Pr[G_1]$.

GAME $\mathbf{G}_2$: This game is identical to $\mathbf{G}_1$, except for each encryption query we program the random oracle so that for the j-th encryption query on message m_j, we set $\mathsf{H}(\mathsf{ctxt}_j) := h^{r_j}$, where $\mathsf{ctxt}_j := (m_j \cdot g^{r_j}, g^{r_j})$ for $r_j \leftarrow\$ \mathbb{Z}_p$.

The indistinguishability between $\mathcal{A}$'s view in game $\mathbf{G}_1$ and $\mathbf{G}_2$ is a crucial step of our proof. We prove this indistinguishability assuming the hardness of DDH in $\mathbb{G}$. More formally,

Lemma 10. *Let $\varepsilon_{\mathsf{ddh}}$ be the advantage of breaking DDH in $\mathbb{G}$ as defined in Assumption 1, then $|\Pr[G_1] - \Pr[G_2]| \leq \varepsilon_{\mathsf{ddh}} + 1/p$.*

Proof. Recall the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ defined in lemma 1. The only difference between the two games is that $(h, \{u_j, \mathsf{H}(\mathsf{ctxt}_j)\}_{j\in[q_h]})$ in game $\mathbf{G}_1$ is sampled from $\mathcal{D}_0$ and in game $\mathbf{G}_2$ is sampled from $\mathcal{D}_1$. The rest of the game is identical. Therefore, by lemma 1, $|\Pr[G_1] - \Pr[G_2]| \leq \varepsilon_{\mathsf{ddh}} + 1/p$.

GAME $\mathbf{G}_3$: This game is identical to $\mathbf{G}_2$, except that we use simulated proofs for correctness of partial decryption instead of actual NIZK proofs. Looking ahead, we switch to simulated NIZK proofs in this game to later argue in game $\mathbf{G}_5$ that the NIZK proofs do not reveal any information about the secret keys. This is crucial to argue the indistinguishability between game $\mathbf{G}_4$ and $\mathbf{G}_5$. We have the following lemma.

Lemma 11. *Let q_e be the upper bound on the number of encryption queries, $\varepsilon_{\mathsf{ZK}}$ be the distinguishing advantage of a simulated NIZK proof, then,*

$$|\Pr[G_3] - \Pr[G_2]| \leq q_e \cdot n \cdot \varepsilon_{\mathsf{ZK}} \tag{19}$$

Proof. The NIZK protocol (Figure 4) we use is Zero-Knowledge. Hence, conditioned on the successful simulation of the NIZK proofs, the views of the $\mathcal{A}$ in games $\mathbf{G}_2$ and $\mathbf{G}_3$ are identically distributed. Since $\mathcal{A}$ makes at most q_e encryption queries, and we need to simulate at most n partial decryptions per encryption query, the lemma follows from a simple union bound.

GAME $\mathbf{G}_4$: This game is identical to $\mathbf{G}_3$ except we sample the secret key polynomial $v(\cdot)$ such that $v(0) \leftarrow\$ \mathbb{Z}_p^*$. This is another crucial step in our proof where we introduce a rigged pk

Lemma 12. $\Pr[G_3] = \Pr[G_4]$

Proof. To help us prove this lemma, we define a distinguishing game $\mathbf{G}_{\mathsf{VSS}}$. We show that distinguishing between game $\mathbf{G}_3$ and game $\mathbf{G}_4$ can be reduced to winning game $\mathbf{G}_{\mathsf{VSS}}$. In $\mathbf{G}_{\mathsf{VSS}}$, a distinguisher $\mathcal{B}$ interacts with a challenger Chal that works as follows.

1. Sample $g, h \leftarrow\$ \mathbb{G}$ and send (g, h) to $\mathcal{B}$.
2. Sample $x \leftarrow\$ \mathbb{Z}_p$ and a random degree t polynomial $x(\cdot)$ such that $x(0) = x$.
3. Sample a random bit $\mu \leftarrow\$ \{0, 1\}$. If $\mu = 0$, let $y := 0$, otherwise, let $y \leftarrow\$ \mathbb{Z}_p$. Sample degree t random polynomial $y(\cdot)$ such that $y(0) = y$.
4. Send $(g^x h^y), \{g^{x(i)} h^{y(i)}\}_{i\in[n]})$ to $\mathcal{B}$.
5. Upon receiving an index $i \in [n]$ from $\mathcal{B}$ for the j-th time for $j \in [t]$, respond with $(x(i), y(i))$.

