Adaptive TDFs from Injective TDFs

Xinyu Mao * Hongxu Yi †

June 1, 2025

Abstract

Adaptive trapdoor functions (ATDFs) and tag-based ATDFs (TB-ATDFs) are variants of trapdoor functions proposed by Kiltz, Mohassel, and O'Neill (EUROCRYPT 2010). They are both sufficient for constructing chosen-ciphertext secure public-key encryption (CCA-secure PKE), and their definitions are closely related to CCA-secure PKE. Hohenberger, Koppula, and Waters (CRYPTO 2020) showed that CCA-secure PKE can be constructed from injective TDFs; however, the relations among TDF, ATDF, and TB-ATDF remain unclear.

We provide black-box constructions of ATDFs and TB-ATDFs from injective TDFs, answering the question posed by Kiltz, Mohassel, and O'Neill (EUROCRYPT 2010). Our results indicate that ATDF, TB-ATDF, and TDF are equivalent under mild restrictions.

1 Introduction

Trapdoor function (TDF) is a fundamental primitive in public-key cryptography [DH76, RSA78]. Roughly speaking, a TDF is a function family where each function is indexed by an evaluation key and associated with a trapdoor; the function is easy to compute given the evaluation key, and easy to invert given the trapdoor. The security requirement is that the function should be hard to invert given only the evaluation key.

For public-key encryption (PKE), security under chosen ciphertext attack (CCA) is necessary in various applications, which provides security guarantees against active adversaries. This raises an immediate question: Can we construct CCA-secure PKE from TDFs? Towards answering this question, Kiltz, Mohassel, and O'Neill introduced the notion of *adaptive TDF (ATDF)*, where the adversary can access the inversion oracle except for the challenge image; they also considered an extension called *tag-based* ATDF (TB-ATDF) and presented simple black-box constructions of CCA-secure PKE from both ATDF and TB-ATDF [KMO10]. In a beautiful work, Hohenberger, Koppula, and Waters (henceforth HKW [HKW20]) addressed the question conclusively, giving a black-box construction of CCA-secure PKE from injective TDFs.

Though (injective) TDF is sufficient for CCA-secure PKE, ATDF is still an instructive notion. First, in the random oracle model, TDF, ATDF, and TB-ATDF are equivalent. Second, several constructions of CCA-secure PKE (e.g., [PW08, RS09, Wee12]) can be unified into the TB-ATDF framework: One starts with constructing a TB-ATDF and then uses the transformation in [KMO10] to get

*Thomas Lord Department of Computer Science, University of Southern California. Email: xinyumao@usc.edu

†School of Cyber Science and Technology, Shandong University, Qingdao 266237, China. Email: tcs.hongxu.yi@mail.sdu.edu.cn

a CCA-secure PKE. ATDF is also strictly weaker than some other notions of TDF, like correlated-product TDF [RS09] or lossy TDF [PW08].

On the complexity and constructions of ATDF and TB-ATDF. As noted in [KMO10], the relations among TDF, ATDF, and TB-ATDF are unclear; moreover, ATDF is strictly weaker than some other notions of TDF, including correlated TDF [RS09] and lossy TDF [PW08]. The constructions of ATDF and TB-ATDF in [KMO10] are based on a (non-standard) variant of RSA assumption. Kitagawa, Matsuda, and Tanaka constructed ATDF/TB-ATDF from PKE with pseudorandom ciphertexts plus secret-key encryption with key-dependent message (KDM) security [KMT22], which can be instantiated from standard assumptions such as CDH, DDH, LPN, and LWE (e.g., [ACPS09, BHHI10, HL17, BLSV18]).

ATDF is similar to CCA-secure PKE in the sense that they both provide security when the adversary has access to an oracle on all inputs except for the challenge input, where the oracle breaks the security for its input (by inversion or decryption). Owing to the similarity, the result of HKW suggests the potential of constructing ATDF from TDF, and hence we revisit this question:

Can we construct ATDF and TB-ATDF from (injective) TDF?

In this paper, we answer this question affirmatively, showing that all three primitives are equivalent (modulo mild restrictions).

1.1 Our Results

For $\mathsf{X} \in \{$TDF, ATDF, TB-ATDF$\}$, we say X is ***canonical*** if it is injective, perfectly correct, and the domain of X is an Abelian group that only depends on the security parameter κ (e.g., $\{0,1\}^{\ell(\kappa)}$). In a nutshell, we present black-box constructions of ATDF and TB-ATDF from canonical TDF:

Theorem 1.1. *There exists a black-box construction of ATDF/TB-ATDF from a canonical TDF.*

Remark 1.2. The resulting ATDF and TB-ATDF are not canonical. The ATDF in [KMT22] is canonical, but requires additional assumptions; if we wish to adopt a similar approach, we have to assume that the images of the TDF also form an Abelian group, which seems to be an overly restrictive assumption.

Remark 1.3. The "perfect correctness" requirement can be relaxed to *almost-all-key correctness*, i.e., with overwhelming probability over the generation of evaluation key ek and trapdoor td, it holds that $\mathsf{Inv}(\mathsf{Eval}(\mathsf{ek}, x)) = x$ for all input x, where $\mathsf{Eval}, \mathsf{Inv}$ are the evaluation and inversion algorithm for the TDF respectively.

ATDF and TB-ATDF trivially imply TDF, and the resulting TDF is canonical if one starts with a canonical ATDF or TB-ATDF. Therefore, we establish that

Theorem 1.4. *For* $\mathsf{X}, \mathsf{Y} \in \{$*TDF, ATDF, TB-ATDF*$\}$, *there exists a black-box construction of* Y *from a canonical* X.

This completes the picture of the relations between variants of trapdoor functions in [KMO10], as shown in fig. 1.

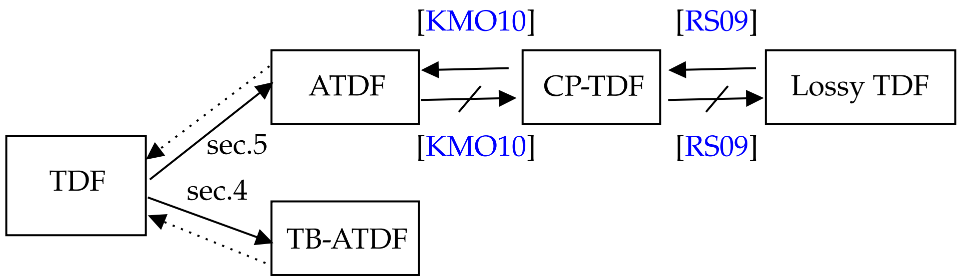

Figure 1: Relations among the variants of trapdoor functions. "$\rightarrow$" denotes an implication and "$\not\rightarrow$" denotes a black-box separation; dashed lines indicate trivial implications. Constructions in this paper require the starting primitive to be canonical. CP-TDF stands for correlated-product TDF.

1.2 Technical Overview

We start with a tag-based version of the CCA-secure PKE construction due to HKW. The construction uses a *randomness-recoverable*, CPA-secure PKE $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Recover})$. That is, besides decryption, the message m can also be recovered using the randomness r underlying the ciphertext ct, i.e., $m = \mathsf{Recover}(\mathsf{pk}, \mathsf{ct}, r)$ if $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m; r)$; moreover, we also require the decryption algorithm to recover r as well as m. For example, Yao's construction of PKE from TDF is randomness-recoverable [Yao82].

Consider encrypting the message using N key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}$, namely, $\mathsf{ct} := (\mathsf{ct}_i)_{i\in[N]}$ where $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, m; r_i)$. The decryption algorithm decrypts all N ciphertexts and if all messages are the same (well-formedness check), it outputs the common message. Imagine that we want to prove the CCA-security of this naive construction, then one step should be switching ct_i to a dummy ciphertext by a reduction to the CPA-security of the i-th key pair. The reduction algorithm has no access to sk_i, so it cannot conduct the well-formedness check; however, it has to answer the decryption queries made by the adversary in the CCA game. Therefore, the crux of the proof of HKW is to devise a mechanism that allows one to simulate the decryption oracle using only $(N-1)$ key pairs.

This is done by introducing a correlation between random coins $r_1, \ldots, r_N$. Note that with $(N-1)$ key pairs $(\mathsf{sk}_j)_{j\neq i}$, the reduction algorithm can recover the random coins $(r_j)_{j\neq i}$. Hence, as long as the correlation allows us to recover r_i from $(r_j)_{j\neq i}$, we can use the randomness-recovery mechanism to "decrypt" ct_i by $z_i := \mathsf{Recover}(\mathsf{pk}_i, \mathsf{ct}_i, r_i)$. A naive attempt is to choose $r_1, \ldots, r_N$ such that $\bigoplus_{i=1}^{N} r_i = 0^\ell$ (where $r_1, \ldots, r_N \in \{0,1\}^\ell$ and $\oplus$ denotes bit-wise XOR). However, this correlation hinders the reduction to CPA-security since the random coins for encryption are no longer chosen uniformly at random. Instead, HKW uses a statistically weak correlation: choose a subset $S \subseteq [N]$ of size B uniformly at random, and sample $r_1, \ldots, r_N \leftarrow \{0,1\}^\ell$ conditioned on $\bigoplus_{i\in S} r_i = 0^\ell$. For proper choice of parameter (i.e., $\binom{N}{B} \gg 2^\ell$), the distribution of $r_1, \ldots, r_N$ is statistically close to uniform.

To make this correlation useful, we also modify the construction to indicate in the plaintext whether $i \in S$: for $i \in S$, $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, 1\|m; r_i)$; otherwise, $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, 0\|m; r_i)$; the decryption aborts unless there are exactly B ciphertexts that decrypt to $1\|m$ and the randomness underlying these ciphertexts XOR to zero. Call an index $i \in [N]$ *active* if ct_i decrypts to $1\|m$. If ct is well-formed in the sense that there are *at most* B active indices, we can simulate decryption using only $(\mathsf{sk}_j)_{j\neq i}$

as follows: Let $U := \{j \in [N] \setminus \{i\} : j \text{ is active}\}$.

- If $|U| = B$, the decryption is successful if the randomness underlying $(\mathsf{ct}_i)_{i\in U}$ XOR to zero; there is no need to decrypt ct_i.
- If $|U| = B - 1$, set $r_i = \bigoplus_{j\in U} r_j$ and $z_i := \mathsf{Recover}(\mathsf{pk}_i, \mathsf{ct}_i, r_i)$; the decryption is successful if $z_i = 1\|m$ and $\mathsf{Enc}(\mathsf{pk}_i, 1\|m; r_i) = \mathsf{ct}_i$.
- If $|U| < B - 1$, outputs $\perp$; the real decryption algorithm must also abort in this case.

The above simulation coincides with the real decryption algorithm on well-formed ciphertexts.

The last missing piece is a tagged mechanism, called *tagged set commitment*, that allows us to enforce well-formedness on all tags except for the challenge tag T^*. More specifically, we want to violate the well-formedness on tag T^* so that later we can make all indices in the challenge ciphertext ct^* active, removing the information about S^* (the B-size set used in the generation of ct^*) in the plaintexts. Once the information about S^* is only leaked by the correlation among the random coins used for generating ct^*, the correlation statistically vanishes as mentioned earlier.

Tagged set commitment. A tagged set commitment (TSC) is a scheme that allows one to commit to a B-size set $S \subseteq [N]$ with a tag T (where N, B are given to generate public parameters pp). That is, $\mathsf{Commit}(\mathsf{pp}, S, T)$ outputs a commitment com and openings $(\sigma_i)_{i\in S}$ for each element. The verification algorithm checks the opening σ_i to verify that $i \in S$ under tag T. The soundness property roughly says any commitment has at most B valid openings. The scheme supports an alternative setup algorithm $\mathsf{AltSetup}$ that takes in a tag T^* and produces public parameters together with a special commitment com^* and openings $(\sigma_i^*)_{i\in[N]}$ for all N elements; it violates the soundness property under tag T^* while retaining soundness for other tags. We require that the two modes of setup are indistinguishable for efficient adversaries. HKW showed that TSC can be constructed from pseudorandom generators.

A tag-based version of the HKW construction is obtained by integrating TSC into the aforementioned construction:

- The encryption algorithm, on input tag T and message m, chooses $S, r_1, \ldots, r_N$, computes $(\mathsf{com}, (\sigma_i)_{i\in S}) \leftarrow \mathsf{Commit}(\mathsf{pp}, S, T)$, and outputs $(\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})$ where $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \sigma_i\|m)$ for $i \in S$ and for $i \notin S$, ct_i is a dummy ciphertext.
- The decryption algorithm, on input tag T and ciphertext $(\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})$, decrypts $z_i := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ and also recover the randomness r_i; if there are exactly B indices i such that (1) $z_i = \sigma_i\|m$, (2) σ_i is an opening of com under tag T, and (3) these r_i's XOR to zero, then it outputs m; otherwise, it outputs $\perp$.

From tag-based CCA-secure PKE to TB-ATDF. First, we observe that, in the TSC construction proposed by HKW, the randomness used by Commit can be fully recovered from the openings $(\sigma_i)_{i\in S}$. At first glance, the above tag-based CCA-secure PKE is directly a TB-ATDF—all randomness used in encryption can be recovered during decryption. Nevertheless, when simulating the inversion oracle (in a reduction to the CPA-security), the reduction cannot always recover all the randomness. Consider simulating the inversion oracle with $(\mathsf{sk}_j)_{j\neq i}$ and an inversion query $(T, \mathsf{com}, (\mathsf{ct}_j)_{j\in[N]})$. If the index i is active in the sense that ct_i encrypts an opening of com, then we can recover r_i from $(r_j)_{j\neq i}$. However, if i is not active, then there is no way to recover r_i from ct_i without sk_i.

To address this issue, we note that the PKE constructed from a canonical TDF has the property that the encryption of a random message is uniformly distributed over the ciphertext space C. Our remedy is as follows: for $i \notin S$, we set $\mathsf{ct}_i \leftarrow C$ instead of a dummy ciphertext, and put $(\mathsf{ct}_i)_{i\in[N]\setminus S}$ directly into the input. In other words, the input of our TB-ATDF is of the form

$$x = \left(r_{\mathsf{com}}, S, (\mathsf{ct}_i)_{i\in[N]\setminus S}, (r_{i_j})_{j\in[B-1]}\right),$$

where $S = \{i_1, \ldots, i_B\} \subseteq [N]$ and r_{com} is the randomness used for commitment. And to evaluate on input x under tag T, one sets $r_{i_B} := \bigoplus_{j<B} r_{i_j}$ and compute

$$y = (\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})$$

as the function value, where $(\mathsf{com}, (\sigma_i)_{i\in S}) \leftarrow \mathsf{Commit}(\mathsf{pp}, S, T; r_{\mathsf{com}})$, and $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma_i; r_i)$ for $i \in S$. This is reminiscent of the ATDF construction in [KMT22], but we adopt the HKW-style construction to allow simulation of the inversion oracle with partial secret keys, avoiding the use of KDM-secure SKE.

Remark 1.5. Our TB-ATDF construction requires a domain sampling algorithm to sample $\mathsf{ct}_i \leftarrow C$ for $i \notin S$ (by encrypting a random message). Therefore, it is important that the output of our domain sampling algorithm is uniformly distributed over the domain; without the restriction of uniform domain sampling, there are trivial (and meaningless) constructions of TDFs [HKW20].

