Chosen Ciphertext Security from Injective Trapdoor Functions

Susan Hohenberger*
Johns Hopkins University
susan@cs.jhu.edu

Venkata Koppula†
Weizmann Institute of Science
venkata.koppula@weizmann.ac.il

Brent Waters‡
University of Texas at Austin and NTT Research
bwaters@cs.utexas.edu

June 20, 2020

Abstract

We provide a construction of chosen ciphertext secure public-key encryption from (injective) trapdoor functions. Our construction is black box and assumes no special properties (e.g. "lossy", "correlated product secure") of the trapdoor function.

1 Introduction

A public-key encryption system is said to be chosen ciphertext attack (CCA) secure [NY90, RS91, DDN00] if no polynomial-time attacker can distinguish whether a challenge ciphertext ct^* is an encryption of m_0 or m_1 *even* when given access to a decryption oracle for all ciphertexts except ct^*. In most deployed encryptions systems, CCA security is necessary to protect against an active attacker that might induce a user to decrypt messages of its choosing or even gain leverage from just the knowledge that an attempted decryption failed. See Shoup [Sho98] for an excellent discussion on the importance of CCA security.

Over time the cryptographic community has become rather adept at achieving CCA security from many of the same assumptions that can be used to achieve chosen plaintext attack (CPA) security for public-key encryption, where the adversary is not given access to a decryption oracle. For instance we now have practical CCA secure encryption schemes from the Decisional [CS98, CS02] and Search [CKS09] Diffie-Hellman, the difficulty of factoring [HK09, HK08], Learning with Errors (LWE) [PW08] and Learning Parity with Noise (LPN) [DMN12, KMP14] assumptions.

Despite the success in these ad-hoc number-theoretic rooted approaches, there is a strong drive to be able to understand CCA security from the perspective of general assumptions with an ultimate goal of showing that the existence of CPA secure public-key encryption implies CCA secure public-key encryption. In this work we make significant progress in this direction by showing that CCA secure public-key encryption can be built from any (injective) trapdoor function. Recall that a trapdoor function is a primitive in which any user given a public key $\mathsf{tdf.pk}$ can evaluate the input $\mathbf{x}$ by calling $\mathsf{TDF.Eval}(\mathsf{tdf.pk}, \mathbf{x}) \rightarrow \mathbf{y}$. And a user with the secret key $\mathsf{tdf.sk}$ can can recover $\mathbf{x}$ from $\mathbf{y}$ as $\mathsf{TDF.Invert}(\mathsf{tdf.sk}, \mathbf{y}) \rightarrow \mathbf{x}$. However, a polynomial-time

*Supported by NFS CNS-1414023, NSF CNS-1908181, the Office of Naval Research N00014-19-1-2294, and a Packard Foundation Subaward via UT Austin.

†Supported by the Binational Science Foundation (Grant No. 2016726), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701). This work was done in part while the author was visiting the Simons Institute for the Theory of Computing.

‡Supported in part by NSF CNS-1414082, NSF CNS-1908611, a Simons Investigator Award and a Packard Foundation Fellowship.

attacker without the secret key should not be able to output $\mathbf{x}$ given $\mathbf{y} = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, \mathbf{x})$ for a randomly chosen $\mathbf{x}$. By injective, we require a one-to-one mapping of the function input and evaluation spaces.

There is a strong lineage connecting trapdoor functions with chosen ciphertext security. Fujisaki and Okamoto [FO99] showed how in the random oracle model any CPA secure encryption scheme can be transformed into a CCA secure scheme. Their transformation implicitly creates a trapdoor function (in a spirit similar to the random oracle based TDF construction of [BHSV98]) where the decryption algorithm recovers encryption randomness and re-encrypts to test ciphertext validity. If we allow the trapdoor function to be a "doubly enhanced" permutation [Gol11], then they can be used to create non-interactive zero knowledge proofs which are known to give chosen ciphertext security via non-black box constructions [NY90, DDN00]. Peikert and Waters [PW08] introduced the notion of lossy trapdoor functions and showed that this primitive also gives rise to chosen ciphertext secure public-key encryption. Other works (e.g., [MY10, HO09]) extended and generalized this notion including Rosen and Segev [RS10] who showed that a "correlated product secure" TDF gives rise to CCA security. In each of these (standard model) cases an additional property of the trapdoor function (i.e., permutation and doubly enhanced, lossy, correlated product secure) was required and critical for achieving chosen ciphertext security leaving open the problem of building chosen ciphertext secure encryption by only assuming injective trapdoor functions.

Finally, Koppula and Waters [KW19] recently showed how to achieve chosen ciphertext security from CPA secure public-key encryption and a newly introduced "Hinting PRG" which is a pseudorandom generator that has a special form of circular security.[1] Their construction can be viewed as a "partial trapdoor" where the decryption process recovers some, but not all of the randomness used to encrypt the ciphertext and re-encrypts parts of the ciphertext to check for validity. They show how Hinting PRGs can be constructed from number theoretic assumptions such as CDH and LWE using techniques similar to [DG17b, DG17a, CDG$^+$17, BLSV18, DGHM18, GH18].

Our Results

In this work we show a black box approach to construct chosen ciphertext security using just injective trapdoor functions (in addition to primitives known to be implied by TDFs.) We outline our approach, which begins with two abstractions that we will use as building blocks in our construction. These abstractions are called (1) encryption with randomness recovery, and (2) tagged set commitments. We build the first generically from injective trapdoor functions and the latter from pseudorandom generators, which are known to be implied by TDFs. These abstractions are intentionally simple, but useful for building intuition.

Encryption with Randomness Recovery The "Encryption with Randomness Recovery" abstraction is simply an IND-CPA secure public-key encryption where (1) the decryption algorithm recovers both the message and the encryption randomness r and (2) where there is also a $\mathsf{Recover}$ algorithm which can recover the message from a ciphertext given the encryption randomness r. That is, when $\mathsf{Enc}(\mathsf{pk}, m, r) \to \mathsf{ct}$, then $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to (m, r)$ and $\mathsf{Recover}(\mathsf{pk}, \mathsf{ct}, r) \to m$. We formally define this abstraction in Section 3, followed by an immediate construction of Encryption with Randomness Recovery from injective trapdoor functions. Notably Yao's method [Yao82] of achieving encryption from trapdoor functions is actually Encryption with Randomness Recovery for 1-bit messages, where many such ciphertexts can be concatenated together to encrypt many bits.

Tagged Set Commitments The "Tagged Set Commitment" abstraction is a commitment scheme that commits to a B-sized set of indices $S \in [N]$ with a tag tg (where N and B are inputs to a trusted setup algorithm) by producing a commitment together with a membership proof for each $i \in S$; that is, $\mathsf{Commit}(\mathsf{pp}, S, \mathsf{tg}) \to (\mathsf{com}, (\sigma_i)_{i \in S})$. The verification algorithm checks the membership proof to verify that $i \in S$ under tag tg. These algorithms take in a set of public parameters pp generated by a Setup algorithm with a bound B that enforces (the maximum) size of S. Additionally, for proof purposes, the scheme must

[1] Kitagawa and Matsuda [KM19] show how the Hinting PRG assumption can alternatively be replaced with the assumption of symmetric key encryption with key-dependent security.

support an alternative setup algorithm $\mathsf{AltSetup}$ that takes in a tag tg and produces public parameters together with a special commitment and a proof of membership for this commitment for *every* element in the committing domain (which will exceed the bound B that all other commitments must abide by). In addition to the regular soundness property, we will require that no polynomial-time adversary can distinguish between when the parameters were generated by the regular or the alternative setup algorithm. We formally define this abstraction in Section 4, followed by a construction from pseudorandom generators. This abstraction is related to a number of prior works. It can be viewed as a generalization of the commitment scheme used in [KW19] to achieve a generic CCA compiler for attribute-based encryption schemes, which was itself related to Naor's commitment from pseudorandom generators [Nao89].

Our CCA Construction Our construction uses three building blocks: a one-time signature scheme, a CPA-secure encryption scheme with randomness recovery and tagged set commitments. Our construction will create a CCA key that includes N CPA keys. To encrypt a message a user will encrypt it to a subset of the keys. Decryption will then follow the paradigm of recovering randomness from (some of) the CPA encryptions and then re-encrypting to check for validity. Conceptually, it is critical for us to perform a type of balancing act when encrypting the ciphertexts in order to prove security. At one step in the proof we want to have enough redundancy in the way randomness is chosen so that one can decrypt given any $N-1$ of the private keys. However, at a later stage in the proof we want the fact that we choose any redundancy at all to statistically wash away. We sketch our construction below and show how we find this balance.

We begin by noting the parameterization of our scheme. The driving factor will be the length of randomness $\ell_{\mathrm{rnd}} = \ell_{\mathrm{rnd}}(\lambda)$ of the underlying encryption with randomness recovery scheme for security parameter λ. We will choose integers N, B such that $N > B$ and $\binom{N}{B} > 2^{\ell_{\mathrm{rnd}}+\lambda}$. For example, we could let $N = 2(\ell_{\mathrm{rnd}}+\lambda)$ and $B = N/2$.

The CCA setup algorithm initially chooses N key pairs from the CPA with randomness recovery scheme as $(\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$. In addition, it samples the tagged set commitment as $\mathsf{tsc.pp} \leftarrow \mathsf{TSC.Setup}(1^\lambda, 1^N, 1^B, 1^t)$ where t is the length of a verification key in the one-time signature scheme.

To encrypt one first chooses a uniformly random B-size subset $S \subset [N]$. Next, choose a signing/verification key $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \mathsf{Sig.Setup}(1^\lambda)$. And then get a commitment to the set elements as $(\mathsf{tsc.com}, (\mathsf{tsc.}\sigma_i)_{i\in S}) \leftarrow \mathsf{TSC.Commit}(\mathsf{tsc.pp}, S, \mathsf{sig.vk})$. At this point the encryptor will select the randomness used for encryption. For all $i \in S$ choose $r_i \in \{0,1\}^{\ell_{\mathrm{rnd}}}$ uniformly at random with the constraint that these values XOR to $0^{\ell_{\mathrm{rnd}}}$. Observe that this slight redundancy implies that for a correctly formed ciphertext if we are given the set S along with the r_i values for $B-1$ of the indices in S, then we can derive the last one by simply XORing all the others together. For $i \notin S$ simply choose r_i at random.

To finalize encryption for $i \in [N]$, if $i \in S$ encrypt the message along with proof for index i as $\mathsf{cpa.ct}_i = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|\mathsf{tsc.}\sigma_i|m; r_i)$. Otherwise for $i \notin S$ encrypt the all 0's string as $\mathsf{cpa.ct}_i = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 0^{\ell_{\mathsf{cpa}}}; r_i)$. Finally, sign $\left(\mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$ with $\mathsf{sig.sk}$ to get $\mathsf{sig.}\sigma$ and output the ciphertext ct as $\left(\mathsf{sig.vk}, \mathsf{sig.}\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$.

The decryption algorithm on a ciphertext $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig.}\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$ will first verify the signature and reject if that fails. Next, it will initialize a set $U = \emptyset$ and use the $\mathsf{cpa.sk}_i$ to decrypt all $\mathsf{cpa.ct}_i$ using the respective $\mathsf{cpa.sk}_i$. For each $i \in [N]$, it gets a message y_i which is parsed as $g_i|\sigma_i|m_i$ and randomness r_i. The decryption algorithm adds (i, y_i) to U if decryption is successful and (1) $\mathsf{TSC.Verify}(\mathsf{tsc.pp}, \mathsf{tsc.com}, i, \mathsf{tsc.}\sigma_i, \mathsf{sig.vk}) = 1$ and (2) $\mathsf{cpa.ct}_i = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, y_i; r_i)$. It then checks that there are exactly B entries in the set U, they all encrypt the same message and that $\oplus_{(i,y_i)\in U}\ r_i = 0^\ell$. If so, it outputs the message. We emphasize that the decryption algorithm both checks the well formness of ciphertext components in U via re-encryption and checks for the redundancy in randomness via the XOR operation. However, ciphertext components outside of the set U are not verified in this way. Indeed, the algorithm will allow decryption to proceed even it "knows" some components outside of U were malformed.

Our proof is given as a sequence of games where we show that for any poly-time attacker the advantage of the attacker must be negligibly close in successive games. We sketch the proof at high level here and refer the reader to the main body for details.

1. In the first step of our proof the decryption algorithm rejects all ciphertexts that come with a signature under $\mathsf{sig.vk}^*$ where $\mathsf{sig.vk}^*$ is the signing key of the challenge ciphertext. This step is proven via a standard reduction to a *strongly* secure one-time signature scheme.

2. In the next security game the set commitment parameters are chosen via the alternate setup algorithm as: $\left(\mathsf{tsc.com}^*, (\mathsf{tsc}.\sigma_i)_{i\in[N]}\right) \leftarrow \mathsf{AltSetup}(1^\lambda, 1^N, 1^B, 1^t, \mathsf{sig.vk}^*)$. This means that for the tag $\mathsf{sig.vk}^*$ (and only the tag $\mathsf{sig.vk}^*$) proof values exist for every single index in $[N]$. However, in the challenge ciphertext $\mathsf{tsc}.\sigma_i$ are only used for $i \in S^*$ where S^* is the set used in creating the challenge ciphertext.

3. In our proof for all indices $i \notin S^*$ we will want to change $\mathsf{cpa.ct}_i$ from an encryption of the all 0's string to an encryption of $1|\mathsf{tsc}.\sigma_i^*|m_b$. We will change these one at a time. Suppose we want to argue that no attacker can detect such a change on the j-th index. To prove this we need a reduction that will not have access to the j-th secret key $\mathsf{cpa.sk}_j$, but will still be able to decrypt in an equivalent (but not identical) manner to the original decryption algorithm. To do this the alternative decryption algorithm uses all $N-1$ secret keys that it has to build a partial set U as in the actual decryption algorithm above. It then branches its behavior on the size of U: (1) If $|U| > B$, then reject. In this case the missing j-th component can only add to the size of U which is already too big and will be rejected. (2) If $|U| < B-1$, then reject. The missing j-th component can make the set size at most $B-1$ which is too small and will be rejected. (3) If $|U| = B$, then proceed with the remaining checks of decryption using the set U and ignore the j-th component. By soundness of the tagged set commitment scheme, this could not have contained $\mathsf{tsc}.\sigma_j$ for a tag $\mathsf{sig.vk} \neq \mathsf{sig.vk}^*$ so we can safely ignore the j-th component. (4) If $|U| = B-1$, compute $r_j = \oplus_{i\in U} r_i$ and use this candidate randomness to decrypt $\mathsf{cpa.ct}_j$ in lieu of the key $\mathsf{cpa.sk}_j$. Once this step is done, the result can be added (or not) to the set U and the rest of decryption proceeds as before. We can show that the required redundancy checks make this decryption case equivalent to the original as well.

   Once this proof step has occurred for all $j \in [N]$ we have that each message is $1|\mathsf{tsc}.\sigma_i^*|m_b$, but that the challenge ciphertext has the redundancy in the randomness $\oplus_{i\in S^*} r_i = 0^{\ell_{\mathrm{rnd}}}$.