At the end of the interaction, $\mathcal{B}$ outputs a bit. Let $\varepsilon_{\mathsf{VSS}}$ denote $\mathcal{B}$'s distinguishing advantage, i.e., $\varepsilon_{\mathsf{VSS}} := |\Pr[\mathcal{B} \Rightarrow 1|\mu = 1] - \Pr[\mathcal{B} \Rightarrow 1|\mu = 0]|$.

Intuitively, in game $\mathbf{G}_{\mathsf{VSS}}$, $\mathcal{B}$ receives the commitment of some x, y and the commitments of all Shamir sharings of x, y. Then $\mathcal{B}$ adaptively corrupts at most t shares of x, y. Lastly, $\mathcal{B}$ attempts to distinguish if $y = 0$.

Next, we show that the difference between games G_3 and G_4 is exactly $\varepsilon_{\mathsf{VSS}}$ by building a reduction adversary $\mathcal{B}$ against the $\mathbf{G}_{\mathsf{VSS}}$ game from $\mathcal{A}$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

- Upon receiving $(g^x h^y, \{g^{x(i)} h^{y(i)}\}_{i\in[n]})$, $\mathcal{B}$ sends them to $\mathcal{A}$ as pk and $\{\mathsf{pk}_i\}_{i\in[n]}$.
- When $\mathcal{A}$ queries the encryption oracle on message m, $\mathcal{B}$ responds with $\mathsf{ctxt} := (m \cdot \mathsf{pk}^r, g^r), \{\hat{m}_i := \mathsf{pk}_i^r\}_{i\in[n]}$ for $r \leftarrow\$ \mathbb{Z}_p$.
- when $\mathcal{A}$ queries the corruption oracle on party i, $\mathcal{B}$ sends i to Chal and receives $(x(i), y(i))$. $\mathcal{B}$ sends $(x(i), y(i))$ to $\mathcal{A}$ as party i's secret key.
- When $\mathcal{A}$ challenges m_0, m_1, $\mathcal{B}$ samples random coin $\mu' \leftarrow\$ \{0, 1\}$. $\mathcal{B}$ encrypts $m_{\mu'}$ as $\mathsf{ctxt} := (m_{\mu'} \cdot \mathsf{pk}^r, g^r)$ for $r \leftarrow\$ \mathbb{Z}_p$. $\mathcal{B}$ sends ctxt to $\mathcal{A}$ and receives bit $\mu_{\mathcal{A}} \in \{0, 1\}$. If $\mu_{\mathcal{A}} = \mu'$, $\mathcal{B}$ outputs 1. $\mathcal{B}$ outputs 0 if otherwise.

When $\mu = 0$, i,e., $y := 0$, $\mathcal{A}$'s view is identical to that of game $\mathbf{G}_3$. Similarly, when $\mu = 1$, i.e., $y \leftarrow\$ \mathbb{Z}_p$, $\mathcal{A}$'s view is identical to that of game $\mathbf{G}_4$. Then we have $|\Pr[G_3] - \Pr[G_4]| := |\Pr[\mathcal{B} \Rightarrow 1|\mu = 0] - \Pr[\mathcal{B} \Rightarrow 1|\mu = 1]| := \varepsilon_{\mathsf{VSS}}$.

Lastly, we show that $\varepsilon_{\mathsf{VSS}} := 0$ and thus $\Pr[G_3] = \Pr[G_4]$. To prove this, we rely on the H-coefficient technique [24, 5]. Without loss of generality, we can assume that the optimal adversary $\mathcal{B}$ is unbounded and deterministic.

Let W_0 and W_1 denote the world when $\mu = 0$ and $\mu = 1$, respectively. Let T be the random variables denoting the transcripts of the interaction of $\mathcal{D}$ in either W_0 or W_1 for fixed queries from $\mathcal{B}$. From Lemma 9, $\Pr[T = \tau|W_0] = \Pr[T = \tau|W_1]$. Then by H-coefficient, W_0 and W_1 are perfectly indistinguishable, i.e., $\Pr[\mathcal{B} \Rightarrow 1|\mu = 1] = \Pr[\mathcal{B} \Rightarrow 1|\mu = 0]$.