From TB-ATDF to ATDF by using stronger properties of TSC. The transformation in [KMO10] from tag-based CCA-secure PKE to one without tags uses a strong one-time signature (OTS) scheme, and HKW also adopts this transformation. The transformation works as follows: The encryption algorithm generates a signing key sk and a verification key vk for the OTS, and uses vk as tag in the tag-based scheme to get ciphertext c, then it signs c with sk and outputs the signature along with $(\mathsf{vk}, \mathsf{c})$. The decryption algorithm uses vk as tag and proceeds as the tag-based scheme if the signature verifies, and it aborts if the signature does not verify. Although strong OTS can be based on one-way functions due to a classic result by Lamport [Lam79], Lamport's scheme is not randomness-recoverable: Some random coins used in the key generation of $(\mathsf{sk}, \mathsf{vk})$ cannot be recovered given vk and a valid signature. Hence, one cannot draw on this transformation to get ATDF from TB-ATDF.

One possible approach to address this issue is replacing the OTS with target collision-resistant (TCR) hash functions (also known as universal one-way hash functions), which can also be based on one-way functions [Rom90, HHR+10, MMZ23]. Specifically, one can use $\mathsf{H}(\mathsf{ct}_1\| \cdots \|\mathsf{ct}_N)$ as tag for TSC, where H is a TCR hash function. The replacement of OTS by TCR hash functions is used in [MH15] and [KMT22], which requires the underlying tag-based scheme to have special structures. We show that this replacement can be applied to our HKW-style TB-ATDF construction, as long as the TSC satisfies the following stronger properties:

- Adaptive indistinguishability of setup. Roughly speaking, we require that no efficient adversary can distinguish the two modes of setup, even if it submits the challenge tag *after seeing openings* $(\sigma_i)_{i\in S}$.

- Uniqueness. For each B-size set S and openings $(\sigma_i)_{i\in S}$, there is a unique commitment com such that the following holds: For $\mathsf{com}' \neq \mathsf{com}$, there exists some $i^* \in S$ such that σ_{i^*} does not verify under com', i.e.,

$$\mathsf{com}' \neq \mathsf{com} \implies \exists i^* \in S\ \mathsf{TSC.Verify}(\mathsf{pp}, \mathsf{com}', i^*, \sigma_{i^*}, T) = 0.$$

  In other words, com is the unique commitment such that all σ_i's verify.

We show that the TSC scheme by HKW does satisfy both properties, and thus we conclude that ATDF can be constructed from a canonical TDF.

Remarkably, we observe that TCR hash functions are not necessary in our case. This is because the TSC scheme by HKW has the property that the length of openings $(\sigma_i)_{i\in S}$ does not depend on the length of the tag, and hence we can directly use $\mathsf{ct}_1\|\cdots\|\mathsf{ct}_N$ as tag without hashing.

1.3 Discussion

On the almost-all-keys correctness of TDFs. It is unclear how to transform a TDF with ordinary correctness into one with almost-all-keys correctness. Such a transformation exists for PKE [DNR04], but it does not generalize to TDF.

Domain of TDFs. Our constructions require that the domain of TDF only depends on the security parameter and does not depend on the evaluation key ek, which is also assumed in [HKW20]. This requirement is met by many TDF constructions, such as those from the LWE assumption [PW08] and from the DDH assumption [GGH19, DGI+19, DGH+19]. It is not always the case: For example, the RSA trapdoor permutation does not satisfy the requirement, since its domain is $\mathbb{Z}_n^*$ and n is part of the evaluation key. It is an interesting question whether the CCA-secure PKE construction by HKW and our constructions can be adapted to allow more flexible domains.

2 Preliminaries

Notation. We use $[N]$ to denote $\{1, 2, \ldots, N\}$ and $\binom{[N]}{B}$ to denote the set of all B-size subsets of $[N]$. We use $\leftarrow$ to denote sampling from a distribution, choosing an element from a set uniformly at random, or collecting the output of a randomized algorithm. For two distributions χ_1 and χ_2, we write $\chi_1 \approx_s \chi_2$ if χ_1 and χ_2 are statistically close; and $\chi_1 \approx_c \chi_2$ means that they are computationally indistinguishable. For a function $\nu : \mathbb{N} \to [0, 1]$, we write $\nu = \mathtt{negl}(\kappa)$ if for every $c \in \mathbb{N}$, $\nu(\kappa) \leq 1/(c\kappa^c)$ for sufficiently large κ. PPT stands for probabilistic polynomial time.

2.1 Randomness-Recoverable Public-Key Encryption

Let $\ell_{\mathsf{msg}} : \mathbb{N} \to \mathbb{N}$ be a length function. A randomness-recoverable public-key encryption scheme (RR-PKE) with message space $\{0,1\}^{\ell_{\mathsf{msg}}}$ is a tuple of four algorithms $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Recover})$ with the following syntax.

- $\mathsf{Gen}(1^\kappa) \mapsto (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm takes as input the security parameter κ and outputs a public key pk and secret key sk. Public key pk implicitly defines a randomness space $\mathsf{Rnd}_{\mathsf{pk}}$ and efficient procedures to recognize and uniformly sample from $\mathsf{Rnd}_{\mathsf{pk}}$ given pk.

- $\mathsf{Enc}(\mathsf{pk}, m; r) \mapsto \mathsf{ct}$: Then encryption algorithm takes as input a public key pk, a message $m \in \{0,1\}^{\ell_{\mathsf{msg}}}$ and randomness $r \in \mathsf{Rnd}_{\mathsf{pk}}$, and outputs a ciphertext ct. Let C_{pk} denote the set of all possible ciphertext, i.e., $C_{\mathsf{pk}} \stackrel{\text{def}}{=} \left\{\mathsf{Enc}(\mathsf{pk}, m; r) : m \in \{0,1\}^{\ell_{\mathsf{msg}}}, r \in \mathsf{Rnd}_{\mathsf{pk}}\right\}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \mapsto (m, r) \in \{0,1\}^{\ell_{\mathsf{msg}}} \times \mathsf{Rnd}\ /\bot$: The (deterministic) decryption algorithm takes as input a secret key sk and a ciphertext ct, and either outputs $\bot$ or a message $m \in \{0,1\}^{\ell_{\mathsf{msg}}}$.
- $\mathsf{Recover}(\mathsf{pk}, \mathsf{ct}, r) \mapsto z \in \{0,1\}^{\ell_{\mathsf{msg}}}\ /\bot$: The (deterministic) recovery algorithm takes as input a public key pk and a ciphertext ct and randomness $r \in \mathsf{Rnd}_{\mathsf{pk}}$, and outputs $\bot$ or a message $m \in \{0,1\}^{\ell_{\mathsf{msg}}}$.

Definition 2.1 (ε-Almost-all-keys perfect correctness)**.** We say PKE is *ε-almost-all-keys perfectly correct* if there exists a negligible function $\varepsilon(\cdot)$ such that for all $\kappa \in \mathbb{N}$,

$$\Pr_{(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)}\left[\exists m \in \{0,1\}^{\ell_{\mathsf{msg}}}, r \in \mathsf{Rnd} :\ \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r)) \neq (m, r)\right] \leq \varepsilon(\kappa)$$

and

$$\Pr_{(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)}\left[\exists m \in \{0,1\}^{\ell_{\mathsf{msg}}}, r \in \mathsf{Rnd} :\ \mathsf{Rec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) \neq m\right] \leq \varepsilon(\kappa).$$

We sometimes omit ε if it is a negligible function.

Definition 2.2 (IND-CPA security)**.** We say PKE is *IND-CPA secure* if for every PPT adversray $\mathcal{A}$, it holds that $\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(\kappa) \stackrel{\text{def}}{=} \left|\Pr\left[\mathsf{Expr}^{\mathsf{ind\text{-}cpa},0}_{\mathsf{PKE},\mathcal{A}}(\kappa) \Rightarrow 1\right] - \Pr\left[\mathsf{Expr}^{\mathsf{ind\text{-}cpa},1}_{\mathsf{PKE},\mathcal{A}}(\kappa) \Rightarrow 1\right]\right| \leq \mathtt{negl}(\kappa)$, where $\mathsf{Expr}^{\mathsf{ind\text{-}cpa},b}_{\mathsf{PKE},\mathcal{A}}(\kappa)$ is the following experiment:

1. Challenger generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$ and sends pk to $\mathcal{A}$.
2. $\mathcal{A}$ submits $m_0, m_1 \in \{0,1\}^{\ell_{\mathsf{msg}}}$ to challenger.
3. Challenger runs $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$ and sends ct_b to $\mathcal{A}$.
4. $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$; the experiment outputs 1 iff $b' = 1$.

Definition 2.3 (Uniform ciphertext for random message)**.** We say PKE has ***uniform ciphertext for random message*** if for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$, $\mathsf{Enc}(\mathsf{pk}, m; r)$ is uniformly distributed over C_{pk} where $m \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}, r \leftarrow \mathsf{Rnd}_{\mathsf{pk}}$.

We recall the Goldreich-Levin Lemma, which is useful for building PKE from TDF.

Lemma 2.4 (Goldreich-Levin hardcore)**.** *Let $\ell : \mathbb{N} \to \mathbb{N}$ be a length function and let $\mathcal{F} = (\mathcal{F}_\kappa)_{\kappa \in \mathbb{N}}$ be a keyed function where $\mathcal{F}_\kappa = \left\{f_K : \mathcal{X}_\kappa \to \{0,1\}^*\right\}_{K \in \mathcal{K}_\kappa}$ and $\mathcal{X}_\kappa$ is a subset of $\{0,1\}^{\ell(\kappa)}$. If $\mathcal{F}$ is one-way, i.e., for every PPT algorithm $\mathcal{I}$,*

$$\mathbf{Adv}^{ow}_{\mathcal{F},\mathcal{I}}(\kappa) \stackrel{\text{def}}{=} \Pr_{K \leftarrow \mathcal{K}_\kappa, x \leftarrow \mathcal{X}_\kappa}\left[f_K(\mathcal{I}(f_K(x))) = f_K(x)\right] = \mathtt{negl}(\kappa),$$

then for every PPT adversary $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathcal{A}(1^\kappa, K, f_K(x), r, \langle x, r\rangle) = 1\right] - \Pr\left[\mathcal{A}(1^\kappa, K, f_K(x), r, b) = 1\right]\right| = \mathtt{negl}(\kappa),$$

where $K \leftarrow \mathcal{K}_\kappa, x \leftarrow \mathcal{X}_\kappa, r \leftarrow \{0,1\}^{\ell(\kappa)}, b \leftarrow \{0,1\}$.

Remark 2.5. The domain of f_K could be arbitrary, as long as its elements can be encoded injectively into $\{0,1\}^\ell$, and both encoding and decoding are efficient.

2.2 Trapdoor Functions and Variants

When it comes to trapdoor functions and their variants, we always assume they are injective.

Trapdoor functions. An (injective) trapdoor function (TDF) is a collection $\mathcal{T} = \{T_\kappa\}_{\kappa\in\mathbb{N}}$ where each T_κ is probability distribution over a set of injective trapdoor functions indexed by the evaluation key ek. With input space $\mathtt{Dom}_{\mathsf{ek}}$ and output space $\mathtt{Im}_{\mathsf{ek}}$, an injective trapdoor function $\mathsf{TDF} = (\mathsf{Setup}, \mathsf{Samp}, \mathsf{Eval}, \mathsf{Inv})$ consists of four PPT algorithms with the following syntax.

- $\mathsf{Setup}(1^\kappa) \mapsto (\mathsf{ek}, \mathsf{td})$: The setup algorithm takes as input a security parameter κ and outputs an evaluation key ek and a trapdoor key td. The evaluation key ek is public and implicitly determines necessary public parameters $\mathtt{Dom}_{\mathsf{ek}}$ and $\mathtt{Im}_{\mathsf{ek}}$.
- $\mathsf{Samp}(\mathsf{ek}, 1^\kappa) \mapsto x$: The sample algorithm takes as input a security parameter κ and an evaluation key ek and outputs a uniformly distributed element x of $\mathtt{Dom}_{\mathsf{ek}}$.
- $\mathsf{Eval}(\mathsf{ek}, x) \mapsto y$: The evaluation algorithm takes as input a domain element $x \in \mathtt{Dom}_{\mathsf{ek}}$ and an evaluation key ek, and outputs $y \in \mathtt{Im}_{\mathsf{ek}}$.
- $\mathsf{Inv}(\mathsf{td}, y) \mapsto x\,/\perp$: The inversion algorithm takes as an image element $y \in \mathtt{Im}_{\mathsf{ek}}$ and a trapdoor td, and outputs x, which is either $\perp$ or an element of $\mathtt{Dom}_{\mathsf{ek}}$.

Definition 2.6 (ν-almost-all-keys perfect correctness)**.** We say TDF has *ν-almost-all-keys perfect correctness*, if there exists a negligible function $\nu(\cdot)$ such that for all $\kappa \in \mathbb{N}$,

$$\mathbf{Pr}\left[\exists\, x \in \mathtt{Dom}_{\mathsf{ek}} : \mathsf{Inv}(\mathsf{td}, \mathsf{Eval}(\mathsf{ek}, x)) \neq x \mid (\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\kappa);\ x^* \leftarrow \mathsf{Samp}(\mathsf{ek}, 1^\kappa)\right] \leq \nu(\kappa).$$

Definition 2.7 (One-wayness)**.** We say an injective TDF is *one-way* if for any PPT adversary $\mathcal{A}$, there exists a negligible $\mathtt{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$,

$$\mathbf{Adv}^{\mathsf{ow}}_{\mathsf{TDF},\mathcal{A}}(\kappa) = \mathbf{Pr}\left[x = x^* \;\middle|\; \begin{array}{c} (\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\kappa);\ x^* \leftarrow \mathsf{Samp}(\mathsf{ek}, 1^\kappa) \\ y^* \leftarrow \mathsf{Eval}(\mathsf{ek}, x^*);\ x \leftarrow \mathcal{A}(\mathsf{ek}, y^*) \end{array}\right] \leq \mathtt{negl}(1^\kappa).$$

Compared to traditional TDFs, adaptive trapdoor functions (ATDFs) require stronger security, which demands one-wayness holds *even when* the adversary may query an inverse oracle on images except for the challenge image.

α-almost-all-keys ν-correctness ([DNR04]). We say TDF has α-almost-all-keys ν-correctness, if for all $\kappa \in \mathbb{N}$,

$$\mathop{\mathbf{Pr}}_{(\mathsf{ek},\mathsf{td})\leftarrow\mathsf{Setup}(1^\kappa)}\left[\mathbf{Pr}\left[\mathsf{Inv}(\mathsf{td}, \mathsf{Eval}(\mathsf{ek}, x)) \neq x \mid\ x^* \leftarrow \mathsf{Samp}(\mathsf{ek}, 1^\kappa)\right] \geq \nu(\kappa)\right] \leq \alpha(\kappa).$$

Definition 2.8 (Adaptive one-wayness)**.** We say TDF is *adaptively one-way* if for any PPT adversary $\mathcal{A}$, there exists a negligible $\mathtt{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$,

$$\mathbf{Adv}^{\mathsf{aow}}_{\mathsf{TDF},\mathcal{A}}(\kappa) = \mathbf{Pr}\left[x = x^* \;\middle|\; \begin{array}{c} (\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\kappa);\ x^* \leftarrow \mathsf{Samp}(\mathsf{ek}, 1^\kappa) \\ y^* \leftarrow \mathsf{Eval}(\mathsf{ek}, x^*);\ x \leftarrow \mathcal{A}^{\mathsf{Inv}(\mathsf{td},\cdot)}(\mathsf{ek}, y^*) \end{array}\right] \leq \mathtt{negl}(\kappa).$$

where we demand that $\mathcal{A}$ does not make a query $\mathsf{Inv}(\mathsf{td}, y^*)$ to its oracle.