4. For the next game we want to remove the redundancy in the randomness so that r_i is chosen uniformly at random for all indices i. It turns out that by the setting of our parameters this is statistically already done! A random set of r_i variables will have a $\frac{1}{2^{\ell_{\mathrm{rnd}}}}$ chance of XORing to 0^ℓ_{rnd}. Thus, we could then expect there will be approximately $\binom{N}{B} \cdot \frac{1}{2^{\ell_{\mathrm{rnd}}}}$ sets of size B that satisfy this condition if all r_i are chosen randomly. Recall that since we set $\binom{N}{B} > 2^{\ell_{\mathrm{rnd}}+\lambda}$ we might then expect there to be an exponential amount of sets meeting this condition. Therefore we would intuitively expect that planting a single set S^* with this condition and choosing all r_i randomly will be statistically close. In the main body, we formalize this intuition by applying the Leftover Hash Lemma [HILL99].

5. Now that the randomness in the challenge ciphertext is uncorrelated we want to change all encryptions from $1|\mathsf{tsc}.\sigma_i^*|m_b$ to 0^ℓ_{rnd}. This can be done by a hybrid over all j from 1 to N. (At this point in the security game there is no set S^*.) This is done by again using an alternative decryption algorithm that can decrypt using all but the j-th secret key.

Stepping back we can see that the XORing to 0^ℓ_{rnd} condition on S gave enough redundancy where one could decrypt with all but one of the keys allowing Steps 3 and 5 of the proof above to proceed. However, the redundant condition was limited enough where it could be statistically washed away in Step 4 of the proof.

A further comparison to Koppula-Waters (CRYPTO 2019) We provide a closer comparison between our work and that of Koppula and Waters [KW19]. To do so we will imagine modifying our scheme above and arrive at something analogous to [KW19]. Suppose that instead of choosing the values r_i in the set S at random with $\oplus_{i\in S^*} r_i = 0^{\ell_{\mathrm{rnd}}}$, we instead ran a pseudorandom generator (of output length $B \cdot \ell_{\mathrm{rnd}}$) on S as $\mathsf{PRG}(S)$ to determine the r_i values for $i \in S$. The r_i for $i \notin S$ are random as before.

Using this encryption algorithm, one can create an analogous decryption algorithm that first recovers a candidate set U almost as before. However, instead of getting the random coins r_i from decryption once it has U, the decryption algorithm can run $\mathsf{PRG}(U)$ to determine the candidate set of r_i values. At this point it can perform the same re-encryption and other checks as we outlined above. Indeed, the underlying encryption system does not even need to have randomness recovery and thus is not necessarily trapdoor based.

If we try to prove this system secure, we can mostly march along the same steps as above, but we hit a roadblock at Step 4. In our construction we argue that choosing random r_i is statistically close to embedding the XOR condition. Is this true in the modified construction? Let's imagine an arbitrary B-sized subset S of indices with randomly chosen r_i. The probability that $\mathsf{PRG}(S)$ outputs these r_i values is $2^{-\ell_{\mathrm{rnd}} \cdot B}$. Even though there are $\binom{N}{B}$ sets of size B, the chances of there being just one of these subsets that meets this condition is still negligibly small. Thus we cannot make a statistical argument.

To get past Step 4 in the modified construction then, we will be forced to contrive an assumption that these two distributions are computationally indistinguishable. Conceptually, this assumption is very analogous to the "Hinting PRG" assumption introduced by Koppula and Waters. Altogether, our techniques address the main limitation of [KW19] which was the need for a "Hinting PRG" by creating an encryption scheme with less redundancy in the randomness. This allows us to bridge over a critical proof step with a statistical argument.

1.1 Context on Trapdoor Functions

We conclude by providing some more context on trapdoor functions.

Constructions For many years the only known standard model technique for getting trapdoor functions was to use an assumption like RSA [RSA78] that immediately gives a trapdoor function. Peikert and Waters [PW08] gave the first standard model constructions for trapdoor functions from the DDH and the LWE assumptions. More recently, Garg and Hajiabadi [GH18] and Garg, Gay and Hajiabadi [GGH] gave constructions from the Computational Diffie-Hellman assumption.

On (Im)Perfect Correctness We observe that our security argument above relies on the trapdoor function to be perfectly correct when switching from the original decryption algorithm to the alternative decryption algorithm. Otherwise, an attacker could potentially detect the change by constructing a ciphertext component which is well formed, but does not decrypt correctly. (Even if the encryption with randomness recovery correctness error is negligible for randomly sampled coins, it might be easy to adversarially discover bad ciphertexts.) This creates an issue for schemes such as [GH18, GGH] that are not perfectly correct.

To address this issue, we recall the notion of almost-all-keys perfect correctness in encryption schemes, introduced by Dwork et al. [DNR04]. In an almost-all-keys perfectly correct scheme, the key generation algorithm $\mathsf{Setup}(1^\lambda)$ will sample a public, private key pair $(\mathsf{pk}, \mathsf{sk})$ such that, with all but negligible probability, these particular keys will work perfectly. That is, any message m and coins r used for encryption by pk will decrypt to m using sk. (This is a stronger notion of (imperfect) correctness than the usual one where potentially every public, secret key pair has a messages and coin pairs that cause decryption failures.) We observe that almost-all-keys correctness is sufficient for our proof of security to go through. Since the attacker has no influence on the key generation algorithm, with all but negligible probability, he/she will be stuck with a keypair that has perfect correctness.

The CDH based scheme of Garg, Gay and Hajiabadi [GGH] satisfies almost-all-keys perfect correctness. However, for the scheme of Garg and Hajiabadi [GH18], it is not clear if the above approach can directly work.[2] One might hope to use the transformation of [DNR04] to go from an imperfectly correct encryption

[2] In [GH18], it appears that it is computationally difficult for an attacker to discover a TDF input $\mathbf{x}$ where $\mathbf{y} = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, \mathbf{x})$ and $\mathsf{TDF.Invert}(\mathsf{tdf.sk}, \mathbf{y}) \neq \mathbf{x}$. We believe this property is also sufficient for our CCA transformation, but do not show this formally.

scheme to one that satisfies almost-all-keys perfect correctness. Unfortunately this does not appear to work as we require encryption *with randomness recovery.*

TDFs with a Sample Algorithm The work of Bellare et. al.[BHSV98] as well as the Katz-Lindell [KL08] textbook provide an alternative definition to trapdoor functions. In the standard definition the domain is simply all the strings of length ℓ_{inp} and the security experiment chooses $\mathbf{x} \in \{0,1\}^{\ell_{\mathsf{inp}}}$ to evaluate the trapdoor function on. In the alternative "sampling" definition there is an additional algorithm Sample that takes the public key along with random coins and outputs an element $\mathbf{x}$ in the domain. The TDF evaluation algorithm can then be run on $\mathbf{x}$ to give $\mathsf{TDF.Eval}(\mathsf{tdf.pk}, \mathbf{x}) \to \mathbf{y}$. Notably, the domain can depend on the public key and while correctness stipulates that $\mathsf{TDF.Invert}(\mathsf{tdf.sk}, \mathbf{y}) \to \mathbf{x}$, there is *no requirement to recover the coins of the* Sample *algorithm.*

At first glance it might appear that the differences in these two definitions is conceptually minor. However, these nuances are actually very important. As observed by Pandey [Pan13] there exists a trivial construction of the sampling form of trapdoor functions from public key encryption. The public and secret key of the trapdoor function will come from the PKE key generation algorithm. The Sample algorithm will choose a random message m of sufficient length and output an encryption ct of m under the public key to give $\mathbf{x} = (\mathsf{ct}, m)$. The $\mathsf{TDF.Eval}$ algorithm can simply drop m. That is $\mathsf{TDF.Eval}(\mathsf{tdf.pk}, \mathbf{x} = (\mathsf{ct}, m)) \to \mathsf{ct}$. And the inversion algorithm can recover (ct, m) from ct by simply decrypting. Security follows immediately from the IND-CPA security of the underlying encryption scheme.

If we want the Sample algorithm to sample uniformly in the domain, we will need two additional properties of the encryption algorithm. First, that for every public key pk and every pair of messages (m_1, m_2) the number of distinct ciphertexts that can be generated from encrypting m_1 under pk is the same as the number that can be generated by encrypting m_2 under pk. And that for any pk and message m the likelihood of any ciphertext that is in the support of encrypting m under pk is the same.

This construction feels like a cheat as it does not match our intuitive concept of what a trapdoor function is. It takes advantage of the fact that one is not required to recover the random coins used in the Sample algorithm. Thus the definition essentially allows for one to dispense with the recovery of coins requirement and seems to loose the spirit of trapdoor functions. An interesting question is whether such a transformation could be done in a definition where the Sample algorithm only took as input the security parameter and not the TDF's public key.

Looking Forward It is interesting to think what implications our work might have on the ultimate question of whether chosen plaintext security implies chosen ciphertext security. An immediate barrier is that there are black box separations on building TDFs from PKE [GMR01]. However, it might be possible to leverage our construction or lessons from it from an abstraction that delivers "most" of the properties of a TDF.

2 Preliminaries

For any positive integer n, let $[n]$ denote the set of integers $\{1, 2, \ldots, n\}$. For any prime p and positive integer ℓ, let $\mathbb{F}_{p^\ell}$ denote the (unique) field of order p^ℓ. We will use bold letters to denote a vector/array of elements, and subscript i denotes the i^{th} element (e.g. if $\mathbf{w} \in \{0,1\}^n$, then w_i denotes the i^{th} bit). Given two distributions $\mathcal{D}_1, \mathcal{D}_2$ over finite domain $\mathcal{X}$, let $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2)$ denote the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$.

Definition 2.1 (Pseudorandom Generator)**.** Let $n, \ell \in \mathbb{N}$ and let PRG be a deterministic polynomial-time algorithm such that for any $s \in \{0,1\}^n$, $\mathsf{PRG}(s, 1^\ell)$ outputs a string of length ℓ. (Here, we will not require that ℓ be polynomial in n.) We say that PRG is a *pseudorandom generator* if for all probabilistic polynomial-time distinguishers D, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $n, \ell, \lambda \in \mathbb{N}$,

$$|\Pr[D(r) = 1] - \Pr\left[D(\mathsf{PRG}(s, 1^\ell)) = 1\right]| \leq \mathsf{negl}(\lambda),$$

where r is chosen uniformly at random from $\{0,1\}^\ell$, s is chosen uniformly at random from $\{0,1\}^n$, and the probabilities are taken over the choice of r and s and the coins of D.

Definition 2.2 (Strongly Unforgeable One-Time Signature [Lam79])**.** Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-time signature scheme for the message space M. Consider the following probabilistic experiment $\mathsf{SU\text{-}OTS}(\Sigma, \mathcal{A}, \lambda)$ with $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and $\lambda \in \mathbb{N}$:

$\mathsf{SU\text{-}OTS}(\Pi, \mathcal{A}, \lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m, z) \leftarrow \mathcal{A}_1(\mathsf{pk})$ s.t. $m \in M$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}_2(\sigma, z)$
Output 1 if ($m \neq m^*$ and $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$) or ($\sigma \neq \sigma^*$ and $\mathsf{Verify}(\mathsf{pk}, m, \sigma^*) = 1$) and 0 otherwise.

Signature scheme Σ is $\mathsf{SU\text{-}OTS}$-secure if $\forall$ p.p.t. algorithms $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr\left[\mathsf{SU\text{-}OTS}(\Pi, \mathcal{A}, \lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where this probability is taken over all random coins used in the experiment.

Definition 2.3 ($\mathsf{IND\text{-}CPA}$ [GM84])**.** Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme for the message space M. Consider the following probabilistic experiment $\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \lambda)$ with $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and $\lambda \in \mathbb{N}$:

$\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m_0, m_1, z) \leftarrow \mathcal{A}_1(\mathsf{pk})$ s.t. $m_0, m_1 \in M$
$y \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow \mathcal{A}_2(y, z)$
Output 1 if $b' = b$ and 0 otherwise.

Encryption scheme Π is $\mathsf{IND\text{-}CPA}$-secure if $\forall$ p.p.t. algorithms $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr\left[\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where this probability is taken over all random coins used in the experiment.

Definition 2.4 ($\mathsf{IND\text{-}CCA}$ [NY90, RS91, DDN00])**.** Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme for the message space M and let experiment $\mathsf{IND\text{-}CCA}(\Pi, \mathcal{A}, \lambda)$ be identical to $\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \lambda)$ except that both $\mathcal{A}_1$ and $\mathcal{A}_2$ have access to an oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$ that returns the output of the decryption algorithm and $\mathcal{A}_2$ cannot query this oracle on input y. Encryption scheme Π is $\mathsf{IND\text{-}CCA}$-secure if $\forall$ p.p.t. algorithms $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr\left[\mathsf{IND\text{-}CCA}(\Pi, \mathcal{A}, \lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where this probability is taken over all random coins used in the experiment.

Injective Trapdoor Functions An injective trapdoor function family $\mathcal{T}$ with input space $\{0,1\}^{\ell_{\mathsf{inp}}}$ and output space $\{0,1\}^{\ell_{\mathrm{out}}}$, where ℓ_{inp} and ℓ_{out} are polynomial functions of the security parameter λ, consists of three PPT algorithms with syntax:

- $\mathsf{TDF.Setup}(1^\lambda) \to (\mathsf{tdf.pk}, \mathsf{tdf.sk})$: The setup algorithm takes as input security parameter λ and outputs a public key $\mathsf{tdf.pk}$ and secret key $\mathsf{tdf.sk}$.
- $\mathsf{TDF.Eval}(\mathsf{tdf.pk}, x \in \{0,1\}^{\ell_{\mathsf{inp}}},) \to y$: The evaluation algorithm takes as input an input $x \in \{0,1\}^{\ell_{\mathsf{inp}}}$ and public key $\mathsf{tdf.pk}$, and outputs $y \in \{0,1\}^{\ell_{\mathrm{out}}}$.
- $\mathsf{TDF.Invert}(\mathsf{tdf.sk}, y \in \{0,1\}^{\ell_{\mathrm{out}}}) \to x \in \{0,1\}^{\ell_{\mathsf{inp}}} \cup \{\perp\}$: The inversion algorithm takes as input $y \in \{0,1\}^{\ell_{\mathrm{out}}}$ and secret key $\mathsf{tdf.sk}$, and outputs x, which is either $\perp$ or a ℓ_{inp}-bit string.

Almost-all-keys Injectivity We require that for nearly all public/secret keys, inversion works for all inputs. More formally, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\exists\ x \text{ s.t. } \mathsf{TDF.Invert}(\mathsf{TDF.Eval}(\mathsf{tdf.pk}, x), \mathsf{tdf.sk}) \neq x\right] \leq \mathsf{negl}(\lambda),$$

where this probability is over the choice of $(\mathsf{tdf.pk}, \mathsf{tdf.sk}) \leftarrow \mathsf{TDF.Setup}(1^\lambda)$.

Definition 2.5. An injective trapdoor family is hard-to-invert if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[x \leftarrow \mathcal{A}(\mathsf{tdf.pk}, y) : \begin{array}{c}(\mathsf{tdf.pk}, \mathsf{tdf.sk}) \leftarrow \mathsf{TDF.Setup}(1^\lambda) \\ x \leftarrow \{0,1\}^{\ell_{\mathsf{inp}}}, y = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, x)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Define $(r \cdot x) = \oplus_{i=1}^{n} r_i \cdot x_i$ where $r = r_1 \ldots r_n$ and $x = x_1 \ldots x_n$. The Goldreich-Levin theorem for hard-core predicates [GL89] states that no polynomial time algorithm can compute $(r \cdot x)$ given a random r, the TDF public key $\mathsf{tdf.pk}$ and evaluation $\mathsf{TDF.Eval}(\mathsf{tdf.pk}, x)$ on random input x, where $|r| = |x|$.