GAME $\mathbf{G}_5$: In this game, we change the challenge ciphertext, i.e., instead of sending a valid encryption, we send $\mathcal{A}$ a random tuple. More precisely, $\mathcal{A}$, after challenging m_0, m_1, receives $\mathsf{ctxt}_\mu := (g^\beta, m_\mu \cdot (g^\beta)^x (g^\zeta)^y)$ where $\mu \leftarrow\$ \{0,1\}$, $g^\beta, g^\zeta \leftarrow\$ \mathbb{G}$. Recall that in game $\mathbf{G}_4$, the challenge ciphertext is of the form $\mathsf{ctxt}_\mu := (g^\beta, m_\mu \cdot (g^\beta)^x (g^{\alpha\beta})^y)$ for some random $\alpha, \beta \leftarrow\$ \mathbb{Z}_p$. In both games $\mathbf{G}_4$ and $\mathbf{G}_5$, $\mathsf{pk} := g^x h^y$, where $h := g^\alpha$.

Lemma 13. $|\Pr[G_4] - \Pr[G_5]| \leq \varepsilon_{\mathsf{ddh}}$

Proof. We will prove this by building an adversary against DDH assumption, $\mathcal{A}_{\mathsf{ddh}}$ from $\mathcal{A}$. $\mathcal{A}_{\mathsf{ddh}}$ receives a tuple $(g^\alpha, g^\beta, g^\zeta)$ and interacts with $\mathcal{A}$ as described. Then if $\mathcal{A}_{\mathsf{ddh}}$'s input is a DDH tuple, then $\mathcal{A}$'s view is exactly that of game $\mathbf{G}_6$. If $\mathcal{A}_{\mathsf{ddh}}$'s input is a random tuple, then $\mathcal{A}$'s view is exactly that of game $\mathbf{G}_7$.

We have the following:

$$\Pr[\mathcal{A}_{\mathsf{ddh}}(g^x, g^y, g^z) = 1|z = xy] = \Pr[G_4] \tag{20}$$

$$\Pr[\mathcal{A}_{\mathsf{ddh}}(g^x, g^y, g^z) = 1|z \leftarrow\$ \mathbb{Z}_p] = \Pr[G_5] \tag{21}$$

Hence, $\mathcal{A}_{\mathsf{ddh}}$'s advantage is $|\Pr[G_4] - \Pr[G_5]| \leq \varepsilon_{\mathsf{ddh}}$. □

In the last part of our analysis, we will prove that $\mathcal{A}$ wins in game $\mathbf{G}_5$ with probability $1/2$. Intuitively, $\mathcal{A}$ receives a challenge ciphertext independent of m_0 and m_1; thus, the probability of $\mathcal{A}$ winning is equivalent to random guessing.

Lemma 14. $\Pr[G_5] = 1/2$.

Proof. The only part in $\mathcal{A}$'s view that depends on the hidden bit μ is ctxt_μ. In game $\mathbf{G}_5$, $\mathcal{A}$ receives $\mathsf{ctxt}_\mu = (c, u) = (m_\mu \cdot g^{\beta x} \cdot g^{\zeta y}, g^\beta)$ from the game. Since $\beta, \zeta \leftarrow\$ \mathbb{Z}_p$, ctxt_μ is uniformly randomly distributed in $\mathbb{G}$ for both m_0 and m_1, independent of μ. There is exactly one (β, ζ) that can result in any feasible (c, u) for both $\mu = 0$ and $\mu = 1$. Thus, for random variable (c, u), all feasible (c^*, u^*):

$$\begin{aligned} \Pr[(c,u) = (c^*,u^*)|\mu = 0] &= \Pr[(c,u) = (c^*,u^*)|\mu = 1] \\ &= \Pr[\beta = \beta^*] \cdot \Pr[\zeta = \zeta^*] = \frac{1}{p^2} \end{aligned} \tag{22}$$

Therefore, we conclude that

$$\Pr[\mathcal{A} = 1|b = 0] = \Pr[\mathcal{A} = 1|b = 1] = \Pr[G_5] = 1/2.$$ □

We now prove the following main theorem.