Tag-based adaptive trapdoor functions. A tag-based adaptive trapdoor function (TB-ATDF) extends the notion of ATDF by incorporating a tag space $\{0,1\}^t$ where t is the length function of the tag. With input space $\mathtt{Dom}_{\mathsf{ek}}$ and output space $\mathtt{Im}_{\mathsf{ek}}$, a tag-based adaptive trapdoor function $\mathsf{tATDF} = (\mathsf{Setup}, \mathsf{Samp}, \mathsf{Eval}, \mathsf{Inv})$ consists of four PPT algorithms with the following syntax.

- $\mathsf{tSetup}(T, 1^\kappa) \mapsto (\mathsf{ek}, \mathsf{td})$: The setup algorithm takes as input a security parameter κ and a tag $T \in \{0,1\}^t$, and outputs an evaluation key ek and a trapdoor key td. The evaluation key ek is public and implicitly determines necessary public parameters, such as $\mathtt{Dom}_{\mathsf{ek}}$ and $\mathtt{Im}_{\mathsf{ek}}$.
- $\mathsf{Samp}(\mathsf{ek}, 1^\kappa) \mapsto x$: The sample algorithm takes as input a security parameter κ and an evaluation key ek and outputs a uniformly distributed element x of $\mathtt{Dom}_{\mathsf{ek}}$.
- $\mathsf{tEval}(T, \mathsf{ek}, x) \mapsto y$: The evaluation algorithm takes as input a tag $T \in \{0,1\}^t$, a domain element $x \in \mathtt{Dom}_{\mathsf{ek}}$ and an evaluation key ek, and outputs $y \in \mathtt{Im}_{\mathsf{ek}}$.
- $\mathsf{tInv}(T, \mathsf{td}, y) \mapsto x\ /\bot$: The inversion algorithm takes as input a tag $T \in \{0,1\}^t$, $y \in \mathtt{Im}_{\mathsf{ek}}$ and a trapdoor td, and outputs x, which is either $\bot$ or an element of $\mathtt{Dom}_{\mathsf{ek}}$.

α-almost-all-keys ν-correctness. We say a tATDF has *α-almost-all-keys ν-correctness*, if for all $\kappa \in \mathbb{N}$, $T \in \{0,1\}^t$, it holds that

$$\Pr_{(\mathsf{ek},\mathsf{td})\leftarrow\mathsf{Setup}(1^\kappa)}\left[\Pr\left[\mathsf{Inv}(T, \mathsf{td}, \mathsf{tEval}(T, \mathsf{ek}, x)) \neq x \mid\ x^* \leftarrow \mathsf{Samp}(\mathsf{ek}, 1^\kappa)\right] \geq \nu(\kappa)\right] \leq \alpha(\kappa)$$

Definition 2.9 (Tag-based adaptive one-wayness)**.** We say a tATDF has *(tag-based) adaptive one-wayness* if for any PPT adversary $\mathcal{A}$, it holds that

$$\mathbf{Adv}^{\mathsf{taow}}_{\mathsf{tATDF},\mathcal{A}}(\kappa) = \mathbf{Pr}\left[x = x^* \;\middle|\; \begin{array}{c}(T^*, st) \leftarrow \mathcal{A}_1(1^\kappa);\ (\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{tSetup}(T^*, 1^\kappa);\\ x^* \leftarrow \mathsf{Samp}(\mathsf{ek}, 1^\kappa);\ y^* \leftarrow \mathsf{tEval}(T^*, \mathsf{ek}, x^*);\\ x \leftarrow \mathcal{A}_2^{\mathsf{tInv}(\cdot,\mathsf{td},\cdot)}(\mathsf{ek}, t^*, y^*, st)\end{array}\right] = \mathtt{negl}(1^\kappa).$$

where we demand that $\mathcal{A}_2$ does not make any query of the form $\mathsf{tInv}(t^*, \mathsf{td}, \cdot)$ to the oracle.

2.3 RR-PKE from Injective TDFs

The following construction is essentially the IND-CPA secure PKE construction in [Yao82].

Construction 2.10. Let TDF be an injective TDF associated with Goldreich-Levin hardcore $h_{\mathsf{ek}} : \mathtt{Dom}_{\mathsf{ek}} \to \{0,1\}$ (see lemma 2.4).

- $\mathsf{Gen}(1^\kappa) \mapsto (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm chooses $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\kappa)$. The public key is set to be $\mathsf{pk} := \mathsf{ek}$, and the secret key is $\mathsf{sk} := \mathsf{td}$.
- $\mathsf{Enc}(\mathsf{pk} = (\mathsf{ek}, x), m = (m_1, \cdots, m_{\ell_{\mathsf{msg}}})) \mapsto \mathsf{ct}$: For each $i \in [\ell_{\mathsf{msg}}]$, the encryption algorithm proceeds as follows.
  - chooses a random string $x_i \leftarrow \mathsf{Samp}(1^\kappa, \mathsf{ek})$.
  - sets $\mathsf{ct}_{1,i} = h(x_i) \oplus m_i$ and $\mathsf{ct}_{2,i} = \mathsf{Eval}(\mathsf{ek}, x_i)$.

  Let $\mathbf{ct}_b = (\mathsf{ct}_{b,1}, \cdots, \mathsf{ct}_{b,\ell_{\ell_{\mathsf{msg}}}})$ for $b \in \{0,1\}$, and outputs $(\mathbf{ct}_1, \mathbf{ct}_2)$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (\mathbf{ct}_1, \mathbf{ct}_2)) \mapsto (m, r)$: For each $i \in [\ell_{\mathsf{msg}}]$, the decryption algorithm computes $x_i = \mathsf{Inv}(\mathsf{td}, \mathsf{ct}_{2,i})$; if $x_i =\perp$, it outputs $\perp$ and aborts; otherwise, it sets $m_i = \mathsf{ct}_{1,i} \oplus h(x_i)$. Finally, it outputs $m = (m_1, \cdots, m_{\ell_{\mathsf{msg}}})$ and $r = (x_1, \ldots, x_{\ell_{\mathsf{msg}}})$.

- $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct} = (\mathbf{ct}_1, \mathbf{ct}_2), r = (x_1, \cdots, x_{\ell_{\mathsf{msg}}})) \mapsto z\ /\perp$: The recovery algorithm performs the following subroutine for each $i \in [\ell_{\mathsf{msg}}]$:

  - $y_i := \mathsf{Eval}(\mathsf{pk}, x_i)$.
  - If $y_i \neq \mathsf{ct}_{2,i}$, outputs $\perp$ and aborts; otherwise sets $m_i = \mathsf{ct}_{1,i} \oplus h(x_i)$.

  Finally, it outputs $m = (m_1, \cdots, m_{\ell_{\mathsf{msg}}})$.

If TDF is a canonical TDF, the above construction satisfies the following properties.

- Almost-all-keys perfect correctness and IND-CPA security.

- It has uniform ciphertext for random message due to the injectivity of TDF.

- It also has a key-independent randomness space, which is an Abelian group. [1]

3 Tagged Set Commitment with Randomness Opening

Let $\ell_\sigma : \mathbb{N} \to \mathbb{N}$ be a length function. A tagged set commitment with randomness opening and opening space $\{0,1\}^{\ell_\sigma}$ is a tuple of four algorithms $\mathsf{TSC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify}, \mathsf{AltSetup})$ with the following syntax.

- $\mathsf{Setup}(1^\kappa, 1^N, 1^B, 1^t) \mapsto \mathsf{pp}$: The setup algorithm takes as input the security parameter κ, the universe size N, bound B on committed sets and tag length t; it outputs public parameters pp.

- $\mathsf{Commit}(\mathsf{pp}, S, T \in \{0,1\}^t, (\sigma_i)_{i\in S}) \mapsto \mathsf{com}$: The commit algorithm takes as input the public parameter pp, a subset $S \subset [N]$ of size B, a tag T, and randomness $(\sigma_i)_{i\in S}$; it *deterministically* outputs a commitemnt com. Each $\sigma_i \in \{0,1\}^{\ell_\sigma}$ will be used as opening for $i \in S$.

- $\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, i, \sigma_i, T) \mapsto 0/1$: The verification algorithm takes as input the public parameters, an index $i \in [N]$, an opening $\sigma_i \in \{0,1\}^{\ell_\sigma}$, and a tag $T \in \{0,1\}^t$; it outputs 1 to indicate acceptance and 0 otherwise.

- $\mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T, (\sigma_i)_{i\in[N]}) \mapsto (\mathsf{pp}, \mathsf{com})$: The alternative setup algorithm takes as input the security parameter κ, a set S of size B, tag $T \in \{0,1\}^t$, openings $(\sigma_i)_{i\in N}$; it (could use additional randomness and) outputs public parameters pp and a commitment com.

Remark 3.1. ℓ_σ, the length of openings, is independent of the parameters N, B, t and only depends on the security parameter.

[1] This is because Goldreich-Levin hardcore only adds uniform random bits to the random coins, and we assumed the domain of TDF is an Abelian group that does not depend on the evaluation key.

Correctness. A TSC scheme should satisfy the following correctness requirements.

1. Correctness of Setup and Commit: For all $\kappa, N, t \in \mathbb{N}$, $B \le N$, $T \in \{0,1\}^t$ and set $S \subset [N]$ of size B, it holds that for all $i \in S$,

$$\mathbf{Pr}\left[\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, i, \sigma_i, T) = 1 \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^N, 1^B, 1^t);\ (\sigma_i)_{i \in S} \leftarrow (\{0,1\}^{\ell_\sigma})^B \\ \mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{pp}, S, T, (\sigma_i)_{i \in S}) \end{array}\right] = 1.$$

2. Correctness of AltSetup: For all $\kappa, N, t \in \mathbb{N}$, $B \le N$ and $T \in \{0,1\}^t$, it holds that for any $i \in [N]$:

$$\mathbf{Pr}\left[\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, i, \sigma_i, T) = 1 \;\middle|\; \begin{array}{c} (\sigma_i)_{i \in [N]} \leftarrow (\{0,1\}^{\ell_\sigma})^N; \\ (\mathsf{pp}, \mathsf{com}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T, (\sigma_i)_{i \in [N]}) \end{array}\right] = 1.$$

Remark 3.2. The above two correctness requirements define perfect correctness. Similarly, we can define almost-all-keys perfect correctness for TSC as well. A TSC with almost-all-keys perfect correctness can be easily transformed into one with perfect correctness: The commit algorithm can check whether each commitment verifies using the public verification algorithm, as noted in [HKW20]. If the check fails, the commitment algorithm can *fall back to a trivial scheme* that is perfectly correct, but has no security guarantee. This fallback mechanism only adds an extra negligible probability to the adversary's advantage. Therefore, without loss of generality, adopt perfect correctness in the definition above.

TSC should satisfy the following two security properties.

Soundness. A tagged set commitment scheme TSC = (Setup, Commit, Verify, AltSetup) enjoys *soundness* if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathtt{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$,

$$\mathbf{Pr}\left[\begin{array}{c} T' \neq T \wedge S' \subset [N] \wedge |S'| > B \\ \forall\, i \in S' : \\ \mathsf{Verify}(\mathsf{pp}, \mathsf{com}', i, \sigma_i', T') = 1 \end{array} \;\middle|\; \begin{array}{c} (1^N, 1^B, 1^t, T, st) \leftarrow \mathcal{A}_1(1^\kappa),\ \text{s.t. } B \le N, T \in \{0,1\}^t \\ (\sigma_i)_{i \in [N]} \leftarrow (\{0,1\}^{\ell_\sigma})^N \\ (\mathsf{pp}, \mathsf{com}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T, (\sigma_i)_{i \in [N]}) \\ (T', S', (\sigma_i)_{i \in S'}) \leftarrow \mathcal{A}_2(st, \mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in [N]}) \end{array}\right] \le \mathtt{negl}(\kappa).$$

Adaptive indistinguishability of Setup**.** A tagged set commitment scheme TSC satisfies *adaptive indistinguishability of setup* if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathtt{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$,

$$\mathbf{Adv}^{\mathsf{ind\text{-}setup}}_{\mathsf{TSC},\mathcal{A}}(\kappa) \stackrel{\text{def}}{=} \left|\mathbf{Pr}\left[\mathsf{Expr}^{\mathsf{ind\text{-}setup},0}_{\mathsf{TSC},\mathcal{A}}(\kappa) \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Expr}^{\mathsf{ind\text{-}setup},1}_{\mathsf{TSC},\mathcal{A}}(\kappa) \Rightarrow 1\right]\right| \le \mathtt{negl}(\kappa),$$

where $\mathsf{Expr}^{\mathsf{ind\text{-}setup},b}_{\mathsf{TSC},\mathcal{A}}(\kappa)$ is defined as follows:

1. On input 1^κ, $\mathcal{A}$ sends $(1^N, 1^B, 1^t, S)$ with $B \le N$ and $|S| = B$ to the challenger.
2. Challenger samples $(\sigma_i)_{i \in [N]} \leftarrow (\{0,1\}^{\ell_\sigma})^N$ and sends $(\sigma_i)_{i \in S}$ to $\mathcal{A}$.
3. $\mathcal{A}$ receives $(\sigma_i)_{i \in S}$ and sends a tag $T \in \{0,1\}^t$ to the challenger.
4. Challenger proceeds according to b:

- $b = 0$: Compute $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^N, 1^B, 1^t)$ and $\mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{pp}, S, T, (\sigma_i)_{i\in S})$.
- $b = 1$: Run $(\mathsf{pp}, \mathsf{com}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T, (\sigma_i)_{i\in[N]})$.

5. Challenger sends $(\mathsf{pp}, \mathsf{com})$ to $\mathcal{A}$.
6. $\mathcal{A}$ outputs its guess b'; the experiment outputs b'.

Our definition strengthens the indistinguishability of Setup defined in [HKW20] by adding a level of adaptivity: The challenger chooses the challenge tag T *after seeing openings* $(\sigma_i)_{i\in S}$.

3.1 Construction from PRG

The scheme in [HKW20], as presented below, can be adapted to fit into the syntax of TSC with randomness opening.

Construction 3.3 (TSC with randomness opening from PRG)**.** Let $\mathsf{PRG} : (\{0,1\}^\kappa, 1^\ell) \to \mathbb{F}_{2^\ell}$ be a pseudorandom generator. Let emb be an injective and efficiently-computable function that maps strings in $\{0,1\}^\ell$ (tags) to elements in $\mathbb{F}_{2^\ell}$. Below the notation $p \leftarrow \mathbb{F}_{2^\ell}[x]^{B-1}$ means that p is set to be a random degree $B-1$ polynomial over variable x, where p is represented in canonical form with B randomly chosen coefficients in $\mathbb{F}_{2^\ell}$.

- $\mathsf{Setup}(1^\kappa, 1^N, 1^B, 1^t) \mapsto \mathsf{pp}$: The setup algorithm sets $\ell = 2t + (B+1)\cdot \log N + \kappa\cdot(B+1)+\kappa$, and chooses N random elements $A_i, D_i \leftarrow \mathbb{F}_{2^\ell}$ for all $i \in [N]$. The public parameters is set to be $\mathsf{pp} = (1^\ell, (A_i, D_i)_{i\in[N]})$.

