SO-CCA secure PKE from pairing based all-but-many lossy trapdoor functions

Dingding Jia, Benoît Libert

SO-CCA Secure PKE from Pairing based ABM-LTFs

Dingding Jia[1,2] and Benoît Libert[2,3]

[1] State Key Laboratory of Information Security, Institute of Information Engineering of Chinese Academy of Sciences, Beijing China
[2] CNRS, Laboratoire LIP, France
[3] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL), France
jiadingding@iie.ac.cn,benoit.libert@ens-lyon.fr

Abstract In a selective-opening chosen ciphertext (SO-CCA) attack on an encryption scheme, an adversary A has access to a decryption oracle, and after getting a number of ciphertexts, can then adaptively corrupt a subset of them, obtaining the plaintexts and corresponding encryption randomness. SO-CCA security requires the privacy of the remaining plaintexts being well protected. There are two flavors of SO-CCA definition: the weaker indistinguishability-based (IND) and the stronger simulation-based (SIM) ones. In this paper, we study SO-CCA secure PKE constructions from all-but-many lossy trapdoor functions (ABM-LTFs) in pairing-friendly prime order groups. Concretely,

– we construct two ABM-LTFs with $O(n/\log\lambda)$ size tags for n bits inputs and security parameter λ, which lead to IND-SO-CCA secure PKEs with ciphertext size $O(n/\log\lambda)$ to encrypt n bits messages. In addition, our second ABM-LTF enjoys tight security, so as the resulting PKE.
– by equipping a lattice trapdoor for opening randomness, we show our ABM-LTFs are SIM-SO-CCA compatible.

Keywords: public key encryption, all-but-many lossy trapdoor functions, selective-opening security, chosen-ciphertext secure, tight security

1 Introduction

Selective-opening attacks (SO) considers a scenario involving a receiver and Q senders. They encrypt (possibly correlated) messages $(M_1, \ldots, M_Q)$ under the receiver's public key PK, and upon receiving the ciphertexts $(C_1, \ldots, C_Q)$, the adversary corrupts a subset of the senders by choosing $I \subset [Q]$. It then obtains the messages $\{M_i\}_{i\in I}$ as well as the random coins $\{r_i\}_{i\in I}$ for which $C_i = \mathsf{Encrypt}(PK, M_i; r_i)$. The goal of the attacker is to break the security of the unopened ciphertext $\{C_i\}_{i\in[Q]\setminus I}$. The main difficulty towards proving selective-opening security arises from the fact that encrypted messages $\{M_i\}_{i\in I}$ could be correlated and the adversary obtains the random coins $\{r_i\}_{i\in I}$ of corrupted senders.

While traditional security notions seem to imply security under selective openings at the first glance, it surprisingly turned out [BDWY12,HRW16] that all bets are off when ordinary IND-CCA secure encryption schemes are subject to sender corruptions in the absence of reliable erasures. Even the strong notion of IND-CCA security [HRW16] was found not to guarantee security in the sense of a weak definition of indistinguishability-based SO security.

Two formalizations of selective-opening security have been considered in the literature. The first one is indistinguishability-based [BHY09,BY09] and requires unopened plaintexts $\{M_i\}_{i\in[Q]\setminus I}$ to be indistinguishable from messages $\{M_i'\}_{i\in[Q]\setminus I}$ that are independently resampled. However, this indistinguishability-based (IND-SO) formalization has several drawbacks. We need the resampling operation to be conditioned on the adversary's view in order to make the adversary's task non-trivial. Hence, the challenger is only efficient if the conditional resampling operation can be done efficiently, which is much more restrictive than only asking for efficient samplability: Indeed, message distributions where some messages are one-way functions of other messages are not efficiently resamplable.

To overcome the limitations of IND-SO security, Bellare *et al.* [BY09,BHY09] proposed a stronger, simulation-based notion of selective-opening (SIM-SO) security. In this model, we require the output of the adversary (after having seen the ciphertexts as well as the corrupted messages and randomness) to be efficiently simulatable from *only the corrupted plaintexts* $\{M_i\}_{i\in I}$, and thus in particular without seeing the other ciphertexts nor the public key. Unlike the indistinguishability-based definition, this strong notion does not induce any restriction on the message distribution besides being efficiently samplable. On the other hand, SIM-SO security has proven to be much harder to achieve. In particular, it is not implied by IND-SO security in general, as proven in [BHK12].

SIM-SO security naturally extends to the chosen-ciphertext (CCA) setting, where the adversary is additionally allowed to make decryption queries. While SIM-SO-CPA secure public-key encryption schemes are known under various number theoretic assumptions (e.g., Quadratic [BY09] and Composite Residuosity [HLOV11], DDH [BY09,HJR16]), achieving SIM-SO-CCA security turns out to be considerably more challenging. Indeed, in the standard model, most constructions based on standard assumptions [FHKW10,HLQ13,LDL$^+$14,LP15,LLHG18] encrypt messages bit-by-bit. In other words, they encrypt '0' and '1' in indistinguishable but different way, and finally authenticate the encryption of all bits together. So the resulting SIM-SO-CCA schemes have ciphertexts of size $O(n)$ for an n bits message.

As an alternative way to build SO-CCA secure PKE, Hofheinz [Hof12] introduced a primitive called *all-but-many lossy trapdoor functions* (ABM-LTF). For the time being, only three constructions achieve SIM-SO-CCA security with more compact ciphertext [Hof12,Fuj14,LSSS17a] in the standard model, all via ABM-LTF. The solutions of [Hof12,Fuj14] rely on a non-standard variants of the Composite Residuosity assumption [Pai99]. Under the standard Learning-With-Errors (LWE) assumption [Reg05], SIM-SO-CCA security was actually achieved [LSSS17a] while encrypting many bits at once. On the downside, the scheme of Libert *et al.* [LSSS17a] is rather expensive in terms of computation as its encryption algorithm appeals to the fully homomorphic encryption (FHE) scheme of Gentry *et al.* [GSW13] to homomorphically evaluate a pseudorandom function. In addition, the LWE-based ABM-LTF in [LSSS17a] achieves tight security. Tight security concerns about the security loss when one reduces the ability of breaking the scheme to that of solving the underlying hard-solved problem. If the simulator, that runs in similar time as the adversary $\mathcal{A}$, can transfer $\mathcal{A}$'s ability that has advantage ε_1 of attacking the scheme to an algorithm that solves the underlying problem with probability ε_2, then the security loss is defined as $L = \frac{\varepsilon_1}{\varepsilon_2}$. As bigger ε_2 will result in smaller parameter size and more efficient computation, while smaller ε_1 has better security guarantee, we hope the security loss L as small as possible.

Besides the LWE based construction via ABM-LTF, Lyu *et al.* [LLHG18] also constructed tightly secure SIM-SO-CCA secure PKEs in the standard model, based on the DDH assumption, via the bit-by-bit style. We wonder whether there exist other efficient approaches to achieve tightly SIM-SO-CCA secure encryptions, like ABM-LTFs.

1.1 Our Contributions

In this paper, we give two SIM-SO-CCA secure PKEs based on SXDH-alike assumptions in pairing-friendly, prime order groups. In addition, our second construction is (almost) tightly secure. To achieve this,

- Firstly we construct an SO-CCA compatible ABM-LTF with tag size $O(n/\log\lambda)$ for n bits input and security parameter λ.
- Then by embedding an almost tightly secure MAC [LQ19] to replace Waters signature scheme, we get an ABM-LTF with almost tight security. Our ABM-LTF is the first tightly secure one based on a standard, not lattice-based assumption.
- Finally, by combining with the LTF given in Appendix E of [LSSS17a], and setting a lattice trapdoor to enable randomness opening for the simulation phase, we give two SIM-SO-CCA secure encryptions. In addition, our second PKE is tightly SIM-SO-CCA secure.

1.2 Technical Overview

Our scheme builds on the lossy encryption paradigm [BHY09,BY09], where normal public keys are indistinguishable from lossy public keys, for which ciphertexts are statistically independent of encrypted messages. In order to prove SIM-SO-CCA security, we endow our scheme with a weak efficient opening algorithm. In short, *weak efficient opening* [BHY09] means that lossy ciphertexts should be equivocable in the same way as a chameleon hash function: Namely, a trapdoor information should make it possible to efficiently find collisions $(M_0, r_0), (M_1, r_1)$ such that $\mathsf{Encrypt}(PK, M_0; r_0) = \mathsf{Encrypt}(PK, M_1; r_1)$ when PK is a lossy public key. Bellare *et al.* [BHY09,BY09] showed that any IND-SO secure encryption scheme also enjoys simulation-based security when such a weak efficient opening algorithm exists.

Our lossy encryption scheme relies on lossy trapdoor functions (LTFs) [PW08] and their generalization called *all-but-many lossy trapdoor functions* (ABM-LTFs) [Hof12]. LTFs are function families where injective functions are computationally indistinguishable from many-to-one functions, which have a much smaller image size. ABM-LTFs are an extension of LTFs, introduced by Hofheinz [Hof12], where each function is parametrized by a tag t which determines if $g(t, X)$ is injective or lossy as a function of X. Each tag $t = (t_a, t_c)$ is actually comprised of an auxiliary component t_a (which can be an arbitrary string) and a core component t_c. For any t_a, there exists at least one t_c such that $t = (t_a, t_c)$ induces a lossy function $g(t, \cdot)$. At the same time, lossy tags should be computationally indistinguishable from random tags (a property known as *indistinguishability*) and they should be hard to compute without a trapdoor (another property termed *evasiveness* in [Hof12]). The key feature of ABM-LTFs is that, unlike all-but-one trapdoor functions defined in [PW08], each function has a super-polynomial number of lossy tags, although they are sparse in the tag space. Hofheinz proved in [Hof12] that any ABM-LTF $g(\cdot, \cdot)$ can be generically combined with an LTF $f(\cdot)$ to build an encryption scheme with IND-SO-CCA security. Specifically, the ciphertext of the resulting PKE consists of:

$$(t_c, f(X), g(t, X), AE.E(h(X), m)),$$

here X is the encryption randomness, h is a hard-core function for f and g, $f(X)$ serves as the auxiliary tag t_a, and AE is a symmetric authentication encryption. In the security proof, one will change f and tags for g to lossy, then $h(X)$ will be completely random distributed, thus assures the privacy of m. In the security proof, f is changed to be lossy, and t_c in the challenge ciphertext is modified to a lossy core tag. This change should be undetected by the adversary, which means that t_c should be opened as a random core tag. We will refer to this property as *explainable* for the rest of the paper. Note that here the adversary may submit decryption queries, while the reduction can only answer for those with injective tags, the evasiveness property prohibits the adversary from producing a valid query with lossy tag. Hence only an ABM-LTF with tight evasiveness can leads to a tightly secure SO-CCA PKE.

In this paper, we concentrate on constructions based on the DDH-type assumptions, in prime order groups and with smaller tag size. In the following we firstly review previous constructions, then show how to get our final result step by step.

ABM-LTF construction: initialization. In [Hof12], Hofheinz observed that the evasiveness requirement for ABM-LTF is similar to the unforgeability of signatures, then they associate core tag part with the Boneh-Boyen signatures [Wat05,BB08] in their pairing based construction; and to provide indistinguishability, they blind the signatures with random group elements.

The construction in [Hof12] runs in pairing groups with composite order $N = p_1 p_2$. Subgroups with order p_1 is used to embed the Boneh-Boyen signature scheme, and subgroup of order p_2 is used for two purposes: to hide signatures to achieve indistinguishability, and for the inversion process. As the Boneh-Boyen signature scheme, they employed a non-standard q-type assumption. And the tag size is $O(n^2)$ for n bits input.

ABM-LTF construction: in prime order group. To construct an ABM-LTF in prime order groups, two things should be settled: one is to get signatures properly blinded, the other is to find an inversion key to recover the input from output. Libert and Qian [LQ19] noticed that a construction of lossy algebraic filters (LAF) implicitly gives a solution of that. LAF is proposed by Hofheinz [Hof13] to construct circular CCA secure PKE schemes, which also operates with injective or lossy tags, and embraces evasiveness and indistinguishability as ABM-LTF. The difference lies in that, in LAF, for injective tags, inversion is no longer required; for lossy tags, the output should always be the same linear combination of the input. The LAF given in [Hof13] runs in prime order groups, based on the standard decisional linear assumption (DLIN). Instead of using another subgroup to achieve indistinguishability, they used a fixed group element to hide the signatures, thus solved the first problem. Then [LQ19] indicates that, by modifying the evaluation algorithm from $\mathbf{M} \circ \mathbf{x}$ to $\mathbf{M}^T \circ \mathbf{x}$, there exists an implicitly inversion key from the key generation process of the LAF in [Hof13]. In this way one can actually get an ABM-LTF in prime order groups, but still with tag size of $O(n^2)$.

ABM-LTF construction: tag size from $O(n^2)$ to $O(n)$. In [Hof12,Hof13], to evaluate a function with n bits input, they employ an n by n tag related matrix. And the tag size is $O(n^2)$ as all matrix elements are given explicitly, which is similar to the structure of the evaluation key of the DDH based LTF given by Peikert and Waters [PW08]. One could notice that the off-diagonal elements have the same distribution for both lossy and injective tags, the diagonal elements mark the lossiness of tags. So the central part of shrinking the tag size is to shrink the off-diagonal parts. In 2010, Boyen and Waters proposed two methods to shrink the evaluation key size for LTF from $O(n^2)$ to $O(n)$ [BW10]. The first method is to give the off-diagonal elements explicitly as the common reference strings (CRS), resulting a size $O(n^2)$ CRS, then the evaluation key only consists of the diagonal elements. The second one is inspired by the revocation IBE system [LSW10], so that one can re-construct the $O(n^2)$ off-diagonal elements from $4n$ elements. Inspired by the second key size shrinking method, Libert and Qian [LQ19] shrink the tag size of the LAF from $O(n^2)$ to $O(n)$. Furthermore, by replacing the underlying signature scheme with a tightly secure one, they obtained an LAF with tags of size $O(n)$ and (almost) tight evasiveness.

Although the LAFs given by Libert and Qian can be modified to ABM-LTFs, their constructions are not compatible with SO-CCA requirement. That is because they introduce some algebraic structure for computing $n^2 - n$ off-diagonal elements from $4n$ group elements, hence lossy tags cannot be opened as random tags without detecting. So our goal is to shrink the tag size without introducing such structures.

Our first contribution: SO-CCA compatible ABM-LTF with tag size $O(n)$ in prime order group. As mentioned above, there are two methods to reduce the evaluation key size for LTF in [BW10]. While the algebraic structure from the second method hinder the resulting ABM-LTF from achieving SO-CCA PKE, luckily, the first method of [BW10] does not introduce any algebraic structure, it gives the off-diagonal elements explicitly in the CRS. By adapting this method to the ABM-LTF construction in [LQ19], and embedding a blind version of Waters signatures, we get an SO-CCA compatible ABM-LTF. Since we employ a blind version of Waters signature scheme, evasiveness of our scheme is reduced to the 2-3-Diffie-Hellman assumption [KP06], as that in [LQ19]. Compared with the constructions in [LQ19], our construction has larger evaluation key size of $O(n^2)$ group elements, but the tag size keeps small of $O(n)$.

Our second contribution: achieving (almost) tight evasiveness. As all previous ABM-LTF achieves evasiveness via embedding a signature scheme implicitly in tags, one natural approach to achieving tight evasiveness is to replace Waters signatures with a signature scheme with tight unforgeability. For the signature scheme, as the validity of signatures can be publicly checked, one-time security and multi-time security is equivalent. However, for tags with indistinguishability, there is a gap between the reduction for one-time and multi-time security. As here we need multi-time security for evasiveness, i.e. the adversary could access to an oracle that helps to check the lossiness of the tag, for multi-time, conventional tightly secure signature scheme cannot be used directly here. Luckily, in [LQ19], they adapted a trapdoor to the MAC in [BKP14], so that the reduction can always check the validity of the tag, thus achieves tight security for multi-challenge. Then by embedding the MAC given in [LQ19] to our ABM-LTF, we finally get a construction with (almost) tight evasiveness[4], which is the first one with tight security and based on a standard, non-lattice assumption. With the above ABM-LTFs, we can get IND-SO-CCA secure constructions with compacter ciphertext of $O(\ell/\log\lambda)$ group elements for encrypting ℓ bits messages.