Theorem 3 (Adaptively IND-TCPA Secure Threshold Encryption). *Let λ be the security parameter and $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$ be a group of prime order p. Let $n \in \mathbb{N}$ and $t < n$. Consider any PPT adversary that makes at most q_e encryption queries, q_h random oracle queries to H. Let $\varepsilon_{\mathsf{ZK}}$ be the adversary's advantage in distinguishing simulated NIZK proofs. Then, assuming hardness of DDH in $\mathbb{G}$ and the random oracle model, the adversary has a guessing advantage in the $\mathsf{IND\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ game (Section A) for our scheme in Figure 3 at most $\varepsilon_{\text{IND-TCPA}}$ where*

$$\varepsilon_{\text{IND-TCPA}} \leq 2\varepsilon_{\mathsf{ddh}} + \varepsilon_{\mathsf{ZK}} \cdot q_e \cdot n + \frac{1}{p} \tag{23}$$

where $\varepsilon_{\mathsf{ddh}}$ is the advantage an adversary in breaking DDH in $\mathbb{G}$.

Proof. From the above series of games and lemma 14, we get

$$|\Pr[G_0] - \frac{1}{2}| \leq \varepsilon_{\text{IND-TCPA}} \leq 2\varepsilon_{\mathsf{ddh}} + \varepsilon_{\mathsf{ZK}} \cdot q_e \cdot n + \frac{1}{p} \tag{24}$$

which is $\mathcal{A}$'s guessing advantage against our threshold encryption scheme. □

D Analysis of NIZK Protocol

In this section, we show that our NIZK protocols defined in Figure 4 and Figure 7 that are used in our IND-TCPA secure and IND-TCCA secure threshold decryption schemes, respectively, satisfy completeness, soundness, and zero-knowledge defined in Definition 1.

Lemma 15. *The proposed NIZK protocol in Figure 4, which is used in our IND-CPA secure threshold decryption scheme, satisfies completeness, soundness, and zero knowledge.*

Proof. Completeness is straightforward. We now focus on soundness and zero-knowledge.

Soundness. We show that our NIZK protocol satisfies soundness by showing that its underlying Σ-protocol satisfies *knowledge soundness*, i.e., there exists an efficient witness extractor $\mathcal{E}$ that interacts with a $\mathcal{P}$ (that produces an accepting proof) and extracts x such that $(x, y) \in \mathcal{R}$. $\mathcal{E}$ interacts with the prover twice, using two different challenges e and e' on the same input. The prover returns two accepting proofs, $(\gamma, \psi, f_x, f_y), (\gamma, \psi, f'_x, f'_y)$, where $e \neq e'$. The witness $x = \frac{f_x - f'_x}{e - e'}$ and $y = \frac{f_y - f'_y}{e - e'}$ can be extracted from the two proofs. Since knowledge soundness implies soundness [21, §3.2], our NIZK protocol satisfies soundness.

Zero-knowledge. We prove zero-knowledge by constructing a simulator $\mathcal{S}$ such that $\mathcal{A}$'s view in its interaction with $\mathcal{S}$ is distinguishable from its view in the real execution of the protocol with negligible probability $\varepsilon_{\mathsf{ZK}}$. Additionally, $\mathcal{A}$ at most q_{FS} queries to H_{FS}. To answer $\mathcal{A}$'s random oracle queries, $\mathcal{S}$ samples responses

uniformly at random. To simulate a NIZK proof, $\mathcal{S}$ samples $(e, f_x, f_y) \leftarrow\$ \mathbb{Z}_p^3$, and computes (γ, ψ) as,

$$\gamma = g^{f_x} h^{f_y} \mathsf{pk}^{-e}, \qquad \psi = u^{f_x} H^{f_y} \hat{m}^{-e} \tag{25}$$

Finally, $\mathcal{S}$ programs the random oracle $\mathsf{H_{FS}}$ such that $\mathsf{H_{FS}}(\gamma, \psi, \mathsf{pk}, g, h, u, \hat{m}) := e$, and outputs (γ, ψ, f_x, f_y), which is an accepting transcript with the correct distribution, provided the bad event Coll does not occur. Coll occurs if $\mathcal{A}$ queries $\mathsf{H_{FS}}$ on $(\gamma, \psi, \mathsf{pk}, g, h, u, \hat{m})$ before $\mathcal{S}$ outputs (γ, ψ, f_x, f_y) as the simulated proof. If Coll occurs, $\mathcal{S}$ cannot program $\mathsf{H_{FS}}$ to the desired e, causing the simulation to fail.