Theorem 2.1 (Goldreich-Levin Hardcore Bit [GL89])**.** Assuming TDF is an injective trapdoor family, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left|\Pr\left[b \leftarrow \mathcal{A}(\mathsf{tdf.pk}, s, y, z_b) : \begin{array}{c}(\mathsf{tdf.pk}, \mathsf{tdf.sk}) \leftarrow \mathsf{TDF.Setup}(1^\lambda) \\ x, s \leftarrow \{0,1\}^{\ell_{\mathsf{inp}}}, y = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, x) \\ z_0 = s \cdot x, z_1 \leftarrow \{0,1\}, b \leftarrow \{0,1\}\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

3 Encryption Scheme with Randomness Recovery

An encryption scheme with randomness recovery is an IND-CPA secure encryption scheme with two additional properties: (a) the decryption algorithm can be used to recover the message as well as the randomness used for encryption (b) the randomness used for encryption can be used to decrypt the ciphertext. Formally, it consists of four PPT algorithms with the following syntax. Here the message length ℓ_{msg} and the length of the randomness ℓ_{rnd} are polynomial functions of the security parameter λ.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: The setup algorithm takes as input the security parameter λ and outputs a public key pk and secret key sk.

$\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$: The encryption algorithm is randomized; it takes as input a public key pk and a message m, uses ℓ_{rnd} bits of randomness and outputs a ciphertext ct. We will sometimes write $\mathsf{Enc}(\mathsf{pk}, m; r)$, which runs $\mathsf{Enc}(\mathsf{pk}, m)$ using r as the randomness.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to z \in \left(\{0,1\}^{\ell_{\mathrm{msg}}} \times \{0,1\}^{\ell_{\mathrm{rnd}}}\right) \cup \{\perp\}$: The decryption algorithm takes as input a secret key sk and a ciphertext ct, and either outputs $z = \perp$ or $z = (m, r)$ where $m \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $r \in \{0,1\}^{\ell_{\mathrm{rnd}}}$.

$\mathsf{Recover}(\mathsf{pk}, \mathsf{ct}, r) \to z \in \{0,1\}^{\ell_{\mathrm{msg}}} \cup \{\perp\}$: The recovery algorithm takes as input a public key pk, a ciphertext ct and string $r \in \{0,1\}^{\ell_{\mathrm{rnd}}}$. It either outputs $\perp$ or a message $m \in \{0,1\}^{\ell_{\mathrm{msg}}}$.

These algorithms must satisfy the following almost-all-keys perfect correctness property.

Almost-all-keys Perfect Correctness We require perfect correctness of decryption and recovery for all but a negligible fraction of $(\mathsf{pk}, \mathsf{sk})$ pairs. More formally, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any security parameter λ,

$$\Pr\left[\exists\ m, r \text{ s.t. } \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r)) \neq (m, r)\right] \leq \mathsf{negl}(\lambda) \quad \text{and}$$
$$\Pr\left[\exists\ m, r \text{ s.t. } \mathsf{Recover}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m; r), r) \neq m\right] \leq \mathsf{negl}(\lambda)$$

where $m \in \{0,1\}^{\ell_{\mathsf{cpa}}}$, $r \in \{0,1\}^{\ell_{\mathrm{rnd}}}$, and the probability is over the choice of $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$.

Koppula and Waters [KW19] defined a notion of "recovery *from* randomness" which has the above almost-all-keys perfect correctness requirement on the $\mathsf{Recover}$ algorithm, but not also on the Dec algorithm.

3.1 Construction: Encryption Scheme with Randomness Recovery from Injective TDFs

We show an IND-CPA secure encryption scheme with randomness recovery for messages of length ℓ_{msg} where encryption uses ℓ_{rnd}-bits of randomness based on injective trapdoor functions. This construction is closely related to the CPA-secure encryption scheme of Yao [Yao82]. Let $\mathsf{tdf} = (\mathsf{TDF.Setup}, \mathsf{TDF.Eval}, \mathsf{TDF.Invert})$ be an injective trapdoor function (see Section 2) with input space $\{0,1\}^{\ell_{\mathsf{inp}}}$ and output space $\{0,1\}^{\ell_{\text{out}}}$. Here ℓ_{inp}, ℓ_{out}, ℓ_{msg} and $\ell_{\text{rnd}} = \ell_{\text{msg}} \cdot \ell_{\mathsf{inp}}$ are polynomial functions in the security parameter λ.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: The setup algorithm first chooses $(\mathsf{tdf.pk}, \mathsf{tdf.sk}) \leftarrow \mathsf{TDF.Setup}(1^\lambda)$. Next, it choses a uniformly random string $t \leftarrow \{0,1\}^{\ell_{\mathsf{inp}}}$. The public key is set to be $\mathsf{pk} = (\mathsf{tdf.pk}, t)$ and the secret key is $\mathsf{sk} = (\mathsf{tdf.sk}, t)$.

$\mathsf{Enc}\left(\mathsf{pk} = (\mathsf{tdf.pk}, t), \mathbf{m} = (m_1, \ldots, m_{\ell_{\text{msg}}})\right) \to \mathsf{ct}$: For each $i \in [\ell_{\text{msg}}]$, the encryption algorithm:

- chooses a random string $r_i \leftarrow \{0,1\}^{\ell_{\mathsf{inp}}}$.
- sets $\mathsf{ct}_{1,i} = (r_i \cdot t) + m_i$ and $\mathsf{ct}_{2,i} = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, r_i)$.

For $w \in \{0,1\}$, it sets $\mathbf{ct}_w = (\mathsf{ct}_{w,1}, \ldots, \mathsf{ct}_{w,\ell_{\text{msg}}})$ and outputs $(\mathbf{ct}_1, \mathbf{ct}_2)$.

$\mathsf{Dec}\left(\mathsf{sk} = (\mathsf{tdf.sk}, t), \mathsf{ct} = (\mathbf{ct}_1, \mathsf{ct}_2)\right) \to z$: For each $i \in [\ell_{\text{msg}}]$, the decryption algorithm computes $r_i = \mathsf{TDF.Invert}(\mathsf{tdf.sk}, \mathsf{ct}_{2,i})$. If $r_i = \perp$, it outputs $\perp$ and aborts. Else, it sets $m_i = \mathsf{ct}_{1,i} + (r_i \cdot t) (\text{mod } 2)$.

Finally, it outputs $\mathbf{m} = (m_1, \ldots, m_{\ell_{\text{msg}}})$ and $\mathbf{r} = (r_1, \ldots, r_{\ell_{\text{msg}}})$.

$\mathsf{Recover}\left(\mathsf{pk} = (\mathsf{tdf.pk}, t), \mathsf{ct} = (\mathbf{ct}_1, \mathbf{ct}_2), \mathbf{r}\right) \to z$: The recovery algorithm performs the following for each $i \in [\ell_{\text{msg}}]$: it computes $z_i = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, r_i)$. If $z_i \neq \mathsf{ct}_{2,i}$, it outputs $\perp$ and aborts. Else it sets $m_i = \mathsf{ct}_{1,i} + z_i (\text{mod } 2)$.

Finally it outputs $\mathbf{m} = (m_1, \ldots, m_{\ell_{\text{msg}}})$.

Almost-all-keys perfect correctness follows from the almost-all-keys perfect injectivity TDFs.

Encrypting long messages In the construction above the number of random bits, ℓ_{rnd} required by encryption grows linearly in the message size as $\ell_{\text{rnd}} = \ell_{\text{msg}} \cdot \ell_{\mathsf{inp}}$. We observe that to encrypt long messages we could instead use the system above to encrypt a PRG seed $k \in \{0,1\}^\lambda$ and then encrypt the message itself as $\mathsf{PRG}(k) \oplus m$ for a pseudorandom generator of appropriate output length. This hybrid encryption method would maintain the randomness recovery property, but the growth of the random coins would be independent of the message length.

3.1.1 IND-CPA Security

Theorem 3.1. The Section 3.1 construction is IND-CPA-secure (per Definition 2.3) assuming TDF is a hard-to-invert injective trapdoor family (per Definition 2.5).

The proof of security follows via a simple sequence of hybrid experiments $\{H_j\}_{j \in \{0,\ldots,\ell_{\text{msg}}+1\}}$ defined as follows. H_0 corresponds to the IND-CPA experiment of the construction in Section 3.1, While in hybrid $H_{\ell_{\text{msg}}+1}$, the adversary will have advantage 0.

Hybrid H_j with security parameter λ, for $j \in \{1, \ldots, \ell_{\text{msg}} + 1\}$:

- The challenger chooses $(\mathsf{tdf.pk}, \mathsf{tdf.sk}) \leftarrow \mathsf{TDF.Setup}(1^\lambda)$ and a random string $t \leftarrow \{0,1\}^{\ell_{\mathsf{inp}}}$. It sends $(\mathsf{tdf.pk}, t)$ to the adversary.
- On receiving challenge messages $\mathbf{m_0}, \mathbf{m_1} \in \{0,1\}^{\ell_{\text{msg}}}$ from the adversary, the challenger chooses $b \leftarrow \{0,1\}$. Next, it chooses $r_i \leftarrow \{0,1\}^{\ell_{\mathsf{inp}}}$ for all $i \in [\ell_{\text{msg}}]$. For $i < j$, it sets $\mathsf{ct}_{1,i}$ uniformly at random. For $i \geq j$, it sets $\mathsf{ct}_{1,i} = (r_i \cdot t) + m_{b,i}$. In either case, it sets $\mathsf{ct}_{2,i} = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, r_i)$ It sends $(\mathbf{ct}_1, \mathbf{ct}_2)$ to the adversary and receives guess b'. Adversary wins if $b = b'$.

Analysis: For any PPT adversary $\mathcal{A}$, let $\mathsf{adv}_{\mathcal{A},j}(\lambda)$ denote the advantage of $\mathcal{A}$ in H_j (with sec. par. λ).

Claim 3.1. Assuming the hard-to-invert property of the injective TDF family $\mathcal{T}$ (see Definition 2.5), for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and $j \in [0, \ell_{\mathrm{msg}}]$, $\mathsf{adv}_{\mathcal{A},j}(\lambda) - \mathsf{adv}_{\mathcal{A},j+1}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ and $0 \leq j \leq \ell_{\mathrm{msg}}$ such that $\mathsf{adv}_{\mathcal{A},j} - \mathsf{adv}_{\mathcal{A},j+1} = \epsilon$, where ϵ is non-negligible.[3] Then there exists a PPT algorithm $\mathcal{B}$ that breaks the hardcore bit property of $\mathcal{T}$, since this property follows from the hard-to-invert property of $\mathcal{T}$ and Theorem 2.1, we have a contradiction. The algorithm $\mathcal{B}$ receives $(\mathsf{tdf.pk}, s, y, z)$ from the challenger, where $y = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, x)$ for a uniformly random $x \in \{0,1\}^{\ell_{\mathsf{inp}}}$, and z is either $(s \cdot x)$ or a uniformly random bit. $\mathcal{B}$ sets $t = s$ and sends $(\mathsf{tdf.pk}, t)$ to $\mathcal{A}$, and receives challenge messages $\mathbf{m_0}, \mathbf{m_1}$. The reduction then chooses $w \leftarrow \{0,1\}$. For all $i \neq j$, the challenge ciphertext components $\mathsf{ct}_{1,i}, \mathsf{ct}_{2,i}$ are identically distributed, and the reduction algorithm can compute them using $\mathsf{tdf.pk}$ and t. It sets $\mathsf{ct}_{j,1} = z + m_{w,j}$ and $\mathsf{ct}_{j,2} = y$, and sends $(\mathbf{ct}_1, \mathbf{ct}_2)$ to $\mathcal{A}$. The adversary sends its guess w'. If $w = w'$, then the reduction $\mathcal{B}$ outputs 0 (indicating that $z = s \cdot x$), else it outputs 1 (indicating that z is uniformly random).

Note that if $y = \mathsf{TDF.Eval}(\mathsf{tdf.pk}, x)$ and $z = (s \cdot x)$, then this corresponds to H_j; if z is uniformly random, then this corresponds to H_{j+1}. Let $\mathsf{adv}^{\mathcal{T}}_{\mathcal{B}}$ denote $\mathcal{B}$'s advantage in the hardcore bit experiment against $\mathcal{T}$.

$$\begin{aligned}\mathsf{adv}^{\mathcal{T}}_{\mathcal{B}} &= \Pr[\mathcal{B} \text{ outputs } 0 \mid z = (s \cdot x)] - \Pr[\mathcal{B} \text{ outputs } 0 \mid z \text{ is random}] \\ &= \Pr[\mathcal{A} \text{ wins in } H_j] - \Pr[\mathcal{A} \text{ wins in } H_{j+1}] = \epsilon.\end{aligned}$$

□

4 Tagged Set Commitment

We introduce an abstraction called a "tagged set commitment" and show that it can be constructed generically from a pseudorandom generator. We employ this abstraction shortly in our Section 5 construction.

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

$\mathsf{Commit}(\mathsf{pp}, S \subseteq [N], \mathsf{tg} \in \{0,1\}^t) \to (\mathsf{com}, (\sigma_i)_{i \in S})$: The commit algorithm is randomized; it takes as input the public parameters pp, set S of size B and string tg, and outputs a commitment com together with 'proofs' σ_i for each $i \in S$. [4]

$\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, i \in [N], \sigma_i, \mathsf{tg} \in \{0,1\}^t) \to \{0,1\}$: The verification algorithm takes as input the public parameters, an index i, a proof σ_i, and tg. It outputs 0/1.

$\mathsf{AltSetup}(1^\lambda, 1^N, 1^B, 1^t, \mathsf{tg}) \to (\mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in [N]})$: The scheme also has an 'alternate setup' which is used in the proof. It takes the same inputs as Setup together with a special tag tg, and outputs public parameters pp, commitment com together with proofs σ_i for all $i \in [N]$.

These algorithms must satisfy the following perfect correctness requirements:

Correctness of Setup and Commit: For all λ, N, $B \leq N$, t, $\mathsf{tg} \in \{0,1\}^t$ and set $S \subseteq [N]$ of size B, if $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^B, 1^t)$ and $(\mathsf{com}, (\sigma_i)_{i \in S}) \leftarrow \mathsf{Commit}(\mathsf{pp}, S, \mathsf{tg})$, then for all $i \in S$, $\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, i, \sigma_i, \mathsf{tg}) = 1$.

Correctness of AltSetup: For all λ, N, $B \leq N$, t, $\mathsf{tg} \in \{0,1\}^t$, if $(\mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in [N]}) \leftarrow \mathsf{AltSetup}(1^\lambda, 1^N, 1^B, 1^t, \mathsf{tg})$, then for all $i \in [N]$, $\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, i, \sigma_i, \mathsf{tg}) = 1$.

[3] We drop dependence on λ for notational convenience.

[4] We require S to be of size exactly B for simplicity of presentation, however, one could generalize this to allow S to be of size at most B.

Security We require two security properties of a tag set commitment.

Indistinguishability of Setup : In this experiment, the adversary chooses a tag tg, set S and receives either public parameters, together with commitments for (S, tg), or receives public parameters and commitment/proofs (corresponding to set S) generated by $\mathsf{AltSetup}$ (for tag tg). The scheme satisfies indistinguishability of setup if no PPT adversary can distinguish between the two scenarios. This experiment is formally defined below.