Our third contribution: SIM-SO-CCA secure PKE. To achieve SIM-SO-CCA security, one has to find an appropriate opening algorithm to explain $f(X), g(t, X)$ as function evaluation of another X'. Note that the DDH-based ABM-LTF given in [Hof12] is not SIM-SO-CCA compatible for the lack of efficient opening algorithm. In [LSSS17a], they use a lattice trapdoor to open a DDH-based LTF, and finally get an SIM-SO-CPA secure PKE. The centre our ABM-LTF opening algorithm is the same as the DDH-based LTF: one need to find a *short* vector $\mathbf{x}$ such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ given a fixed $\mathbf{b}$ and matrix $\mathbf{A}$. Then, as in [LSSS17a], with a trapdoor $T_{\mathbf{A}}$ generated together with $\mathbf{A}$, one can sample such an $\mathbf{x}$ from the Gaussian distribution. With the help of such a trapdoor, by modifying the evaluation of our ABM-LTFs a little bit, and restrict the input to Gaussian distribution, one can get SIM-SO-CCA compatible ABM-LTFs.

[4] We suppose that the tightly multi-pesudorandom MAC given in [HJP18] can also be used here, however, the security loss of their construction is larger than that of the MAC in [LQ19], although in the same level.

With the ABM-LTFs we constructed above, we can finally extend the DDH-based SIM-SO-CPA secure PKE in [LSSS17a] to two SIM-SO-CCA constructions. Both are in prime order, pairing-friendly groups. And our second construction achieves (almost) tight security. In Table 1, we give a comparison of the currently known SIM-SO-CCA secure PKE constructions. For the use of lattice trapdoor, our final construction needs $O(\ell)$ group elements to encrypt ℓ bits messages, as that in [LLHG18]. This shows that the lossy encryption approach can perform asymptotically as well as the bit-by-bit KEM-based approach of [LP15,LLHG18] (which is the most efficient blueprint so far) when it comes to constructing SIM-SO-CCA-secure encryption. The main caveat is that the lossy encryption approach requires pairings for now. We leave it as open problems to construct ABM-LTF with shorter evaluation keys, tighter evasiveness, say $O(\log q)$, and in pairing-free prime order groups.

Approach	compactness	Scheme	tightness	Assump.	Remark
bit-by-bit	–	FHKW10	–	DCR,DDH,QR	from HPS
		LP15	–	HPS,Dlin,iO	from KEM
		LLHG18	√	MDDH	from tightly secure KEM
ABM-LTF	√	Hof12a	√	DCR+No-mul	non-standard assump.
	√	Hof12b	√	SXDH+SD+strong DH	composite order, paring,non-standard assum.
	√	LSSS17	√	LWE	use homomorphic encryption, low efficiency
	–	Ours	– √	wR3DH+SXDH+2-3-CDH	prime order, pairing, widely-used assum.

Table 1. Comparison of the currently known SIM-SO-CCA secure PKE. Here we assume the encrypted message is of length n.

1.3 Related Work

The non-triviality of selective-opening security was first identified by Dwork *et al.* [DNRS99] in 1999. The first positive results on the feasibility of SO-secure public-key encryption were given by Bellare, Hofheinz and Yilek [BHY09,BY09] a decade later. They proved that IND-SO-CPA security can be achieved using lossy trapdoor functions and, more efficiently, under the standard DDH assumption. At the expense of encrypting messages bitwise, they also showed how to prove simulation-based (SIM-SO-CPA) security under the Quadratic Residuosity and DDH assumptions. In particular, they proved that the Goldwasser-Micali cryptosystem [GM84] is actually secure in the SIM-SO-CPA sense and their result was immediately extended to Paillier [HLOV11]. Under the DDH assumption, Hofheinz, Jager and Rupp [HJR16] obtained compact ciphertexts while retaining simulation-based security. Bellare, Waters and Yilek [BWY11] extended and realized the notion of SIM-SO-CPA security to the identity-based setting. Hoang *et al.* [HKOZ16] analyzed the feasibility of SO security using deficient randomness.

Selective-opening chosen-ciphertext security was first considered by Fehr *et al.* [FHKW10] and attracted much attention since then [HLQ13,Hof12,LDL^{+}14], [LP15,BL17,LSSS17a,LLHG18]. Of these works, [BL17] only considers the weaker IND-SO-CCA security. Except for realizations [Hof12,Fuj14] based on (variants of) the Composite Residuosity assumption and the FHE-based construction of [LSSS17a], all of these schemes process messages bit-by-bit, thus incurring an expansion factor $\Omega(\lambda)$. In the random oracle model, Heuer *et al.* [HJKS15] gave much more efficient constructions by showing that several practical schemes like RSA-OAEP [BR95] are secure in the SIM-SO-CCA sense. In the ideal cipher model, Heuer and Poettering [HP16] generically realized SIM-SO-CCA security using hybrid encryption.

Until 2015, selective-opening security was mostly considered for corruptions at the senders. The receiver corruption (RSO) setting was fleshed out by Hazay *et al.* [HPW15] who gave constructions under various assumptions. In particular, they achieved simulation-based RSO security from receiver non-committing encryption [JL00,CHK05]. [JLL16,JLL17] extends RSO security to the CCA setting, but they only consider the IND-based definition. The SIM-based RSO security was recently extended to the chosen-ciphertext scenario by Hara *et al.* [HKM+18,HLC+19,**?**], who gave solutions under standard assumptions.

2 Preliminaries

Notations. For any $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers with addition and multiplication modulo q. We always set q as a prime integer. If $\mathbf{x}$ is a vector over $\mathbb{R}$, then $\|\mathbf{x}\|$ denotes its Euclidean norm. If X is a random variable over a countable domain, the min-entropy of X is defined as $H_\infty(X) = \min_x(-\log_2 \Pr[X = x])$. If X and Y are distributions over the same domain, then $\Delta(X, Y)$ denotes their statistical distance. We let $\sigma_n(\mathbf{M})$ denote the least singular value of matrix $\mathbf{M}$, where n is the rank of $\mathbf{M}$. We use $U(S)$ to denote the uniform distribution on the set S. We use $x \xleftarrow{\$} S$ to denote choosing x uniformly random from the set S, and $x \leftarrow D$ picking an element according to the distribution D.

2.1 Randomness Extraction and Chameleon Hash Function

We first recall the Leftover Hash Lemma, as it was stated in [ABB10].

Lemma 1 ([ABB10]). *Let $\mathcal{H} = \{h : X \to Y\}_{h \in \mathcal{H}}$ be a family of universal hash functions, for countable sets X, Y. For any random variable T taking values in X, we have*

$$\Delta\big((h, h(T)), (h, U(Y))\big) \leq \frac{1}{2} \cdot \sqrt{2^{-H_\infty(T)} \cdot |Y|}.$$

More generally, let $(T_i)_{i \leq k}$ be independent random variables with values in X, for some $k > 0$. We have

$$\Delta\big((h, (h(T_i))_{i \leq k}), (h, (U(Y)_i)_{i \leq k})\big) \leq \frac{k}{2} \cdot \sqrt{2^{-H_\infty(T)} \cdot |Y|}.$$

A consequence of Lemma 1 was used by Agrawal *et al.* [ABB10] to re-randomize matrices over $\mathbb{Z}_q$ by multiplying them with small-norm matrices.

Lemma 2 ([ABB10]). *Let us assume that $m > 2n \cdot \log q$, for some prime $q > 2$. For any integer $k \in \mathsf{poly}(n)$, if $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{k\times n}$, $\mathbf{R} \xleftarrow{\$} \{-1,1\}^{k\times m}$, the distributions $(\mathbf{A}, \mathbf{R}\cdot\mathbf{A})$ and $(\mathbf{A}, \mathbf{B})$ are within $2^{-\Omega(n)}$ statistical distance.*

Definition 1 (Chameleon hash function [KR00]). *A chameleon hash function has three algorithms $\mathsf{CH} := (\mathsf{CH.gen}, \mathsf{CH.eval}, \mathsf{CH.switch})$:*

- *The key generation algorithm $\mathsf{CH.gen}(\lambda)$ returns the evaluation/trapdoor key $(\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}})$.*
- *The evaluation algorithm $\mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, x; \mathsf{R}_{\mathsf{CH}})$ returns an image y on input x with randomness R_{CH}.*
- *The equivocation algorithm $\mathsf{CH.switch}(\mathsf{sk}_{\mathsf{CH}}, x, \mathsf{R}_{\mathsf{CH}}, x')$ outputs a new randomness $\mathsf{R}'_{\mathsf{CH}}$ such that $\mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, x; \mathsf{R}_{\mathsf{CH}}) = \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, x'; \mathsf{R}'_{\mathsf{CH}})$.*

The collision-resistance of chameleon hash means that it is difficult to output a collision $((x, \mathsf{R}_{\mathsf{CH}}), (x', \mathsf{R}'_{\mathsf{CH}}))$ for any adversary without the trapdoor $\mathsf{sk}_{\mathsf{CH}}$.

Definition 2 (Collision-resistance). *We say a family of chameleon hash functions CH is (t, ε)-collision-resistant (CR) if for all adversaries $\mathcal{A}$ that run in time t,*

$$\Pr[(x, \mathsf{R}_{\mathsf{CH}}) \neq (x', \mathsf{R}'_{\mathsf{CH}}) \wedge \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, x; \mathsf{R}_{\mathsf{CH}}) = \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, x'; \mathsf{R}'_{\mathsf{CH}})$$
$$\mid (\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}}) \xleftarrow{\$} \mathsf{CH.gen}(\lambda), (x, x', \mathsf{R}_{\mathsf{CH}}, \mathsf{R}'_{\mathsf{CH}}) \leftarrow \mathcal{A}(\mathsf{pk}_{\mathsf{CH}})] \leq \varepsilon.$$

2.2 Computational Assumptions

Let Ggen be a probabilistic polynomial time (PPT) algorithm that on input 1^λ returns a description $\mathcal{G} := (\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T, q, g_1, g_2, e)$ of asymmetric pairing groups, where G_1, G_2, G_T are cyclic groups of order q for a λ-bit prime q, g_1 and g_2 are generators of G_1 and G_2, respectively, and $e : \mathsf{G}_1 \times \mathsf{G}_2$ is an efficiently computable (non-degenerated) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in G_T. In this paper, we only consider Type III pairings, where $\mathsf{G}_1 \neq \mathsf{G}_2$ and there is no efficient homomorphism between them.

Firstly we will review the definition of SXDH assumption, which is actually the widely used DDH assumption in the source groups of the asymmetric pairing. We give the formal description of the assumption as DDHι for $\iota \in \{1, 2\}$.

Definition 3 (DDHι). *We say that the* First/Second Decision Diffie-Hellman *(DDH1/DDH2) assumption is (t, ε)-hard relative to Ggen in group G_ι if for all adversaries $\mathcal{A}$ with running time t, it holds that*

$$|\Pr[\mathcal{A}(\mathcal{G}, g_\iota^a, g_\iota^b, g_\iota^{ab}) = 1] - \Pr[\mathcal{A}(\mathcal{G}, g_\iota^a, g_\iota^b, g_\iota^z) = 1]| \leq \varepsilon,$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{Ggen}(1^\lambda)$ and $a, b, z \xleftarrow{\$} \mathbb{Z}_q$.

The above SXDH assumption is randomizable, which means that hardness of the Q-fold SXDH assumption is equal to that of the SXDH assumption [EHK+13].

In this paper we will also use weaker versions of 3-party Diffie-Hellman assumption. In the following we give definitions in two source groups separately, following that of [LQ19].

Definition 4 (wD3DHι [LQ19]). *For $\iota = 1, 2$, we say that the* Decision weak 3-Party Diffie-Hellman *(wD3DHι) assumption is (t, ε)-hard relative to* Ggen *in group* G_ι *if for all adversaries* $\mathcal{A}$ *with running time t, it holds that*

$$|\Pr[\mathcal{A}(\mathcal{G}, g_\iota^a, g_\iota^b, g_\iota^c, g_{3-\iota}^b, g_{3-\iota}^c, g_\iota^{abc}) = 1] - \Pr[\mathcal{A}(\mathcal{G}, g_\iota^a, g_\iota^b, g_\iota^c, g_{3-\iota}^b, g_{3-\iota}^c, g_\iota^z) = 1]| \leq \varepsilon,$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{Ggen}(1^\lambda)$ *and* $a, b, c, z \xleftarrow{\$} \mathbb{Z}_q$.

Definition 5 (2-3-CDH [LQ19,KP06]). *We say that the* 2-out-of-3 Computational Diffie-Hellman *(2-3-CDH) assumption is (t, ε)-hard relative to* Ggen *in group* G_2 *if for all adversaries* $\mathcal{A}$ *with running time t, it holds that*

$$\Pr[\mathcal{A}(\mathcal{G}, g_1^a, g_1^b, g_2^a, g_2^b) \Rightarrow (g_2^r, g_2^{r \cdot ab} \textit{ with } r \neq 0)] \leq \varepsilon,$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{Ggen}(1^\lambda)$ *and* $a, b \xleftarrow{\$} \mathbb{Z}_q$.

Next we review the randomized version of wD3DHι denoted as R-wD3DHι assumption, the hardness of which can be tightly reduced to the wD3DHι plus SXDH assumptions.

Definition 6 (R-wD3DHι [LQ19]). *For $\iota = 1, 2$, we say that the R-wD3DHι assumption is (t, ε)-hard relative to* Ggen *in group* G_ι *if for all adversaries* $\mathcal{A}$ *with running time t, it holds that*
$|\Pr[\mathcal{A}(\mathcal{G}, \{g_\iota^{a_i}\}_{i\in[Q]}, g_\iota^b, g_\iota^c, g_{3-\iota}^b, g_{3-\iota}^c, \{g_\iota^{a_i bc}\}_{i\in[Q]}) = 1] - \Pr[\mathcal{A}(\mathcal{G}, \{g_\iota^{a_i}\}_{i\in[Q]}, g_\iota^b,$ $g_\iota^c, g_{3-\iota}^b, g_{3-\iota}^c, \{g_\iota^{z_i}\}_{i\in[Q]}) = 1]| \leq \varepsilon,$
where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{Ggen}(1^\lambda)$ *and* $b, c, \{a_i, z_i\}_{i\in[Q]} \xleftarrow{\$} \mathbb{Z}_q$.

Lemma 3 (SXDH+wD3DHι⇒R-wD3DHι,[LQ19]). *For $\iota \in \{1, 2\}$, if the DDHι problem is (t_1, ε_1)-hard and the wD3DHι problem is (t_2, ε_2)-hard relative to* Ggen *in group* G_ι, *then the R-wD3DHι problem is $(t, \varepsilon_1 + \varepsilon_2)$-hard, where $t_1 \approx t_2 \approx t + \mathsf{poly}(\lambda)$.*

2.3 Lattice Background

Since we will use a lattice trapdoor to perform the opening algorithm to achieve the SIM-SO-CCA security, here we introduce some basic facts of lattices. Let $\mathbf{\Sigma} \in R^{n\times n}$ be a symmetric definite positive matrix, and $\mathbf{c} \in R^n$. We define the Gaussian function on R^n by $\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) = \exp(-\pi(\mathbf{x}-\mathbf{c})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c}))$ and if $\mathbf{\Sigma} = \sigma^2 \cdot \mathbf{I}_n$ and $\mathbf{c} = \mathbf{0}$ we denote it by ρ_σ. For an n-dimensional lattice Λ, we define the Gaussian distribution $D_{\Lambda,\sigma}$ on Λ as: $\Pr[\mathbf{x} = \mathbf{a}] = \frac{\rho_\sigma(\mathbf{a})}{\Sigma_{\mathbf{b}\in\Lambda}\rho_\sigma(\mathbf{b})}$. Gaussian distribution over the support $\Lambda + \mathbf{x}'$ with parameters $\mathbf{\Sigma}, \mathbf{c}$ is denoted as $D_{\Lambda+\mathbf{x}',\mathbf{\Sigma},\mathbf{c}} \sim \rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x})$ for all $\mathbf{x} \in \Lambda + \mathbf{x}'$. The smoothing parameter $\eta_\varepsilon(\Lambda)$ is defined as the smallest $r > 0$ such that $\rho_{1/r}(\widehat{\Lambda} \setminus \mathbf{0}) \leq \varepsilon$ with $\widehat{\Lambda}$ denoting the dual of Λ, for any $\varepsilon \in (0,1)$. In particular, we have $\eta_{2^{-n}}(\mathbb{Z}^n) \leq O(\sqrt{n})$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, we define $\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x}^\top \cdot \mathbf{A} = \mathbf{0} \bmod q\}$ and $\Lambda^\perp_{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x}^\top \cdot \mathbf{A} = \mathbf{u}^\top \bmod q\}$.