The simulation fails with the same probability that event Coll occurs. Lastly, we analyze the probability of event Coll. Since $\gamma, \psi \leftarrow\$ \mathbb{G}$ is sampled uniformly at random, Coll occurs with probability at most q_{FS}/p^3. Thus $\varepsilon_{\mathsf{ZK}} \leq q_{\mathsf{FS}}/p^3$, which is negligible. □

Lemma 16. *The proposed NIZK protocol in Figure 7, which is used in our IND-CCA secure threshold decryption scheme, satisfies completeness, soundness, and zero knowledge.*

Proof. The completeness, soundness, and zero-knowledge of our NIZK scheme defined in Figure 7 follow similarly to Lemma 15. □

E Analysis of Correctness and Robustness

In this section, we provide analysis of the correctness and robustness properties of our threshold decryption schemes.

Our IND-CPA secure threshold decryption scheme satisfies the correctness property and the robustness property defined in Definition 5 and definition 7, respectively. We use the NIZK protocol defined in Definition 1 to ensure these two properties of our scheme. Recall that in our scheme, during PDec of ciphertext ctxt, each party i acts as the prover and computes a NIZK proof π_i for the correctness of the partial decryption $\hat{m}_i$ in relation to the per-party public key pk_i as part of the decryption share. More specifically, each party i computes π_i for the following statement: party i knows secret key (x_i, y_i) such that $\mathsf{pk}_i = g^{x_i} h^{y_i}$ and $\hat{m}_i = u^{x_i} \mathsf{H}(\mathsf{ctxt})^{y_i}$.

The combiner, upon receiving a partial decryption in the form of $(\hat{m}_i, \pi_i)$, acts as the verifier in Definition 1 and verifies π_i. Only $\hat{m}_i$ which corresponding π_i is accepted (verified to true) is used in recovering the plaintext.

The correctness of our scheme is due to the completeness of the NIZK protocol since all honestly computed $(\hat{m}_i, \pi_i)$ pairs will be accepted by the combiner.

The robustness of our scheme follows from the soundness property of the NIZK protocol. Intuitively, the soundness property of the NIZK protocol ensures that dishonestly computed decryption shares do not contribute to the recovery of the plaintext with high probability. More formally, we have the following lemma:

Lemma 17 (TCPA-Robustness). *Assuming the NIZK protocol satisfies soundness defined in Definition 1, the threshold decryption in Figure 3 is* $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ *secure.*

Proof. Let $\mathcal{B}$ denote an adversary against the soundness game defined in definition 1. Let $\mathcal{A}$ be an adversary against the $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ game provided in Section A. $\mathcal{B}$ interacts with $\mathcal{A}$ by honestly providing pk and $\{\mathsf{pk}_i\}_{i\in[n]}$ by sampling secret polynomials $x(\cdot), y(\cdot)$. $\mathcal{B}$ also honestly answers all oracle queries according to the protocol specification. Observe that $\mathcal{A}$ wins the $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ game in two ways. The first way is by violating the correctness property, defined in definition 5, of our scheme. More precisely, $\mathcal{A}$ wins the $\mathsf{RB\text{-}TCPA}^{\mathcal{A}}_{\mathsf{TD}}$ game by finding a set $\mathcal{S}'$ of parties whose honestly computed partial decryptions combine to a plaintext that differs from the one produced by set $\mathcal{S}$. The second way is by forging at least one accepted π_i for a dishonestly computed $\hat{m}_i$. Thus, $\mathcal{B}$ wins the soundness game by identifying and outputting such an accepted but invalid π_i, statement pair. Since $\mathcal{B}$ knows the witness, i.e., the secret key of party i, it can verify that π_i proves an incorrect statement. □

Our IND-CCA secure threshold decryption scheme satisfies the correctness property defined in Definition 5, which follows from the completeness property of our NIZK protocol.

Our adaptively secure threshold Shoup-Gennaro decryption scheme satisfies the robustness property defined in Definition 6 through reasoning similar to that in Lemma 17, except that the reduction adversary additionally answers decryption oracle queries.