Definition 4.1. A tagged set commitment scheme $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify}, \mathsf{AltSetup})$ satisfies indistinguishability of setup if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathsf{Expt\text{-}Ind\text{-}Setup}_{\mathcal{A}}(\lambda)] - 1/2| \leq \mathsf{negl}(\lambda)$, where $\mathsf{Expt\text{-}Ind\text{-}Setup}_{\mathcal{A}}$ is defined in Figure 1.

$\mathsf{Expt\text{-}Ind\text{-}Setup}_{\mathcal{A}}(\lambda)$

1. Adversary $\mathcal{A}$ receives input 1^λ and sends $1^N, 1^B, 1^t, \mathsf{tg}, S$ such that $B \leq N$, $\mathsf{tg} \in \{0,1\}^t$ and $|S| = B$.
2. Challenger chooses $b \leftarrow \{0,1\}$. It computes $\mathsf{pp}^0 \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^B, 1^t)$ and $(\mathsf{com}^0, (\sigma_i^0)_{i \in S}) \leftarrow \mathsf{Commit}(\mathsf{pp}, S, \mathsf{tg})$, and $(\mathsf{pp}^1, \mathsf{com}^1, (\sigma_i^1)_{i \in [N]}) \leftarrow \mathsf{AltSetup}(1^\lambda, 1^N, 1^B, 1^t, \mathsf{tg})$. It sends $(\mathsf{pp}^b, \mathsf{com}^b, (\sigma_i^b)_{i \in S})$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs its guess b'. The experiment outputs 1 iff $b = b'$.

Figure 1: Experiment for Indistinguishability of Setup

Soundness Security : The soundness property informally states that if public parameters are generated for bound B (using either regular setup or $\mathsf{AltSetup}$), then no PPT adversary can produce a commitment with greater than B 'proofs'. However, for our CCA application, we need a stronger guarantee: if the challenger generates the public parameters for a tag tg using $\mathsf{AltSetup}$ and the adversary gets all N proofs, even then it cannot generate a commitment with $B+1$ proofs for a different tag tg'.

Definition 4.2. A tagged set commitment scheme $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify}, \mathsf{AltSetup})$ satisfies soundness security if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[1 \leftarrow \mathsf{Expt\text{-}Sound}_{\mathcal{A}}(\lambda)] \leq \mathsf{negl}(\lambda)$, where $\mathsf{Expt\text{-}Sound}_{\mathcal{A}}$ is defined in Figure 2.

$\mathsf{Expt\text{-}Sound}_{\mathcal{A}}(\lambda)$

1. Adversary $\mathcal{A}$ receives input 1^λ, sends $1^N, 1^B, 1^t, \mathsf{tg}$ such that $B \leq N$, $\mathsf{tg} \in \{0,1\}^t$.
2. Challenger computes $(\mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in [N]}) \leftarrow \mathsf{AltSetup}(1^\lambda, 1^N, 1^B, 1^t, \mathsf{tg})$, sends $(\mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in [N]})$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs $\mathsf{tg}' \neq \mathsf{tg}$, set $S \subseteq [N]$ of size greater than B, commitment com' and proofs $(\sigma_i')_{i \in S}$. The experiment outputs 1 iff for all $i \in S$, $\mathsf{Verify}(\mathsf{pp}, \mathsf{com}', i, \sigma_i', \mathsf{tg}') = 1$.

Figure 2: Experiment for Soundness Security

4.1 Construction of Tagged Set Commitment

In this section, we will present a Tagged Set Commitment scheme TSC whose security is based on PRG security. Let $\mathsf{PRG} : (\{0,1\}^\lambda, 1^\ell) \to \mathbb{F}_{2^\ell}$ be a pseudorandom generator. Let emb be an injective and efficiently-computable function that maps strings in $\{0,1\}^t$ (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 $\mathbb{F}_{2^\ell}$.

$\mathsf{Setup}(1^\lambda, 1^N, 1^B, 1^t)$: The setup algorithm sets $\ell = 2t + (B+1) \cdot \log N + \lambda \cdot (B+1) + \lambda$. Next it 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 $\mathsf{pp} = (1^\ell, (A_i, D_i)_{i \in [N]})$.

$\mathsf{Commit}(\mathsf{pp} = (1^\ell, (A_i, D_i)_i), S \subseteq [N], \mathsf{tg})$: The commitment algorithm first chooses $s_i \leftarrow \{0,1\}^\lambda$ for each $i \in S$. Next, it chooses the degree $B-1$ polynomial $p(\cdot)$ over $\mathbb{F}_{2^\ell}$ such that for all $i \in S$, $p(i) = \mathsf{PRG}(s_i, 1^\ell) + A_i + D_i \cdot \mathsf{emb}(\mathsf{tg})$. (Since we fix B points, there is a unique degree $B-1$ polynomial p, which is described in canonical form using B coefficients in $\mathbb{F}_{2^\ell}$.) The commitment com is the polynomial p, and the proof $\sigma_i = s_i$ for each $i \in S$.

$\mathsf{Verify}(\mathsf{pp} = (1^\ell, (A_i, D_i)_i), \mathsf{com} = p, i, \sigma_i, \mathsf{tg})$: The verification algorithm outputs 1 iff $p(i) = \mathsf{PRG}(\sigma_i, 1^\ell) + A_i + D_i \cdot \mathsf{emb}(\mathsf{tg})$.

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

The correctness properties follow immediately from the construction.

4.1.1 Security Proofs

We need to show that the scheme satisfies indistinguishability of setup and soundness security (Definition 4.2).

Lemma 4.1. Assuming PRG is a secure pseudorandom generator (Definition 2.1), TSC satisfies indistinguishability of setup (Definition 4.1).

Proof. We will prove this using a sequence of hybrid experiments $H_0, \ldots, H_3$ where H_0 corresponds to the challenger using Setup and H_3 is the challenger using $\mathsf{AltSetup}$.

Hybrid H_0 : In this experiment, the challenger runs Setup and Commit.

1. The adversary sends $1^N, 1^B, \mathsf{tg}$ and $S \subset [N]$ of size B.
2. The challenger performs the following steps:
   (a) It chooses $A_i, D_i \leftarrow \mathbb{F}_{2^\ell}$ for each $i \in [N]$ and sets $\mathsf{pp} = (1^\ell, (A_i, D_i)_i)$.
   (b) Next, it chooses $s_i \leftarrow \{0,1\}^\lambda$ for each $i \in S$, sets $d_i = \mathsf{PRG}(s_i, 1^\ell) + A_i + D_i \cdot \mathsf{emb}(\mathsf{tg})$.
   (c) It computes the polynomial $p \in \mathbb{F}_{2^\ell}[x]^{B-1}$ such that $p(i) = d_i$ for all $i \in S$ and sets $\mathsf{com} = p$, $\sigma_i = s_i$ for each $i \in S$.
   (d) The challenger sends $\mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in S}$.
3. The adversary sends its guess b'.

Hybrid H_1 : This experiment is identical to the previous one, except that the challenger first chooses the d_i values uniformly at random for each $i \in S$ and then sets the corresponding A_i values appropriately.

1. The adversary sends $1^N, 1^B, \mathsf{tg}$ and $S \subset [N]$ of size B.
2. The challenger performs the following steps:
   (a) It chooses $D_i \leftarrow \mathbb{F}_{2^\ell}$ for each $i \in [N]$, $A_i \leftarrow \mathbb{F}_{2^\ell}$ for each $i \in [N] \setminus S$ and $s_i \leftarrow \{0,1\}^\lambda, d_i \leftarrow \mathbb{F}_{2^\ell}$ for each $i \in S$. It sets $A_i = d_i - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg})$ for each $i \in S$, and sets $\mathsf{pp} = (\ell, (A_i, D_i)_i)$.
   (b) It computes the polynomial $p \in \mathbb{F}_{2^\ell}[x]^{B-1}$ such that $p(i) = d_i$ for all $i \in S$ and sets $\mathsf{com} = p$, $\sigma_i = s_i$ for each $i \in S$.
   (c) The challenger sends $\mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in S}$.
3. The adversary sends its guess b'.

Hybrid H_2 : Here, the challenger first chooses a random degree $B-1$ polynomial, and then sets the d_i values according to this polynomial.

1. The adversary sends $1^N, 1^B, \mathsf{tg}$ and $S \subset [N]$ of size B.
2. The challenger performs the following steps:
   (a) It chooses $D_i \leftarrow \mathbb{F}_{2^\ell}$ for each $i \in [N]$, $A_i \leftarrow \mathbb{F}_{2^\ell}$ for each $i \in [N] \setminus S$ and $s_i \leftarrow \{0,1\}^\lambda$ for each $i \in S$. It chooses a random polynomial $p \leftarrow \mathbb{F}_{2^\ell}[x]^{B-1}$, sets $d_i = p(i)$, $A_i = d_i - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg})$ for each $i \in S$, and sets $\mathsf{pp} = (\ell, (A_i, D_i)_i)$.
   (b) It sets $\mathsf{com} = p$, $\sigma_i = s_i$ for each $i \in S$.
   (c) The challenger sends $\mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in S}$.
3. The adversary sends its guess b'.

Hybrid H_3 : In this experiment, the challenger uses $\mathsf{AltSetup}$. The only difference between this hybrid and the previous one is with respect to the choice of A_i for $i \in [N] \setminus S$. In the previous experiment, the challenger chooses A_i uniformly at random for all $i \in [N] \setminus S$. In this one, it sets $A_i = p(i) - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg})$. The experiment is described formally below.

1. The adversary sends $1^N, 1^B, \mathsf{tg}$ and $S \subset [N]$ of size at most B.
2. The challenger performs the following steps:
   (a) It chooses $D_i \leftarrow \mathbb{F}_{2^\ell}$ and $s_i \leftarrow \{0,1\}^\lambda$ for each $i \in [N]$. It chooses a random polynomial $p \leftarrow \mathbb{F}_{2^\ell}[x]^{B-1}$, sets $A_i = p(i) - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg})$ for each $i \in [N]$, and sets $\mathsf{pp} = (\ell, (A_i, D_i)_i)$.
   (b) It sets $\mathsf{com} = p$, $\sigma_i = s_i$ for each $i \in S$.
   (c) The challenger sends $\mathsf{pp}, \mathsf{com}, (\sigma_i)_{i \in S}$.
3. The adversary sends its guess b'.

Analysis : We will now show that the above hybrids are computationally indistinguishable. For any PPT adversary $\mathcal{A}$, let $\mathsf{pr}_{\mathcal{A},x}(\lambda)$ denote the probability that the adversary outputs 1 in hybrid H_x.

Claim 4.1. For any adversary $\mathcal{A}$, $\mathsf{pr}_{\mathcal{A},0}(\lambda) = \mathsf{pr}_{\mathcal{A},1}(\lambda)$.

Proof. The only difference between H_0 and H_1 is as follows: in H_0, the challenger chooses $(A_i)_{i \in S}$ uniformly at random and then sets $d_i = \mathsf{PRG}(s_i, 1^\ell) + A_i + D_i \cdot \mathsf{emb}(\mathsf{tg})$; in H_1, the challenger chooses $(d_i)_{i \in S}$ uniformly at random and sets $A_i = d_i - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg})$. Clearly, both these experiments are identical. □

Claim 4.2. For any adversary $\mathcal{A}$, $\mathsf{pr}_{\mathcal{A},1}(\lambda) = \mathsf{pr}_{\mathcal{A},2}(\lambda)$.

Proof. The only difference between H_1 and H_2 is as follows: in H_1, the challenger chooses $(d_i)_{i \in S}$ uniformly at random and then chooses the polynomial p which satisfies the restriction that $p(i) = d_i$ for all $i \in S$. In H_2, the challenger first chooses a random degree $B-1$ polynomial and sets $d_i = p(i)$ for all $i \in S$. Since there is a bijective mapping between $\mathbb{F}_{2^\ell}^B$ and $\mathbb{F}_{2^\ell}[x]^{B-1}$, it follows that the outputs of these two experiments are identically distributed. □

Claim 4.3. Assuming PRG is a secure pseudorandom generator, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathsf{pr}_{\mathcal{A},2}(\lambda) - \mathsf{pr}_{\mathcal{A},3}(\lambda)| \leq \mathsf{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{pr}_{\mathcal{A},2}(\lambda) - \mathsf{pr}_{\mathcal{A},3}(\lambda)$ is non-negligible. We will use $\mathcal{A}$ to build an algorithm $\mathcal{B}$ that breaks the PRG security. (Recall, per Definition 2.1 for our case, no efficient distinguisher can distinguish between a random element in $\mathbb{F}_{2^\ell}$ and the output of $\mathsf{PRG}(s, 1^\ell)$ for a random seed $s \in \{0,1\}^\lambda$. Suppose we consider an extension of this experiment where the distinguisher submits an integer q (polynomial in λ) and then receives either q random elements in $\mathbb{F}_{2^\ell}$ or the q values $\{\mathsf{PRG}(a_i, 1^\ell)\}_i$ for random $a_i \in \{0,1\}^\lambda$. If no efficient distinguisher has a non-negligible probability of distinguishing the first experiment, then that will also hold for this polynomial repetition of it by a simple hybrid argument.)

The algorithm $\mathcal{B}$ first receives $1^N, 1^B, 1^t, \mathsf{tg}, S$ from $\mathcal{A}$. It chooses $D_i \leftarrow \mathbb{F}_{2^\ell}$ for each $i \in [N]$, a random polynomial $p \leftarrow \mathbb{F}_{2^\ell}[x]^{B-1}$, random strings $s_i \leftarrow \{0,1\}^\lambda$ for $i \in S$, and sets $A_i = p(i) - \mathsf{PRG}(s_i, 1^\ell) - \mathsf{emb}(\mathsf{tg}) \cdot D_i$ for all $i \in S$. Next, it sends integer $N - B$ queries to our extended PRG challenger above and receives $(z_i)_{i \in [N] \setminus S}$ as the responses (we index each response with an element in $[N] \setminus S$). The reduction algorithm $\mathcal{B}$ sets $A_i = p(i) - z_i - D_i \cdot \mathsf{emb}(\mathsf{tg})$. Finally, it sets $\mathsf{pp} = (\ell, (A_i, D_i))$, $\mathsf{com} = p$, and sends $\mathsf{pp}, \mathsf{com}, (s_i)_{i \in S}$ to $\mathcal{A}$. If the adversary outputs 1, then $\mathcal{B}$ guesses that the PRG challenger's outputs are random, else it guesses that they are pseudorandom. □

□

Lemma 4.2. TSC satisfies soundness security (Definition 4.2).

Proof. This proof is a statistical argument. Suppose there exists an adversary $\mathcal{A}$ s.t. $\Pr[1 \leftarrow \mathsf{Expt\text{-}Sound}_{\mathcal{A}}(\lambda)] = \epsilon(\lambda)$. The adversary $\mathcal{A}$ sends tg, receives $(\mathsf{pp}, \mathsf{com}, (s_i)_{i \in [N]}) \leftarrow \mathsf{AltSetup}(1^\lambda, 1^N, 1^B, \mathsf{tg})$, and outputs tg' together with $\mathsf{com}' = p'$, S and $(s'_i)_{i \in S}$ such that $|S| > B$ and $\mathsf{Verify}(\mathsf{pp}, \mathsf{com}', i, s'_i, \mathsf{tg}') = 1$ for all $i \in S$. This means, $\mathsf{com}' \in \mathbb{F}_{2^\ell}[x]^{B-1}$, and for all $i \in S$, $A_i = p'(i) - \mathsf{PRG}(s'_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}')$. It suffices to show that the following probability, and therefore $\epsilon(\lambda)$, is bounded by $\mathsf{negl}(\lambda)$ [5]:

$$\rho(\lambda) = \Pr\left[\begin{array}{c} \exists p' \in \mathbb{F}_{2^\ell}[x]^{B-1}, S \subseteq [N], |S| = B+1, (s'_i)_i, \mathsf{tg} \neq \mathsf{tg}' \text{ such that} \\ \forall i \in S, p(i) - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}) = p'(i) - \mathsf{PRG}(s'_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}') \end{array}\right]$$

where the probability is over the choice of $D_i \leftarrow \mathbb{F}_{2^\ell}$, $s_i \leftarrow \{0,1\}^\lambda$ for all $i \in [N]$ and $p \leftarrow \mathbb{F}_{2^\ell}[x]^{B-1}$.