Lemma 4 (Poisson summation formula, Lemma 2.14 of [Reg05]). *For any lattice* $\mathbf{L}$ *and any function* $f : R^n \to C$*, there is*

$$f(\mathbf{L}) = \det(\mathbf{L}^*)\hat{f}(\mathbf{L}^*),$$

where $\hat{f}$ *is the Fourier transform of* f*.*

Lemma 5 (The min-entropy of $D_{\mathbb{Z}^n,\sigma}$). $H_\infty(D_{\mathbb{Z}^n,\sigma}) \geq n\log\sigma$.

Proof. $H_\infty(D_{\mathbb{Z}^n,\sigma}) = -\log\frac{\rho_\sigma(\mathbf{0})}{\rho_\sigma(\mathbb{Z}^n)}$. For Gaussian distribution, there is $\hat{\rho}_\sigma = \sigma^n \cdot \rho_{1/\sigma}$, then according to Lemma 4, $\rho_\sigma(\mathbb{Z}^n) = \det((\mathbb{Z}^n)^*) \cdot \sigma^n \rho_{1/\sigma}(\mathbb{Z}^{n*}) \geq \sigma^n$. Hence $H_\infty(D_{\mathbb{Z}^n,\sigma}) \geq n\log\sigma$. ☐

Lemma 6 (Adapted from [BLP+13]). *There exists a PPT algorithm that, given a basis* $(\mathbf{b}_i)_{i\leq n}$ *of a full-rank lattice* Λ*,* $\mathbf{x}', \mathbf{c} \in R^n$ *and* $\mathbf{\Sigma} \in R^{n\times n}$ *symmetric definite positive such that* $\Omega(\sqrt{\log n}) \cdot \max_i \|\mathbf{\Sigma}^{-1/2} \cdot \mathbf{b}_i\| \leq 1$*, returns a sample from* $D_{\Lambda+\mathbf{x}',\mathbf{\Sigma},\mathbf{c}}$*.*

Lemma 7 (Adapted from [MR04]). *For any* n*-dimensional lattice* Λ*,* $\mathbf{x}', \mathbf{c} \in R^n$ *and symmetric positive definite* $\mathbf{\Sigma} \in R^{n\times n}$ *satisfying* $\sigma_n(\sqrt{\mathbf{\Sigma}}) \geq \eta_{2^{-n}}(\Lambda)$*, we have* $\Pr_{\mathbf{x}\leftarrow D_{\Lambda+\mathbf{x}',\mathbf{\Sigma},\mathbf{c}}}[\|\mathbf{x}-\mathbf{c}\| \geq \sqrt{n}\cdot\|\sqrt{\mathbf{\Sigma}}\|] \leq 2^{-n+2}$*.*

In [MP12], Micciancio and Peikert described a trapdoor mechanism for Gaussian sampling. Their technique uses a "gadget" matrix $\mathbf{G} \in \mathbb{Z}_q^{m\times n}$ for which anyone can publicly sample short vectors $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{x}^\top\mathbf{G} = \mathbf{v}^\top$ for any $\mathbf{v}^\top$. As in [MP12], we call $\mathbf{R} \in \mathbb{Z}^{m\times m}$ a $\mathbf{G}$-trapdoor for a matrix $\mathbf{A} \in \mathbb{Z}_q^{2m\times n}$ if $[\mathbf{R} \mid \mathbf{I}_m] \cdot \mathbf{A} = \mathbf{G} \cdot \mathbf{H}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ which is referred to as the trapdoor tag. If $\mathbf{H} = \mathbf{0}$, then $\mathbf{R}$ is called a "punctured" trapdoor for $\mathbf{A}$.

Lemma 8 ([MP12, Section 5]). *Assume that* $m \geq 2n\log q$*. There exists a PPT algorithm* GenTrap *that takes as inputs matrices* $\bar{\mathbf{A}} \in \mathbb{Z}_q^{m\times n}$*,* $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ *and outputs matrices* $\mathbf{R} \in \{-1,1\}^{m\times m}$ *and*

$$\mathbf{A} = \begin{bmatrix} \bar{\mathbf{A}} \\ -\mathbf{R}\bar{\mathbf{A}} + \mathbf{G}\mathbf{H} \end{bmatrix} \in \mathbb{Z}_q^{2m\times n}$$

such that if $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ *is invertible, then* $\mathbf{R}$ *is a* $\mathbf{G}$*-trapdoor for* $\mathbf{A}$ *with tag* $\mathbf{H}$*; and if* $\mathbf{H} = \mathbf{0}$*, then* $\mathbf{R}$ *is a punctured trapdoor.*

Further, in case of a $\mathbf{G}$*-trapdoor, one can efficiently compute from* $\mathbf{A}, \mathbf{R}$ *and* $\mathbf{H}$ *a basis* $(\mathbf{b}_i)_{i\leq 2m}$ *of* $\Lambda^\perp(\mathbf{A})$ *such that* $\max_i \|\mathbf{b}_i\| \leq O(m^{3/2})$*.*

Micciancio and Peikert also showed that a $\mathbf{G}$-trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{2m\times n}$ can be used to output a Gaussian distribution $\mathbf{x}$ such that $\mathbf{x}^\top\mathbf{A} = \mathbf{u}^\top$ for any $\mathbf{u}$.

Lemma 9 ([MP12, Theorem 5.5]). *There exists a PPT algorithm* Gausam *that takes as inputs matrices* $\mathbf{R} \in \mathbb{Z}^{m \times m}$, $\mathbf{A} \in \mathbb{Z}_q^{2m \times n}$, $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ *such that* $\mathbf{R}$ *is a* $\mathbf{G}$*-trapdoor for* $\mathbf{A}$ *with invertible tag* $\mathbf{H}$*, and a vector* $\mathbf{u}$*, then outputs* $\mathbf{x} \leftarrow D_{\Lambda_q^{\mathbf{u}},\sigma}$ *that is statistically close to* $(\mathbf{x} \leftarrow D_{\mathbb{Z}^m,\sigma} | \mathbf{x}^\top \mathbf{A} = \mathbf{u}^\top)$ *with* $\sigma > \eta_\epsilon(\Lambda(\mathbf{G}^\perp))$.

2.4 All-but-many Lossy Trapdoor Functions

In the following we will review the definitions of lossy trapdoor functions (LTF) [PW08] and all-but-many lossy trapdoor functions ($\mathsf{ABM\text{-}LTF}$) [Hof12]. Here we will use the adapted definitions given by Libert *et al.* [LSSS17a], in which the input is not necessarily chosen in a uniform way, but according to proper distributions, and the input domain could be a bit larger than the output domain of the inversion algorithm.

Definition 7 (LTF). *A family of lossy trapdoor functions* (LTF) *has four algorithms* $\mathsf{LTF} := (\mathsf{LTF.Igen}, \mathsf{LTF.Lgen}, \mathsf{LTF.Eval}, \mathsf{LTF.Invert})$ *with the following properties:*

- *The injective key generation algorithm* $\mathsf{LTF.Igen}(\lambda)$ *returns the evaluation/inversion key* $(\mathsf{ek}, \mathsf{ik})$*. We assume that* ek *implicitly defines the distribution* D_x *on input domain* DomE*, and the inversion domain* $\mathsf{DomD} \subset \mathsf{DomE}$*.*
- *The lossy key generation algorithm* $\mathsf{LTF.Lgen}(\lambda)$ *returns the evaluation key* ek*.*
- *The evaluation algorithm* $\mathsf{LTF.Eval}(\mathsf{ek}, x)$ *returns an image* y*.*
- *The deterministic inversion algorithm* $\mathsf{LTF.Invert}(\mathsf{ik}, y)$ *returns an* $x \in \mathsf{DomD}$ *or abort symbol* $\perp$*.*

Security requirements.

Inversion correctness. *For an injective key pair* $(\mathsf{ek}, \mathsf{ik}) \xleftarrow{\$} \mathsf{LTF.Igen}(\lambda)$*, we have, except with negligible probability over* $(\mathsf{ek}, \mathsf{ik})$*, that for all inputs* $x \in \mathsf{DomD}$*,* $x = \mathsf{LTF.Invert}(\mathsf{ik}, \mathsf{LTF.Eval}(\mathsf{ek}, x))$*.*

Sampling correctness. *For* $x \leftarrow D_x$*, we have* $x \in \mathsf{DomD}$ *except with negligible probability.*

ℓ-Lossiness. *We say an* LTF *is with* ℓ*-lossiness, if for all* $\mathsf{ek} \leftarrow \mathsf{LTF.Lgen}(\lambda)$ *and* $x \leftarrow D_x$*, it holds that* $H_\infty(x | (\mathsf{ek}, \mathsf{LTF.Eval}(\mathsf{ek}, x) = y)) \geq \ell$ *except for a negligible probability.*

(t, ε)-Indistinguishability. *An* LTF *is* (t, ε)*-indistinguishable, if for any adversary* $\mathcal{A}$ *that runs in time* t*,*

$$|\Pr[\mathcal{A}(\mathsf{ek}) = 1 | (\mathsf{ek}, \mathsf{ik}) \leftarrow \mathsf{LTF.Igen}(\lambda)] - \Pr[\mathcal{A}(\mathsf{ek}) = 1 | \mathsf{ek} \leftarrow \mathsf{LTF.Lgen}(\lambda)]| \leq \varepsilon.$$

In an $\mathsf{ABM\text{-}LTF}$, functions are computed with the evaluation key and an associated tag. There are two kinds of indistinguishable tags: lossy and injective tags, which lead to lossy and injective functions respectively. For injective tags, it proceeds the same way as trapdoor one way function. For lossy tags, the range size of the function is strictly smaller than the domain size. Lossy tags can be sampled with a special trapdoor, but for adversaries who do not have the trapdoor, it is hard to generate fresh lossy tags, even given access to polynomially many lossy tags. This property is called the evasiveness property. The formal definition is as follows.

Definition 8 (ABM-LTF). *A family of all-but-many lossy trapdoor functions* (ABM-LTF) *has four algorithms* (ABM.Gen, ABM.LGen, ABM.Eval, ABM.Inver) *with the following properties:*

- *The key generation algorithm* $\mathsf{ABM.Gen}(\lambda)$ *returns the evaluation, inversion and trapdoor keys* $(\mathsf{ek}, \mathsf{ik}, \mathsf{tk})$. *We assume that* ek *implicitly defines the distribution* D_x *on input domain* DomE, *and the inversion domain* $\mathsf{DomD} \subset \mathsf{DomE}$. *We also assume that* ek *defines the tag space* $\mathcal{T} := \mathcal{T}_a \times \mathcal{T}_c$ *containing the disjoint sets of lossy tags* $\mathcal{T}_{\mathsf{loss}}$ *and* $\mathcal{T}_{\mathsf{inj}}$, *here* $\mathcal{T}_a$ *denotes the auxiliary tag space and* $\mathcal{T}_c$ *denotes the core tag space*[5].
- *The lossy tag generation algorithm* $\mathsf{ABM.LGen}(\mathsf{tk}, t_a)$ *returns the core tag part* t_c *such that* $t := (t_a, t_c) \in \mathcal{T}_{\mathsf{loss}}$.
- *The evaluation algorithm* $\mathsf{ABM.Eval}(\mathsf{ek}, t, x)$ *returns an image* y *with respect to a tag* t *and input* x.
- *The deterministic inversion algorithm* $\mathsf{ABM.Inver}(\mathsf{ik}, t, y)$ *returns an* $x \in \mathsf{DomD}$ *or an abort symbol* $\perp$.

Security requirements.

Inversion correctness. *We require that for all pairs* $(\mathsf{ek}, \mathsf{ik}, \mathsf{tk}) \xleftarrow{\$} \mathsf{ABM.Gen}(\lambda)$, *for all* $t \in \mathcal{T}_{\mathsf{inj}}$, *for all* $x \in \mathsf{DomD}$, $\Pr[\mathsf{ABM.Inver}(\mathsf{ik}, t, \mathsf{ABM.Eval}(\mathsf{ek}, t, x)) = x] \geq 1 - \varepsilon$, *where* ε *is negligible in* λ.

Sampling correctness. *For* $x \leftarrow D_x$, *we have* $x \in \mathsf{DomD}$ *except with negligible probability.*

Explainable tags. *We say an* ABM-LTF *is with explainable tags, if there exists a PPT algorithm* Resam, *such that for any* $t_c \in \mathcal{T}_c$, $\mathsf{Resam}(t_c) \to \delta$, *such that* δ *is uniformly distributed in the set* $\{\delta' : \mathsf{Samp}(\delta') = t_c\}$, *where* Samp *is a PPT algorithm to sample uniformly random elements in* $\mathcal{T}_c$, *and* δ' *denotes the randomness used by the* Samp *algorithm. In general, this means that any core tag part can be explained as a randomly chosen tag by showing the sampling randomness.*

ℓ-Lossiness. *We say an* ABM-LTF *is with* ℓ*-lossiness, if for all* $(\mathsf{ek}, \mathsf{ik}, \mathsf{tk}) \leftarrow \mathsf{ABM.Gen}$, *all* $t \in \mathcal{T}_{\mathsf{loss}}$ *and* $x \leftarrow D_x$, *it holds that* $H_\infty(x|(\mathsf{ek}, \mathsf{ABM.Eval}(\mathsf{ek}, t, x) = y)) \geq \ell$ *except for negligible probability.*

$(q_{\mathsf{in}}, t, \varepsilon)$-Indistinguishability. *An* ABM-LTF *is* $(q_{\mathsf{in}}, t, \varepsilon)$*-indistinguishable, if for any adversary* $\mathcal{A}$ *that runs in time* t,

$$\left|\Pr[\mathcal{A}(\mathsf{ek})^{\textsc{Loss}(\cdot)} = 1] - \Pr[\mathcal{A}(\mathsf{ek})^{\mathrm{U}(\cdot)} = 1]\right| \leq \varepsilon,$$

where on input t_a, Loss *returns a* $t_c \leftarrow \mathsf{ABM.LGen}(\mathsf{tk}, t_a)$ *and* U *returns a random* $t_c \xleftarrow{\$} \mathcal{T}_c$, *the adversary may make at most* q_{in} *queries.*

$(q_{\mathsf{eva}}, q_{\mathsf{ver}}, t, \varepsilon)$-Evasiveness. *An* ABM-LTF *is* $(q_{\mathsf{eva}}, q_{\mathsf{ver}}, t, \varepsilon)$*-evasiveness, if for any adversary* $\mathcal{A}$ *that runs in time* t,

$$\Pr[\mathcal{A}(\mathsf{ek})^{\textsc{Loss}(\cdot),\textsc{Ver}(\cdot)} \Rightarrow (t_a^*, t_c^*) \in \mathcal{T} \backslash \mathcal{T}_{\mathsf{inj}}] \leq \varepsilon,$$

[5] Here note that $\mathcal{T} \supset \mathcal{T}_{\mathsf{loss}} \cup \mathcal{T}_{\mathsf{inj}}$, there may exist a tag $t \in \mathcal{T}$ but $t \notin \mathcal{T}_{\mathsf{loss}} \cup \mathcal{T}_{\mathsf{inj}}$.

where on input (t_a, t_c), Ver *returns 1 iff $(t_a, t_c) \in \mathcal{T} \backslash \mathcal{T}_{\mathsf{inj}}$, and we make the trivial restriction that (t_a^*, t_c^*) is different from that returned by* Loss*, the adversary may make at most q_{eva} queries to the* Loss *oracle and at most q_{ver} queries to the* Ver *oracle.*

2.5 Selective Opening Security

In the following we give the formal definition of simulation-based selective opening and chosen-ciphertext attacks (SIM-SO-CCA) for PKE. In general, it requires that the view of an adversary that accesses to the decryption oracle, sees the challenge ciphertexts and can adaptively open some of them, can be simulated by a simulator that only gets the opened messages. The formal definition is as follows.