- $\mathsf{Commit}(\mathsf{pp} = (1^\ell, (A_i, D_i)_{i\in[N]}), S \subset [N], T, (\sigma_i)_{i\in S}) \mapsto \mathsf{com}$: The commitment algorithm first chooses the degree $B-1$ degree polynomial $p(\cdot)$ over $\mathbb{F}_{2^\ell}$ such that for all $i \in S$, $p(i) = \mathsf{PRG}(\sigma_i, 1^\ell) + A_i + D_i \cdot \mathsf{emb}(T)$. The commitment com is the polynomial p, and $(\sigma_i)_{i\in S}$ is the opening proofs for the set S.

- $\mathsf{Verify}(\mathsf{pp} = (1^\ell, (A_i, D_i)_{i\in[N]}), \mathsf{com} = p, i, \sigma_i, T) \mapsto \{0,1\}$: The verification algorithm outputs 1 iff $p(i) = \mathsf{PRG}(\sigma_i, 1^\ell) + A_i + D_i\cdot\mathsf{emb}(T)$.

- $\mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T, (\sigma_i)_{i\in[N]}) \mapsto (\mathsf{pp}, \mathsf{com})$: The alternative setup algorithm chooses random strings $s_i \leftarrow \{0,1\}^\kappa, D_i \leftarrow \mathbb{F}_{2^\ell}$ for each $i \in [N], p \leftarrow \mathbb{F}_{2^\ell}[x]^{B-1}$ and sets $A_i = p(i) - \mathsf{PRG}(\sigma_i, 1^\ell) - D_i\cdot\mathsf{emb}(T)$.

The above construction satisfies *correctness* and *soundness*; for detailed proofs, we refer the reader to [HKW20]. In the rest of this section, we prove that it also satisfies our notion of adaptive indistinguishability of Setup.

Theorem 3.4. *If* $\mathsf{PRG} : (\{0,1\}^\kappa, 1^\ell) \to \mathbb{F}_{2^\ell}$ *is a pseudorandom generator, then construction 3.3 satisfies adaptive indistinguishability of* Setup.

Proof. Let TSC denote the scheme in construction 3.3. Let $\mathcal{A}$ be an adversary attacking the adaptive indistinguishability of Setup of TSC. We shall devise an adversary $\mathcal{B}$ such that

$$\left|\mathbf{Pr}\left[\mathsf{Expr}^{\mathsf{prg},0}_{\mathsf{PRG},\mathcal{B}}(\kappa) \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Expr}^{\mathsf{prg},1}_{\mathsf{PRG},\mathcal{B}}(\kappa) \Rightarrow 1\right]\right| \geq \mathbf{Adv}^{\mathsf{ind\text{-}setup}}_{\mathsf{TSC},\mathcal{A}}(\kappa), \tag{1}$$

where the experiment $\mathsf{Expr}^{\mathsf{prg},b}_{\mathsf{PRG},\mathcal{B}}(\kappa)$ is the following experiment:

1. On input 1^κ, $\mathcal{B}$ sends a number Q and ℓ to the challenger.

2. Challenger proceeds according to $b \in \{0,1\}$:

   - if $b = 0$, it samples $a_i \leftarrow \mathbb{F}_{2^\ell}$ for $i \in [Q]$, and sends $(a_i)_{i\in[Q]}$ to $\mathcal{B}$.
   - if $b = 1$, it samples $s_i \leftarrow \{0,1\}^\kappa$ for $i \in [Q]$, and sends $(a_i)_{i\in[Q]}$ to $\mathcal{B}$, where $a_i = \mathsf{PRG}(s_i, 1^\ell)$.

3. $\mathcal{B}$ outputs a bit $b' \in \{0,1\}$ and the experiment outputs b'.

It is easy to see that the above experiment captures the pseudorandomness of PRG.

Consider the following adversary $\mathcal{B}$:

1. $\mathcal{B}$ sends 1^κ to $\mathcal{A}$.

2. On receiving $(1^N, 1^B, 1^t, S)$ from $\mathcal{A}$ with $B \leq N$ and $|S| = B$ from $\mathcal{A}$, it sets $Q := N - B$ and $\ell := 2t + (B+1)\cdot \log N + \kappa \cdot (B+1) + \kappa$ and submit (Q, ℓ) to the challenger.

3. On receiving $(a_i)_{i\in[Q]}$ from the challenger, $\mathcal{B}$ samples $\sigma_i \leftarrow \{0,1\}^{\ell_\sigma}$ for $i \in S$, and sends $(\sigma_i)_{i\in S}$ to $\mathcal{A}$; $\mathcal{B}$ also renames $(a_i)_{i\in[Q]}$ as $(d_i)_{i\in[N]\setminus S}$. (Note that $|[N] \setminus S| = Q$.)

4. On receiving T from $\mathcal{A}$, $\mathcal{B}$ chooses a degree-$(B-1)$ polynomial $p \leftarrow \mathbb{F}_{2^\ell}[X]^{B-1}$ uniformly at random and samples $D_i \leftarrow \mathbb{F}_{2^\ell}$ for $i \in [N]$; then it sets $A_i := p(i) - \mathsf{PRG}(\sigma_i, 1^\ell) - D_i \cdot \mathsf{emb}(T)$ for $i \in S$, and $A_i := p(i) - d_i - D_i \cdot \mathsf{emb}(T)$ for $i \in [N] \setminus S$. $\mathcal{B}$ sends $(\mathsf{pp} = (1^\ell, (A_i, D_i)_{i\in[N]}), \mathsf{com} = p)$ to $\mathcal{A}$.

5. Finally, $\mathcal{A}$ outputs a bit b' and $\mathcal{B}$ outputs b'.

For $b \in \{0,1\}$, if $\mathcal{B}$ is in the experiment $\mathsf{Expr}^{\mathsf{prg},b}_{\mathsf{PRG},\mathcal{B}}(\kappa)$, it perfectly simulates $\mathsf{Expr}^{\mathsf{ind\text{-}setup},b}_{\mathsf{TSC},\mathcal{A}}(\kappa)$ for $\mathcal{A}$. This finishes the proof. □

4 TB-ATDFs from Canonical TDFs

This section presents our construction of TB-ATDF.

4.1 Construction

Our TB-ATDF uses building blocks TSC, PKE with the following properties.

1. TSC is a TSC with randomness opening; let $\ell_\sigma = \ell_\sigma(\kappa)$ denote the length of the openings.

2. $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec})$ is a randomness-recoverable PKE with message space $\{0,1\}^{\ell_{\mathsf{msg}}}$, where $\ell_{\mathsf{msg}} \stackrel{\mathsf{def}}{=} \kappa + \ell_\sigma$. We require PKE to have a key-independent randomness space $\mathsf{Rnd} = (\mathsf{Rnd}_\kappa)_{\kappa\in\mathbb{N}}$ that does not depend on the public key, and each Rnd_κ is an Abelian group.

Construction 4.1 (TB-ATDF)**.** Let ℓ_{tag} be the length of tags. Choose parameters $N = N(\kappa), B = B(\kappa)$ used for TSC such that $\binom{N}{B} \geq |\mathsf{Rnd}| \cdot 2^\kappa$. We construct a TB-ATDF with tag space $\{0,1\}^{\ell_{\mathsf{tag}}}$ as follows.

- $\mathsf{tSetup}(T \in \{0,1\}^{\ell_{\mathsf{tag}}}, 1^\kappa) \mapsto (\mathsf{ek}, \mathsf{td})$:

  1. $\mathsf{pp} \leftarrow \mathsf{TSC.Setup}(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}})$.

Check

- **Hardwired**: $\mathsf{ek}, T, y = (\mathsf{com}, (\mathsf{ct}_j)_{j\in[N]})$.
- **Input:** $i \in [N], z \in (\{0,1\}^\kappa \times \{0,1\}^{\ell_{\mathsf{msg}}}) \cup \{\perp\}, r \in \mathsf{Rnd}$.

Output 1 if and only if the following conditions are satisfied:

(a) $z \neq \perp$. Parse $z = g\|\sigma$ where $g \in \{0,1\}^\kappa$.

(b) $g = 1^\kappa$.

(c) $\mathsf{TSC.Verify}(\mathsf{pp}, \mathsf{com}, i, \sigma, T) = 1$.

(d) $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, \sigma; r)$.

Figure 2: Subroutine $\mathsf{Check}(i, z, r)$

2. Generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ for $i \in [N]$.
3. Return $\mathsf{ek} = \big(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\big)$ and $\mathsf{td} = (\mathsf{sk}_i)_{i\in[N]}$.

- $\mathsf{Samp}(\mathsf{ek}, 1^\kappa) \mapsto x$:
  1. Choose a size-B subset $S \subset [N]$ uniformly at random. Let $S = \{i_1, i_2, \cdots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$. Then sample $r_{i_j} \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$.
  2. $\sigma_i \leftarrow \{0,1\}^{\ell_\sigma}$ for $i \in S$.
  3. For $i \in [N] \setminus S$, $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, m_i)$ where $m_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
  4. Return $\Big(S, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in S}, (\mathsf{ct}_i)_{i\in[N]\setminus S}\Big)$.

- $\mathsf{tEval}(T, \mathsf{ek}, x) \mapsto y$:
  1. Parse $x = \Big(S, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in S}, (\mathsf{ct}_i)_{i\in[N]\setminus S}\Big)$, $\mathsf{ek} = \Big(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\Big)$.
  2. $\mathsf{com} \leftarrow \mathsf{TSC.Commit}(\mathsf{pp}, S, T, (\sigma_i)_{i\in S})$.
  3. Sets $r_{i_B} = -\sum_{j=1}^{B-1} r_{i_j}$.
  4. For $i \in S$, computes $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, 1^\kappa\|\sigma_i; r_i)$.
  5. Return $\Big(\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]}\Big)$.

- $\mathsf{tInv}(T, \mathsf{td}, y) \mapsto x\ /\perp$:
  1. Parse $\mathsf{td} = (\mathsf{sk}_i)_{i\in[N]}$ and $y = \Big(\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]}\Big)$.
  2. For each $i \in [N]$, $(z_i, r_i) := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$.
  3. Initialize a set $U := \emptyset$. For each $i \in [N]$, add i into U if $\mathsf{Check}(i, z_i, r_i) = 1$, where Check is defined in fig. 2.
  4. If the set $|U| \neq B$, output $\perp$.
  5. If $\sum_{i\in U} r_i \neq 0$, output $\perp$.
  6. For $i \in U$, parse $z_i = 1^\kappa\|\sigma_i$; let $U = \{i_1, \ldots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$.

7. Return $\left(U, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in U}, (\mathsf{ct}_i)_{i\in[N]\setminus U}\right)$.

Regarding the correctness and security of construction 4.1, we have the next two theorems.

Theorem 4.2 (Correctness)**.** *If* PKE *satisfies ε-almost-all-keys correctness, then construction 4.1 satisfies α-almost-all-keys ν-correctness, where $\alpha = N \cdot \varepsilon$ and $\nu = (N-B)\cdot 2^{-\kappa}$.*

Theorem 4.3 (Adaptive one-wayness)**.** *Assume* TSC *is a TSC with randomness opening, and* PKE *is an RR-PKE with (i) almost-all-keys perfect correctness and (ii) uniform ciphertext for random message. Then construction 4.1 satisfies adaptive one-wayness.*

We prove theorem 4.2 and theorem 4.3 in section 4.2 and section 4.3 respectively.

Uniform domain sampling. Assume that PKE has uniform ciphertext for random message. For a perfectly correct key-pair $(\mathsf{pk}, \mathsf{sk})$, we have $|C_{\mathsf{pk}}| = 2^{\ell_{\mathsf{msg}}} \cdot |\mathsf{Rnd}_\kappa|$, which is independent of pk. If all N key pairs are perfectly correct, each input is sampled by Samp with equal probability, namely,

$$\frac{1}{\binom{N}{B}} \cdot \frac{1}{(|\mathsf{Rnd}_\kappa|)^{B-1}} \cdot (2^{-\ell_\sigma})^B \cdot \frac{1}{(2^{\ell_{\mathsf{msg}}} \cdot |\mathsf{Rnd}_\kappa|)^{N-B}}.$$

Thus, we achieve uniform domain sampling (with overwhelming probability over the generation of $(\mathsf{ek}, \mathsf{td})$).

4.2 Proof of Theorem 4.2: Correctness

Let $\mathtt{GoodKey}$ be the event that all N key pairs are error-free. By the ε-almost-all-keys perfect correctness of PKE and union bound, we have

$$\mathbf{Pr}\left[\mathtt{GoodKey}\right] \geq 1 - N \cdot \varepsilon.$$

Conditioned on $\mathtt{GoodKey}$, the decryption errors are solely owing to the bad $y = (\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})$ evaluation where

- x is sampled uniformly from $\mathtt{Dom}$ and $y = \mathsf{Eval}(\mathsf{ek}, x)$.
- There exists $i \in [N] \setminus S$, $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, m_i; r_i)$ such that $\mathsf{Check}(i, m_i, r_i) = 1$.

We use $\mathtt{GoodMsg}$ to denote that the above bad y evaluation does not happen. Observe that $(m_i)_{i\in[N]\setminus S}$ are chosen uniformly at random. The first κ-bits of m_i happens to be κ-bits ones with probability $2^{-\kappa}$, which is a sub-check in Check. Then by union bound, we have

$$\mathbf{Pr}\left[\mathtt{GoodMsg}\right] \geq 1 - (N-B)\cdot 2^{-\kappa}.$$

Therefore, the theorem theorem 4.2 holds with $\alpha = N \cdot \varepsilon$ and $\nu = (N-B)\cdot 2^{-\kappa}$.

4.3 Proof of Theorem 4.3: Adaptive One-Wayness

Let tATDF denote the TB-ATDF in construction 4.1 and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary that aims at attacking the adaptive one-wayness of tATDF. The proof proceeds via a sequence of security games Game_i ($i \in [3]$), where Game_1 is exactly the adaptive one-wayness experiment. Therefore,

$$\begin{aligned}\mathbf{Adv}^{\mathsf{taow}}_{\mathsf{tATDF},\mathcal{A}}(\kappa) &= \mathbf{Pr}\left[\mathsf{Game}_1 \Rightarrow 1\right] \\ &\leq \mathbf{Pr}\left[\mathsf{Game}_3 \Rightarrow 1\right] + \sum_{j\in[2]} \left|\mathbf{Pr}\left[\mathsf{Game}_i \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_{i+1} \Rightarrow 1\right]\right|. \end{aligned} \tag{2}$$

We finish the proof of theorem 4.3 by showing that all terms in eq. (2) are negligible in the following lemmas.

Game_1. This game corresponds to the adaptive one-wayness experiment.