We will prove something stronger: consider the following probability where $s_i \in \{0,1\}^\lambda$, $p \in \mathbb{F}_{2^\ell}[x]^{B-1}$ are arbitrary and the probability is only over the choice of $D_i \leftarrow \mathbb{F}_{2^\ell}$. Let $\rho_{\mathbf{s},p}$ denote the probability $\rho(\lambda)$ for some fixed λ, $\mathbf{s} = (s_i)_i$ and p.

Claim 4.4. If $\ell > t + (B+1) \cdot \log N + \lambda \cdot (B+1) + \lambda$, then $\rho_{\mathbf{s},p} \leq 2^{-\lambda}$.

Proof. The proof follows via the union bound. For any tuple $(\lambda, N, B, \mathsf{tg})$, let $R(\lambda, N, B, \mathsf{tg})$ denote the set of all potentially-winning responses sent by the adversary after receiving $(\mathsf{pp}, \mathsf{com}, (s_i)_{i \in [N]})$; that is, it consists of all tuples $(\mathsf{tg}', p', S, (s'_i)_{i \in S})$ such that $\mathsf{tg}' \neq \mathsf{tg}$, $p' \in \mathbb{F}_{2^\ell}[x]^{B-1}$, $|S| = B+1$ and the s'_i strings are λ bits long.[6] Then, note that we can express $\rho_{\mathbf{s},p}$ in terms of $R(\lambda, N, B, \mathsf{tg})$ as follows:

$$\rho_{\mathbf{s},p} = \Pr\left[\begin{array}{c} \forall i \in [N], \text{ choose } D_i \text{ uniformly at random} \\ \mathcal{A} \text{ sends } (\mathsf{tg}', p', S, (s'_i)_{i \in S}) \in R(\lambda, N, B, \mathsf{tg}) \\ \forall i \in S, p(i) - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}) = p'(i) - \mathsf{PRG}(s'_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}'). \end{array}\right]$$

First, we note that the size of $R(\lambda, N, B, \mathsf{tg})$ can be bounded as follows.

Observation 4.1. For any $\lambda, N, B, \mathsf{tg}$, $|R(\lambda, N, B, \mathsf{tg})| = (2^t - 1) \cdot \binom{N}{B+1} \cdot 2^{\lambda \cdot (B+1)} \cdot 2^{\ell \cdot B}$.

Proof. The number of tags tg' satisfying $\mathsf{tg} \neq \mathsf{tg}'$ is $2^t - 1$. The number of sets S of size $B+1$ is $\binom{N}{B+1}$. The set $\mathbb{F}_{2^\ell}[x]^{B-1}$ has size $2^{\ell \cdot B}$ and the set $(s_i)_{i \in S}$ has size $2^{\lambda \cdot (B+1)}$.

Note that in this proof, we use the fact that degree of p' is $B - 1$. □

[5] Note that in the following probability, we only look at sets S of size $B+1$. Clearly, if an adversary can output a set S of size greater than B, then it can output a set of size exactly $B+1$.

[6] We call it a potentially-winning response because the definition of this set does not have the main requirement for winning: $\forall i \in S, p(i) - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}) = p'(i) - \mathsf{PRG}(s'_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}')$

Next, we observe the following:

Observation 4.2. For any $\lambda, N, B, \mathsf{tg}$, strings $(s_i)_{i\in[N]}$, polynomial $p \in \mathbb{F}_{2^\ell}[x]^{B-1}$ and $r = (\mathsf{tg}', p', S, (s'_i)) \in R(\lambda, N, B, \mathsf{tg})$,

$$\rho^r_{\mathbf{s},p} = \Pr\left[\begin{array}{c} \forall i \in [N],\ \text{choose } D_i \text{ uniformly at random} \\ \forall i \in S, p(i) - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}) = p'(i) - \mathsf{PRG}(s'_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}'). \end{array}\right] = 2^{-\ell\cdot(B+1)}$$

Proof. The above expression can be rewritten as follows:

$$\Pr\left[\begin{array}{c} \forall i \in [N],\ \text{choose } D_i \text{ uniformly at random} \\ \forall i \in S, D_i = (\mathsf{emb}(\mathsf{tg}) - \mathsf{emb}(\mathsf{tg}'))^{-1} \cdot (\mathsf{PRG}(s'_i, 1^\ell) - \mathsf{PRG}(s_i, 1^\ell) + p(i) - p'(i)) \end{array}\right]$$

Here, we use the fact that $\mathsf{tg} \neq \mathsf{tg}'$, and hence this probability precisely captures the event where $B+1$ uniformly random strings (of length ℓ each) are equal to some fixed $B+1$ strings.

Note that in this part, we do not use the fact that p' has degree $B-1$. □

Since the above expression $\rho^r_{\mathbf{s},p}$ is independent of the tuple r, let us call this $\rho^{\text{fixed}}_{\mathbf{s},p}$. Finally, we show a bound on $\rho_{\mathbf{s},p}$.

Observation 4.3. For any adversary $\mathcal{A}$ that sends $\lambda, N, B, \mathsf{tg}$, any strings $(s_i)_{i\in[N]}$, polynomial $p \in \mathbb{F}_{2^\ell}[x]^{B-1}$, $\rho_{\mathbf{s},p} \leq |R(\lambda, N, B, \mathsf{tg})| \cdot \rho^{\text{fixed}}_{\mathbf{s},p}$.

Proof. First, note that $\rho_{\mathbf{s},p} \leq \rho'_{\mathbf{s},p}$, where

$$\rho'_{\mathbf{s},p} = \Pr\left[\begin{array}{c} \forall i \in [N],\ \text{choose } D_i \text{ uniformly at random} \\ \exists r = (\mathsf{tg}', p', S, (s'_i)_{i\in S}) \in R(\lambda, N, B, \mathsf{tg}) \\ \forall i \in S, p(i) - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}) = p'(i) - \mathsf{PRG}(s'_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}'). \end{array}\right].$$

(We are simply setting the best possible response that the adversary can send).

Next, we can bound the expression above by a union bound as follows:

$$\rho'_{\mathbf{s},p} \leq \sum_{r\in R(\lambda,N,B,\mathsf{tg})} \underbrace{\Pr\left[\begin{array}{c} \forall i \in [N],\ \text{choose } D_i \text{ uniformly at random} \\ \forall i \in S, p(i) - \mathsf{PRG}(s_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}) = p'(i) - \mathsf{PRG}(s'_i, 1^\ell) - D_i \cdot \mathsf{emb}(\mathsf{tg}'). \end{array}\right]}_{= \rho^{\text{fixed}}_{\mathbf{s},p} \ = \ 2^{\ell\cdot(B+1)} \text{(as discussed in Observation 4.2)}}.$$

Hence, we conclude that $\rho_{\mathbf{s},p} \leq \rho'_{\mathbf{s},p} \leq |R(\lambda, N, B, \mathsf{tg})| \cdot \rho^{\text{fixed}}_{\mathbf{s},p}$.

□

The proof follows by setting $\ell \geq t + (B+1)\cdot\log N + \lambda\cdot(B+1) + \lambda$. □

Since we have shown that $\rho_{\mathbf{s},p} \leq 2^{-\lambda}$ for all $\mathbf{s}, p$, it follows that it holds even for randomly chosen $\mathbf{s}$ and p. This concludes the proof of the lemma.

□

5 Our CCA Secure Encryption Scheme

In this section, we will present a CCA secure encryption scheme with message space $\{0,1\}^{\ell_{\mathsf{cca}}}$ satisfying almost-all-keys perfect correctness. We require the following parameters/notations for our construction.

- λ: security parameter
- N: number of ciphertext components of underlying CPA scheme
- B: size of set for 'selected' ciphertext components
- $\ell_{\mathsf{tsc}.\sigma}$: size of proofs output by tagged set commitment scheme

- ℓ_{cpa}: message space for underlying CPA scheme
- ℓ_{rnd}: number of random bits used by CPA scheme to encrypt ℓ_{cpa} bit message
- ℓ_{vk}: size of verification key of signature scheme

The construction uses the following primitives, which are defined in Sections 2, 3 and 4 respectively:

- A Strongly Unforgeable One-Time Signature Scheme $P_1 = (\mathsf{Sig.Setup}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$.
- A CPA Secure almost-all-keys perfectly correct Encryption Scheme with Randomness Recovery $P_2 = (\mathsf{CPA.Setup}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec}, \mathsf{CPA.Recover})$, parameterized by polynomials ℓ_{cpa} (denoting the message space) and ℓ_{rnd} (denoting the number of random bits used for encryption).[7]
- A Tagged Set Commitment Scheme $P_3 = (\mathsf{TSC.Setup}, \mathsf{TSC.Commit}, \mathsf{TSC.Verify}, \mathsf{TSC.AltSetup})$, parameterized by polynomials $\ell_{\mathsf{tsc}.\sigma}$ (denoting the length of proof for each index) and ℓ_{com} (denoting the length of commitment).

These parameters must satisfy the following constraints:

- $\ell_{\mathsf{cpa}} = 1 + \ell_{\mathsf{tsc}.\sigma} + \ell_{\mathsf{cca}}$
- $\log\left(\binom{N-1}{B-1}\right) > \ell_{\mathrm{rnd}} + 2\lambda$

$\mathsf{Setup}(1^\lambda)$: The setup algorithm performs the following steps:

1. It first chooses public parameters for the commitment scheme. Let $\mathsf{tsc.pp} \leftarrow \mathsf{TSC.Setup}(1^\lambda, 1^N, 1^B, 1^{\ell_{\mathsf{vk}}})$.
2. Next, it chooses N public/secret keys for the encryption scheme. Let $(\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$.
3. It sets $\mathsf{pk} = \left(\mathsf{tsc.pp}, (\mathsf{cpa.pk}_i)_{i\in[N]}\right)$ and $\mathsf{sk} = (\mathsf{cpa.sk}_i)_{i\in[N]}$.

$\mathsf{Enc}(\mathsf{pk}, m)$: The encryption algorithm takes as input $\mathsf{pk} = \left(\mathsf{tsc.pp}, (\mathsf{cpa.pk}_i)_{i\in[N]}\right)$ and $m \in \{0,1\}^{\ell_{\mathsf{cca}}}$, and performs the following steps:

1. It chooses a uniformly random B size subset $S \subset [N]$. Let $S = \{i_1, i_2, \ldots, i_B\}$ where $i_1 < i_2 < \ldots < i_B$.
2. Next, it chooses a signing/verification key $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \mathsf{Sig.Setup}(1^\lambda)$.
3. It then commits to the set S using $\mathsf{sig.vk}$ as the tag. It computes $(\mathsf{tsc.com}, (\mathsf{tsc}.\sigma_i)_{i\in S}) \leftarrow \mathsf{TSC.Commit}(\mathsf{tsc.pp}, S, \mathsf{sig.vk})$.
4. For all $i \neq i_B$, it chooses random values $r_i \leftarrow \{0,1\}^{\ell}_{\mathrm{rnd}}$, and sets $r_{i_B} = \oplus_{j<B}\, r_{i_j}$.
5. Using the r_i values, the encryption algorithm computes N ciphertext components. For $i \in [N]$ if $i \in S$, it computes $\mathsf{cpa.ct}_i = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|\mathsf{tsc}.\sigma_i|m; r_i)$. Else it sets $\mathsf{cpa.ct}_i = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 0^{\ell_{\mathsf{cpa}}}; r_i)$.
6. Finally, the algorithm computes a signature $\mathsf{sig}.\sigma \leftarrow \mathsf{Sig.Sign}\left(\mathsf{sig.sk}, \left(\mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)\right)$ and outputs $\left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Let $\mathsf{sk} = (\mathsf{cpa.sk}_i)_{i\in[N]}$ and $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$. The decryption algorithm performs the following steps:

1. It first verifies the signature $\mathsf{sig}.\sigma$. If $0 \leftarrow \mathsf{Sig.Verify}\left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \left(\mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)\right)$ then decryption outputs $\perp$.
2. Next, it initializes a set U to be $\emptyset$. For each $i \in [N]$ it does the following:

Check

Hardwired: pk, ct
Input: $i \in [N], y \in \left(\{0,1\} \times \{0,1\}^{\ell_{\mathsf{tsc}.\sigma}} \times \{0,1\}^{\ell_{\mathsf{cca}}}\right) \cup \{\perp\}, r \in \{0,1\}^{\ell_{\mathrm{rnd}}}$
Output: 0/1.

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

- $y \neq \perp$. Parse $y = (g, \mathsf{tsc}.\sigma, m)$
- $g = 1$.
- $\mathsf{TSC.Verify}(\mathsf{tsc.pp}, \mathsf{tsc.com}, i, \mathsf{tsc}.\sigma, \mathsf{sig.vk}) = 1$.
- $\mathsf{cpa.ct}_i = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, y; r)$.

Figure 3: Routine Check for checking if tuple (i, y) should be added to set U

(a) Let $(y_i, r_i) = \mathsf{CPA.Dec}(\mathsf{cpa.sk}_i, \mathsf{cpa.ct}_i)$.[8] The decryption algorithm adds (i, y_i) to U if $\mathsf{Check}(i, y_i, r_i) = 1$, where Check is defined in Figure 3.

3. If the set U does not have exactly B elements then the decryption algorithm outputs $\perp$.
4. If $\oplus_{(i,y_i)\in U}\ r_i \neq 0^{\ell_{\mathrm{rnd}}}$, it outputs $\perp$.
5. Finally, the decryption algorithm checks that for all $(i, r_i) \in U$, the m_i values recovered from y_i are the same. If not, it outputs $\perp$. Else it outputs this common m_i value as the decryption.

Perfect Correctness The message space is $\{0,1\}^{\ell_{\mathsf{cca}}}$, where ℓ_{cca} is a polynomial function in the security parameter λ. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any security parameter λ,

$$\Pr\left[\exists\ m, r \in \{0,1\}^{\ell_{\mathsf{cca}}} \text{ s.t. } \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) \neq m\right] \leq \mathsf{negl}(\lambda)$$

where the probability is over the choice of $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the random coins of Enc.

The almost-all-keys perfect correctness of the CCA scheme follows from the almost-all-keys perfect correctness of the CPA scheme and the (perfect) correctness of the signature and tagged set commitment schemes.

Remark 5.1. Any signature or tagged set commitment scheme with negligible correctness error can be transformed into one with perfect correctness. A signer or committer can check whether the respective signature or commitment verifies using the public verification algorithm. If it does not, the signing algorithm can fall back to a trivial signature that is perfectly correct, but has no security against forgeries. In the case of commitments use a trivial scheme that is binding, but is not hiding. Since the correctness error is negligible, this will only happen with negligible probability in the security argument.