Definition 9 (SIM-SO-CCA Security [BHY09]). *A PKE is $(q_{\mathsf{dec}}, N, t, t', \epsilon)$-SO-CCA-secure if for all N-message sampler* dist *that takes $\alpha \in \{0,1\}^*$ as input and returns $\mathbf{m} := (\mathsf{m}_1, \ldots, \mathsf{m}_N)$, all randomized relation $\mathcal{R}$, for any $\mathcal{A}$ in* SOCCAreal *that runs in time t, makes at most q_{dec} decryption queries, there exists a simulator $\mathcal{S}$ in* SOCCArand *that runs in time t', such that*

$$|\Pr[\mathsf{SOCCAreal}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{SOCCArand}^{\mathcal{S}} \Rightarrow 1]| \leq \epsilon,$$

where the security games are defined as in Figures 1 and 2, and in both games the adversary must make one Chal *query before one* Curr *query.*

<u>Init:</u>	<u>Chal(α):</u> // one query	<u>Curr(I):</u> // one query
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Keygen}(\lambda)$	$\mathbf{m} \xleftarrow{\$} \mathsf{dist}(\alpha)$	Return $(\mathbf{m}_I, \mathfrak{r}_I)$
Return pk	For $i \in [N]$:	
	$\mathfrak{r}_i \xleftarrow{\$} \mathcal{R}$,	<u>Finalize(OUT):</u>
<u>Dec(c):</u> // at most q_{dec} queries	$\mathsf{c}_i := \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_i; \mathfrak{r}_i)$	Return $\mathcal{R}(\mathbf{m}, I, \mathsf{OUT})$.
If $\mathsf{c} \in \mathcal{C}_{\mathsf{enc}}$, return $\perp$	$\mathfrak{ct} := (\mathsf{c}_1, \ldots, \mathsf{c}_N)$	
$\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{c})$;	$\mathcal{C}_{\mathsf{enc}} := \{\mathsf{c}_1, \ldots, \mathsf{c}_N\}$	
Return m	Return $\mathfrak{ct}$	

Figure 1. Security game SOCCAreal

<u>Init$_{\mathcal{S}}$:</u>	<u>Chal$_{\mathcal{S}}$(α):</u> // one query	<u>Curr$_{\mathcal{S}}$(I):</u> // one query
Return ϵ	$\mathbf{m} \xleftarrow{\$} \mathsf{dist}(\alpha)$	Return $\mathbf{m}_I$
	Return ϵ	
		<u>Finalize(OUT):</u>
		Return $\mathcal{R}(\mathbf{m}, I, \mathsf{OUT})$.

Figure 2. Security game SOCCArand

3 Constructions of ABM-LTF with Linear-Size Tags

In this section we will give two constructions of ABM-LTF with linear-size tags. Our constructions are inspired by that of lossy algebraic filter (LAF) and ABM-LTF given in [LQ19]. They employed the revocation technique [LSW10,BW10] to compute n^2 elements from $4n$ elements, hence shrinking the tag length. However, there is a special and publicly recognizable structure in their tags. With such structure, lossy tags cannot be explained as random tags, hence incompatible with the selective opening requirement. To eliminate the structure in the tags for ABM-LTF, here we put all off-diagonal elements in the public key. Although this results in larger public key size compared with that of [LQ19], in this way the lossy tags are pseudorandom and can be explained to random tags. Then by combining with the Waters signatures and the tightly secure MAC in [LQ19] respectively, we get two ABM-LTFs with explainable tags, hence compatible with the SO-CCA scenario. And the second one is with (almost) tight indistinguishability and evasiveness.

Furthermore, we adapt the evaluation algorithm of the ABM-LTFs, and make the function output be efficiently openable, so that it can be used to build SIM-SO-CCA secure PKE. To do this, we firstly make a change on the evaluation algorithm, then we set the function input to be small-norm integer vectors from a discrete Gaussian distribution, then with the help of a lattice trapdoor, we can open the lossy function value to any input.

In the following subsections, we denote $\mathsf{DomD} := \{\mathbf{x} \in \mathbb{Z}^n | \|\mathbf{x}\| \leq \sigma\sqrt{n}\}$ and $\mathsf{DomE} := \{\mathbf{x} \in \mathbb{Z}^n | \|\mathbf{x}\| \leq \gamma \cdot \sigma\sqrt{n}\}$ with $\sigma \geq \Omega(n)$ and $\gamma \geq 3$, where DomD is the inversion domain and DomE is the input domain.

3.1 An ABM-LTF based on Waters Signatures

In this subsection we firstly give an ABM-LTF based on Waters signatures, and then adapt to a new ABM-LTF, such that the output for lossy tags is efficiently openable. These two ABM-LTFs only disagree in the evaluation and inversion algorithms, while share same parameters and tags, so we show correctness and lossiness separately, and prove indistinguishability and evasiveness for once.

ABM.Gen: Choose bilinear groups $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T$ of prime order q with $e : \mathsf{G}_1 \times \mathsf{G}_2 \to \mathsf{G}_T$. Choose random elements $g_1 \xleftarrow{\$} \mathsf{G}_1, g_2 \xleftarrow{\$} \mathsf{G}_2$ and denote $g_T := e(g_1, g_2)$.

1. Choose a chameleon hash function $\mathsf{CH} := (\mathsf{CH.gen}, \mathsf{CH.eval}, \mathsf{CH.switch})$ with $\mathsf{CH.eval} : \{0,1\}^* \times \mathcal{R}_{\mathsf{CH}} \to \{0,1\}^L$. Generate $(\mathsf{pk_{CH}}, \mathsf{sk_{CH}}) \leftarrow \mathsf{CH.gen}$.
2. Pick random $r_i, s_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [n]$ and $w_0, \ldots, w_L \xleftarrow{\$} \mathbb{Z}_q$, where n is the input length, compute $R_i := g_1^{r_i},\ S_i := g_1^{s_i},\ U_{ij} := g_1^{r_i s_j}$ for $i \neq j \in [n]$ and $W_{\iota,k} := g_\iota^{w_k}$ for $\iota = 1, 2, k \in [0, L]$, for any $\mathsf{m} \in \{0,1\}^L$, we define $H_{\mathsf{G}_\iota}(\mathsf{m}) := g_\iota^{w_0 + \sum_{k=1}^{L} w_k \cdot \mathsf{m}_k}$ for $\iota = 1, 2,$.
3. $\mathsf{ek} := (\mathsf{pk_{CH}}, \{W_{\iota,k}\}_{\iota \in [2], k \in [0,L]}, \{R_i, S_i\}_{i \in [n]}, \{U_{ij}\}_{i \neq j \in [n]})$.
   $\mathsf{ik} := (\mathsf{ek}, \{s_i\}_{i \in [n]})$, $\mathsf{tk} := (\mathsf{sk_{CH}}, \{r_i, s_i\}_{i \in [n]}, \{w_k\}_{k \in [0,L]})$.

The tag space is defined as follows: tags are of the form $t = (t_a, t_c)$, where $t_a \in \{0,1\}^*$ is the auxiliary part, and $t_c := (\{B_i, D_i, E_i\}_{i\in[n]}, \mathsf{R}_{\mathsf{CH}}) \in \mathbb{G}_2^{3n} \times \mathcal{R}_{\mathsf{CH}}$ is the core tag part. Random t_c are uniformly random elements in the core tag space. The input distribution D_x is the uniform distribution over $(\{0,1\}^{\log\lambda})^n$.

$\mathsf{ABM.LGen}(\mathsf{tk}, t_a)$**:** The lossy core tag part $t_c := (\{B_i, D_i, E_i\}_{i\in[n]}, \mathsf{R}_{\mathsf{CH}})$ is computed as:

1. For $i \in [n]$, pick $b_i, \rho_i \xleftarrow{\$} \mathbb{Z}_q$, compute $B_i := g_2^{b_i}$, $D_i := g_2^{r_i b_i s_i} H_{\mathsf{G}_2}(\tau)^{\rho_i}$ and $E_i := g_2^{\rho_i}$, where $\tau := \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, (t_a, \{B'_i, D'_i, E'_i\}_{i\in[n]}); \mathsf{R}'_{\mathsf{CH}}) \in \{0,1\}^L$, where $B'_i, D'_i, E'_i, \mathsf{R}'_{\mathsf{CH}}$ are randomly chosen.
2. Compute $\mathsf{R}_{\mathsf{CH}} := \mathsf{CH.switch}(\mathsf{sk}_{\mathsf{CH}}, (t_a, \{B'_i, D'_i, E'_i\}_{i\in[n]}), \mathsf{R}'_{\mathsf{CH}}, (t_a, \{B_i, D_i, E_i\}_{i\in[n]}))$.

Each tag is corresponding to a matrix $(\mathbf{M}_{ij})$ with

$$\mathbf{M}_{ij} := \begin{cases} e(U_{ij}, B_i) = g_T^{r_i s_j b_i} & \text{if} \quad i \neq j \\ \frac{e(g_1, D_i)}{e(H_{\mathsf{G}_1}(\tau), E_i)} & \text{else} \end{cases} \tag{1}$$

We say a tag $t = (t_a, t_c)$ is lossy if $\mathbf{M}_{ii} = g_T^{r_i s_i b_i}$ for each $i \in [n]$. A tag is injective if $\mathbf{M}_{ii} \neq g_T^{r_i s_i b_i}$ for all $i \in [n]$. Note that there exist tags that are neither lossy nor injective.

$\mathsf{ABM.Eval}(\mathsf{ek}, t, \mathbf{x})$**:** For input $\mathbf{x} := (x_1, \ldots, x_n)$ with $x_i \in \{0,1\}^{\log\lambda}$, output $\mathbf{y} := (y_0, \ldots, y_n)$ as:

1. Compute $\mathbf{M}$ according to Equation (1).
2. Compute $y_0 := \prod_{j\in[n]} e(R_j, B_j)^{x_j} = g_T^{\sum_{j\in[n]} r_j b_j x_j}$ and $y_i := \prod_{j\in[n]} \mathbf{M}_{ji}^{x_j}$.

$\mathsf{ABM.Inver}(\mathsf{ik}, \mathbf{y})$**:** Compute $\mathbf{x}$ as:

1. Compute $M'_i := \frac{\mathbf{M}_{ii}}{g_T^{r_i b_i s_i}}$ for each $i \in [n]$. If there exists $i \in [n]$ such that $M'_i = g_T^0$, return $\perp$.
2. Compute $Z_i := y_i / y_0^{s_i}$ for each $i \in [n]$.
3. Search $x_i \in \{0,1\}^{\log\lambda}$ such that $M_i'^{x_i} = Z_i$.

Correctness. When the tag $t = (t_a, t_c)$ is injective, we have $M'_i \neq 1_T$ for each $i \in [n]$. Then

$$\begin{aligned} Z_i = y_i / y_0^{s_i} &= \frac{\prod_{j\in[n]} \mathbf{M}_{ji}^{x_j}}{g_T^{s_i \sum_{j\in[n]} r_j b_j x_j}} \\ &= \frac{\prod_{j\neq i} g_T^{r_j x_j s_i b_j} \cdot \mathbf{M}_{ii}^{x_i}}{g_T^{s_i \sum_{j\in[n]} r_j b_j x_j}} = M_i'^{x_i}, \end{aligned}$$

hence x_i can be recovered correctly.

Lossiness. For the lossy tag, $y_i = g_T^{s_i(\sum_j r_j b_j x_j)}$, the output is completely determined by $\sum_j r_j b_j x_j \bmod q$, so that the function has image size no larger than $\log q$, hence the lossiness $\ell = n\log\lambda - \log q$.

Explainable tags. Our construction has explainable tags as soon as the employed group G_2 is efficiently samplable and explainable.

In comparison with the construction given in [LQ19], the above construction has larger evaluation keys of size $O(n^2)$. On the other hand, its lossy tags are pseudorandom and thus make the ABM-LTF amenable to the design of a PKE scheme with IND-SO-CCA security. Next, we will give a variant of the above construction which can be used as a building block to achieve stronger and more natural simulation-based (SIM-SO-CCA) security. The key generation and the lossy tag generation algorithms remain exactly identical. However, the tag related matrix $\mathbf{M}$ is defined differently and the evaluation and inversion algorithms also need some modifications. The main observation is that, given $\delta = \sum_j (r_j b_j x_j)$, we don't know how to find $\mathbf{x}'$ such that $\delta = \sum_j (r_j b_j x'_j)$. On the other hand, if we change the linear combination of $\mathbf{x}$ independent of tags, say $\delta = \sum_j (r_j x_j)$, and embedding a lattice trapdoor when generating $\mathbf{r}$, we can efficiently sample an $\mathbf{x}'$. The concrete description is as below.

The input distribution is modified as: D_x is the Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$ with the restriction that the sampled $\mathbf{x}$ satisfies $\mathbf{x} \in \mathsf{DomE}$.

The computation of tag related matrix $(\mathbf{M}_{ij})$ is modified to:

$$\mathbf{M}_{ij} := \begin{cases} e(U_{ji}, B_i) = g_T^{r_j s_i b_i} & \text{if} \quad i \neq j \\ \frac{e(g_1, D_i)}{e(H_{\mathsf{G}_1}(\tau), E_i)} & \text{else} \end{cases} \tag{2}$$

We say a tag $t = (t_a, t_c)$ is lossy if $\mathbf{M}_{ii} = g_T^{r_i s_i b_i}$ for each $i \in [n]$. And a tag is injective if $\mathbf{M}_{ii} \neq g_T^{r_i s_i b_i}$ for all $i \in [n]$. Note that there exist tags that are neither lossy nor injective.

$\mathsf{ABM.Eval}(\mathsf{ek}, t, \mathbf{x})$**:** For input $\mathbf{x} := (x_1, \ldots, x_n) \in \mathsf{DomE}$, output $\mathbf{y} := (y_{0,1}, y_{1,1}, \ldots, y_{0,n}, y_{1,n})$ as:

1. Compute $\mathbf{M}$ according to Equation (2).
2. Compute $y_{0,i} := \prod_{j \in [n]} e(R_j, B_i)^{x_j} = g_T^{b_i \sum_{j \in [n]} r_j x_j}$ and $y_{1,i} := \prod_{j \in [n]} \mathbf{M}_{ij}^{x_j}$.

$\mathsf{ABM.Inver}(\mathsf{ik}, \mathbf{y})$**:** Compute x_i as:

1. Compute $M'_i := \frac{\mathbf{M}_{ii}}{g_T^{r_i s_i b_i}}$ for each $i \in [n]$. If there exists $i \in [n]$ such that $M'_i = g_T^0$, return $\perp$.
2. Compute $Z_i := y_{1,i} / y_{0,i}^{s_i}$ for each $i \in [n]$.
3. Search $\mathbf{x} \in \mathsf{DomD}$ such that $M_i'^{x_i} = Z_i$ for all $i \in [n]$.