- Choosing the challenge tag: The adversary $\mathcal{A}_1$ sends a tag T^* to the challenger as the challenge tag, and maintains an internal state st for $\mathcal{A}_2$, i.e., $(T^*, st) \leftarrow \mathcal{A}_1(1^\kappa)$.
- Generating the challenge image: In this stage, the challenger runs the setup, sampling, and evaluation algorithms of the TB-ATDF scheme.
  1. $\mathsf{pp} \leftarrow \mathsf{TSC.Setup}(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}})$.
  2. For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$
  3. $\mathsf{ek} := \left(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\right)$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
  4. Choose $S^* = \{i_1, \ldots, i_B\} \subset [N]$ uniformly at random where $i_1 < \cdots < i_B$.
  5. Sample $r^*_{i_j} \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r^*_{i_B} := -\sum_{j=1}^{B-1} r^*_{i_j}$.
  6. For $i \in S^*$, choose $\sigma^*_i \leftarrow \{0,1\}^{\ell_\sigma}$ and $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, 1^\kappa \| \sigma^*_i; r^*_i)$; for $i \in [N] \setminus S^*$, $\mathsf{ct}^*_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m^*_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
  7. $\mathsf{com}^* := \mathsf{Commit}(\mathsf{pp}, S^*, T^*, (\sigma^*_i)_{i\in S^*})$.
  8. $x^* := \left(S^*, (r^*_{i_j})_{j\in[B-1]}, (\sigma^*_i)_{i\in S^*}, (\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}\right), y^* := \left(\mathsf{com}^*, (\mathsf{ct}^*_i)_{i\in[N]}\right)$.
- Answering inversion queries: In this stage, the challenger runs $\mathcal{A}_2(st, y^*)$ and each inversion query $y = (\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})$ is answered as follows:
  1. Compute $(z_i, r_i) := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ for all $i \in [N]$.
  2. Initialize $U = \emptyset$; for $i \in [N]$, add i to U if $\mathsf{Check}(i, z_i, r_i) = 1$.
  3. If $|U| \neq B$ or $\sum_{i\in U} r_i \neq 0$, return $\perp$.
  4. For each $i \in U$, parse $z_i = 1^\kappa | \sigma_i$; let $U = \{i_1, \ldots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$.
  5. Return $\left(U, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in U}, (\mathsf{ct}_i)_{i\in[N]\setminus U}\right)$.
- Deciding winning condition: Finally, $\mathcal{A}_2$ outputs x' and the game outputs 1 if and only if $x' = x^*$.

Game_2. This game is identical to Game_1 except that the challenger runs AltSetup rather than Setup during the stage of generating the challenge image. Moreover, it puts the index opening proofs and commitment generated by AltSetup into x^*.

- Generating the challenge image:

  1. For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$
  2. Choose $S^* = \{i_1, \ldots, i_B\} \subset [N]$ uniformly at random where $i_1 < \cdots < i_B$.
  3. Sample $r^*_{i_j} \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r^*_{i_B} := -\sum_{j=1}^{B-1} r^*_{i_j}$.
  4. For $i \in [N]$, choose $\sigma^*_i \leftarrow \{0,1\}^{\ell_\sigma}$; for $i \in S^*$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma^*_i; r^*_i)$; for $i \in [N] \setminus S^*$, $\mathsf{ct}^*_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m^*_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
  5. $(\mathsf{pp}, \mathsf{com}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T^*, (\sigma^*_i)_{i\in[N]})$.
  6. $\mathsf{ek} := (\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]})$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
  7. $x^* := \left(S^*, (r^*_{i_j})_{j\in[B-1]}, (\sigma^*_i)_{i\in S^*}, (\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}\right), y^* := (\mathsf{com}^*, (\mathsf{ct}^*_i)_{i\in[N]})$.

Game_3. This game is identical to Game_2 except that for $i \in [N]\setminus S^*$, ct^*_i is switched to an encryption of σ^*_i during the stage of enerating the challenge image. That is, we additionally pick $r^*_i \leftarrow \mathsf{Rnd}$ for $i \in [N] \setminus S^*$ and generate $(\mathsf{ct}^*_i)_{i\in[N]}$ as follows: for $i \in [N]$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma^*_i; r^*_i)$.

- Generating the challenge image:

  1. For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$
  2. Choose $S^* = \{i_1, \ldots, i_B\} \subset [N]$ uniformly at random where $i_1 < \cdots < i_B$.
  3. Sample $r^*_{i_j} \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r^*_{i_B} := -\sum_{j=1}^{B-1} r^*_{i_j}$; sample $r^*_i \leftarrow \mathsf{Rnd}$ for $i \in [N]\setminus S^*$.
  4. For $i \in [N]$, choose $\sigma^*_i \leftarrow \{0,1\}^{\ell_\sigma}$; for $i \in [N]$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma^*_i; r^*_i)$.
  5. $(\mathsf{pp}, \mathsf{com}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T^*, (\sigma^*_i)_{i\in[N]})$.
  6. $\mathsf{ek} := (\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]})$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
  7. $x^* := \left(S^*, (r^*_{i_j})_{j\in[B-1]}, (\sigma^*_i)_{i\in S^*}, (\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}\right), y^* := (\mathsf{com}^*, (\mathsf{ct}^*_i)_{i\in[N]})$.

We first show that $\mathcal{A}$ has negligible advantage in Game_3:

Lemma 4.4. $\mathbf{Pr}[\mathsf{Game}_3 \Rightarrow 1] \leq \frac{|\mathsf{Rnd}|}{\binom{N}{B}} \leq 2^{-\kappa}$.

Proof. We define an auxiliary game Game_4: This game is identical Game_3 except that we choose $(r^*_i)_{i\in S^*}$ uniformly at random and use them to generate the challenge image. At this point, y^* contains no information about S^*; however, for Game_4 to output 1, $\mathcal{A}$ must guess S^* correctly, as S^* is part of x^*. Therefore, $\mathbf{Pr}[\mathsf{Game}_4 \Rightarrow 1] \leq \frac{1}{\binom{N}{B}}$.

It suffice to show that $\mathbf{Pr}[\mathsf{Game}_4 \Rightarrow 1] \geq 1/|\mathsf{Rnd}| \cdot \mathbf{Pr}[\mathsf{Game}_3 \Rightarrow 1]$ holds. Fix $S^* = \{i_1, i_2, \cdots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$. Since $r^*_{i_B}$ in Game_4 is sampled from Abelian group Rnd uniformly at random, it holds that

$$\mathop{\mathbf{Pr}}_{r^*_{i_1},\ldots,r^*_{i_B} \leftarrow \mathsf{Rnd}}\left[r^*_{i_B} = -\sum_{j=1}^{B-1} r^*_{i_j}\right] = 1/|\mathsf{Rnd}|.$$

In Game_4, if the condition $r^*_{i_B} = -\sum_{j=1}^{B-1} r^*_{i_j}$ holds, the adversary $\mathcal{A}$ behaves identically to its behavior in Game_3, i.e.,

$$\mathbf{Pr}\left[\mathsf{Game}_4 \Rightarrow 1 \;\middle|\; r^*_{i_B} = -\sum_{j=1}^{j=B-1} r^*_{i_j}\right] = \mathbf{Pr}\left[\mathsf{Game}_3 \Rightarrow 1\right].$$

Then lemma 4.4 follows from the law of total probability. □

Lemma 4.5. *There exists a PPT adversary $\mathcal{B}_1$ attacking the indistinguishability of* TSC*'s setup such that*

$$\left|\mathbf{Pr}\left[\mathsf{Game}_2 \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_1 \Rightarrow 1\right]\right| = \mathbf{Adv}^{\textit{ind-setup}}_{\mathsf{TSC},\mathcal{B}_1}(\kappa).$$

Proof. Consider the following adversary $\mathcal{B}_1$ attacking the adaptive indistinguishability of TSC's setup, where C_1 is the challenger in the experiment $\mathsf{Expr}^{\mathsf{ind\text{-}setup},b}_{\mathsf{TSC},\mathcal{B}_1}(\kappa)$.

1. $\mathcal{B}_1$ receives input $\left(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}}\right)$ and runs $(T^*, st) \leftarrow \mathcal{A}_1(1^\kappa, 1^{\ell_{\mathsf{tag}}})$ to obtain the challenge tag T^*. $\mathcal{B}_1$ samples $S^* = \{i_1, \ldots, i_B\} \leftarrow \binom{[N]}{B}$ where $i_1 < \cdots < i_B$, and sends $\left(1^N, 1^B, 1^{\ell_{\mathsf{tag}}}, S^*\right)$ to C_1.

2. On receiving $\left(1^N, 1^B, 1^{\ell_{\mathsf{tag}}}, S^*\right)$, C_1 samples $\sigma_i^{b*} \leftarrow \{0,1\}^{\ell_\sigma}$ for all $i \in [N]$ and sends $(\sigma_i^{b*})_{i\in S^*}$ to $\mathcal{B}_1$.

3. $\mathcal{B}_1$ receives $(\sigma_i^{b*})_{i\in S^*}$ and set T^* to C_1.

4. On receiving T^*, C_1 proceeds according to b:
   - If $b = 0$, C_1 runs $\mathsf{pp}^0 \leftarrow \mathsf{Setup}(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}})$ and computes
   $$\mathsf{com}^{0*} \leftarrow \mathsf{TSC.Commit}(\mathsf{pp}^0, S^*, T^*, (\sigma_i^{0*})_{i\in S})).$$
   - If $b = 1$, C_1 samples $\sigma_i^{1*} \leftarrow \{0,1\}^{\ell_\sigma}$ for $i \in [N]$ and computes
   $$(\mathsf{pp}^1, \mathsf{com}^{1*}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}}, T^*, (\sigma_i^{1*})_{i\in[N]}).$$

   Next, C_1 sends $\left(\mathsf{pp}^b, \mathsf{com}^{b*}\right)$ to $\mathcal{B}_1$.

5. $\mathcal{B}_1$ receives $\left(\mathsf{pp}^b, \mathsf{com}^{b*}\right)$, and computes x^* and y^* as follows:
   - For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$
   - Sample $r^*_{i_j} \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r^*_{i_B} := -\sum_{j=1}^{B-1} r^*_{i_j}$.
   - For $i \in S^*$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma_i^{b*}; r^*_i)$; for $i \in [N]\setminus S^*$, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m^*_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
   - $\mathsf{ek} := \left(\mathsf{pp}^b, (\mathsf{pk}_i)_{i\in[N]}\right)$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
   - $x^* := \left(S^*, (r^*_{i_j})_{j\in[B-1]}, (\sigma_i^{b*})_{i\in S^*}, (\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}\right), y^* := \left(\mathsf{com}^{b*}, (\mathsf{ct}^*_i)_{i\in[N]}\right)$.

6. $\mathcal{B}_1$ runs $\mathcal{A}_2(st, y^*)$ and uses all key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}$ as td to simulate the inversion oracle $\mathsf{Inv}(\cdot, \mathsf{td}, \cdot)$ for $\mathcal{A}_2(st, y^*)$. Finally, $\mathcal{B}_1$ obtains output x from $\mathcal{A}_2(y^*, st)$.

7. If $x = x^*$, $\mathcal{B}_1$ sends 1 to C_1; otherwise, $\mathcal{B}_1$ sends 0 to C_1.

It is easy to see that from the point of $\mathcal{A}$, if $b = 0$, $\mathcal{B}_1$ perfectly simulates Game_1; if $b = 1$, $\mathcal{B}_1$ perfectly simulates Game_2. This concludes the proof. □

Lemma 4.6. *There exists PPT adversaries $\mathcal{B}_2$ and $\mathcal{B}_3$ such that*

$$\begin{aligned}|\mathbf{Pr}[\mathsf{Game}_2 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_3 \Rightarrow 1]| = 2(N-B)\cdot \mathbf{Adv}^{\mathsf{sound}}_{\mathsf{TSC},\mathcal{B}_2}(\kappa) + 2(N-B)N\cdot\varepsilon(\kappa)\\ + (N-B)\cdot\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}_3}(\kappa).\end{aligned}$$

Proof. For ease of presentation, we define intermediate games $\mathsf{Game}_{2,j}$ for $j \in [N+1]$:

- $\mathsf{Game}_{2,j}$ is identical to Game_2 except that $(\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}$ is generated as follows: for $i \in [N]\setminus S^*$, if $i < j$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma^*_i; r^*_i)$; if $i \geq j$, $\mathsf{ct}^*_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m^*_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.

Clearly, $\mathsf{Game}_{2,1}$ is identical to Game_2, and $\mathsf{Game}_{2,N+1}$ is identical to Game_3. For $j \in [N]$, we further define intermediate games $\mathsf{Game}_{2,j,\mathsf{Alt},0}$ and $\mathsf{Game}_{2,j,\mathsf{Alt},1}$ to facilitate the switch from $\mathsf{Game}_{2,j}$ to $\mathsf{Game}_{2,j+1}$:

- $\mathsf{Game}_{2,j,\mathsf{Alt},0}$ is identical to $\mathsf{Game}_{2,j}$ except that the inversion oralce is replaced by $\textsc{AltInv}_{-j}$ defined in fig. 3, which only uses $(\mathsf{sk}_i)_{i\neq j}$.

- $\mathsf{Game}_{2,j,\mathsf{Alt},1}$ is identical to $\mathsf{Game}_{2,j,\mathsf{Alt},0}$ except that $(\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}$ is generated as follows: for $i \in [N]\setminus S^*$, if $i \leq j$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma^*_i; r^*_i)$; if $i > j$, $\mathsf{ct}^*_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.

A More detailed description of these intermediate games can be found in appendix A. In the following three lemmas (lemma 4.7, lemma 4.8, and lemma 4.9), we establish that

$$\mathsf{Game}_{2,j} \approx_c \mathsf{Game}_{2,j,\mathsf{Alt},0} \approx_c \mathsf{Game}_{2,j,\mathsf{Alt},1} \approx_c \mathsf{Game}_{2,j+1}$$

for all $j \in [N]$; and this finishes the proof of lemma 4.6. □

Lemma 4.7. *Suppsee that* PKE *satisfies $\varepsilon(\kappa)$-almost-all-keys perfect correctness. For $v \in [N]$, there exists a PPT adversary $\mathcal{B}_2$ against the soundness of* TSC *such that*

$$\left|\mathbf{Pr}\left[\mathsf{Game}_{2,v} \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1\right]\right| \leq \mathbf{Adv}^{\mathsf{sound}}_{\mathsf{TSC},\mathcal{B}_2}(\kappa) + N\cdot\varepsilon(\kappa).$$

Proof. In $\mathsf{Game}_{2,v}$, the challenger uses $\mathsf{td} = (\mathsf{sk}_i)_{i\in[N]}$ to simulate the inversion oracle Inv, while in $\mathsf{Game}_{2,v,\mathsf{Alt},0}$, the challenger uses $(\mathsf{sk}_i)_{i\neq v}$ to simulate the alternative inversion oracle $\textsc{AltInv}_{-v}$. We use $\mathtt{GoodKey}$ to denote the event that all key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}$ sampled from Gen are perfectly correct. The event $\mathtt{GoodKey}$ only depends on the Gen, and happens with overwhelming probability. By union bound,

$$\mathbf{Pr}[\mathtt{GoodKey}] \geq 1 - N\cdot\varepsilon(\kappa).$$

Conditioned on $\mathtt{GoodKey}$, $\mathcal{A}$'s behavior diverges between $\mathsf{Game}_{2,v}$ and $\mathsf{Game}_{2,v,\mathsf{Alt},0}$ solely for queries $(T \neq T^*, (\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]}))$ where exactly B+1 indices, especially including v, pass the check $\mathsf{Check}(i, z_i, r_i) = 1$. We use $\mathtt{Bad}$ to denote this bad event, and we have

$$\left|\mathbf{Pr}\left[\mathsf{Game}_{2,v} \Rightarrow 1 \wedge \mathtt{GoodKey}\right] - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1 \wedge \mathtt{GoodKey}\right]\right| \leq \mathbf{Pr}\left[\mathtt{Bad} \wedge \mathtt{Goodkey}\right].$$

Therefore, conditioned on $\mathtt{GoodKey}$, if $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ can distinguish between $\mathsf{Game}_{2,v}$ and $\mathsf{Game}_{2,v,\mathsf{Alt},0}$, then the $\mathtt{Bad}$ event happens with non-negligible probability in $\mathsf{Game}_{2,v}$, which can be utilized to attack the soundness of TSC. Consider the following adversary $\mathcal{B}_2$ attacking the soundness of TSC, where C_2 is the challenger in the soundness game of TSC.