5.1 Proof of Security

Theorem 5.1. The above construction is IND-CCA-secure (per Definition 2.4) and almost-all-keys perfectly correct, assuming P_1 is a strongly unforgeable one-time signature scheme (Definition 2.2), P_2 is an IND-CPA-secure encryption scheme (Definition 2.3) with randomness recovery with almost-all-keys perfect correctness (Section 3) and P_3 is a secure tagged set commitment scheme (Definitions 4.1 and 4.2).

The following result follows immediately from the above theorem, Theorem 3.1, Lemma 4.1, and known constructions of other building blocks from injective trapdoor functions.

Corollary 5.1 (IND-CCA-secure Public-Key Encryption is Implied by (Injective) Trapdoor Functions)**.** The above construction is IND-CCA-secure (per Definition 2.4) assuming injective trapdoor functions.

Proof of the main theorem proceeds via a sequence of hybrid experiments.

[7] For security parameter λ, the scheme will support $\ell_{\mathsf{cpa}}(\lambda)$ bit messages, and the encryption algorithm will use $\ell_{\mathrm{rnd}}(\lambda)$ bits of randomness. We will drop the dependence on λ when it is clear from context.

[8] Recall the decryption algorithm also recovers the randomness used for encryption.

Hybrid H_0 This experiment corresponds to the CCA experiment. Here, we spell out the setup and encryption algorithms again in order to set up notations for the proof.

- Setup phase: This is identical to the scheme's setup.
  1. The challenger first chooses $\mathsf{tsc.pp} \leftarrow \mathsf{TSC.Setup}(1^\lambda, 1^N, 1^B, 1^t)$.
  2. Next, it chooses $(\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for all $i \in [N]$.
  3. It sends $\mathsf{pk} = \left(\mathsf{tsc.pp}, (\mathsf{cpa.pk}_i)_{i\in[N]}\right)$ to $\mathcal{A}$ and uses $\mathsf{sk} = (\mathsf{cpa.sk}_i)_{i\in[N]}$ for handling decryption queries.
- Pre-challenge decryption queries: The adversary makes polynomially many decryption queries. For each query $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig.}\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$, the challenger outputs $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$.
- Challenge ciphertext: The adversary sends two challenge messages $m_0, m_1 \in \{0,1\}^{\ell_{\mathsf{cca}}}$. The challenger chooses a bit b and does the following.
  1. It chooses a uniformly random B size subset $S^* = \{i_j\}_{j\in[B]} \subset [N]$.
  2. Next, it chooses a signing/verification key $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*) \leftarrow \mathsf{Sig.Setup}(1^\lambda)$.
  3. It then commits to the set S using $\mathsf{sig.vk}^*$ as the tag. It computes $(\mathsf{tsc.com}^*, (\mathsf{tsc.}\sigma_i^*)_{i\in S}) \leftarrow \mathsf{TSC.Commit}(\mathsf{tsc.pp}, S^*, \mathsf{sig.vk}^*)$.
  4. For all $i \neq i_B$, it chooses $r_i \leftarrow \{0,1\}^{\ell_{\mathsf{rnd}}}$, and sets $r_{i_B} = \oplus_{j<B}\, r_{i_j}$.
  5. Using the r_i values, the encryption algorithm computes N ciphertexts. If $i \in S$, it computes $\mathsf{cpa.ct}_i^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|\mathsf{tsc.}\sigma_i^*|m_b; r_i)$. Else it sets $\mathsf{cpa.ct}_i^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 0^{\ell_{\mathsf{cpa}}}; r_i)$.
  6. Finally, the challenger computes a signature $\mathsf{sig.}\sigma^* \leftarrow \mathsf{Sig.Sign}\left(\mathsf{sig.sk}^*, \left(\mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)\right)$ and outputs $\left(\mathsf{sig.vk}^*, \mathsf{sig.}\sigma^*, \mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)$.
- Post-challenge decryption queries: Same as pre-challenge decryption queries, but challenge ciphertext not allowed as a decryption query.
- Guess: The adversary sends bit b' and wins if $b = b'$.

Hybrid H_1 : This experiment is identical to the previous one except that the challenger chooses $\mathsf{sig.vk}^*$ and S^* during setup, and uses these to compute the challenge ciphertext.

- Setup phase:
  1. The challenger first chooses $\mathsf{tsc.pp} \leftarrow \mathsf{TSC.Setup}(1^\lambda, 1^N, 1^B, 1^t)$.
  2. Next, it chooses $(\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for all $i \in [N]$.
  3. Then it chooses a uniformly random B size subset $S^* = \{i_j\}_{j\in[B]} \subset [N]$ and $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*) \leftarrow \mathsf{Sig.Setup}(1^\lambda)$.
  4. It sends $\mathsf{pk} = \left(\mathsf{tsc.pp}, (\mathsf{cpa.pk}_i)_{i\in[N]}\right)$ to $\mathcal{A}$ and uses $\mathsf{sk} = (\mathsf{cpa.sk}_i)_{i\in[N]}$ for handling decryption queries.

Hybrid H_2 : In this experiment, the challenger outputs $\perp$ during the decryption queries if the queried ciphertext $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig.}\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$ is such that $\mathsf{sig.vk} = \mathsf{sig.vk}^*$.

- Pre-challenge decryption queries: The adversary makes polynomially many decryption queries. For each query $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig.}\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$, if $\mathsf{sig.vk} = \mathsf{sig.vk}^*$, then the challenger outputs $\perp$, else it outputs $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$.
- Post-challenge decryption queries: Same as pre-challenge decryption queries, but challenge ciphertext not allowed as a decryption query.

Hybrid H_3 : Here, the challenger runs $\mathsf{TSC.AltSetup}$ instead of $\mathsf{TSC.Setup}$ during the setup phase. During the challenge phase, it uses the commitment and proofs generated by $\mathsf{TSC.AltSetup}$ instead of computing them using $\mathsf{TSC.Commit}$.

- Setup phase:
  1. The challenger first chooses $(\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i) \leftarrow \mathsf{CPA.Setup}(1^\lambda)$ for all $i \in [N]$.
  2. Next, it chooses a uniformly random B size subset $S^* \subset [N]$ and $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*) \leftarrow \mathsf{Sig.Setup}(1^\lambda)$.
  3. Then it chooses $\left(\mathsf{tsc.com}^*, (\mathsf{tsc.}\sigma_i)_{i\in[N]}\right) \leftarrow \mathsf{TSC.AltSetup}(1^\lambda, 1^N, 1^B, 1^t, \mathsf{sig.vk}^*)$.
  4. It sends $\mathsf{pk} = \left(\mathsf{tsc.pp}, (\mathsf{cpa.pk}_i)_{i\in[N]}\right)$ to $\mathcal{A}$ and uses $\mathsf{sk} = (\mathsf{cpa.sk}_i)_{i\in[N]}$ for handling decryption queries.
- Challenge phase: Note that the signature keys $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*)$, set S^* and commitment $\mathsf{tsc.com}^*$ together with proofs $(\mathsf{tsc.}\sigma_i)_{i\in[N]}$ were chosen during setup. Below we include the full challenge phase for readability.
  1. For all $i \neq i_B$, it chooses $r_i \leftarrow \{0,1\}^{\ell_{\mathrm{rnd}}}$, and sets $r_{i_B} = \oplus_{j<B}\, r_{i_j}$.
  2. Using the r_i values, the encryption algorithm computes N ciphertexts. If $i \in S$, it computes $\mathsf{cpa.ct}_i^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|\mathsf{tsc.}\sigma_i^*|m_b; r_i)$. Else it sets $\mathsf{cpa.ct}_i^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 0^{\ell_{\mathsf{cpa}}}; r_i)$.
  3. Finally, the challenger computes a signature $\mathsf{sig.}\sigma^* \leftarrow \mathsf{Sig.Sign}\left(\mathsf{sig.sk}^*, \left(\mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)\right)$ and outputs $\left(\mathsf{sig.vk}^*, \mathsf{sig.}\sigma^*, \mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)$.

Hybrid H_4 : In this experiment, the challenger modifies the challenge ciphertext. Instead of encrypting $0^{\ell_{\mathsf{cpa}}}$ at $N - B$ positions, the challenger encrypts $1|\mathsf{tsc.}\sigma_i|m_b$ at position i for all $i \in [N]$.

- Challenge phase:
  1. For all $i \neq i_B$, it chooses $r_i \leftarrow \{0,1\}^{\ell_{\mathrm{rnd}}}$, and sets $r_{i_B} = \oplus_{j<B}\, r_{i_j}$.
  2. Using the r_i values, the encryption algorithm computes N ciphertexts. For all $i \in [N]$, it computes $\mathsf{cpa.ct}_i^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|\mathsf{tsc.}\sigma_i^*|m_b; r_i)$.
  3. Finally, the challenger computes a signature $\mathsf{sig.}\sigma^* \leftarrow \mathsf{Sig.Sign}\left(\mathsf{sig.sk}^*, \left(\mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)\right)$ and outputs $\left(\mathsf{sig.vk}^*, \mathsf{sig.}\sigma^*, \mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)$.

Hybrid H_5 : In this experiment, the challenger encrypts at all positions using true randomness.

- Challenge phase:
  1. For all $i \in [N]$, the challenger chooses $r_i \leftarrow \{0,1\}^{\ell_{\mathrm{rnd}}}$.
  2. Using the r_i values, the encryption algorithm computes N ciphertexts. For all $i \in [N]$, it computes $\mathsf{cpa.ct}_i^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 1|\mathsf{tsc.}\sigma_i^*|m_b; r_i)$.
  3. Finally, the challenger computes a signature $\mathsf{sig.}\sigma^* \leftarrow \mathsf{Sig.Sign}\left(\mathsf{sig.sk}^*, \left(\mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)\right)$ and outputs $\left(\mathsf{sig.vk}^*, \mathsf{sig.}\sigma^*, \mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)$.

Hybrid H_6 : In the final hybrid experiment, the challenger switches all challenge ciphertext components to encryptions of $0^{\ell_{\mathsf{cpa}}}$. As a result, in this hybrid, the adversary has advantage 0.

- Challenge phase:
  1. For all $i \in [N]$, it chooses $r_i \leftarrow \{0,1\}^{\ell_{\mathrm{rnd}}}$.
  2. Using the r_i values, the encryption algorithm computes N ciphertexts. For all $i \in [N]$, it computes $\mathsf{cpa.ct}_i^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}_i, 0^{\ell_{\mathsf{cpa}}}; r_i)$.
  3. Finally, the challenger computes a signature $\mathsf{sig}.\sigma^* \leftarrow \mathsf{Sig.Sign}\left(\mathsf{sig.sk}^*, \left(\mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)\right)$ and outputs $\left(\mathsf{sig.vk}^*, \mathsf{sig}.\sigma^*, \mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)$.

5.1.1 Analysis

Lemma 5.1. For all $\lambda \in \mathbb{N}$, and any adversary $\mathcal{A}$, $\mathsf{pr}_{\mathcal{A},0}(\lambda) - \mathsf{pr}_{\mathcal{A},1}(\lambda) = 0$.

Proof. In game H_0 the challenge phase is used to choose a random $S^* = \{i_j\}_{j\in[B]} \subset [N]$ and sample $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*) \leftarrow \mathsf{Sig.Setup}(1^\lambda)$. Both of these samplings will use a fresh set of coins and their distribution will be completely independent of any attacker actions including the challenge messages selected by the attacker. Therefore the attacker's sampling them in the challenge phase as in H_0 or earlier as in H_1 is identical. □

Lemma 5.2. Assuming that P_1 is a strongly-unforgeable one-time signature scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. for all $\lambda \in \mathbb{N}$, and any ppt. adversary $\mathcal{A}$, $\mathsf{pr}_{\mathcal{A},1}(\lambda) - \mathsf{pr}_{\mathcal{A},2}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. In game H_1, the challenger answers all decryption queries, except when queried on the challenge ciphertext $\mathsf{ct}^* = \left(\mathsf{sig.vk}^*, \mathsf{sig}.\sigma^*, \mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)$. In game H_2, the challenger will not respond to decryption queries on ct^* and returns $\perp$ on any decryption query for $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$ where $\mathsf{ct} \neq \mathsf{ct}^*$ and $\mathsf{sig.vk} = \mathsf{sig.vk}^*$. However, there are two cases to explore. First, if $\mathsf{ct} \neq \mathsf{ct}^*$ and $\mathsf{sig.vk} = \mathsf{sig.vk}^*$ but the signature $\mathsf{sig}.\sigma$ does not verify under $\mathsf{sig.vk}$ on message $(\mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]})$, then the challenger of game H_2 immediately outputs $\perp$ and the challenger of game H_1 would also have returned $\perp$ (via rejection of this ciphertext by the regular decryption algorithm) and the two responses are identical.

Second if $\mathsf{ct} \neq \mathsf{ct}^*$ and $\mathsf{sig.vk} = \mathsf{sig.vk}^*$, but the signature *does* verify, then the adversary's view of these two games differ, but we argue that due to the strong unforgeability of the one-time signature scheme P_1, this case occurs with only negligible probability. To see this, we argue that any adversary with non-negligble $\mathsf{pr}_{\mathcal{A},1}(\lambda) - \mathsf{pr}_{\mathcal{A},2}(\lambda)$ can be used to break P_1 as follows. The reduction generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pk to $\mathcal{A}$. It receives $\mathsf{sig.vk}^*$ from the SU-OTS challenger. If $\mathsf{sig.vk}^*$ appears as the signing key in any phase I decryption query, then it aborts. Since $\mathcal{A}$ has no information about $\mathsf{sig.vk}^*$ at this point, this can happen with probability at most the number of decryption queries (polynomial) divided by the size of the public key space for the signature scheme (exponential), so with negligible probability. Once $\mathcal{A}$ outputs challenge messages m_0, m_1, the reduction selects one of these messages randomly and encrypts it according to the normal encryption algorithm, except it uses $\mathsf{sig.vk}^*$ as the verification key and obtains the corresponding signature $\mathsf{sig}.\sigma^*$ by calling the SU-OTS challenger to sign the message $(\mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]})$ (this message is computed according to the normal encryption algorithm). It passes this properly-distributed ciphertext $\mathsf{ct}^* = \left(\mathsf{sig.vk}^*, \mathsf{sig}.\sigma^*, \mathsf{tsc.com}^*, (\mathsf{cpa.ct}_i^*)_{i\in[N]}\right)$ back to $\mathcal{A}$. When $\mathcal{A}$ issues a Phase II decryption query $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$ where $\mathsf{ct} \neq \mathsf{ct}^*$, $\mathsf{sig.vk} = \mathsf{sig.vk}^*$ and $\mathsf{sig}.\sigma$ verifies, then the reduction outputs $((\mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}), \mathsf{sig}.\sigma)$ to win the SU-OTS challenge. □