Decryption correctness. When the tag $t = (t_a, t_c)$ is injective, we have $M'_i \neq 1_T$ for each $i \in [n]$. Then

$$\begin{aligned} Z_i = y_{1,i} / y_{0,i}^{s_i} &= \frac{\prod_{j \in [n]} \mathbf{M}_{ij}^{x_j}}{g_T^{s_i b_i \sum_{j \in [n]} r_j x_j}} \\ &= \frac{\prod_{j \neq i} g_T^{r_j x_j s_i b_i} \cdot \mathbf{M}_{ii}^{x_i}}{g_T^{s_i \sum_{j \in [n]} r_j b_i x_j}} = M_i'^{x_i}, \end{aligned}$$

hence x_i can be recovered correctly.
Sampling correctness. According to Lemmas 6 and 7, if $\sigma \geq \Omega(\sqrt{n})$, the distribution $D_{\mathbb{Z}^n,\sigma}$ is efficiently samplable, and $\Pr[\mathbf{x} \in \mathsf{DomD} | \mathbf{x} \leftarrow D_{\mathbb{Z}^n,\sigma}] \geq 1 - 2^{-\Omega(\lambda)}$.
Lossiness. For the lossy tag, $y_{1,i} = g_T^{s_i b_i (\sum_j r_j x_j)}$, the output is completely determined by $\sum_j r_j x_j \bmod q$, so that the function has image size no larger than q. Then applying the following lemma in [DRS04], lossiness is proved.

Lemma 10 ([DRS04]). *Let x, y be two variables that y has at most s possible values, then $H_\infty(x|y) \geq H_\infty(x) - \log s$.*

Since a vector $\mathbf{x}$ sampled from the distribution $D_{\mathbb{Z}^n,\sigma}$ has at least $n \log \sigma$ bits of min-entropy according to Lemma 5, we have lossiness $\ell := H_\infty(\mathbf{x}|(\mathbf{y}_0, \mathbf{y}_1)) \geq n \log \sigma - \log q$.

Security. For the above two constructions, although the evaluation and inversion algorithms are different, the condition for lossy and injective tags is the same. So we give the indistinguishable and evasiveness proof once.

As in [LQ19], we prove the indistinguishability based on the R-wD3DH2 assumption. One difference is that in the first construction of [LQ19], they give the proof based on the non-randomized version of wD3DH2 assumption and prove via a sequence of $q_{\mathsf{in}} \cdot n$ games, here we employ the randomized version assumption, hence only use n game hops. Note that R-wD3DH2 assumption can be tightly reduced to the wD3DH2 plus CDH assumptions in type 3 asymmetric pairing groups. And for symmetric pairing groups, there exists a security proof reduced to the wD3DH2 assumption with security loss the same as that of [LQ19].
Indistinguishability. We prove the indistinguishable property in n steps. At step k, we modify D_k to be randomly distributed for all queries, so that all tags are randomly distributed in the final game. We employ the R-wD3DH2 assumption to assure that the modification is undetectable. To do this, the reduction embeds the challenge in R_k, S_k and (B_k, D_k)s for every query at step k.

Theorem 1 (Indistinguishability). *If the R-wD3DH2 problem is (t_1, ε_1)-hard, then the above* $\mathsf{ABM\text{-}LTF}$ *is $(q_{\mathsf{in}}, t_{\mathcal{A}}, \varepsilon)$-indistinguishable, where $t_1 \approx t_{\mathcal{A}} + \mathsf{poly}(\lambda)$, and $\varepsilon \leq n \cdot \varepsilon_1$.*

Proof. Note that for any tag $t = (t_a, t_c)$, the core tag part $t_c = (\{B_i, D_i, E_i\}_{i\in[n]}, \mathsf{R}_{\mathsf{CH}})$. If $D_i = g_2^{r_i b_i s_i} H_{\mathsf{G}_2}(\tau)^{\rho_i}$ for all $i \in [n]$, then t is a lossy tag; and if D_i is randomly distributed for all $i \in [n]$, then t is randomly distributed. We prove the indistinguishablity via a sequence of n games. In the initial game of G_0, the adversary has access to the real lossy tag oracle $\textsc{Loss}(\mathsf{tk}, \cdot)$. In the final game, the adversary has access to an oracle $O_{T_c}(\cdot)$ that always returns random tags. Concretely,

$\mathsf{G}_k (k \in [n])$**:** In this game, the answer to the ξth query is set as follows: the first k D_i are uniformly random and the last $n - k$ $D_i = g_2^{r_i b_i s_i} H_{\mathsf{G}_2}(\tau)^{\rho_i}$ are set as the lossy oracle answer. Next we prove that difference between adjacent games is bounded by the R-wD3DH2 assumption.

Lemma 11. *We assume that an adversary runs in time $t_{\mathcal{A}}$ has advantage ε_k in G_k. Then if the R-wD3DH2 problem is (t_1, ε_1)-hard, then $|\varepsilon_k - \varepsilon_{k-1}| \leq \varepsilon_1$ for $k \geq 1$, and $t_1 \approx t_{\mathcal{A}} + \mathsf{poly}(\lambda)$.*

Proof. On receiving an R-wD3DH2 challenge $(g_1, g_2, g_1^b, g_1^c, \{g_2^{a_\xi}\}_\xi, g_2^b, g_2^c, \{T_\xi\}_\xi)$ for $\xi = 1, \ldots, q_{\mathsf{in}}$, the reduction $\mathcal{B}$'s task is to decide whether $T_\xi = g_2^{a_\xi bc}$ or T_ξ is randomly chosen from G_2. To do this, $\mathcal{B}$ proceeds as follows:

To generate ek, $\mathcal{B}$ firstly generates $(\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}}) \leftarrow \mathsf{CH.gen}$ as in the real game, then it picks $r_i, s_i \xleftarrow{\$} \mathbb{Z}_q$ and computes $R_i := g_1^{r_i},\ S_i := g_1^{s_i}$ for $i \in [n] \setminus k$, it also sets $R_k := g_1^b$, $S_k := g_1^c$ and $U_{ik} := g_1^{r_i c}$, $U_{ki} := g_1^{bs_i}$, in this way it implicitly sets $r_k := b$ and $s_k := c$. Then it picks $w_0, ..., w_L \xleftarrow{\$} \mathbb{Z}_q$, computes $W_{\iota,i} := g_\iota^{w_i}$ for $\iota = 1, 2, i \in [0, L]$. It is obvious that ek is distributed exactly as in the real game.

When the adversary proposes the ξth query $t_a^{(\xi)}$ to the Loss, the core tag part $t_c^{(\xi)}$ is answered as follows: $\mathcal{B}$ firstly computes a τ with random input. Then for $i < k$, (B_i, D_i, E_i) are randomly picked. And

$$B_k := g_2^{a_\xi},\ D_k := T_\xi \cdot H_{\mathsf{G}_2}(\tau)^{\rho_k},\ E_k := g_2^{\rho_k}.$$

with $\rho_k \xleftarrow{\$} \mathbb{Z}_q$.

For $i > k$, $\mathcal{B}$ picks $b_i,\ \rho_i \xleftarrow{\$} \mathbb{Z}_q$ and sets

$$B_i := g_2^{b_i},\ D_i := g_2^{r_i s_i b_i} \cdot H_{\mathsf{G}_2}(\tau)^{\rho_i},\ E_i := g_2^{\rho_i}.$$

Finally, $\mathcal{B}$ uses the trapdoor $\mathsf{sk}_{\mathsf{CH}}$ for the chameleon hash to find coins R_{CH} such that $\tau = \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, t_a, \{B_i, D_i, E_i\}_{i=1}^n, \mathsf{R}_{\mathsf{CH}})$.

It is obvious that, if $T_\xi = g_2^{a_\xi bc}$, then t_c is distributed as in G_{k-1}; and if T_ξ is randomly picked, then t_c is distributed as in G_k. $\square$

$\square$

Evasiveness. To prove evasiveness, we proceed in two steps: firstly we modify the verification oracle Ver to reject tags with non-fresh τ^*, then we use a similar proof as Waters signatures to bound the success probability of breaking evasiveness. To reject the non-fresh τ^*, we should firstly remove the use of $\mathsf{sk}_{\mathsf{CH}}$ for the chameleon hash, then use the collision-resistant property to reject the non-fresh τ^*, which is referred to as a 'deferred analysis' technique [Hof12].

Theorem 2 (Evasiveness). *If the* CH *is (t_1, ε_1)-collision-resistant; the R-wD3DH2 problem is (t_2, ε_2)-hard and the 2-3-CDH problem is (t_3, ε_3)-hard, then the above* $\mathsf{ABM\text{-}LTF}$ *is $(q_{\mathsf{eva}}, q_{\mathsf{ver}}, t_{\mathcal{A}}, \varepsilon)$-evasiveness, where $t_1 \approx t_2 \approx t_3 \approx t_{\mathcal{A}} + \mathsf{poly}(\lambda)$, and $\varepsilon \leq \varepsilon_1 + n\varepsilon_2 + O(n \cdot q_{\mathsf{eva}} \cdot q_{\mathsf{ver}} \cdot \sqrt{L})\varepsilon_3$.*

The proof is similar to that in [LQ19] and we defer it to supporting material Appendix A.

3.2 A Tightly Secure ABM-LTF Scheme

In this section, we replace the Waters Signatures with the tightly secure MAC given in [LQ19], then get an ABM-LTF with tight evasiveness. We give a review of the underlying MAC in supporting material B.

$\mathsf{ABM.Gen}$**:** Choose bilinear groups $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T$ of prime order q with asymmetric pairing $e : \mathsf{G}_1 \times \mathsf{G}_2 \to \mathsf{G}_T$. Choose random elements $g_1 \xleftarrow{\$} \mathsf{G}_1, g_2 \xleftarrow{\$} \mathsf{G}_2$ and denote $g_T := e(g_1, g_2)$.

1. Choose a chameleon hash function $\mathsf{CH} := (\mathsf{CH.gen}, \mathsf{CH.eval}, \mathsf{CH.switch})$ with $\mathsf{CH.eval} : \{0,1\}^* \times \mathcal{R}_{\mathsf{CH}} \to \{0,1\}^L$. Generate $(\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}}) \leftarrow \mathsf{CH.gen}$.
2. Pick random $\alpha_i, \beta_i, \theta_i, s_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [n]$ and compute $R_i := g_1^{\alpha_i + \theta_i \beta_i}$, $S_i := g_1^{s_i}$, $U_{ij} := g_1^{(\alpha_i + \theta_i \beta_i) s_j}$ for $i \neq j \in [n]$.
3. For each $\mu \in \{0,1\}$, $i \in [n]$, choose vectors $\mathbf{x}_{i,\mu} := (x_{i,1,\mu}, \ldots, x_{i,L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$ and $\mathbf{y}_{i,\mu} := (y_{i,1,\mu}, \ldots, y_{i,L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$, compute $\mathbf{z}_{i,\mu} := \mathbf{x}_{i,\mu} + \theta_i \mathbf{y}_{i,\mu}$ and $\mathbf{Z}_{i,\mu} := g_1^{\mathbf{z}_{i,\mu}} = (g_1^{z_{i,1,\mu}}, \ldots, g_1^{z_{i,L,\mu}})$.
4. $\mathsf{ek} := (\mathsf{pk}_{\mathsf{CH}}, \{\mathbf{Z}_{i,\mu}\}_{\mu \in \{0,1\}, i \in [n]}, \{g_1^{\theta_i}, R_i, S_i\}_{i \in [n]}, \{U_{ij}\}_{i \neq j \in [n]})$.
   $\mathsf{ik} := (\mathsf{ek}, \{s_i, \alpha_i + \theta_i \beta_i\}_{i \in [n]})$, $\mathsf{tk} := (\mathsf{sk}_{\mathsf{CH}}, \{\alpha_i, \beta_i, s_i\}_{i \in [n]}, \{\mathbf{x}_{i,\mu}, \mathbf{y}_{i,\mu}\}_{i \in [n], \mu \in \{0,1\}})$.

The tag space is defined as follows: tags are of the form $t = (t_a, t_c)$, where $t_a \in \{0,1\}^*$ is the auxiliary part, and $t_c := (\{B_i, D_i, E_i, F_i\}_{i \in [n]}, \mathsf{R}_{\mathsf{CH}}) \in \mathsf{G}_2^{4n} \times \mathcal{R}_{\mathsf{CH}}$ is the core tag part. Random t_c are uniformly random elements in the core tag space. The input distribution D_x is the Gaussian distribution $D_{\mathbb{Z}^n, \sigma}$ with the restriction that sampled $\mathbf{x} \in \mathsf{DomE}$.

$\mathsf{ABM.LGen}(\mathsf{tk}, t_a)$**:** The lossy core tag part $t_c := (\{B_i, D_i, E_i, F_i\}_{i \in [n]}, \mathsf{R}_{\mathsf{CH}})$ is computed as:

1. For $i \in [n]$, pick $b_i, \rho_i \xleftarrow{\$} \mathbb{Z}_q$, compute $B_i := g_2^{b_i}$, $D_i := g_2^{\alpha_i b_i s_i} g_2^{\rho_i x_{i,\tau}}$, $E_i := g_2^{\beta_i b_i s_i} g_2^{\rho_i y_{i,\tau}}$ and $F_i := g_2^{\rho_i}$, where $\tau := \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, (t_a, \{B'_i, D'_i, E'_i, F'_i\}_{i \in [n]}); \mathsf{R}'_{\mathsf{CH}}) \in \{0,1\}^L$, where $B'_i, D'_i, E'_i, F'_i, \mathsf{R}'_{\mathsf{CH}}$ are randomly chosen.
2. Compute $\mathsf{R}_{\mathsf{CH}} := \mathsf{CH.switch}(\mathsf{sk}_{\mathsf{CH}}, (t_a, \{B'_i, D'_i, E'_i, F'_i\}_{i \in [n]}), \mathsf{R}'_{\mathsf{CH}}, (t_a, \{B_i, D_i, E_i, F_i\}_{i \in [n]}))$.

Each tag is corresponding to a matrix $(\mathbf{M}_{ij})$ with

$$\mathbf{M}_{ij} := \begin{cases} e(U_{ji}, B_i) = g_T^{(\alpha_j + \theta_j \beta_j) s_i b_i} & \text{if} \quad i \neq j \\ \frac{e(g_1, D_i) \cdot e(g_1^{\theta_i}, E_i)}{e(Z_{i,\tau}, F_i)} & \text{else} \end{cases} \tag{3}$$

where $Z_{i,\tau} := \Pi_{k=1}^{L} Z_{i,k,\tau[k]}$.

We say a tag $t = (t_a, t_c)$ is lossy if $\mathbf{M}_{ii} = g_T^{(\alpha_i + \theta_i \beta_i) s_i b_i}$ for every $i \in [n]$. And a tag is injective if $\mathbf{M}_{ii} \neq g_T^{(\alpha_i + \theta_i \beta_i) s_i b_i}$ for every $i \in [n]$. Note that there exist tags that are neither lossy nor injective.

$\mathsf{ABM.Eval}(\mathsf{ek}, t, \mathbf{x})$**:** For input $\mathbf{x} := (x_1, \ldots, x_n) \in \mathsf{DomE}$, output $\mathbf{y} := (y_{0,1}, y_{1,1}, \ldots, y_{0,n}, y_{1,n})$ as:

1. Compute $\mathbf{M}$ according to Equation (3).
2. Compute $y_{0,i} := \prod_{j \in [n]} e(R_j, B_i)^{x_j} = g_T^{b_i \sum_{j \in [n]} (\alpha_j + \theta_j \beta_j) x_j}$ and $y_{1,i} := \prod_{j \in [n]} \mathbf{M}_{ij}^{x_j}$.

$\mathsf{ABM.Inver}(\mathsf{ik}, \mathbf{y})$**:** Compute x_i as:

1. Compute $M'_i := \frac{\mathbf{M}_{ii}}{g_T^{(\alpha_i + \theta_i \beta_i) s_i b_i}}$ for each $i \in [n]$. If there exists $i \in [n]$ such that $M'_i = g_T^0$, return $\perp$.
2. Compute $Z_i := y_{1,i} / y_{0,i}^{s_i}$ for each $i \in [n]$.
3. Search $x_i \in \mathsf{DomD}$ such that $M_i'^{x_i} = Z_i$ for all $i \in [n]$.