$\textsc{AltInv}_{-j}$

- **Hardwired**: $\mathsf{pk}, (\mathsf{sk}_i)_{i \neq j}, T^*$.
- **Input:** $\big(T, y = \big(\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]}\big)\big)$.
- **Operations:**
  1. If $T = T^*$, return $\perp$.
  2. Compute $(z_i, r_i) := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ for all $i \in [N] \setminus \{j\}$.
  3. Initialize $U = \emptyset$; for $i \in [N] \setminus \{j\}$, add i to U if $\mathsf{Check}(i, z_i, r_i) = 1$.
  4. If $|U| = B - 1$, set $r_j = -\sum_{i\in U} r_i$ and compute $z_j := \mathsf{Recover}(\mathsf{pk}_j, \mathsf{ct}_j, r_j)$; if $\mathsf{Check}(j, z_j, r_j) = 1$, add j to U.
  5. If $|U| \neq B$ or $\sum_{i\in U} r_i \neq 0$, return $\perp$.
  6. For each $i \in U$, parse $z_i = \sigma_i$; let $U = \{i_1, \ldots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$.
  7. Return $\left(U, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in U}, (\mathsf{ct}_i)_{i\in[N]\setminus U}\right)$.

Figure 3: Inversion oralce $\textsc{AltInv}_{-j}$

1. $\mathcal{B}_2$ receives input $\left(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}}\right)$ and runs $(T^*, st) \leftarrow \mathcal{A}_1(1^\kappa, 1^{\ell_{\mathsf{tag}}})$ to obtain the challenge tag T^*. $\mathcal{B}_2$ samples $S^* = \{i_1, \ldots, i_B\} \leftarrow \binom{[N]}{B}$ where $i_1 < \cdots < i_B$, and sends $\left(1^N, 1^B, 1^{\ell_{\mathsf{tag}}}, T^*\right)$ to the soundness game's challenger C_2.
2. C_2 receives $\left(1^N, 1^B, 1^{\ell_{\mathsf{tag}}}, T^*\right)$, then computes $(\mathsf{pp}, \mathsf{com}^*, (\sigma_i^*)_{i\in[N]}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}}, T^*)$, and sends $(\mathsf{pp}, \mathsf{com}^*, (\sigma_i^*)_{i\in[N]})$.
3. $\mathcal{B}_2$ receives $\left(\mathsf{pp}, \mathsf{com}^*, (\sigma_i^*)_{i\in[N]}\right)$, and computes x^* and y^* as follows:
   - For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$.
   - Sample $r_{i_j}^* \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r_{i_B}^* := -\sum_{j=1}^{B-1} r_{i_j}^*$.
   - For $i \in S^* \cup [v-1]$, $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma_i^*; r_i^*)$; for $i \in [N] \setminus (S^* \cup [v-1]), \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i^*)$ where $m_i^* \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
   - $\mathsf{ek} := \left(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\right)$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
   - $x^* := \left(S^*, (r_{i_j}^*)_{j\in[B-1]}, (\sigma_i^*)_{i\in S^*}, (\mathsf{ct}_i^*)_{i\in[N]\setminus S^*}\right), y^* := \left(\mathsf{com}^*, (\mathsf{ct}_i^*)_{i\in[N]}\right)$.
4. $\mathcal{B}_2$ runs $\mathcal{A}_2(st, y^*)$ and uses all key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}$ as td to simulate the inversion oracle $\mathsf{Inv}(\cdot, \mathsf{td}, \cdot)$ for $\mathcal{A}_2(st, y^*)$. Simultaneously, $\mathcal{B}_2$ monitors every query $\left(T \neq T^*, (\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})\right)$ from $\mathcal{A}_2$, decrypts to obtain corresponding (z_i, r_i), and checks if the event $\mathtt{Bad}$ happens.
5. Finally, if the event $\mathtt{Bad}$ happens, $\mathcal{B}_2$ sends the corresponding $(T \neq T^*, U, (\sigma_i)_{i\in U})$ to C_2 where $|U| = B+1$, $v \in U$ and $\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, i, \sigma_i, T) = 1$ (implied by $\mathsf{Check}(i, z_i, r_i) = 1$) for all $i \in U$; otherwise, $\mathcal{B}_2$ sends $\perp$ to C_2.

It is easy to see that

$$\mathbf{Pr}\left[\mathtt{Bad} \wedge \mathtt{GoodKey}\right] \leq \mathbf{Adv}^{\mathsf{Sound}}_{\mathsf{TSC},\mathcal{B}_2}(\kappa).$$

Then we have

$$\begin{aligned}
&\left|\mathbf{Pr}\left[\mathsf{Game}_{2,v} \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1\right]\right| \\
&= \Big|\mathbf{Pr}\left[\mathsf{Game}_{2,v} \Rightarrow 1 \wedge \mathtt{GoodKey}\right] + \mathbf{Pr}\left[\mathsf{Game}_{2,v} \Rightarrow 1 \wedge \overline{\mathtt{GoodKey}}\right] \\
&\quad - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1 \wedge \mathtt{GoodKey}\right] - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1 \wedge \overline{\mathtt{GoodKey}}\right]\Big| \\
&\leq \left|\mathbf{Pr}\left[\mathsf{Game}_{2,v} \Rightarrow 1 \wedge \mathtt{GoodKey}\right] - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1 \wedge \mathtt{GoodKey}\right]\right| \\
&\quad + \left|\mathbf{Pr}\left[\mathsf{Game}_{2,v} \Rightarrow 1 \wedge \overline{\mathtt{GoodKey}}\right] - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1 \wedge \overline{\mathtt{GoodKey}}\right]\right| \\
&\leq \mathbf{Pr}\left[\mathtt{Bad} \wedge \mathtt{GoodKey}\right] + \mathbf{Pr}\left[\overline{\mathtt{GoodKey}}\right] \\
&\leq \mathbf{Adv}^{\mathsf{Sound}}_{\mathsf{TSC},\mathcal{B}_2}(\kappa) + N \cdot \varepsilon(\kappa).
\end{aligned}$$

□

Lemma 4.8. *For every $v \in [N]$, there exists a PPT adversary $\mathcal{B}_3$ against the IND-CPA security of* PKE *such that*

$$\left|\mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},1} \Rightarrow 1\right]\right| \leq \mathbf{Adv}^{\textit{ind-cpa}}_{\mathsf{PKE},\mathcal{B}_3}(\kappa).$$

Proof. Observe that if $v \in S^*$, $\mathsf{Game}_{2,v,\mathsf{Alt},0}$ is identical to $\mathsf{Game}_{2,v,\mathsf{Alt},1}$, i.e.,

$$\mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},0} \Rightarrow 1 \,\middle|\, v \in S^*\right] = \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},1} \Rightarrow 1 \,\middle|\, v \in S^*\right].$$

Consider the following adversary $\mathcal{B}_3$ attacking the IND-CPA security of PKE, where C_3 is the challenger in the experiment $\mathsf{Expr}^{\mathsf{ind\text{-}cpa},b}_{\mathsf{PKE},\mathcal{B}_\ni}(\kappa)$:

1. $\mathcal{B}_3$ receives input $(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}})$, runs $(T^*, st) \leftarrow \mathcal{A}_1(1^\kappa, 1^{\ell_{\mathsf{tag}}})$ to obtain the challenge tag T^*, and sends 1^κ to C_3.
2. C_3 runs $(\mathsf{pk}_v, \mathsf{sk}_v) \leftarrow \mathsf{Gen}(1^\kappa)$ sends pk_v to $\mathcal{B}_3$.
3. For $j \in [N] \setminus \{v\}$, $\mathcal{B}_3$ generates $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{Gen}(1^\kappa)$.
4. $\mathcal{B}_3$ samples $S^* = \{i_1, \ldots, i_B\} \leftarrow \binom{[N]}{B}$ where $i_1 < \cdots < i_B$.
5. $\mathcal{B}_3$ amples $r^*_{i_j} \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r^*_{i_B} := -\sum_{j=1}^{B-1} r^*_{i_j}$.
6. For $i \in [N]$, choose $\sigma_i \leftarrow \{0,1\}^{\ell_\sigma}$; for $i \in (S^* \cup [v-1])$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma^*_i; r^*_i)$; for $i \in [N] \setminus (S^* \cup [v])$, $\mathsf{ct}^*_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m^*_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
7. $\mathcal{B}_3$ decides if v belongs to S^*:
   - If $v \in S^*$, $\mathcal{B}_3$ already obtained $\mathsf{ct}^*_v = \mathsf{Enc}(\mathsf{pk}_v, \sigma^*_v; r^*_v)$. Note that in this case, $\mathcal{B}_3$ owns all $(\mathsf{ct}^*_i)_{i \in [N]}$. Therefore, $\mathcal{B}_3$ sends $\perp$ to C_3.
   - If $v \notin S^*$, $\mathcal{B}_3$ samples $m^*_v \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$, and sends (m^*_v, σ^*_v) to C_3:
     - C_3 samples a random bit $b \in \{0,1\}$, and computes ct_v as follows:
       * if $b = 0$, then $\mathsf{ct}^*_v := \mathsf{Enc}(\mathsf{pk}_v, m^*_v)$;

* if $b = 1$, then $\mathsf{ct}_v^* := \mathsf{Enc}(\mathsf{pk}_v, \sigma_v^*)$.

8. $\mathcal{B}_3$ computes $(\mathsf{pp}, \mathsf{com}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T^*, (\sigma_i^*)_{i\in[N]})$ and sets $\mathsf{ek} := \big(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\big)$, $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]\setminus\{v\}}$, $x^* := \Big(S^*, (r_{i_j}^*)_{j\in[B-1]}, (\sigma_i^*)_{i\in S^*}, (\mathsf{ct}_i^*)_{i\in[N]\setminus S^*}\Big)$, and $y^* := \big(\mathsf{com}^*, (\mathsf{ct}_i^*)_{i\in[N]}\big)$.

9. $\mathcal{B}_3$ runs $\mathcal{A}_2(\mathsf{ek}, y^*, st)$ and uses $\big(\mathsf{ek} = \big(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\big), (\mathsf{sk}_i)_{i\neq v}\big)$ to simulate the alternative inversion oracle $\textsc{AltInv}_{-v}$. Finally, $\mathcal{B}_3$ obtains output x from $\mathcal{A}_2(y^*, st)$.

10. If $x = x^*$, $\mathcal{B}_3$ sends 1 to $\mathcal{C}_3$; else, $\mathcal{B}_3$ sends 0 to $\mathcal{C}_3$.

Conditioned on $v \notin S^*$, it is easy to see that from the point of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathcal{B}_3$ perfectly simulates $\mathsf{Game}_{3,v,\mathsf{Alt},b}$ when it is in experiment $\mathsf{Expr}^{\mathsf{ind\text{-}cpa},b}_{\mathsf{PKE},\mathcal{B}_\ni}(\kappa)$.

Note that the event $v \in S^*$ happens with the same probability in the two games. Therefore, by the law of total probability, lemma 4.8 holds. □

Lemma 4.9. *Suppose that* PKE *satisfies $\varepsilon(\kappa)$-almost-all-keys perfect correctness. For $v \in [N]$, there exists a PPT adversary $\mathcal{B}_2$ against the soundness of* TSC *such that*

$$\Big|\mathbf{Pr}\left[\mathsf{Game}_{2,v+1} \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_{2,v,\mathsf{Alt},1} \Rightarrow 1\right]\Big| \leq \mathbf{Adv}^{\mathsf{Sound}}_{\mathsf{TSC},\mathcal{B}_2}(\kappa) + N \cdot \varepsilon(\kappa).$$

Proof. The proof of this lemma is the same as that of lemma 4.7, and thus we omit it here. □

5 ATDFs from Canonical TDFs

This section presents our construction of ATDF.

5.1 Construction

Our ATDF construction uses building blocks TSC, PKE with the following properties.

1. TSC is a TSC with randomness opening; let $\ell_\sigma = \ell_\sigma(\kappa)$ denote the length of the openings. Moreover, we additionally require that TSC satisfies uniqueness (definition 5.1), defined below.

2. $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec})$ is a IND-CPA secure randomness-recoverable PKE with message space $\{0,1\}^{\ell_{\mathsf{msg}}}$, where $\ell_{\mathsf{msg}} \stackrel{\text{def}}{=} \kappa + \ell_\sigma$. We require PKE to have a key-independent randomness space $\mathsf{Rnd} = (\mathsf{Rnd}_\kappa)_{\kappa\in\mathbb{N}}$ that does not depend on the public key, and each Rnd_κ is an Abelian group. We assume that a ciphertext can be encoded by ℓ_{ct} bits.

Definition 5.1 (Uniqueness)**.** We say TSC satisfies uniqueness, if for all $\mathsf{pp} \leftarrow \mathsf{TSC.Setup}(1^\kappa, 1^N, 1^B, 1^t)$, $S \in \binom{[N]}{B}$, $(\sigma_i)_{i\in S}$, tag T, and com', if $\mathsf{com}' \neq \mathsf{TSC.Commit}(\mathsf{pp}, S, T, (\sigma_i)_{i\in S})$, then there exists some $i^* \in S$ such that $\mathsf{TSC.Verify}(\mathsf{pp}, \mathsf{com}', i^*, \sigma_{i^*}, T) = 0$.

We show that construction 3.3 satisfies uniqueness in appendix B.

Construction 5.2. Choose parameters $N = N(\kappa), B = B(\kappa)$ used for TSC such that $\binom{N}{B} \geq |\mathsf{Rnd}| \cdot 2^\kappa$, and let $t \stackrel{\text{def}}{=} N \cdot \ell_{\mathsf{ct}}$. Our ATDF construction is as follows.

- $\mathsf{Setup}(1^\kappa) \mapsto (\mathsf{ek}, \mathsf{td})$:

Check

- **Hardwired**: $\mathsf{ek}, y = ((\mathsf{ct}_j)_{j\in[N]}, \mathsf{com})$.
- **Input:** $i \in [N], z \in \left(\{0,1\}^\kappa \times \{0,1\}^{\ell_{\mathsf{tsc}}}\right) \cup \{\perp\}, r \in \mathsf{Rnd}$.

Output 1 if and only if the following conditions are satisfied:

(a) $z \neq \perp$. Parse $z = g \| \sigma$ where $g \in \{0,1\}^\kappa$.

(b) $g = 1^\kappa$.

(c) $\mathsf{TSC.Verify}(\mathsf{pp}, \mathsf{com}, i, \sigma, T) = 1$ where $T := \mathsf{ct}_1 \| \cdots \| \mathsf{ct}_N$.

(d) $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, z; r)$.

Figure 4: Subroutine $\mathsf{Check}(i, z, r)$

1. $\mathsf{pp} \leftarrow \mathsf{TSC.Setup}(1^\kappa, 1^N, 1^B, 1^t)$.
2. $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ for $i \in [N]$.
3. Return $\mathsf{ek} = \left(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\right)$ and $\mathsf{td} = (\mathsf{sk}_i)_{i\in[N]}$.