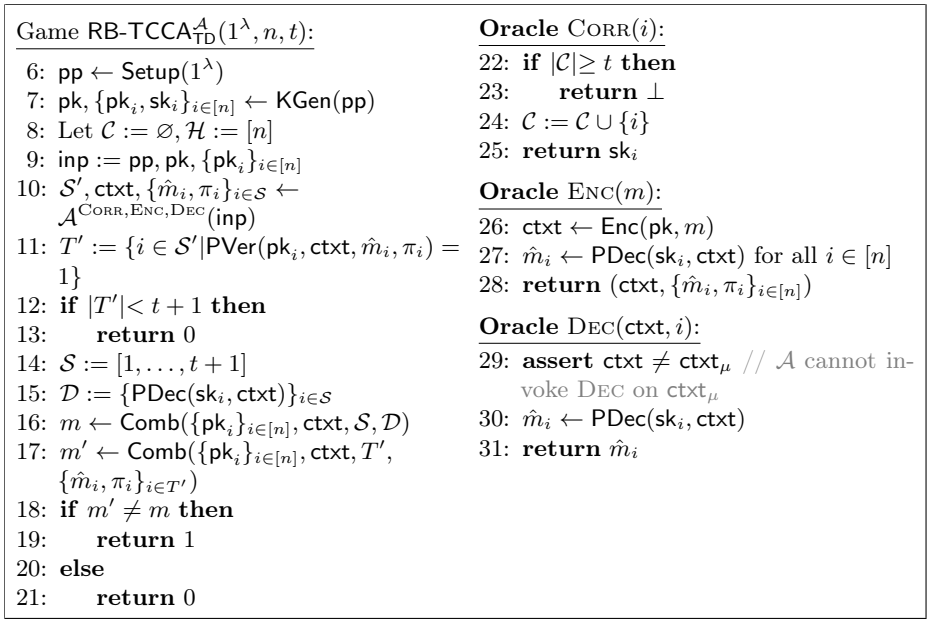

Fig. 5: The robustness security game RB-TCCA$^{\mathcal{A}}_{\mathsf{TD}}$ for a non-interactive threshold decryption $\mathsf{TD} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{PDec}, \mathsf{Comb})$ with an adaptive adversary $\mathcal{A}$.

$\underline{\mathsf{Setup}(1^\lambda, n, t)}$:

1: $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$.
2: Let $h, v \in \mathbb{G}$ be a uniformly random generator of $\mathbb{G}$.
3: Let $\mathsf{H}_1 : \{0,1\}^l \times \mathbb{G}^4 \to \mathbb{Z}_p$, $\mathsf{H}_2, \mathsf{H}_3 : \{0,1\}^l \times \mathbb{G}^2 \times \mathbb{Z}_p^2 \to \mathbb{G}$ and $\mathsf{H}_{\mathsf{FS}} : \{0,1\}^* \to \mathbb{Z}_p$ be four hash functions modeled as random oracle.
4: **return** $\mathsf{pp} := (\mathbb{G}, p, n, t, g, h, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_{\mathsf{FS}})$.

// We assume all algorithms implicitly take pp as input. We use H_{FS} in SigmaProve and SigmaVer.

$\underline{\mathsf{KGen}(\mathsf{pp})}$:

5: Let $x(\cdot), y(\cdot), z(\cdot) \leftarrow\!\!\$\ \mathbb{Z}_p[x]$ be three polynomials of degree t with $y(0) := 0, z(0) := 0$.
6: $\mathsf{pk} := g^{x(0)} h^{y(0)} v^{z(0)} = g^{x(0)}$
7: **for** each $i \in [n]$ **do**
8: $\quad \mathsf{sk}_i := (x(i), y(i), z(i))$
9: $\quad \mathsf{pk}_i := g^{x(i)} h^{y(i)} v^{z(i)}$
10: **return** $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{sk}_j)$ to party j for all $j \in [n]$

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

11: $r, s \leftarrow\!\!\$\ \mathbb{Z}_p$
12: $(c, u) := (m \cdot \mathsf{pk}^r, g^r)$
13: $w, \bar{u}, \bar{w} := g^s, \bar{g}^r, \bar{g}^s$
14: $e := \mathsf{H}_1(c, u, w, \bar{u}, \bar{w})$
15: $f := s + er$
16: **return** $\mathsf{ctxt} := (c, u, \bar{u}, e, f)$

$\underline{\mathsf{PDec}(\mathsf{sk}_i, \mathsf{ctxt} = (c, u, \bar{u}, e, f))}$:

17: **if** $e \neq \mathsf{H}_1(c, u, w, \bar{u}, \bar{w})$ where $w := \frac{g^f}{u^e}, \bar{w} := \frac{\bar{g}^f}{\bar{u}^e}$ **then**
18: $\quad$ **return** $\perp$
19: $\hat{m}_i := u^{x(i)} \mathsf{H}_2(\mathsf{ctxt})^{y(i)} \mathsf{H}_3(\mathsf{ctxt})^{z(i)}$
20: $\pi_i := \mathsf{SigmaProve}(pk_i, u, \hat{m}_i, \mathsf{sk}_i)$
21: **return** $(\hat{m}_i, \pi_i)$

$\underline{\mathsf{PVer}(\mathsf{pk}_i, c = (c_1, c_2), \hat{m}_i, \pi_i)}$:

22: **return** $\mathsf{SigmaVer}(\mathsf{pk}_i, u, \hat{m}_i, \pi_i)$

$\underline{\mathsf{Comb}(\{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{ctxt}, S, \{(\hat{m}_i, \pi_i)\}_{i\in S})}$:

23: **parse** $(c, u, \bar{u}, e, f) := \mathsf{ctxt}$
24: $T := \varnothing$
25: **for** each $i \in S$ **do**
26: $\quad$ **if** $\hat{m}_i = \perp$ **then**
27: $\quad\quad$ **return** $\perp$
28: $\quad$ **if** $e \neq \mathsf{H}_1(c, u, w, \bar{u}, \bar{w})$ where $w := \frac{g^f}{u^e}, \bar{w} := \frac{\bar{g}^f}{\bar{u}^e}$ **then**
29: $\quad\quad$ **return** $\perp$
30: $\quad$ **if** $\mathsf{PVer}(u, \mathsf{pk}_i, \hat{m}_i, \pi_i)$ **then**
31: $\quad\quad$ $T \cup \{i\}$
32: **assert** $|T| \geq t + 1$
33: Let $L_{i,T}$ be the i-th Lagrange coefficients for T
34: $\hat{m} := \prod_{i\in T} \hat{m}_i^{L_{i,T}}$
35: **return** $m := c \cdot \hat{m}^{-1}$

Fig. 6: Adaptively IND-TCCA secure threshold ElGamal decryption scheme with trusted key generation.

Input: $g, h, v, \mathsf{pk}, H_2, H_3, u, \hat{m}$
Witness: $(x, y, z) \in \mathbb{Z}_p^3$
The prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that it knows $x, y, z \in \mathbb{Z}_p$ such that $\mathsf{pk} = g^x h^y v^z$ and $\hat{m} = u^x {H_2}^y {H_3}^z$.

// We assume that both algorithms implicitly take of $g, h, v, u, H_2, H_3, \hat{m}$ as input

$\underline{\mathsf{SigmaProve}(\mathsf{pk}, u, \hat{m}, x, y, z)}$:
1: Sample $\hat{x}, \hat{y}, \hat{z} \leftarrow\!\!\$\ \mathbb{Z}_p$.
2: $\gamma := g^{\hat{x}} h^{\hat{y}} v^{\hat{z}}, \quad \psi := u^{\hat{x}} {H_2}^{\hat{y}} {H_3}^{\hat{z}}$
3: $e := \mathsf{H}_{\mathsf{FS}}(\gamma, \psi, \mathsf{pk}, u, \hat{m})$
4: $f_x := \hat{x} + x \cdot e, \quad f_y := \hat{y} + y \cdot e, \quad f_z := \hat{z} + z \cdot e.$
5: **return** $\pi := (\gamma, \psi, f_x, f_y, f_z)$.

$\underline{\mathsf{SigmaVer}(\mathsf{pk}, u, \hat{m}, \pi = (\gamma, \psi, f_x, f_y, f_z))}$:
6: $e := \mathsf{H}_{\mathsf{FS}}(\gamma, \psi, \mathsf{pk}, u, \hat{m})$
7: **if** $g^{f_x} h^{f_y} v^{f_z} = \gamma \cdot \mathsf{pk}^e$ and $u^{f_x} {H_2}^{f_y} {H_3}^{f_z} = \psi \cdot \hat{m}^e$ **then**
8: **return** 1
9: **return** 0

Fig. 7: Σ-protocol for computing and verifying the correctness proof for partial decrytpions CCA scheme.