Decryption correctness. When the tag $t = (t_a, t_c)$ is injective, we have $M_i' \neq 1_T$ for every $i \in [n]$. Then

$$Z_i = y_{1,i}/y_{0,i}^{s_i} = \frac{\prod_{j\in[n]} \mathbf{M}_{ij}^{x_j}}{g_T^{s_i b_i \sum_{j\in[n]} r_j x_j}}$$
$$= \frac{\prod_{j\neq i} g_T^{(\alpha_j+\theta_j\beta_j)x_j s_i b_i} \cdot \mathbf{M}_{ii}^{x_i}}{g_T^{s_i b_i \sum_{j\in[n]}(\alpha_j+\theta_j\beta_j)x_j}} = {M_i'}^{x_i},$$

hence x_i can be recovered correctly.
Sampling correctness. According to Lemmas 6 and 7, if $\sigma \geq \Omega(\sqrt{n})$, the distribution $D_{\mathbb{Z}^n,\sigma}$ is efficiently samplable, and $\Pr[\mathbf{x} \in \mathsf{DomD} | \mathbf{x} \leftarrow D_{\mathbb{Z}^n,\sigma}] \geq 1 - 2^{-\Omega(\lambda)}$.
Lossiness. For the lossy tag, $y_{1,i} = g_T^{s_i b_i (\sum_j (\alpha_j+\theta_j\beta_j)x_j)}$, the output is completely determined by $\sum_j (\alpha_j + \theta_j\beta_j)x_j \bmod q$, so that the function has image size no larger than q. Since a vector $\mathbf{x}$ sampled from the distribution $D_{\mathbb{Z}^n,\sigma}$ has at least $n \log\sigma$ bits of min-entropy, by applying Lemma 10, we have lossiness $\ell := H_\infty(\mathbf{x}|(\mathbf{y}_0, \mathbf{y}_1)) \geq n \log\sigma - \log q$.
Explainable tags. Our construction has explainable tags as soon as the employed group G_2 is efficiently samplable and explainable.

Security. As in [LQ19], we prove the indistinguishability based on the R-wD3DH2 and SXDH assumptions.
Indistinguishability. We prove the indistinguishable property in $2n$ steps. At step $2k-1$, we modify D_k to be randomly distributed for all queries. At step $2k$, we modify E_k to be randomly distributed for all queries. Then in the end all tags are randomly distributed. We employ the R-wD3DH2 assumptions to assure the $(2k-1)$th modification is undetectable. To do this, the reduction embeds the challenge in $g_1^{\theta_k}$, R_k, S_k and (B_k, D_k)s for every query. And for the $(2k)$th modification, we use DDH2 assumption to give a bound, embedding the challenge in $g_1^{\beta_k}$ and (B_k, E_k)s for every query.

Theorem 3 (Indistinguishability). *If the R-wD3DH2 problem is (t_1, ε_1)-hard, the DDH2 problem is (t_2, ε_2)-hard, then the above* ABM-LTF *is $(q_{\mathsf{in}}, t_{\mathcal{A}}, \varepsilon)$-indistinguishable, where $t_1 \approx t_2 \approx t_{\mathcal{A}} + \mathsf{poly}(\lambda)$, and $\varepsilon \leq n(\varepsilon_1 + \varepsilon_2) + neg(\lambda)$.*

Proof. Note that for any tag $t = (t_a, t_c)$, the core tag part $t_c = (\{B_i, D_i, E_i, F_i\}_{i\in[n]}, \mathsf{R}_{\mathsf{CH}})$. If $D_i = g_2^{\alpha_i b_i s_i} g_2^{\rho_i x_{i,\tau}}$, $E_i = g_2^{\beta_i b_i s_i} g_2^{\rho_i y_{i,\tau}}$ for all $i \in [n]$, then t is a lossy tag; and if D_i, E_i are randomly distributed for all $i \in [n]$, then t is a random tag. We prove the indistinguishability via a sequence of $2n$ games. In the initial game of G_0, the adversary has access to the real lossy tag oracle $\textsc{Loss}(\mathsf{tk}, \cdot)$. Then we modify the games in the following sequence:

$$\mathsf{G}_0 \rightsquigarrow \mathsf{G}_{1,1} \rightsquigarrow \mathsf{G}_{2,1} \rightsquigarrow \mathsf{G}_{1,2} \rightsquigarrow \cdots \rightsquigarrow \mathsf{G}_{2,n}$$

Concretely,

$\mathsf{G}_{1,k}(k \in [n])$**:** In this game, the answer to every query is set as follows: the first k D_i and the first $k-1$ E_i are uniformly random and other parts of t_c is computed as the lossy tags.

$\mathsf{G}_{2,k}(k \in [n])$**:** In this game, the answer to every query is set as follows: the first k D_i and E_i are uniformly random and the last $n-k$ $D_i = g_2^{\alpha_i b_i s_i + \rho_i x_{i,\tau}}$ and $E_i = g_2^{\beta_i b_i s_i + \rho_i y_{i,\tau}}$ are set as the lossy tags.

Then in the final game, the adversary has access to an oracle $O_{T_c}(\cdot)$ that always returns random tags. Next we prove that differences between adjacent games are bounded by the R-wD3DH2 or DDH2 assumptions. We assume that an adversary runs in time $t_{\mathcal{A}}$ has advantage $\varepsilon_{\iota,k}$ in $\mathsf{G}_{\iota,k}$ for $\iota = 1, 2$.

Lemma 12. *If the R-wD3DH2 problem is* (t_1, ε_1)*-hard, then* $|\varepsilon_{1,k} - \varepsilon_{2,k-1}| \leq \varepsilon_1$ *for* $k \geq 1$*, and* $t_1 \approx t_{\mathcal{A}} + \mathsf{poly}(\lambda)$*. Here* $\mathsf{G}_{2,0} := \mathsf{G}_0$*.*

Proof. On receiving an R-wD3DH2 challenge $(g_1, g_2, g_1^b, g_1^c, \{g_2^{a_\xi}\}_\xi, g_2^b, g_2^c, \{T_\xi\}_\xi)$ for $\xi = 1, \ldots, q_{\mathsf{in}}$, the reduction $\mathcal{B}$'s task is to decide whether $T_\xi = g_2^{a_\xi bc}$ or T_ξ is randomly chosen from G_2. To do this, $\mathcal{B}$ proceeds as follows:

To generate ek, $\mathcal{B}$ firstly generates $(\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}}) \leftarrow \mathsf{CH.gen}$ as in the real game, then it picks $\alpha_i, \theta_i, \beta_i, s_i \xleftarrow{\$} \mathbb{Z}_q$ and computes $R_i := g_1^{\alpha_i + \theta_i \beta_i}$, $S_i := g_1^{s_i}$ for $i \in [n] \setminus k$, it also picks $\overline{\theta}_k, \overline{\beta}_k \xleftarrow{\$} \mathbb{Z}_q$ and sets $g_1^{\theta_k} := (g_1^c)^{\overline{\theta}_k}$, $R_k := g_1^{b + \overline{\theta}_k \cdot \overline{\beta}_k}$, $S_k := g_1^c$ and $U_{ik} := g_1^{(\alpha_i + \theta_i \cdot \beta_i)c}$, $U_{ki} := g_1^{(b + \overline{\theta}_k \cdot \overline{\beta}_k)s_i}$, in this way it implicitly sets $\alpha_k := b$, $\theta_k := c \cdot \overline{\theta}_k$, $\beta_k := \overline{\beta}_k / c$ and $s_k := c$. Then for each $\mu \in \{0,1\}$, $i \in [n]$, choose vectors $\mathbf{x}_{i,\mu} := (x_{i,1,\mu}, \ldots, x_{i,L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$ and $\mathbf{y}_{i,\mu} := (y_{i,1,\mu}, \ldots, y_{i,L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$, compute $\mathbf{z}_{i,\mu} := \mathbf{x}_{i,\mu} + \theta_i \mathbf{y}_{i,\mu}$ and $\mathbf{Z}_{i,\mu} := g_1^{\mathbf{z}_{i,\mu}} = (g_1^{z_{i,1,\mu}}, \ldots, g_1^{z_{i,L,\mu}})$ for $i \neq k$ and

$$\mathbf{Z}_{k,\mu} := g_1^{\mathbf{x}_{k,\mu}} (g_1^c)^{\overline{\theta}_k \cdot \mathbf{y}_{k,\mu}}.$$

It is obvious that ek is distributed exactly as in the real game.

When the adversary proposes the ξth query $t_a^{(\xi)}$ to the LOSS, the core tag $t_c^{(\xi)}$ is answered as follows: for $i < k$, (B_i, D_i, E_i, F_i) is randomly picked. And

$$B_k := g_2^{a_\xi}, \ D_k := T_\xi \cdot g_2^{\rho_k x_{k,\tau}}, \ E_k := (g_2^{a_\xi})^{\overline{\beta}_k} g_2^{\rho_k y_{k,\tau}}, \ F_k := g_2^{\rho_k}.$$

with $\rho_k \xleftarrow{\$} \mathbb{Z}_q$ and $\tau \xleftarrow{\$} \{0,1\}^L$. For $i > k$, $\mathcal{B}$ picks $b_i, \rho_i \xleftarrow{\$} \mathbb{Z}_q$ and sets

$$B_i := g_2^{b_i}, \ D_i := g_2^{\alpha_i s_i b_i} \cdot g_2^{\rho_i x_{i,\tau}}, \ E_i := g_2^{\beta_i s_i b_i} \cdot g_2^{\rho_i y_{i,\tau}}, \ F_i := g_2^{\rho_i}.$$

Finally, $\mathcal{B}$ uses the trapdoor $\mathsf{sk}_{\mathsf{CH}}$ for the chameleon hash to find coins R_{CH} such that $\tau = \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, t_a, \{B_i, D_i, E_i, F_i\}_{i=1}^n; \mathsf{R}_{\mathsf{CH}})$.

It is not difficult to see that E_k is distributed as the lossy tag since

$$E_k = (g_2^{a_\xi})^{\overline{\beta}_k} g_2^{\rho_k y_{k,\tau}} = g_2^{a_\xi (\overline{\beta}_k / c) \cdot c} \cdot g_2^{\rho_k y_{k,\tau}} = g_2^{a_\xi \beta_k \cdot s_k} \cdot g_2^{\rho_k y_{k,\tau}},$$

then if $T_\xi = g_2^{a_\xi bc}$, D_k is distributed as the lossy tag, hence t_c is distributed as in $\mathsf{G}_{2,k-1}$; and if T_ξ is randomly picked, then t_c is distributed as in $\mathsf{G}_{1,k}$. $\square$

Lemma 13. *If the DDH2 problem is (t_2, ε_2)-hard, then $|\varepsilon_{2,k} - \varepsilon_{1,k}| \le \varepsilon_2 + \frac{1}{q}$ for $k \ge 1$, and $t_2 \approx t_{\mathcal{A}} + \mathsf{poly}(\lambda)$.*

Proof. On receiving a DDH2 challenge $(g_2, \{g_2^{a_\xi}\}_\xi, g_2^b, \{T_\xi\}_\xi)$ for $\xi = 1, \ldots, q_{\mathsf{in}}$, the reduction $\mathcal{B}$'s task is to decide whether $T_\xi = g_2^{a_\xi b}$ or T_ξ is randomly chosen from G_2. To do this, $\mathcal{B}$ proceeds as follows:

To generate ek, $\mathcal{B}$ firstly generates $(\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}}) \leftarrow \mathsf{CH.gen}$ as in the real game, then it picks $\alpha_i, \theta_i, \beta_i, s_i \xleftarrow{\$} \mathbb{Z}_q$ and computes $R_i := g_1^{\alpha_i + \theta_i \beta_i}$, $S_i := g_1^{s_i}$ for $i \in [n] \setminus k$, it also picks $\theta_k, r_k, s_k \xleftarrow{\$} \mathbb{Z}_q$ and sets $R_k := g_1^{r_k}$, $S_k := g_1^{s_k}$ and $U_{ij} := (R_i)^{s_j}$ for $i \ne j$. Here it implicitly sets $\beta_k := b$, $\alpha_k := r - \theta_k \cdot b$. Then for each $\mu \in \{0,1\}$, $i \in [n]$, choose vectors $\mathbf{x}_{i,\mu} := (x_{i,1,\mu}, \ldots, x_{i,L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$ and $\mathbf{y}_{i,\mu} := (y_{i,1,\mu}, \ldots, y_{i,L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$, compute $\mathbf{z}_{i,\mu} := \mathbf{x}_{i,\mu} + \theta_i \mathbf{y}_{i,\mu}$ and $\mathbf{Z}_{i,\mu} := g_1^{\mathbf{z}_{i,\mu}} = (g_1^{z_{i,1,\mu}}, \ldots, g_1^{z_{i,L,\mu}})$ for $i \in [n]$. It is obvious that ek is distributed exactly as in the real game.

When the adversary proposes the ξth query $t_a^{(\xi)}$ to the LOSS, the core tag $t_c^{(\xi)}$ is answered as follows: for $i < k$, (B_i, D_i, E_i, F_i) is randomly picked. And

$$B_k := g_2^{a_\xi},\ D_k \xleftarrow{\$} \mathsf{G}_2,\ E_k := (T_\xi)^{s_k} g_2^{\rho_k y_{k,\tau}},\ F_k := g_2^{\rho_k}.$$

with $\rho_k \xleftarrow{\$} \mathbb{Z}_q$ and $\tau \xleftarrow{\$} \{0,1\}^L$. For $i > k$, $\mathcal{B}$ picks $b_i \xleftarrow{\$} \mathbb{Z}_q$ and $\rho_i \xleftarrow{\$} \mathbb{Z}_q$ and sets

$$B_i := g_2^{b_i},\ D_i := g_2^{\alpha_i s_i b_i} \cdot g_2^{\rho_i x_{i,\tau}},\ E_i := g_2^{\beta_i s_i b_i} \cdot g_2^{\rho_i y_{i,\tau}},\ F_i := g_2^{\rho_i}.$$

Finally, $\mathcal{B}$ uses the trapdoor $\mathsf{sk}_{\mathsf{CH}}$ for the chameleon hash to find R_{CH} such that $\tau = \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, t_a, \{B_i, D_i, E_i, F_i\}_{i=1}^n; \mathsf{R}_{\mathsf{CH}})$.

It is not difficult to see that if $T_\xi = g_2^{a_\xi b}$, then E_k is distributed as the lossy tag, hence t_c is distributed as in $\mathsf{G}_{1,k}$; and if T_ξ is randomly picked, then t_c is distributed as in $\mathsf{G}_{2,k}$. □

□

Evasiveness. To prove evasiveness, we proceed in two steps: firstly we modify the VER to reject tags with non-fresh τ^*, then we use a similar proof as the unforgeable proof of the underlying MAC to bound the success probability of breaking evasiveness. To reject the non-fresh τ^*, we firstly remove the use of $\mathsf{sk}_{\mathsf{CH}}$ for the chameleon hash, then use the collision-resistant property to reject the non-fresh τ^*, which is referred to as a 'deferred analysis' technique [Hof12].

Theorem 4 (Evasiveness). *If the* CH *is (t_1, ε_1)-collision-resistant, the R-wD3DH2 problem is (t_2, ε_2)-hard and the DDH2 problem is (t_3, ε_3)-hard, the* MAC *scheme described in [LQ19] is (t_4, ε_4)-unforgeable, then the above* $\mathsf{ABM\text{-}LTF}$ *is $(q_{\mathsf{eva}}, q_{\mathsf{ver}}, t, \varepsilon)$ evasiveness, where $t_1 \approx t_2 \approx t_3 \approx t_4 \approx t + \mathsf{poly}(\lambda)$, and $\varepsilon \le \varepsilon_1 + n(\varepsilon_2 + \varepsilon_3 + \varepsilon_4) + neg(\lambda)$.*

Proof. We prove the evasiveness via a sequence of 2 games. In the initial game of G_0, the adversary proceeds as in the real game. And in the next game G_1, when the adversary proposes tags to the VER oracle, it rejects those tags that generate the same chameleon hash τ^* as that answered by the lossy tag oracle $\text{LOSS}(\mathsf{tk}, \cdot)$. We use bad_i to denote the event that $\mathcal{A}$ manages to output a non-injective tag in G_i for $i = 0, 1$. It is obvious that $\varepsilon = \Pr[\mathsf{bad}_0]$.