- $\mathsf{Samp}(\mathsf{ek}, 1^\kappa) \mapsto x$:

1. Choose a subset $S \subset [N]$ of size B uniformly at random. Let $S = \{i_1, i_2, \cdots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$. Then sample $r_{i_j} \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$.
2. Choose $\sigma_i \leftarrow \{0,1\}^{\ell_\sigma}$ for each $i \in S$.
3. For $i \in [N] \backslash S$, $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}, m_i)$ where $m_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
4. Return $\left(S, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in S}, (\mathsf{ct}_i)_{i\in[N]\backslash S}\right)$.

- $\mathsf{Eval}(\mathsf{ek}, x) \mapsto y$:

1. Parse $x = \left(S, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in S}, (\mathsf{ct}_i)_{i\in[N]\backslash S}\right)$.
2. Set $r_{i_B} := \sum_{j=1}^{B-1} r_{i_j}$
3. For $i \in S$, $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, 1^\kappa \| \sigma_i; r_i)$.
4. $\mathsf{com} \leftarrow \mathsf{TSC.Commit}(\mathsf{pp}, S, T, (\sigma_i)_{i\in S})$, where $T := \mathsf{ct}_1 \| \cdots \| \mathsf{ct}_N$.
5. Return $\left(\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]}\right)$.

- $\mathsf{Inv}(\mathsf{td}, y) \mapsto x\ /\perp$:

1. Parse $y = \left(\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]}\right)$ and $\mathsf{td} = (\mathsf{sk}_i)_{i\in[N]}$.
2. For each $i \in [N]$, compute $(z_i, r_i) := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ for each $i \in [N]$.
3. Initialize a set $U := \emptyset$. For each $i \in [N]$, add i into U if $\mathsf{Check}(i, y_i, r_i) = 1$, where Check is defined in fig. 4.
4. If the set $|U| \neq B$, output $\perp$.
5. If $\sum_{i\in U} r_i \neq 0$, output $\perp$.

6. For each $i \in U$, parse $z_i = 1^\kappa \| \sigma_i$; let $U = \{i_1, \ldots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$.
7. Return $\left(U, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in U}, (\mathsf{ct}_i)_{i\in[N]\setminus U}\right)$.

The correctness and security of construction 5.2 are captured by the following two theorems.

Theorem 5.3 (Correctness)**.** *If* PKE *satisfies ε-almost-all-keys correctness, then construction 5.2 satisfies α-almost-all-keys ν-correctness, where $\alpha = N \cdot \varepsilon$ and $\nu = (N - B) \cdot 2^{-\kappa}$.*

Theorem 5.4 (Adaptive one-wayness)**.** *Assume* TSC *is a TSC with randomness opening satisfying uniqueness, and* PKE *is an RR-PKE with (i) almost-all-keys perfect correctness and (ii) uniform ciphertext for random message. Then construction 5.2 satisfies adaptive one-wayness.*

Since PKE and TSC with aforementioned properties can all be based on canonical TDFs, we conclude that TB-ATDFs and ATDFs can be constructed from canonical TDFs, establishing theorem 1.1.

The proof of theorem 5.3 is direct and almost identical to that of theorem 4.2, and thus we omit it here. We prove theorem 5.4 in the rest of this section.

5.2 Proof of Theorem 5.4: Adaptive One-Wayness

Let ATDF denote the ATDF in construction 5.2 and let $\mathcal{A}$ be an adversary that aims to attack the adaptive one-wayness of ATDF. We define a sequence of games Game_j ($j \in [4]$), where Game_1 is exactly the adaptive one-wayness experiment.

Game_1**.** This is the adaptive one-wayness experiment.

- Challenge generation phase: Generating $(\mathsf{ek}, \mathsf{td})$ and (x^*, y^*).
  1. $\mathsf{pp} \leftarrow \mathsf{TSC.Setup}(1^\kappa, 1^N, 1^B, 1^t)$.
  2. For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$
  3. $\mathsf{ek} := (\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]})$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
  4. Choose $S^* = \{i_1, \ldots, i_B\}$ uniformly at random where $i_1 < \cdots < i_B$.
  5. sample $r^*_{i_j} \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r^*_{i_B} := -\sum_{j=1}^{B-1} r_{i_j}$.
  6. For $i \in S^*$, choose $\sigma^*_i \leftarrow \{0,1\}^{\ell_\sigma}$ uniformly at random and $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, 1^\kappa \| \sigma^*_i; r^*_i)$; for $i \in [N] \setminus S^*$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}, m^*_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
  7. $\mathsf{com} := \mathsf{TSC.Commit}(\mathsf{pp}, S^*, T^*, (\sigma^*_i)_{i\in S^*})$ where $T^* := \mathsf{ct}_1 || \cdots || \mathsf{ct}_N$.
  8. $x^* := \left(S^*, (r^*_{i_j})_{j\in[B-1]}, (\sigma^*_i)_{i\in S^*}, (\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}\right), y^* := (\mathsf{com}^*, (\mathsf{ct}^*_i)_{i\in[N]})$.
- Run $\mathcal{A}(\mathsf{ek}, y^*)$ and each inversion query $y = (\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})$ is answered as follows:
  1. Compute $(z_i, r_i) := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ for all $i \in [N]$.
  2. Initialize $U = \emptyset$; for $i \in [N]$, add i to U if $\mathsf{Check}(i, z_i, r_i) = 1$.
  3. If $|U| \neq B$ or $\sum_{i\in U} r_i \neq 0$, return $\perp$.
  4. For each $i \in U$, parse $z_i = 1^\kappa \| \sigma_i$; let $U = \{i_1, \ldots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$.
  5. Return $\left(U, (r_{i_j})_{j\in[B-1]}, (\sigma_i)_{i\in U}, (\mathsf{ct}_i)_{i\in[N]\setminus U}\right)$.
- $\mathcal{A}$ outputs x' and the game outputs 1 if and only if $x' = x^*$.

Game_2**.** This game is identical to Game_1 except that we add an additional rejection rule in the inversion oracle: if $\mathsf{ct}_1\|\cdots\|\mathsf{ct}_N = \mathsf{ct}_1^*\|\cdots\|\mathsf{ct}_N^*$, return $\bot$.

Game_3**.** This game is identical to Game_2 except that we additionally pick $\sigma_i^* \leftarrow \mathsf{Rnd}$ for $i \in [N] \setminus S^*$ and generate pp and com^* as follows:

- $(\mathsf{pp}, \mathsf{com}^*) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, T^*, (\sigma_i^*)_{i\in[N]})$.

Game_4**.** This game is identical to Game_3 except that for $i \in [N] \setminus S^*$, ct_i^* is switched to an encryption of $1^\kappa|\sigma_i^*$. That is, we additionally pick $r_i^* \leftarrow \mathsf{Rnd}$ for $i \in [N] \setminus S^*$ and generate $(\mathsf{ct}_i^*)_{i\in[N]}$ as follows:

- For $i \in [N]$, $\mathsf{ct}_i^* := \mathsf{Enc}(\mathsf{pk}_i, 1|\sigma_i^*; r_i^*)$.

It holds that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{aow}}_{\mathsf{ATDF},\mathcal{A}}(\kappa) &= \mathbf{Pr}\left[\mathsf{Game}_1 \Rightarrow 1\right] \\ &\le \mathbf{Pr}\left[\mathsf{Game}_4 \Rightarrow 1\right] + \sum_{j\in[3]} \left|\mathbf{Pr}\left[\mathsf{Game}_j \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_{j+1} \Rightarrow 1\right]\right|. \end{aligned} \tag{3}$$

It suffices to show that all terms in eq. (3) are negligible; the following lemmas finish the proof.

Lemma 5.5. *If* PKE *satisfies ε-almost-all-keys perfect correctness, it holds that*

$$|\mathbf{Pr}\left[\mathsf{Game}_1 \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_2 \Rightarrow 1\right]| \le N \cdot \varepsilon + (N - B) \cdot 2^{-\kappa} = \mathtt{negl}(\kappa).$$

Proof. In Game_2, we say an inversion query $y = (\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})$ issued by $\mathcal{A}$ is *tag-bad* if it holds that $\mathsf{ct}_1\|\cdots\|\mathsf{ct}_N = \mathsf{ct}_1^*\|\cdots\|\mathsf{ct}_N^*$ and $\mathsf{Inv}(\mathsf{td}, y) \neq \bot$. Let $\mathtt{TB}$ denote the event that a tag-bad query occurs in Game_2. Since Game_1 and Game_2 are identical except that a tag-bad query occurs, it holds that

$$|\mathbf{Pr}\left[\mathsf{Game}_1 \Rightarrow 1\right] - \mathbf{Pr}\left[\mathsf{Game}_2 \Rightarrow 1\right]| \le \mathop{\mathbf{Pr}}_{\mathsf{Game}_2}\left[\mathtt{TB}\right].$$

It remains to bound $\mathbf{Pr}\left[\mathtt{TB}\right]$ from above.

Consider a tag-bad query $y = (\mathsf{com}, (\mathsf{ct}_i)_{i\in[N]})$. Since $y \neq y^*$ and $\mathsf{ct}_1\|\cdots\|\mathsf{ct}_N = \mathsf{ct}_1^*\|\cdots\|\mathsf{ct}_N^*$, it must be that $\mathsf{com} \neq \mathsf{com}^*$. By the uniqueness of TSC, there exists some $i^* \in S^*$ such that

$$\mathsf{TSC.Verify}(\mathsf{pp}, \mathsf{com}, i^*, \sigma_i^*, T^*) = 0, \tag{4}$$

where $T^* = \mathsf{ct}_1^*\|\cdots\|\mathsf{ct}_N^*$. Let $(z_i, r_i) := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i) = \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i^*)$ and

$$U_y := \{i \in [N] : \mathsf{Check}(i, z_i, r_i) = 1\}.$$

Let $\mathtt{GoodKey}$ be the event that all N key pairs are error-free. By the ε-almost-all-key perfect correctness of PKE, we have

$$\mathbf{Pr}\left[\mathtt{GoodKey}\right] \ge 1 - N \cdot \varepsilon.$$

Let $\mathtt{GoodMsg}$ be the event that for all $i \in [N] \setminus S^*$, the first κ-bits of m_i^* are not all ones. Since m_i^*'s are chosen uniformly at random, by union bound we have

$$\mathbf{Pr}\left[\mathtt{GoodMsg}\right] \ge 1 - (N - B) \cdot 2^{-\kappa}.$$

Conditioned on $\mathtt{GoodMsg}$ and $\mathtt{GoodKey}$, we have the following observations:

- For all $i \notin S^*$, $\mathsf{Check}(i, z_i, r_i) = 0$. This is because, by the perfect correctness of PKE, $z_i = m_i^*$, and thus the first κ-bits of z_i are not all ones.

- For all $i \in S^*$, by the perfect correctness of PKE, we have $z_i = 1^\kappa | \sigma_i^*$. In particular, eq. (4) implies $\mathsf{Check}(i^*, z_{i^*}, r_{i^*}) = 0$, meaning that $i^* \notin U_y$.

Consequently, we have $|U_y| < B$ and thus $\mathsf{Inv}(td, y) = \bot$, contradicting the assumption that y is a tag-bad query. In other words,

$$\Pr_{\mathsf{Game}_2}[\mathtt{TB} \mid \mathtt{GoodKey} \wedge \mathtt{GoodMsg}] = 0.$$

Hence,

$$\begin{aligned}\Pr[\mathtt{TB}] &\le \Pr[\mathtt{TB} \mid \mathtt{GoodKey} \wedge \mathtt{GoodMsg}] + \Pr\left[\overline{\mathtt{GoodKey}} \vee \overline{\mathtt{GoodMsg}}\right] \\ &\le \Pr[\mathtt{TB} \mid \mathtt{GoodKey} \wedge \mathtt{GoodMsg}] + \Pr\left[\overline{\mathtt{GoodKey}}\right] + \Pr\left[\overline{\mathtt{GoodMsg}}\right] \\ &\le 0 + N \cdot \varepsilon + (N - B) \cdot 2^{-\kappa}.\end{aligned}$$

This completes the proof. □

Lemma 5.6. *There exists a PPT adversary $\mathcal{B}_1$ attacking the adaptive indistinguishability of* TSC*'s setup such that*

$$|\Pr[\mathsf{Game}_2 \Rightarrow 1] - \Pr[\mathsf{Game}_1 \Rightarrow 1]| = \mathbf{Adv}^{\textit{ind-setup}}_{\mathsf{TSC},\mathcal{B}_1}(\kappa).$$

Proof. Consider the following adversary $\mathcal{B}_1$ attacking the adaptive indistinguishability of TSC's setup, where C_1 is the challenger in the experiment $\mathsf{Expr}^{\mathsf{ind\text{-}setup},b}_{\mathsf{TSC},\mathcal{B}_1}(\kappa)$.

1. $\mathcal{B}_1$ receives input $(1^\kappa, 1^N, 1^B, 1^t)$ and samples $S^* = \{i_1, \ldots, i_B\} \leftarrow \binom{[N]}{B}$ where $i_1 < \cdots < i_B$, and sends $(1^N, 1^B, 1^{\ell_{\mathsf{tag}}}, S^*)$ to C_1.

2. C_1 samples $(\sigma_i^*)_{i \in [N]} \leftarrow (\{0,1\}^{\ell_\sigma})^N$ and sends $(\sigma_i^*)_{i \in S^*}$ to $\mathcal{B}_1$.

3. On receiving $(\sigma_i^*)_{i \in S^*}$, $\mathcal{B}_1$ does the following pre-computation for generating x^* and y^*:

   - For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$.
   - Sample $r_{i_j}^* \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r_{i_B}^* := -\sum_{j=1}^{B-1} r_{i_j}^*$.
   - For $i \in S^*$, $\mathsf{ct}_i^* := \mathsf{Enc}(\mathsf{pk}_i, \sigma_i^*; r_i^*)$; for $i \in [N] \setminus S^*$, $\mathsf{ct}_i^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i^*)$, where $m_i^* \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.

4. $\mathcal{B}_1$ sends $T^* = \mathsf{ct}_1^* || \cdots || \mathsf{ct}_N^*$ to the C_1.

5. C_1 proceeds according to b:

   - If $b = 0$, C_1 runs $\mathsf{pp}^0 \leftarrow \mathsf{Setup}(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}})$, and computes $\mathsf{com}^{0*} \leftarrow \mathsf{TSC.Commit}(\mathsf{pp}^0, S^*, T^*, (\sigma_i^*)_{i \in [N]})$.
   - If $b = 1$, C_1 computes $(\mathsf{pp}^1, \mathsf{com}^{1*}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^{\ell_{\mathsf{tag}}}, T^*, (\sigma_i^*)_{i \in [N]})$.

   Next, C_1 sends $(\mathsf{pp}^b, \mathsf{com}^{b*})$ to $\mathcal{B}_1$.

6. $\mathcal{B}_1$ receives $(\mathsf{pp}^b, \mathsf{com}^{b*})$, and finishes computing x^* and y^* as follows:

- $\mathsf{ek} := \left(\mathsf{pp}^b, (\mathsf{pk}_i)_{i\in[N]}\right)$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
- $x^* := \left(S^*, (r^*_{i_j})_{j\in[B-1]}, (\sigma^*_i)_{i\in S^*}, (\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}\right), y^* := \left(\mathsf{com}^{b*}, (\mathsf{ct}^*_i)_{i\in[N]}\right)$.