Lemma 5.3. Assuming that P_3 is a tagged set commitment scheme with indistinguishability of setup (Definition 4.1), there exists a negligible function $\mathsf{negl}(\cdot)$ s.t. for all $\lambda \in \mathbb{N}$, and any ppt. adversary $\mathcal{A}$, $\mathsf{pr}_{\mathcal{A},2}(\lambda) - \mathsf{pr}_{\mathcal{A},3}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. In game H_2, the challenger uses $\mathsf{TSC.Setup}$ to generate the public commitment parameters and $\mathsf{TSC.Commit}$ to generate the commitment and proofs of membership (for the B items in $S \subseteq [N]$), whereas in game H_3, the challenger uses $\mathsf{TSC.AltSetup}$ to generate the public parameters, commitment and proofs of membership (for all items in $[N]$). Otherwise, these games are identical. If there exists an efficient $\mathcal{A}$ that can distinguish between H_2 and H_3, then we can use this $\mathcal{A}$ to attack the indistinguishability of setup of P_3. The reduction works as follows. In both games, a random set $S^* \subset [N]$ and a signing/verification key $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*) \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ are chosen at the start of the game. The reduction sets $\mathsf{tg} = \mathsf{sig.vk}^*$. It sends $(1^\lambda, 1^B, 1^t, \mathsf{tg}, S^*)$ to the $\mathsf{Expt\text{-}Ind\text{-}Setup}$ challenger, who responds with $(\mathsf{pp}^*, \mathsf{com}^*, (\sigma_i^*)_{i\in S})$, which are either generated by $\mathsf{TSC.Setup}$ (making this equivalent to H_2 or $\mathsf{TSC.AltSetup}$ (making this equivalent to H_3). The reduction uses $\mathsf{CPA.Setup}(1^\lambda, 1^{\mathsf{CPA}})$ to generate N public/secret key pairs. It sets $\mathsf{pk} = \left(\mathsf{pp}^*, (\mathsf{cpa.pk}_i)_{i\in[N]}\right)$ and $\mathsf{sk} = (\mathsf{cpa.sk}_i)_{i\in[N]}$. It sends pk to $\mathcal{A}$. It answers each decryption query by running the normal decryption algorithm using sk. (In both games, we already have that if any decryption (pre or post challenge) query $\mathsf{sig.vk} = \mathsf{sig.vk}^*$, then the response is $\perp$, so if this somehow happens the response would be identical in both games.) Upon receiving challenge messages m_0, m_1, it chooses a random bit b and encrypts m_b, using S^* (which it chose randomly earlier) in step 1 of the encryption algorithm, setting $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*)$ as the signing/verification key in step 2 (instead of generating a new pair), using the commitment/proofs $(\mathsf{com}^*, (\sigma_i^*)_{i\in S})$ (obtained earlier) in step 3, instead of computing them using $\mathsf{TSC.Commit}$, and then following steps 3-5 as normal to generate ct^*. It sends this challenge ciphertext ct^* to $\mathcal{A}$. It continues to answer decryption queries for $\mathcal{A}$ using sk. Once $\mathcal{A}$ outputs a guess b' if $b = b'$, then it outputs 0 (guessing H_2) and otherwise outputs 1 (guessing H_3). Since our assumption is that $\mathcal{A}$ has a non-negligible advantage in Game H_2 over Game H_3, then this reduction will have a non-negligible advantage in the indistinguishability of setup experiment for the tagged commitment scheme. Thus, we have a contradiction. □

Lemma 5.4. Assuming encryption scheme with randomness recovery P_2 is an $\mathsf{IND\text{-}CPA}$ secure encryption scheme and the tagged set commitment scheme P_3 satisfies statistical soundness (Definition 4.2), for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{adv}_{\mathcal{A},3}(\lambda) - \mathsf{adv}_{\mathcal{A},4}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. First, we define the alternate decryption routine which works without the j^{th} decryption key.

$\mathsf{Dec\text{-}Alt}_j(\mathsf{sk}_{-j}, \mathsf{ct})$: Let $\mathsf{sk} = (\mathsf{cpa.sk}_i)_{i\neq j}$ and $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig.}\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$. The 'alternate' decryption oracle performs the following steps:

1. If $0 \leftarrow \mathsf{Sig.Verify}\left(\mathsf{sig.vk}, \mathsf{sig.}\sigma, \left(\mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)\right)$ then decryption outputs $\perp$.
2. Next, it initializes a set U' to be $\emptyset$. For each $i \neq j$ it computes $(y_i, r_i) = \mathsf{CPA.Dec}(\mathsf{cpa.sk}_i, \mathsf{cpa.ct}_i)$. Parse y_i as $g_i|\sigma_i|m_i$. It adds (i, y_i) to U' if $\mathsf{Check}(i, y_i, r_i) = 1$. [9]
3. If the set U' has $B-1$ elements, then set $r_j = \oplus_{(i,y_i)\in U'} r_i$. Use r_j to recover the message from $\mathsf{cpa.ct}_j$. Let $y_j = \mathsf{CPA.Recover}(\mathsf{cpa.pk}_j, \mathsf{cpa.ct}_j, r_j)$. If $y_j \neq \perp$ and $\mathsf{Check}(j, y_j, r_j) = 1$, then add (j, y_j) to U'.
4. If the set U' does not have exactly B elements then the decryption algorithm outputs $\perp$.
5. If $\oplus_{(i,y_i)\in U'} r_i \neq 0^\ell$, it outputs $\perp$.
6. Finally, the decryption algorithm checks that for all $(i, r_i) \in U'$, the m_i values recovered from y_i are the same. If not, it outputs $\perp$. Else it outputs this common m_i value as the decryption.

We will now show that with overwhelming probability (over the choice of the CPA keys and the output of $\mathsf{TSC.AltSetup}$) there does not exist a ciphertext $\mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig.}\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right)$ with $\mathsf{sig.vk} \neq \mathsf{sig.vk}^*$ such that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \neq \mathsf{Dec\text{-}Alt}_j(\mathsf{sk}_{-j}, \mathsf{ct})$.

[9] Recall, Check was defined in Section 5. It outupts 1 if $y_i \neq \perp$, $g_i = 1$, the commitment verifies and encryption of y_i using public key $\mathsf{cpa.pk}_i$ and randomness r_i outputs $\mathsf{cpa.ct}_i$.

Claim 5.1. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and $j \in [N]$,

$$\Pr\left[\exists \mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right) \text{ s.t. } \begin{array}{l} \mathsf{sig.vk} \neq \mathsf{sig.vk}^* \text{ and} \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \neq \mathsf{Dec\text{-}Alt}_j(\mathsf{sk}_{-j}, \mathsf{ct}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where the probability is over the choice of CPA keys[10] and output of $\mathsf{TSC.AltSetup}$.

Proof. We consider the following cases:

1. Both decryptions output non-bot but distinct messages.

$$\Pr\left[\exists \mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right) \text{ s.t. } \begin{array}{l} \mathsf{sig.vk} \neq \mathsf{sig.vk}^* \text{ and} \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \neq \mathsf{Dec\text{-}Alt}_j(\mathsf{sk}_{-j}, \mathsf{ct}) \text{ and} \\ (\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}), \mathsf{Dec\text{-}Alt}_j(\mathsf{sk}_{-j}, \mathsf{ct})) \neq (\bot, \bot) \end{array}\right] = 0.$$

   This follows directly from the construction of our scheme. Note that Dec and $\mathsf{Dec\text{-}Alt}_j$ agree on $N-1$ of the sub-decryptions. Hence the message recovered must be the same if the output message is non-bot.

2. Decryption using sk outputs $\bot$ but decryption using sk_{-j} outputs non-bot message.

$$\Pr\left[\exists \mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right) \text{ s.t. } \begin{array}{l} \mathsf{sig.vk} \neq \mathsf{sig.vk}^* \text{ and} \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \neq \mathsf{Dec\text{-}Alt}_j(\mathsf{sk}_{-j}, \mathsf{ct}) \text{ and} \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \bot \end{array}\right] \leq \mathsf{negl}(\lambda)$$

   Here we have the following sub-cases, depending on which step of the decryption outputs $\bot$. For each of the sub-cases, we show that $\mathsf{Dec\text{-}Alt}_j$ also outputs $\bot$.

   (a) Step 1 of Dec outputs $\bot$ (that is, signature does not verify). Then Step 1 of $\mathsf{Dec\text{-}Alt}_j$ also outputs $\bot$.

   (b) Step 3 of Dec outputs $\bot$ (that is, the set U constructed by Dec has size not equal to B). If $|U| < B-1$, then Step 4 of $\mathsf{Dec\text{-}Alt}_j$ outputs $\bot$ since the set U' after Step 3 in $\mathsf{Dec\text{-}Alt}_j$ also has size less than B.
   If $|U| > B$, then this can be used to break the statistical soundness security of TSC (see Definition 4.2) because this ciphertext can produce at least $B+1$ commitments for tag $\mathsf{sig.vk} \neq \mathsf{sig.vk}^*$.
   If $|U| = B-1$, then we will show that the size of set U' after Step 3 in $\mathsf{Dec\text{-}Alt}_j$ is also $B-1$, hence $\mathsf{Dec\text{-}Alt}_j$ rejects in Step 4. Suppose on the contrary, the set U' has size B after Step 3. This means (j, y_j) was not added to U in Dec (Step 2), but the same tuple was added to U' in $\mathsf{Dec\text{-}Alt}_j$ (Step 3). Note that this implies $\mathsf{Check}(j, y_j, r_j) = 1$ and therefore, $\mathsf{CPA.Enc}(\mathsf{cpa.pk}_j, y_j; r_j) = \mathsf{cpa.ct}_j$. Using the perfect correctness of the encryption scheme, $\mathsf{CPA.Dec}(\mathsf{cpa.sk}_j, \mathsf{cpa.ct}_j) = (y_j, r_j)$. This leads to a contradiction (as $(j, y_j) \notin U$).

   (c) Step 4 outputs $\bot$. In this case, Step 5 of $\mathsf{Dec\text{-}Alt}_j$ also outputs $\bot$ since the set U recovered by Dec is identical to the set U' recovered by $\mathsf{Dec\text{-}Alt}_j$.

   (d) Step 5 outputs $\bot$. Here again, since the set U recovered by Dec is identical to the set U' recovered by $\mathsf{Dec\text{-}Alt}_j$, Step 6 of $\mathsf{Dec\text{-}Alt}_j$ also rejects here.

3. Decryption using sk_{-j} outputs $\bot$ but decryption using sk outputs non-bot message.

$$\Pr\left[\exists \mathsf{ct} = \left(\mathsf{sig.vk}, \mathsf{sig}.\sigma, \mathsf{tsc.com}, (\mathsf{cpa.ct}_i)_{i\in[N]}\right) \text{ s.t. } \begin{array}{l} \mathsf{sig.vk} \neq \mathsf{sig.vk}^* \text{ and} \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \neq \mathsf{Dec\text{-}Alt}_j(\mathsf{sk}_{-j}, \mathsf{ct}) \text{ and} \\ \mathsf{Dec\text{-}Alt}_j(\mathsf{sk}_{-j}, \mathsf{ct}) = \bot \end{array}\right] \leq \mathsf{negl}(\lambda)$$

   Here we have the following sub-cases, depending on which step of $\mathsf{Dec\text{-}Alt}_j$ outputs $\bot$. For each of the sub-cases, we show that Dec also outputs $\bot$.

[10] For simplicity, we are assuming that the underlying PKE scheme is perfectly correct, instead of almost-all-keys perfect correctness. Note that in an almost-all-keys perfect scheme, there is a negligible probability that the $(\mathsf{pk}, \mathsf{sk})$ output by setup does not satisfy correct decryption on all messages. However, since only a negligible fraction of the keys are 'bad', it suffices to focus our attention on perfectly correct encryption schemes.

(a) Step 1 of $\mathsf{Dec\text{-}Alt}_j$ outputs $\perp \implies$ Step 1 of Dec outputs $\perp$.

(b) Step 4 of $\mathsf{Dec\text{-}Alt}_j$ outputs $\perp$. Let U' be the set after Step 3 in $\mathsf{Dec\text{-}Alt}_j$, and U the set after Step 2 in Dec. If $|U'| > B$, then Step 3 of Dec outputs $\perp$ since U also has size larger than B.

If $|U'| < B-1$, then $|U| < B$, hence Step 3 of Dec outputs $\perp$.

We will now show that if $|U'| = B-1$, then either $|U|$ is $B-1$, or Step 4 of Dec outputs $\perp$. Since $|U'| = B-1$, this means the (j, y'_j, r'_j) tuple[11] extracted in Step 3 does not satisfy $\mathsf{Check}(j, y'_j, r'_j) = 1$. Let us now consider the implications of $|U| = B$ and $\oplus_{(i,y_i)\in U}\ r_i = 0^{\ell_{\mathrm{rnd}}}$. First, note that $U = U' \cup \{(j, y_j)\}$, and hence $r_j = \oplus_{(i,y_i)\in U}\ r_i = r'_j$. Since $\mathsf{Check}(j, y_j, r_j) = 1$, encryption of y_j using randomness r_j outputs $\mathsf{cpa.ct}_j$. Using the perfect correctness of the encryption scheme, it follows that $y'_j = y_j$, but this leads to a contradiction.

(c) Step 5 of $\mathsf{Dec\text{-}Alt}_j$ outputs $\perp \implies$ Step 4 of Dec outputs $\perp$ (the set U' recovered by $\mathsf{Dec\text{-}Alt}_j$ is identical to the set U recovered by Dec).

(d) Step 6 of $\mathsf{Dec\text{-}Alt}_j$ outputs $\perp \implies$ Step 5 of Dec outputs $\perp$ (same reasoning as above).

□

We will now use the alternate decryption algorithm to show that hybrids H_3 and H_4 are computationally indistinguishable. We will first define intermediate hybrid experiments $H_{3,j}$ for $0 \le j \le N$, where $H_{3,0}$ corresponds to H_3 and $H_{3,N}$ corresponds to H_4. In hybrid $H_{3,j}$, for each $i \le j$, the i^{th} challenge ciphertext component $\mathsf{cpa.ct}_i$ is an encryption of $1|\mathsf{tsc}.\sigma^*_i|m_b$. Therefore, it suffices to show that for all $j \in [N]$, $H_{3,j}$ and $H_{3,j-1}$ are computationally indistinguishable.

In order to prove $H_{3,j-1} \approx_c H_{3,j}$, we will introduce two more intermediate hybrid experiments: $H_{\mathsf{alt},j,0}$ and $H_{\mathsf{alt},j,1}$. The experiment $H_{\mathsf{alt},j,0}$ is identical to $H_{3,j-1}$, except that the challenger uses $\mathsf{Dec\text{-}Alt}_j$ instead of Dec for answering decryption queries. Similarly, the experiment $H_{\mathsf{alt},j,1}$ is identical to $H_{3,j}$, except that the challenger uses $\mathsf{Dec\text{-}Alt}_j$ instead of Dec for answering decryption queries. (Note that in both these experiments, the challenger still rejects decryption queries corresponding to $\mathsf{sig.vk}^*$).We will show that $H_{3,j-1} \approx_c H_{\mathsf{alt},j,0}$, $H_{\mathsf{alt},j,0} \approx_c H_{\mathsf{alt},j,1}$ and $H_{\mathsf{alt},j,1} \approx_c H_{3,j}$.

As before, let $\mathsf{adv}_{\mathcal{A},x}$ denote the advantage of adversary $\mathcal{A}$ in hybrid H_x.

Claim 5.2. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$ and any adversary $\mathcal{A}$, $\mathsf{adv}_{\mathcal{A},3,j-1}(\lambda) - \mathsf{adv}_{\mathcal{A},\mathsf{alt},j,0}(\lambda) \le \mathsf{negl}(\lambda)$.

Proof. The proof of this claim follows from Claim 5.1. □

Claim 5.3. Assuming the encryption scheme P_1 is IND-CPA secure, for any ppt. adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{adv}_{\mathcal{A},\mathsf{alt},j,0}(\lambda) - \mathsf{adv}_{\mathcal{A},\mathsf{alt},j,1}(\lambda) \le \mathsf{negl}(\lambda)$.