Lemma 14. *If the* CH *is* (t_1, ε_1)*-collision-resistant, the R-wD3DH2 problem is* (t_2, ε_2)*-hard, the DDH2 problem is* (t_3, ε_3)*-hard, then* $|\Pr[\mathsf{bad}_0] - \Pr[\mathsf{bad}_1]| \leq \varepsilon_1 + n(\varepsilon_2 + \varepsilon_3) + neg(\lambda)$.

Proof. We denote bad_h to denote the event that $\mathcal{A}$ outputs a tag $t = (t_a, (\{B_i, D_i, E_i, F_i\}_{i \in [n]}, \mathsf{R_{CH}}))$ with a hash τ the same as that produced by the LOSS before. It is straightforward that

$$|\Pr[\mathsf{bad}_0] - \Pr[\mathsf{bad}_1]| \leq \Pr[\mathsf{bad}_h \text{ in } \mathsf{G}_1].$$

As in both G_0 and G_1, the chameleon hash trapdoor $\mathsf{sk_{CH}}$ is used to answer LOSS$(\mathsf{tk}, \cdot)$ queries, which makes it difficult to use the collision-resistant property of the chameleon hash to bound $\Pr[\mathsf{bad}_h]$ directly. To solve this problem, we use the "deferred analysis" proof technique [Hof12]. That is, we introduce two intermediate games $\mathsf{G}_{1'}$, $\mathsf{G}_{2'}$ defined as follows:

$\mathsf{G}_{1'}$**:** The same as G_1, except that the VER oracle only checks the freshness of τ computed from the proposed tags.
$\mathsf{G}_{2'}$**:** The same as $\mathsf{G}_{1'}$, except that the LOSS oracle returns random tags instead of lossy tags.

It is obvious that the probability of bad_h is the same in G_1 and $\mathsf{G}_{1'}$. Here as we do not need any secret information to answer VER queries, then we can employ the indistinguishable proof and get

$$|\Pr[\mathsf{bad}_h \text{ in } \mathsf{G}_{1'}] - \Pr[\mathsf{bad}_h \text{ in } \mathsf{G}_{2'}]| \leq n(\varepsilon_2 + \varepsilon_3) + neg(\lambda).$$

Now in $\mathsf{G}_{2'}$, $\mathsf{sk_{CH}}$ is no longer used and we can use collision-resistant property to bound $\Pr[\mathsf{bad}_h \text{ in } \mathsf{G}_{2'}] \leq \varepsilon_1$. ☐

Next we bound bad_1 by the unforgeablity of MAC. $\Pr[\mathsf{bad}_1] \leq n\varepsilon_4$. On receiving the MAC $\mathsf{pp} = (g_1, g_2, h_2, A, R, \mathbf{Z}_0, \mathbf{Z}_1)$ and η, the reduction $\mathcal{B}$'s task is to output a fresh message-tag pair that can pass the verification. $\mathcal{B}$ proceeds as follows:

To generate ek, $\mathcal{B}$ firstly generates $(\mathsf{pk_{CH}}, \mathsf{sk_{CH}}) \leftarrow \mathsf{CH.gen}$ as in the real game, then it picks a random $k \in [n]$ and $\alpha_i, \theta_i, \beta_i, s_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ and computes $R_i := g_1^{\alpha_i + \theta_i \beta_i}$ and $S_i := g_1^{s_i}$ for $i \in [n] \setminus k$, it also sets $R_k := R$, $S_k := g_1^{\eta}$ and $g_1^{\theta_k} := A$, in this way it implicitly sets $s_k := \eta$. Note that $\mathcal{B}$ can compute U_{ij} for $i \neq j$. Then for each $\mu \in \{0,1\}$, $i \in [n]$, it chooses vectors $\mathbf{x}_{i,\mu} := (x_{i,1,\mu}, \ldots, x_{i,L,\mu}) \stackrel{\$}{\leftarrow} \mathbb{Z}_q^L$ and $\mathbf{y}_{i,\mu} := (y_{i,1,\mu}, \ldots, y_{i,L,\mu}) \stackrel{\$}{\leftarrow} \mathbb{Z}_q^L$, computes $\mathbf{z}_{i,\mu} := \mathbf{x}_{i,\mu} + \theta_i \mathbf{y}_{i,\mu}$ and $\mathbf{Z}_{i,\mu} := g_1^{\mathbf{z}_{i,\mu}} = (g_1^{z_{i,1,\mu}}, \ldots, g_1^{z_{i,L,\mu}})$ for $i \neq k$, and $\mathbf{Z}_{k,\mu} := g_1^{\mathbf{x}_{k,\mu}} A^{\mathbf{y}_{k,\mu}}$. Finally $\mathcal{B}$ sets $\mathsf{ek} := (\mathsf{pk_{CH}}, \{\mathbf{Z}_{i,0}, \mathbf{Z}_{i,1}, g_1^{\theta_i}, R_i, S_i\}_{i \in [n]}, \{U_{ij}\}_{i \neq j \in [n]})$. It is obvious that ek is distributed exactly as in the real game.

When the adversary proposes the query t_a to the LOSS, the core tag part t_c is answered as follows:

1. $\mathcal{B}$ samples a random τ in the range of CH. For $i \neq k$, $\mathcal{B}$ picks $b_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ and $\rho_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ and sets

$$B_i := g_2^{b_i},\ D_i := h_2^{\alpha_i s_i b_i} \cdot g_2^{\rho_i x_{i\tau}},\ E_i := h_2^{\beta_i s_i b_i} \cdot g_2^{\rho_i y_{i\tau}},\ F_i := g_2^{\rho_i}.$$

2. For $i = k$, $\mathcal{B}$ transfers τ to its EVAL oracle and gets $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ as response, it sets $B_k := \sigma_1,\ D_k := \sigma_2,\ E_k := \sigma_3,\ F_k := \sigma_4$.
3. Finally, $\mathcal{B}$ uses the trapdoor $\mathsf{sk_{CH}}$ for the chameleon hash to find $\mathsf{R_{CH}}$ such that $\tau = \mathsf{CH.eval}(\mathsf{pk_{CH}}, t_a, \{B_i, D_i, E_i, F_i\}_{i=1}^n; \mathsf{R_{CH}})$.

When the adversary finally outputs q_{ver} tags, $\mathcal{B}$ checks whether there exists one tag such that

$$e(g_1, D_k) \cdot e(A, E_k) = e(R, B_k)^\eta \cdot e(Z_{k,\tau}, F_k),$$

and output $(\tau, (B_k, D_k, E_k, F_k))$ as the forged message-tag pair if the equality holds. Note that if there exists one non-injective tag in the final outputs of $\mathcal{A}$, then $\mathcal{B}$ can find it out, and with probability $\frac{1}{n}$, $\mathcal{B}$ guesses the right k.

□

4 The SIM-SO-CCA Secure Constructions

4.1 The General Construction

In this subsection we will give a general SIM-SO-CCA secure construction tightly from an LTF and an ABM-LTF. Our construction is in general the same as that in [Hof12] except for two differences: the first one is that the encryption randomness can be chosen from a non-uniform distribution to consort with the efficiently opening algorithm; the second difference is that here we use a one-time message authentication code MAC and a universal hash to replace the primitive 'lossy authenticated encryption (LAE)' in [Hof12], this change is only conceptual and for easy expression. The description of our construction (Keygen, Enc, Dec, LKeygen, Lenc, Opener) is as follows, where LKeygen, Lenc and Opener algorithms are only used in the security proof.

Let $\Pi^{\mathsf{LTF}} := (\mathsf{LTF.Igen}, \mathsf{LTF.Lgen}, \mathsf{LTF.Eval}, \mathsf{LTF.Invert})$ be an instance of the LTF. And let $\Pi^{\mathsf{ABM\text{-}LTF}} := (\mathsf{ABM.Gen}, \mathsf{ABM.LGen}, \mathsf{ABM.Eval}, \mathsf{ABM.Inver})$ be an instance of the ABM-LTF. MAC := (MAC.eval, MAC.ver) be a one-time unforgeable MAC. We assume the input domain of both instances are DomE and the inversion domain are both DomD.

Keygen**:** The public key is generated as:
1. Generate the evaluation and inversion keys for LTF: $(\mathsf{ek}_1, \mathsf{ik}_1) \leftarrow \mathsf{LTF.Igen}$.
2. Generate the evaluation key for ABM-LTF: $(\mathsf{ek}_2, \mathsf{ik}_2, \mathsf{tk}_2) \leftarrow \mathsf{ABM.Gen}$.
3. Choose a universal hash function $h : \mathsf{DomE} \to \{0,1\}^{\ell+\ell'}$.

Output $\mathsf{pk} := (\mathsf{ek}_1, \mathsf{ek}_2, h)$ and $\mathsf{sk} := \mathsf{ik}_1$.

Enc**:** To encrypt $\mathsf{m} \in \{0,1\}^\ell$, choose $x \leftarrow D_x$, D_x is a distribution over DomE.
1. Compute $y_1 := \mathsf{LTF.Eval}(\mathsf{ek}_1, x)$.
2. Set $t_a := y_1$ and pick random t_c for ABM-LTF. Then compute $y_2 := \mathsf{ABM.Eval}(\mathsf{ek}_2, (t_a, t_c), x)$.
3. Compute $(k_1, k_2) := h(x)$ with $k_1 \in \{0,1\}^\ell$ and $k_2 \in \{0,1\}^{\ell'}$.
4. Compute $y_3 := k_1 \oplus \mathsf{m}$ and $y_4 := \mathsf{MAC.eval}(k_2, (y_2, y_3))$.
   Output $\mathsf{c} := (y_1, y_2, t_c, y_3, y_4)$.

Dec: To decrypt $\mathsf{c} := (y_1, y_2, t_c, y_3, y_4)$ with sk,

1. Compute $x := \mathsf{LTF.Invert}(\mathsf{ik}_1, y_1)$ and $(k_1, k_2) := h(x)$.
2. Verify if $\mathsf{LTF.Eval}(\mathsf{ek}_1, x) = y_1$, $\mathsf{ABM.Eval}(\mathsf{ek}_2, (t_a, t_c), x) = y_2$, $\mathsf{MAC.ver}(k_2, (y_2, y_3), y_4) = 1$ and $x \in \mathsf{DomD}$, abort if any of the equalities does not hold.
3. Compute and output $\mathsf{m} := y_3 \oplus k_1$.

$\mathsf{LKeygen}$: The lossy public key generation algorithm generates $(\mathsf{ek}_1, \mathsf{ek}_2, h, \mathsf{ik}_2, \mathsf{tk}_2, a)$ and outputs $\mathsf{pk}_l := (\mathsf{ek}_1, \mathsf{ek}_2, h)$. The generated keys satisfy that:

1. ek_1 distributed as a random output of $\mathsf{LTF.Lgen}$.
2. $(\mathsf{ek}_2, \mathsf{ik}_2, \mathsf{tk}_2)$ has the same distribution as the random output of $\mathsf{ABM.Gen}$.
3. h is distributed as a randomly picked universal hash function $\mathsf{DomE} \to \{0,1\}^{\ell+\ell'}$.

Lenc: To generate a lossy ciphertext of m with tk_2, choose $x \leftarrow D_x$ and proceed as follows:

1. Compute $y_1 := \mathsf{LTF.Eval}(\mathsf{ek}_1, x)$.
2. Set $t_a := y_1$ and compute $t_c \xleftarrow{\$} \mathsf{ABM.LGen}(\mathsf{tk}_2, t_a)$. Then compute $y_2 := \mathsf{ABM.Eval}(\mathsf{ek}_2, (t_a, t_c), x)$.
3. Compute $(k_1, k_2) := h(x)$.
4. Compute $y_3 := k_1 \oplus \mathsf{m}$ and $y_4 := \mathsf{MAC.eval}(k_2, (y_2, y_3))$. Output $\mathsf{c} := (y_1, y_2, t_c, y_3, y_4)$.

Opener: The opener algorithm takes as inputs the (pk, a) generated by the $\mathsf{LKeygen}$ algorithm, m, x and c generated by Lenc, and any fixed message m', it outputs x' such that:

1. $y_1 = \mathsf{LTF.Eval}(\mathsf{ek}_1, x')$.
2. $y_2 := \mathsf{ABM.Eval}(\mathsf{ek}_2, (t_a, t_c), x')$.
3. $y_3 = k_1' \oplus \mathsf{m}'$, $y_4 = \mathsf{MAC.eval}(k_2', (y_2, y_3))$ and $H(x') = (k_1', k_2')$.
4. $x' \in \mathsf{DomD}$ and distributed statistically close to D_x.

Correctness. Correctness can be get easily according to the correctness of LTF, MAC and $\mathsf{ABM\text{-}LTF}$.

Remark 1. The existence of the Opener algorithm indicates that given y_1 and y_2, the residence entropy of x is larger than ℓ.

Theorem 5. *For a PKE scheme constructed above, if the underlying* LTF *is* (t_1, ε_1)*-indistinguishable,* $\mathsf{ABM\text{-}LTF}$ *is* $(q_{\mathsf{in2}}, t_2, \varepsilon_2)$*-indistinguishable and* $(q_{\mathsf{eva3}}, q_{\mathsf{ver3}}, t_3, \varepsilon_3)$*-evasive,* MAC *is* (t_4, ε_4)*-unforgeable, then our scheme is* $(q_{\mathsf{dec}}, N, t_{\mathcal{A}}, t', \varepsilon)$*-*$\mathsf{SIM\text{-}SO\text{-}CCA}$ *secure, where* $\varepsilon \leq \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + N \cdot q_{\mathsf{dec}} \cdot \varepsilon_4 + neg(\lambda)$*,* $q_{\mathsf{ver3}} = q_{\mathsf{dec}}$*,* $q_{\mathsf{in2}} = q_{\mathsf{eva3}} = N$ *and* $t' \approx t_1 \approx t_2 \approx t_3 \approx t_4 \approx t_{\mathcal{A}} + \mathsf{poly}(\lambda)$*.*

The proof of Theorem 5 is similar to that in [Hof12] and we put it in supporting material C.

4.2 Instantiation

By combining the MDDH based LTF and efficient opening algorithm given in [LSSS17b], we give an SIM-SO-CCA secure construction in this subsection.

Keygen: Let $\Pi^{\mathsf{LTF}} := (\mathsf{LTF.Igen}, \mathsf{LTF.Lgen}, \mathsf{LTF.Eval}, \mathsf{LTF.Invert})$ be an instance of the LTF given in Appendix E in [LSSS17a]. And let $\Pi^{\mathsf{ABM\text{-}LTF}} := (\mathsf{ABM.Gen}, \mathsf{ABM.LGen}, \mathsf{ABM.Eval}, \mathsf{ABM.Inver})$ be an instance of the ABM-LTF given in Section 3.1(Section 3.2). We assume the input domain of both instances are $\mathsf{DomE} := \{\mathbf{x} \in \mathbb{Z}^n | \|\mathbf{x}\| \le \gamma \cdot \sigma\sqrt{n}\}$ and the inversion domain are both $\mathsf{DomD} := \{\mathbf{x} \in \mathbb{Z}^n | \|\mathbf{x}\| \le \sigma\sqrt{n}\}$ with $\sigma \ge \Omega(n)$ and $\gamma \ge 3$. Then the public key is generated as:
1. Generate the evaluation and inversion keys for LTF: pick $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$, set $\mathsf{ek}_1 := g_T^{\mathbf{A}}$ and $\mathsf{sk} := \mathbf{A}^{-1}$.
2. Generate the evaluation key for ABM-LTF: $(\mathsf{ek}_2, \mathsf{ik}_2, \mathsf{tk}_2) \leftarrow \mathsf{ABM.Gen}$ as in Section 3.1\Section 3.2.
3. Choose a random matrix $\mathbf{H}_{\mathsf{UH}} \xleftarrow{\$} \mathbb{Z}_q^{\ell\times n}$.