7. $\mathcal{B}_1$ runs $\mathcal{A}(st, y^*)$ and uses all key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}$ as td to simulate the inversion oracle $\mathsf{Inv}(\cdot, \mathsf{td}, \cdot)$ for $\mathcal{A}(st, y^*)$. Finally, $\mathcal{B}_1$ obtains output x from $\mathcal{A}(st, y^*)$.
8. If $x = x^*$, $\mathcal{B}_1$ sends 1 to C_1; otherwise, $\mathcal{B}_1$ sends 0 to C_1.

It is easy to see that if $b = 0$, $\mathcal{B}_1$ perfectly simulates Game_1; if $b = 1$, $\mathcal{B}_1$ perfectly simulates Game_2. This finishes the proof. □

Lemma 5.7. *There exists PPT adversaries $\mathcal{B}_3$ and $\mathcal{B}_4$ such that*

$$|\mathbf{Pr}[\mathsf{Game}_3 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_4 \Rightarrow 1]| = 2(N-B)\cdot \mathbf{Adv}^{\mathsf{Sound}}_{\mathsf{TSC},\mathcal{B}_3}(\kappa) + 2(N-B)N\cdot\varepsilon(\kappa) + (N-B)\cdot\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{B}_4}(\kappa).$$

Proof. For each $j \in [N+1]$, we define an intermediate game $\mathsf{Game}_{3,j}$

- $\mathsf{Game}_{3,j}$ is identical to Game_3 except that $(\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}$ is generated as follows: For $i \in [N]\setminus S^*$, if $i < j$, we additionally pick $r^*_i \leftarrow \mathsf{Rnd}$ and set $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, 1^\kappa|\sigma^*_i; r^*_i)$; if $i \geq j$, $\mathsf{ct}^*_i := \mathsf{Enc}(\mathsf{pk}_i, m^*_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.

Observe that $\mathsf{Game}_{3,1} \equiv \mathsf{Game}_3$ and $\mathsf{Game}_{3,N+1} \equiv \mathsf{Game}_4$. For $j \in [N]$, we further define intermediate games $\mathsf{Game}_{3,j,\mathsf{Alt},0}$ and $\mathsf{Game}_{3,j,\mathsf{Alt},1}$ to facilitate the switch from $\mathsf{Game}_{3,j}$ to $\mathsf{Game}_{3,j+1}$:

- $\mathsf{Game}_{3,j,\mathsf{Alt},0}$ is identical to $\mathsf{Game}_{3,j}$ except that the inversion oralce is replaced by $\textsc{AltInv}_{-j}$ defined in fig. 5, which only uses $(\mathsf{sk}_i)_{i\neq j}$.
- $\mathsf{Game}_{3,j,\mathsf{Alt},1}$ is identical to $\mathsf{Game}_{3,j,\mathsf{Alt},0}$ except that $(\mathsf{ct}^*_i)_{i\in[N]\setminus S^*}$ is generated as follows: for $i \in [N]\setminus S^*$, if $i \leq j$, $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma^*_i; r^*_i)$; if $i > j$, $\mathsf{ct}^*_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m^*_i)$ where $m^*_i \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.

It suffices to show that for all $j \in [N]$, we have

$$\mathsf{Game}_{3,j} \approx_c \mathsf{Game}_{3,j,\mathsf{Alt},0} \approx_c \mathsf{Game}_{3,j,\mathsf{Alt},1} \approx_c \mathsf{Game}_{3,j+1}.$$

This follows from the same argument as in the proof of lemma 4.6. □

Lemma 5.8. $\mathbf{Pr}[\mathsf{Game}_4 \Rightarrow 1] \leq \frac{|\mathsf{Rnd}|}{\binom{N}{B}} \leq 2^{-\kappa}$.

The proof of lemma 5.8 is the same as that of lemma 4.4, and thus is omitted.

$\textsc{AltInv}_{-j}$

- **Hardwired**: $\mathsf{ek}, (\mathsf{sk}_i)_{i \neq j}, (\mathsf{ct}_i^*)_{i \in [N]}$.
- **Input:** $y = (\mathsf{com}, (\mathsf{ct}_i)_{i \in [N]})$.
- **Operations:**
  1. If $\mathsf{ct}_1 \| \cdots \| \mathsf{ct}_N = \mathsf{ct}_1^* \| \cdots \| \mathsf{ct}_N^*$, return $\perp$.
  2. Compute $(z_i, r_i) := \mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i)$ for all $i \in [N] \setminus \{j\}$.
  3. Initialize $U = \emptyset$; for $i \in [N] \setminus \{j\}$, add i to U if $\mathsf{Check}(i, z_i, r_i) = 1$.
  4. If $|U| = B - 1$, set $r_j = -\sum_{i \in U} r_i$ and compute $z_j := \mathsf{Recover}(\mathsf{pk}_j, \mathsf{ct}_j, r_j)$; if $\mathsf{Check}(j, z_j, r_j) = 1$, add j to U.
  5. If $|U| \neq B$ or $\sum_{i \in U} r_i \neq 0$, return $\perp$.
  6. For each $i \in U$, parse $z_i = 1^\kappa | \sigma_i$; let $U = \{i_1, \ldots, i_B\}$ where $i_1 < i_2 < \cdots < i_B$.
  7. Return $\left(U, (r_{i_j})_{j \in [B-1]}, (\sigma_i)_{i \in U}, (\mathsf{ct}_i)_{i \in [N] \setminus U}\right)$.

Figure 5: Inversion oralce $\textsc{AltInv}_{-j}$

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 595–618. Springer, 2009. 2

[BHHI10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 423–444. Springer, 2010. 2

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 535–564. Springer, 2018. 2

[DGH+19] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Kevin Liu, and Giulio Malavolta. Rate-1 trapdoor functions from the diffie-hellman problem. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 585–606. Springer, 2019. 6

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2019. 6

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6), 1976. 1

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360. Springer, 2004. 6, 8

[GGH19] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*, pages 33–63. Springer, 2019. 6

[HHR+10] Iftach Haitner, Thomas Holenstein, Omer Reingold, Salil Vadhan, and Hoeteck Wee. Universal one-way hash functions via inaccessible entropy. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 616–637. Springer, 2010. 5

[HKW20] Susan Hohenberger, Venkata Koppula, and Brent Waters. Chosen ciphertext security from injective trapdoor functions. In *Annual International Cryptology Conference*, pages 836–866. Springer, 2020. 1, 5, 6, 11, 12

[HL17] Shuai Han and Shengli Liu. Kdm-secure public-key encryption from constant-noise lpn. In *Information Security and Privacy: 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3–5, 2017, Proceedings, Part I 22*, pages 44–64. Springer, 2017. 2

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 673–692. Springer, 2010. 1, 2, 3, 5

[KMT22] Fuyuki Kitagawa, Takahiro Matsuda, and Keisuke Tanaka. Cca security and trapdoor functions via key-dependent-message security. *Journal of Cryptology*, 35(2):9, 2022. 2, 5

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. 1979. 5

[MH15] Takahiro Matsuda and Goichiro Hanaoka. Constructing and understanding chosen ciphertext security via puncturable key encapsulation mechanisms. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 561–590, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. 5

[MMZ23] Xinyu Mao, Noam Mazor, and Jiapeng Zhang. Non-adaptive universal one-way hash functions from arbitrary one-way functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 502–531. Springer, 2023. 5

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 187–196, 2008. 1, 2, 6

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394, 1990. 5

[RS09] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In *Theory of Cryptography: 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings 6*, pages 419–436. Springer, 2009. 1, 2, 3

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 1

[Wee12] Hoeteck Wee. Dual projective hashing and its applications—lossy trapdoor functions and more. In *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*, pages 246–262. Springer, 2012. 1

[Yao82] Andrew C Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982. 3, 9

A Intermediate Games

Here we present detailed descriptions of intermediate games used in the proof of lemma 4.6.

$\mathsf{Game}_{2,j}$ **(for** $j \in [N+1]$**).** $\mathsf{Game}_{2,j}$ is identical to Game_2 except that $(\mathsf{ct}_i^*)_{i\in[N]\setminus S^*}$ is generated as follows: for $i \in [N] \setminus S^*$, if $i < j$, $\mathsf{ct}_i^* := \mathsf{Enc}(\mathsf{pk}_i, \sigma_i^*; r_i^*)$; if $i \geq j$, $\mathsf{ct}_i^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i^*)$ where $m_i^* \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.

- Generating a challenge image:
  1. For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$
  2. Choose $S^* = \{i_1, \ldots, i_B\} \subset [N]$ uniformly at random where $i_1 < \cdots < i_B$.
  3. Sample $r_{i_j}^* \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r_{i_B}^* := -\sum_{j=1}^{B-1} r_{i_j}^*$.
  4. For $i \in [N]$, choose $\sigma_i^* \leftarrow \{0,1\}^{\ell_\sigma}$; for $i \in (S^* \cup [j-1])$, $\mathsf{ct}_i := \mathsf{Enc}(\mathsf{pk}_i, \sigma_i^*; r_i^*)$; for $i \in [N] \setminus (S^* \cup [j-1])$, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i^*)$, where $m_i^* \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
  5. $(\mathsf{pp}, \mathsf{com}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T^*, (\sigma_i^*)_{i\in[N]})$.
  6. $\mathsf{ek} := \left(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\right)$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
  7. $x^* := \left(S^*, (r_{i_j}^*)_{j\in[B-1]}, (\sigma_i^*)_{i\in S^*}, (\mathsf{ct}_i^*)_{i\in[N]\setminus S^*}\right), y^* := \left(\mathsf{com}^*, (\mathsf{ct}_i^*)_{i\in[N]}\right)$.
- Answering inversion queries: Use $\mathsf{Inv}(\cdot, \mathsf{td}, \cdot)$ algorithm to answer all valid inversion queries $(T_i \neq T^*, y_i)_{i\in[Q]}$, where Q represents the number of $\mathcal{A}$'s inversion queries.

$\mathsf{Game}_{2,j,\mathsf{Alt},0}$ **(for** $j \in [N]$**).** $\mathsf{Game}_{2,j,\mathsf{Alt},0}$ is identical to $\mathsf{Game}_{2,j}$ except that the inversion oralce is replaced by $\textsc{AltInv}_{-j}$ defined in fig. 3, which only uses $(\mathsf{sk}_i)_{i\neq j}$.

- Answering inversion queries: Use $\textsc{AltInv}_{-j}$ to answer all valid inversion queries $(T_i \neq T^*, y_i)_{i\in[Q]}$, where Q represents the number of $\mathcal{A}$'s inversion queries.

$\mathsf{Game}_{2,j,\mathsf{Alt},1}$ **(for** $j \in [N]$**).** $\mathsf{Game}_{2,j,\mathsf{Alt},1}$ is identical to $\mathsf{Game}_{2,j,\mathsf{Alt},0}$ except that $(\mathsf{ct}_i^*)_{i\in[N]\setminus S^*}$ is generated as follows: for $i \in [N] \setminus S^*$, if $i \leq j$, $\mathsf{ct}_i^* := \mathsf{Enc}(\mathsf{pk}_i, \sigma_i^*; r_i^*)$; if $i > j$, $\mathsf{ct}_i^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i^*)$ where $m_i^* \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.

- Generating a challenge image:
  1. For $i \in [N]$, generate $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$
  2. Choose $S^* = \{i_1, \ldots, i_B\} \subset [N]$ uniformly at random where $i_1 < \cdots < i_B$.
  3. Sample $r_{i_j}^* \leftarrow \mathsf{Rnd}$ for $j \in [B-1]$ and set $r_{i_B}^* := -\sum_{j=1}^{B-1} r_{i_j}^*$.
  4. For $i \in [N]$, choose $\sigma_i^* \leftarrow \{0,1\}^{\ell_\sigma}$; for $i \in (S^* \cup [j])$, $\mathsf{ct}_i^* := \mathsf{Enc}(\mathsf{pk}_i, \sigma_i^*; r_i^*)$; for $i \in [N] \setminus (S^* \cup [j])$, $\mathsf{ct}_i^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i^*)$ where $m_i^* \leftarrow \{0,1\}^{\ell_{\mathsf{msg}}}$.
  5. $(\mathsf{pp}, \mathsf{com}) \leftarrow \mathsf{AltSetup}(1^\kappa, 1^N, 1^B, 1^t, T^* \in \{0,1\}^t; (\sigma_i^*)_{i\in[N]})$.
  6. $\mathsf{ek} := \left(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[N]}\right)$ and $\mathsf{td} := (\mathsf{sk}_i)_{i\in[N]}$.
  7. $x^* := \left(S^*, (r_{i_j}^*)_{j\in[B-1]}, (\sigma_i^*)_{i\in S^*}, (\mathsf{ct}_i^*)_{i\in[N]\setminus S^*}\right), y^* := \left(\mathsf{com}^*, (\mathsf{ct}_i^*)_{i\in[N]}\right)$.
- Answering inversion queries: Use $\textsc{AltInv}_{-j}$ to answer all valid inversion queries $(T_i \neq T^*, y_i)_{i\in[Q]}$, where Q represents the number of $\mathcal{A}$'s inversion queries.

B Uniqueness of TSC

We first recall the definition of uniqueness.

Definition B.1 (Definition 5.1, restated)**.** We say TSC satisfies uniqueness, if for all pp generated by $\mathsf{TSC.Setup}(1^\kappa, 1^N, 1^B, 1^t)$, $S \in \binom{[N]}{B}$, $(\sigma_i)_{i \in S}$, tag T, and com', if $\mathsf{com}' \neq \mathsf{TSC.Commit}(\mathsf{pp}, S, T, (\sigma_i)_{i \in S})$, then there exists some $i^* \in S$ such that $\mathsf{TSC.Verify}(\mathsf{pp}, \mathsf{com}', i^*, \sigma_{i^*}, T) = 0$.

Lemma B.2. *The TSC scheme described in construction 3.3 satisfies uniqueness.*

Proof. Let $\mathsf{TSC}_{3.3}$ denote the TSC scheme in construction 3.3. Fix parameters κ, N, B, t, $S \in \binom{[N]}{B}$, $(\sigma_i)_{i \in S}$, and tag T. Recall that $\mathsf{TSC}_{3.3}.\mathsf{Setup}(1^\kappa, 1^N, 1^B, 1^t)$ sets $\ell := 2t + (B+1) \cdot \log N + \kappa \cdot (B+1) + \kappa$ chooses $A_i, D_i \leftarrow \mathbb{F}_{2^\ell}$ for all $i \in [N]$, and outputs $\mathsf{pp} = ((A_i, D_i)_{i \in [N]}, 1^\ell)$. And $\mathsf{TSC}_{3.3}.\mathsf{Commit}(\mathsf{pp}, S, T, (\sigma_i)_{i \in S})$ outputs the *unique* degree-$(B-1)$ polynomial $p \in \mathbb{F}_{2^\ell}[X]$ such that

$$\forall i \in S,\ p(i) = \mathsf{PRG}(\sigma_i, 1^\ell) + A_i + D_i \cdot \mathsf{emb}(T).$$

Let $p' \in \mathbb{F}_{2^\ell}[X]$ be an arbitrary degree-$(B-1)$ polynomial with $p' \neq p$. Since the degree of both p and p' is at most $(B-1)$ and $|S| = B$, there exists some $i^* \in S$ such that $p(i^*) \neq p(i^*)$. Consequently,

$$p'(i^*) \neq \mathsf{PRG}(\sigma_{i^*}, 1^\ell) + A_{i^*} + D_{i^*} \cdot \mathsf{emb}(T),$$

which is equivalent to $\mathsf{TSC}_{3.3}.\mathsf{Verify}(\mathsf{pp}, p', i^*, \sigma_{i^*}, T) = 0$. $\square$