Proof. Suppose there exists a ppt. adversary $\mathcal{A}$ such that $\mathsf{adv}_{\mathcal{A},\mathsf{alt},j,0} - \mathsf{adv}_{\mathcal{A},\mathsf{alt},j,1} \le \mathsf{negl}(\lambda)$. We can use this adversary to build a reduction algorithm $\mathcal{B}$ that breaks the IND-CPA security of the encryption scheme P_1. The main observation here is that in $H_{\mathsf{alt},j,0}$ and $H_{\mathsf{alt},j,1}$, the only component that possibly changes is the j^{th} ciphertext component $\mathsf{cpa.ct}_j$, and we can reduce the computational indistinguishability of these hybrids to the IND-CPA security because both these hybrids do not use the j^{th} secret key $\mathsf{cpa.sk}_j$.

The reduction algorithm receives the public key $\mathsf{cpa.pk}_j$ from the challenger; it chooses a uniformly random B size set S, signing keys $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*)$, CPA scheme's keys $(\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i)_{i\neq j}$, runs $\mathsf{TSC.AltSetup}$ and sends pk to $\mathcal{A}$. The decryption queries are handled using $(\mathsf{cpa.sk}_i)_{i\neq j}$ since both hybrids use $\mathsf{Dec\text{-}Alt}_j$. The adversary sends its challenge messages m_0, m_1, and the reduction algorithm chooses $b \leftarrow \{0,1\}$. If $j \notin S$,[12] the reduction algorithm sends $0^{\ell_{\mathsf{cpa}}}, 1|\mathsf{tsc}.\sigma^*_j|m_b$ to the challenger as challenge messages, and receives $\mathsf{cpa.ct}_j$. It then computes the remaining ciphertext components and sends the ciphertext ct to $\mathcal{A}$. The adversary then makes polynomially many post-challenge decryption queries, and finally sends its guess b'. The reduction algorithm guesses that $\mathsf{cpa.ct}_j$ is encryption of $0^{\ell_{\mathsf{cpa}}}$ iff $b = b'$. □

[11] We use y'_j, r'_j here to distinguish it from y_j, r_j which are computed in Step 2 of Dec.

[12] If $j \in S$, then these two hybrids are identical.

Claim 5.4. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$ and any adversary $\mathcal{A}$, $\mathsf{adv}_{\mathcal{A},\mathsf{alt},j,1}(\lambda) - \mathsf{adv}_{\mathcal{A},3,j}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this claim follows from Claim 5.1. □

□

Lemma 5.5. There exists a negligible function $\mathsf{negl}(\cdot)$ s.t. for all $\lambda \in \mathbb{N}$, and any adversary $\mathcal{A}$, $\mathsf{pr}_{\mathcal{A},4}(\lambda) - \mathsf{pr}_{\mathcal{A},5}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. First, let us consider the distribution $\mathcal{D}$ defined by the following experiment:

- choose a random vector $\mathbf{x} = (x_1, x_2, \ldots, x_{N-1}) \leftarrow \left(\{0,1\}^{\ell_{\mathrm{rnd}}}\right)^{N-1}$.
- choose a random vector $\mathbf{z} \leftarrow \{0,1\}^{N-1}$ of Hamming weight $B-1$.
- output $(\mathbf{x}, \oplus_{i:z_i=1} x_i)$.

Let $\mathcal{U}$ be the uniform distribution over $\left(\{0,1\}^{\ell_{\mathrm{rnd}}}\right)^{N}$.

Claim 5.5.

$$\mathsf{SD}(\mathcal{D},\mathcal{U}) \leq 2^{-\lambda}.$$

Proof. This follows from the Leftover Hash Lemma [HILL99]. Let $h_{\mathbf{x}}$ be a hash function defined by $\mathbf{x} = (x_1, \ldots, x_{N-1})$ which maps $N-1$ bits to ℓ_{rnd} bits as follows: $h_{\mathbf{x}}(z) = \oplus_{i:z_i=1} x_i$. Let $\mathcal{Y}$ denote the uniform distribution over all $N-1$ bit strings of Hamming weight $B-1$. This distribution has min-entropy $H_\infty(\mathcal{Y}) = \log\left(\binom{N-1}{B-1}\right)$. Since the hash function family $\{h_{\mathbf{x}}\}_{\mathbf{x}\in(\{0,1\}^{\ell_{\mathrm{rnd}}})^{N-1}}$ is a pairwise-independent hash function family and $H_\infty(\mathcal{Y}) > \ell_{\mathrm{rnd}} + 2\lambda$, $\mathsf{SD}(\mathcal{D},\mathcal{U}) \leq 2^{-\lambda}$. □

As a corollary, it follows that the following distribution $\mathcal{D}'$ is also close to uniform:

- choose a random vector $\mathbf{z}' \leftarrow \{0,1\}^N$ of Hamming weight B. Let $i_1 < i_2 < \ldots < i_B$ denote the indices such that $z'_{i_j} = 1$.
- for each $i \neq i_B$, choose $x'_i \leftarrow \{0,1\}^{\ell_{\mathrm{rnd}}}$.
- set $x'_{i_B} = \oplus_{j<B} x'_{i_j}$ and output $\mathbf{x}'$.

Corollary 5.2.

$$\mathsf{SD}(\mathcal{D}', U) \leq 2^{-\lambda}.$$

Proof. Given a sample $\mathbf{x}$ which is either from $\mathcal{D}$ or $\mathcal{U}$, one can generate a sample from either $\mathcal{D}'$ or U as follows: choose a random permutation $\pi : [N] \to [N]$, and permute the components of $\mathbf{x}$ according to π; that is, set $x'_i = x_{\pi(i)}$ for all $i \in [N]$. Clearly, if $\mathbf{x}$ is a uniformly random sample from $\left(\{0,1\}^{\ell_{\mathrm{rnd}}}\right)^{N}$, then the resulting vector $\mathbf{x}'$ is also a uniformly random sample.

Suppose $\mathbf{x}$ is a sample from $\mathcal{D}$, and let $\mathbf{z} \in \{0,1\}^{N-1}$ be the random $B-1$ weight vector chosen by $\mathcal{D}$ sampler with 1 at positions $\{i_1, \ldots, i_{B-1}\}$. Let $\mathbf{z}' \in \{0,1\}^N$ be a B weight vector which has 1 at positions $\{\pi(i_1), \ldots, \pi(i_{B-1}), \pi(N)\}$ and 0 elsewhere. Since π is a uniformly random permutation, the vector $\mathbf{z}'$ is a uniformly random B weight vector and the resulting vector $\mathbf{x}'$ is from distribution $\mathcal{D}'$. □

Using this corollary, we can now prove our lemma. Note that the only difference between the two hybrid experiments is the choice of randomness for encryptions. In Hybrid H_4, the challenger chooses a B-size set $S = \{i_1, \ldots, i_B\}$, chooses $r_i \leftarrow \{0,1\}^{\ell_{\mathrm{rnd}}}$ for all $i \neq i_B$ and sets $r_{i_B} = \oplus_{j\in[B]} r_{i_j}$. This corresponds to the distribution $\mathcal{D}'$. In Hybrid H_5, all r_i are chosen uniformly at random.

□

Lemma 5.6. Assuming encryption scheme with randomness recovery P_2 is an IND-CPA secure encryption scheme and the tagged set commitment scheme P_3 satisfies statistical soundness (Definition 4.2), for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{adv}_{\mathcal{A},5}(\lambda) - \mathsf{adv}_{\mathcal{A},6}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is very similar to the proof of Lemma 5.4, the only difference being that there is no set S^* here (that is, we switch all ciphertexts to being encryptions of $0^{\ell_{\mathsf{cpa}}}$; in Lemma 5.4, the ciphertext components corresponding to indices in set S^* were not altered). We include the proof in Appendix A for completeness. □

References

[BHSV98] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In *Annual International Cryptology Conference*, 1998.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *Advances in Cryptology - EUROCRYPT*, pages 535–564, 2018.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In *Advances in Cryptology - CRYPTO*, pages 33–65, 2017.

[CKS09] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. *J. Cryptology*, 22(4):470–504, 2009.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, London, UK, 1998. Springer.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Proceedings of Eurocrypt '02*, volume 2332, pages 45–64, 2002.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Computing*, 30(2):391–437, 2000.

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In *Theory of Cryptography*, pages 372–408, 2017.

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In *Advances in Cryptology - CRYPTO*, pages 537–569, 2017.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In *Public-Key Cryptography - PKC*, pages 3–31, 2018.

[DMN12] Nico Döttling, Jörn Müller-Quade, and Anderson C. A. Nascimento. IND-CCA secure cryptography based on a variant of the LPN problem. In *Advances in Cryptology - ASIACRYPT*, pages 485–503, 2012.

[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, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360. Springer, 2004.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In *International Workshop on Public Key Cryptography*, pages 53–68. Springer, 1999.

[GGH] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. In Yuval Ishai and Vincent Rijmen, editors, *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*, volume 11478 of *Lecture Notes in Computer Science*.

[GH18] Sanjam Garg and Mohammad Hajiabadi. Trapdoor functions from the computational diffie-hellman assumption. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 362–391, 2018.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 126–135. IEEE Computer Society, 2001.

[Gol11] Oded Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 406–421. 2011.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HK08] Goichiro Hanaoka and Kaoru Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. In *Advances in Cryptology - ASIACRYPT*, pages 308–325, 2008.

[HK09] Dennis Hofheinz and Eike Kiltz. Practical chosen ciphertext secure encryption from factoring. In *Advances in Cryptology - EUROCRYPT*, pages 313–332, 2009.

[HO09] Brett Hemenway and Rafail Ostrovsky. Lossy trapdoor functions from smooth homomorphic hash proof systems. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:127, 2009.

[KL08] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2008.

[KM19] Fuyuki Kitagawa and Takahiro Matsuda. Cpa-to-cca transformation for KDM security. In *Theory of Cryptography TCC*, pages 118–148, 2019.

[KMP14] Eike Kiltz, Daniel Masny, and Krzysztof Pietrzak. Simple chosen-ciphertext security from low-noise LPN. In *Public-Key Cryptography - PKC*, pages 1–18, 2014.

[KW19] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In *Advances in Cryptology - CRYPTO*, pages 671–700, 2019.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. *Tech. Report: SRI International Computer Science Laboratory*, 1979.

[MY10] Petros Mol and Scott Yilek. Chosen-ciphertext security from slightly lossy trapdoor functions. In *Public Key Cryptography - PKC*, pages 296–311, 2010.

[Nao89] Moni Naor. Bit commitment using pseudo-randomness. In *Advances in Cryptology - CRYPTO*, pages 128–136, 1989.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437, 1990.

[Pan13] Omkant Pandey. Personal communication, 2013.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 187–196, 2008.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 433–444, 1991.

[RS10] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. *SIAM J. Comput.*, 39(7):3058–3088, 2010.

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

[Sho98] Victor Shoup. Why chosen ciphertext security matters, 1998. IBM TJ Watson Research Center.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, 1982.

A Proof of Lemma 5.6

As in the proof of Lemma 5.4, we will first define intermediate hybrid experiments $H_{5,j}$ for $0 \leq j \leq N$, where $H_{5,0}$ corresponds to H_5 and $H_{5,N}$ corresponds to H_6. In hybrid $H_{5,j}$, the first j ciphertext components are encryptions of $0^{\ell_{\mathsf{cpa}}}$. We will show that for all $j \in [N]$, $H_{5,j} \approx_c H_{5,j-1}$.

We will introduce two more intermediate hybrid experiments: $H_{\mathsf{alt},j,0}$ and $H_{\mathsf{alt},j,1}$.[13] The experiment $H_{\mathsf{alt},j,0}$ is identical to $H_{5,j-1}$, except that the challenger uses $\mathsf{Dec\text{-}Alt}_j$ instead of Dec for answering decryption queries. Similarly, the experiment $H_{\mathsf{alt},j,1}$ is identical to $H_{5,j}$, except that the challenger uses $\mathsf{Dec\text{-}Alt}_j$ instead of Dec for answering decryption queries. (Note that in both these experiments, the challenger still rejects decryption queries corresponding to $\mathsf{sig.vk}^*$).We will show that $H_{5,j-1} \approx_c H_{\mathsf{alt},j,0}$, $H_{\mathsf{alt},j,0} \approx_c H_{\mathsf{alt},j,1}$ and $H_{\mathsf{alt},j,1} \approx_c H_{5,j}$.

As before, let $\mathsf{adv}_{\mathcal{A},x}$ denote the advantage of adversary $\mathcal{A}$ in hybrid H_x.

Claim A.1. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$ and any adversary $\mathcal{A}$, $\mathsf{adv}_{\mathcal{A},5,j-1}(\lambda) - \mathsf{adv}_{\mathcal{A},\mathsf{alt},j,0}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this claim follows from Claim 5.1. □

Claim A.2. Assuming the encryption scheme P_1 is $\mathsf{IND\text{-}CPA}$ secure, for any ppt. adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{adv}_{\mathcal{A},\mathsf{alt},j,0}(\lambda) - \mathsf{adv}_{\mathcal{A},\mathsf{alt},j,1}(\lambda) \leq \mathsf{negl}(\lambda)$.

[13] Note that we are overloading the names of these hybrid names (the same names were used for intermediate hybrids in the proof of Lemma 5.4). This is to avoid additional subscripts in the hybrid name.

Proof. Suppose there exists a ppt. adversary $\mathcal{A}$ such that $\mathsf{adv}_{\mathcal{A},\mathsf{alt},j,0} - \mathsf{adv}_{\mathcal{A},\mathsf{alt},j,1} \leq \mathsf{negl}(\lambda)$. We can use this adversary to build a reduction algorithm $\mathcal{B}$ that breaks the IND-CPA security of the encryption scheme P_1. The main observation here is that in $H_{\mathsf{alt},j,0}$ and $H_{\mathsf{alt},j,1}$, the only component that possibly changes is the j^{th} ciphertext component $\mathsf{cpa.ct}_j$, and we can reduce the computational indistinguishability of these hybrids to the IND-CPA security because both these hybrids do not use the j^{th} secret key $\mathsf{cpa.sk}_j$.

The reduction algorithm receives the public key $\mathsf{cpa.pk}_j$ from the challenger; it chooses signing keys $(\mathsf{sig.sk}^*, \mathsf{sig.vk}^*)$, CPA scheme's keys $(\mathsf{cpa.pk}_i, \mathsf{cpa.sk}_i)_{i \neq j}$, runs TSC.AltSetup and sends pk to $\mathcal{A}$. The decryption queries are handled using $(\mathsf{cpa.sk}_i)_{i \neq j}$ since both hybrids use $\mathsf{Dec\text{-}Alt}_j$. The adversary sends its challenge messages m_0, m_1, and the reduction algorithm chooses $b \leftarrow \{0,1\}$. The reduction algorithm sends $1|\mathsf{tsc}.\sigma_j^*|m_b, 0^{\ell_{\mathsf{cpa}}}$ to the challenger as challenge messages, and receives $\mathsf{cpa.ct}_j$. It then computes the remaining ciphertext components and sends the ciphertext ct to $\mathcal{A}$. The adversary then makes polynomially many post-challenge decryption queries, and finally sends its guess b'. The reduction algorithm guesses that $\mathsf{cpa.ct}_j$ is encryption of $1|\mathsf{tsc}.\sigma_j^*|m_b$ iff $b = b'$. □

Claim A.3. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$ and any adversary $\mathcal{A}$, $\mathsf{adv}_{\mathcal{A},\mathsf{alt},j,1}(\lambda) - \mathsf{adv}_{\mathcal{A},3,j}(\lambda) \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this claim follows from Claim 5.1. □