Output $\mathsf{pk} := (\mathsf{ek}_1, \mathsf{ek}_2, \mathbf{H}_{\mathsf{UH}})$ and sk.

Enc: To encrypt $\mathsf{m} \in \mathbb{Z}_q^{\ell-1}$, choose $\mathbf{x} \leftarrow D_{\mathbb{Z}^n,\sigma}$ and proceeds as follows:
1. Compute $\mathbf{y}_0 := \mathsf{ek}_1^{\mathbf{x}} = g_T^{\mathbf{Ax}}$.
2. Compute $(\mathbf{k}_1^\top, k_2)^\top := \mathbf{H}_{\mathsf{UH}}\mathbf{x}$ and $\mathbf{y}_3 := \mathbf{k}_1 + \mathsf{m} \bmod q$, .
3. Set $t_a := \mathbf{y}_0$ and pick random t_c for ABM-LTF. Then compute $\mathbf{M}$ as Equation (2) (Equation (3) for the tightly secure case) and
$$y_{1,i} := \prod_{j\in[n]} e(R_j, B_i)^{x_j} = g_T^{b_i \sum_{j\in[n]} r_j x_j},\ y_{2,i} := \prod_{j\in[n]} \mathbf{M}_{ij}^{x_j}.$$
Set $\mathbf{y}_\iota := (y_{\iota,1}, ..., y_{\iota,n})$ for $\iota = 1, 2$.
4. Compute $y_4 = \mathsf{MAC.eval}(k_2, (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3))$. Output $\mathsf{c} := (\mathbf{y}_0, t_c, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, y_4)$.

Dec: To decrypt $\mathsf{c} := (\mathbf{y}_0, t_c, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, y_4)$ with sk,
1. Compute $\mathbf{X} := g_T^{\mathbf{A}^{-1}\mathbf{Ax}} = g_T^{\mathbf{x}}$. Use exhaustive search over DomD to find such $\mathbf{x}$, and abort if no $\mathbf{x}$ is found.
2. Verify if $\mathsf{ABM.Eval}(\mathsf{ek}_2, t_a, t_c, \mathbf{x}) = (\mathbf{y}_1, \mathbf{y}_2)$, abort if the equality does not hold.
3. Compute $(\mathbf{k}_1^\top, k_2)^\top := \mathbf{H}_{\mathsf{UH}}\mathbf{x}$, Verify if $\mathsf{MAC.ver}(k_2, (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3), y_4) = 1$ and abort if the equality does not hold.
4. Output $\mathsf{m} := \mathbf{y}_3 - \mathbf{k}_1 \bmod q$.

To illustrate the SIM-SO-CCA security, next we present the lossy key generation algorithm and the efficient opening procedure. As in [LSSS17b], we require $q > 2^\lambda$ and $n > 3(k + \ell + 1) \cdot \lceil \log q \rceil$ to ensure the lossiness.

LKeygen: Choose a random $\bar{\mathbf{a}} \xleftarrow{\$} \mathbb{Z}_q^{n\times 1}$, $\mathbf{w} \in \mathbb{Z}_q^n$, $\mathbf{r} \in \mathbb{Z}_q^n$. Set $\mathbf{A} = \bar{\mathbf{a}} \cdot \mathbf{w}^T$.
1. Choose $\mathbf{C}_0 \xleftarrow{\$} \mathbb{Z}_q^{\bar{n}\times\bar{\ell}}$ with $\bar{\ell} := \ell + 2$ and $\bar{n} := n - \bar{\ell} \cdot \lceil \log q \rceil$. Use the trapdoor generation algorithm in [MP12] to generate $\mathbf{R}_{sim} \xleftarrow{\$} \{-1, 1\}^{\bar{\ell}\cdot\lceil \log q\rceil \times \bar{n}}$ and
$$\mathbf{C}^\top := \begin{pmatrix} \mathbf{C}_0 \\ -\mathbf{R}_{sim}\mathbf{C}_0 + \mathbf{G}_{sim} \end{pmatrix} \in \mathbb{Z}_q^{n\times\bar{\ell}},$$

where $\mathbf{G}_{sim} \in \mathbb{Z}_q^{\bar{\ell}\cdot\lceil\log q\rceil\times\bar{\ell}}$ is the gadget matrix. Then from Lemmas 2 and 8, when $\bar{n} \geq 2\bar{\ell}\cdot\log q$, $\mathbf{C}$ is statistically close to the uniform distribution and $\mathbf{R}_{sim}$ is a trapdoor for $\mathbf{C}$. Then parse $\mathbf{C}$ as:

$$\mathbf{C} = \begin{pmatrix} \mathbf{w}^T \\ \mathbf{r}^\top \\ \mathbf{H}_{\mathsf{UH}} \end{pmatrix} \in \mathbb{Z}_q^{\bar{\ell}\times n},$$

2. Define $\mathsf{ek}_1 := g_T^{\mathbf{A}}$, $R_i := g_1^{r_i}$ for $i = 1,...,n$.
3. Generate other parameters of $\mathsf{ABM\text{-}LTF}$ as real. That is, $(\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}}) \leftarrow \mathsf{CH.gen}$, pick random $s_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [n]$ and $w_0,...,w_L \xleftarrow{\$} \mathbb{Z}_q$, where n is the input length, compute $S_i := g_1^{s_i}$ and $W_{s,k} := g_\iota^{w_k}$ for $s = 1,2, k \in [0,L]$. Set $\mathsf{ek}_2 := (\mathsf{pk}_{\mathsf{CH}}, \{W_{s,k}\}_{s\in[2],k\in[0,L]}, \{R_i, S_i\}_{i\in[n]})$. $\mathsf{ik} := (\mathsf{ek}_2, \{s_i\}_{i\in[n]})$.
   (For tightly secure case, $\mathsf{ek}_2 := (\mathsf{pk}_{\mathsf{CH}}, \{\mathbf{Z}_{i,\mu}\}_{\mu\in\{0,1\},i\in[n]}, \{g_1^{\theta_i}, R_i, S_i\}_{i\in[n]}, \{U_{ij}\}_{i\neq j\in[n]})$ is generated as: $(\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}}) \leftarrow \mathsf{CH.gen}$, for $i \in [n]$, pick random $\beta_i, \theta_i, s_i \xleftarrow{\$} \mathbb{Z}_q$, for $\mu \in \{0,1\}$, choose vectors $\mathbf{x}_{i,\mu} := (x_{i,1,\mu}, \ldots, x_{i,L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$ and $\mathbf{y}_{i,\mu} := (y_{i,1,\mu}, \ldots, y_{i,L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$, implicitly set $\alpha_i = r_i - \theta_i\beta_i$, and compute $S_i := g_1^{s_i}$, $U_{ij} := g_1^{r_i s_j}$ for $i \neq j \in [n]$, $\mathbf{z}_{i,\mu} := \mathbf{x}_{i,\mu} + \theta_i\mathbf{y}_{i,\mu}$ and $\mathbf{Z}_{i,\mu} := g_1^{\mathbf{z}_{i,\mu}} = (g_1^{z_{i,1,\mu}}, \ldots, g_1^{z_{i,L,\mu}})$. $\mathsf{ik} := (\mathsf{ek}_2, \{s_i, r_i\}_{i\in[n]})$).

Return $\mathsf{pk}_l := (\mathsf{ek}_1, \mathsf{ek}_2, \mathbf{H}_{\mathsf{UH}})$ and $\mathsf{sk}_l := (\mathbf{R}_{sim}, \mathbf{C}_0, \overline{\mathbf{a}})$.

Lenc: To encrypt $\mathsf{m} \in \mathbb{Z}_q^{\ell-1}$, choose $\mathbf{x} \leftarrow D_{\mathbb{Z}^n,\sigma_x}$ and proceeds as follows:

1. Compute $\mathbf{y}_0 := \mathsf{ek}_1^{\mathbf{x}} = g_T^{\mathbf{A}\mathbf{x}}$.
2. Compute $(\mathbf{k}_1^\top, k_2)^\top := \mathbf{H}_{\mathsf{UH}}\mathbf{x}$ and $\mathbf{y}_3 := \mathbf{k}_1 + \mathsf{m} \bmod q$.
3. Set $t_a := \mathbf{y}_0$, and generate $t_c \xleftarrow{\$} \mathsf{ABM.LGen}(\mathsf{tk}, t_a)$. Then compute $\mathbf{M}$ as Equation (2) (Equation (3) for the tightly secure case) and

$$y_{1,i} := \prod_{j\in[n]} e(R_j, B_i)^{x_j} = g_T^{b_i \sum_{j\in[n]} r_j x_j},\ y_{2,i} := \prod_{j\in[n]} \mathbf{M}_{ij}^{x_j}.$$

   Set $\mathbf{y}_\iota := (y_{\iota,1}, ..., y_{\iota,n})$ for $\iota = 1,2$.
4. Compute $y_4 = \mathsf{MAC.eval}(k_2, (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3))$. Output $\mathsf{c} := (\mathbf{y}_0, t_c, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, y_4)$.

Opener: Given $\mathbf{x} \in \mathsf{DomD}$ for encrypting m, to find the new randomness $\mathbf{x}'$ to explain the ciphertext to m', do the following:

1. Compute $c_{1,\mathbf{x}} := \mathbf{w}^T\mathbf{x} \in \mathbb{Z}_q$, $c_{2,\mathbf{x}} := \mathbf{r}^\top\mathbf{x} \in \mathbb{Z}_q$ and $\mathbf{c}_{3,\mathbf{x}} := \mathbf{H}_{\mathsf{UH}}\mathbf{x} + (\frac{\mathsf{m}-\mathsf{m}'}{0}) \in \mathbb{Z}_q^\ell$. Define

$$\mathbf{t}_{\mathbf{x}} := [\,c_{1,\mathbf{x}} | c_{2,\mathbf{x}} | \mathbf{c}_{3,\mathbf{x}}^\top]^\top \in \mathbb{Z}_q^{\bar{\ell}}$$

2. Using the trapdoor $\mathbf{R}_{sim}$, sample a small-norm vector $\mathbf{x}' \leftarrow D_{\Lambda_q^{\mathbf{t}_{\mathbf{x}}}(\mathbf{C}),\sigma_x}$, such that

$$\mathbf{C}\cdot\mathbf{x}' = \mathbf{t}_{\mathbf{x}} \bmod q.$$

   If $\mathbf{x}' \in \mathsf{DomD}$, output $\mathbf{x}'$. Otherwise, repeat step 2 until a suitable $\mathbf{x}'$ is found.

Acknowledgments. We thank the anonymous reviewers for their helpful comments. The first author is supported by the National Nature Science Foundation of China (No. 61772515), Beijing Municipal Science & Technology Commission (Project Number: Z191100007119006), and the National Cryptography Development Fund (No. MMJJ20170116). The second author is supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701). This work is also partially supported by Indo French Center for the Promotion of Advanced Research (IFCPAR, project number: 6002-1).

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

BB08. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

BDWY12. Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 645–662, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

BHK12. Florian Böhl, Dennis Hofheinz, and Daniel Kraschewski. On definitions of selective opening security. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 522–539, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.

BHY09. Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.

BKP14. Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

BL17. Xavier Boyen and Qinyi Li. All-but-many lossy trapdoor functions from lattices and applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 298–331, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

BR95. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111, Perugia, Italy, May 9–12, 1995. Springer, Heidelberg, Germany.

BW10. Xavier Boyen and Brent Waters. Shrinking the keys of discrete-log-type lossy trapdoor functions. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 35–52, Beijing, China, June 22–25, 2010. Springer, Heidelberg, Germany.

BWY11. Mihir Bellare, Brent Waters, and Scott Yilek. Identity-based encryption secure against selective opening attack. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 235–252, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany.

BY09. Mihir Bellare and Scott Yilek. Encryption schemes secure under selective opening attack. Cryptology ePrint Archive, Report 2009/101, 2009. `http://eprint.iacr.org/2009/101`.

CHK05. Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 150–168, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany.

DNRS99. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press.

DRS04. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

FHKW10. Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 381–402, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

Fuj14. Eiichiro Fujisaki. All-but-many encryption - A new framework for fully-equipped UC commitments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 426–447, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

GM84. S. Goldwasser and S. Micali. Probabilistic encryption. *J. of Computer and System Sciences*, 28, 1984.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

HJKS15. Felix Heuer, Tibor Jager, Eike Kiltz, and Sven Schäge. On the selective opening security of practical public-key encryption schemes. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 27–51, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany.

HJP18. Dennis Hofheinz, Dingding Jia, and Jiaxin Pan. Identity-based encryption tightly secure under chosen-ciphertext attacks. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 190–220, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

HJR16. Dennis Hofheinz, Tibor Jager, and Andy Rupp. Public-key encryption with simulation-based selective-opening security and compact ciphertexts. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 146–168, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

HK08. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

HKM+18. Keisuke Hara, Fuyuki Kitagawa, Takahiro Matsuda, Goichiro Hanaoka, and Keisuke Tanaka. Simulation-based receiver selective opening CCA secure PKE from standard computational assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 140–159, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany.

HKOZ16. Viet Tung Hoang, Jonathan Katz, Adam O'Neill, and Mohammad Zaheri. Selective-opening security in the presence of randomness failures. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 278–306, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

HLC+19. Zhengan Huang, Junzuo Lai, Wenbin Chen, Man Ho Au, Zhen Peng, and Jin Li. Simulation-based selective opening security for receivers under chosen-ciphertext attacks. *Designs, Codes and Cryptography*, 87(6):1345–1371, 2019.

HLOV11. Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 70–88, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

HLQ13. Zhengan Huang, Shengli Liu, and Baodong Qin. Sender-equivocable encryption schemes secure against chosen-ciphertext attacks revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 369–385, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany.

Hof12. Dennis Hofheinz. All-but-many lossy trapdoor functions. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 209–227, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

Hof13. Dennis Hofheinz. Circular chosen-ciphertext security with compact ciphertexts. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 520–536, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

HP16. Felix Heuer and Bertram Poettering. Selective opening security from simulatable data encapsulation. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 248–277, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

HPW15. Carmit Hazay, Arpita Patra, and Bogdan Warinschi. Selective opening security for receivers. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 443–469, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.

HRW16. Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 121–145, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

JL00. Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

JLL16. Dingding Jia, Xianhui Lu, and Bao Li. Receiver selective opening security from indistinguishability obfuscation. In Orr Dunkelman and Somitra Kumar Sanadhya, editors, *INDOCRYPT 2016*, volume 10095 of *LNCS*, pages 393–410, Kolkata, India, December 11–14, 2016. Springer, Heidelberg, Germany.

JLL17. Dingding Jia, Xianhui Lu, and Bao Li. Constructions secure against receiver selective opening and chosen ciphertext attacks. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 417–431, San Francisco, CA, USA, February 14–17, 2017. Springer, Heidelberg, Germany.

KP06. Sébastien Kunz-Jacques and David Pointcheval. About the security of MTI/C0 and MQV. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 156–172, Maiori, Italy, September 6–8, 2006. Springer, Heidelberg, Germany.

KR00. Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS 2000*, San Diego, CA, USA, February 2–4, 2000. The Internet Society.

LDL+14. Junzuo Lai, Robert H. Deng, Shengli Liu, Jian Weng, and Yunlei Zhao. Identity-based encryption secure against selective opening chosen-ciphertext attack. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 77–92, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

LLHG18. Lin Lyu, Shengli Liu, Shuai Han, and Dawu Gu. Tightly SIM-SO-CCA secure public key encryption from standard assumptions. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 62–92, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

LP15. Shengli Liu and Kenneth G. Paterson. Simulation-based selective opening CCA security for PKE from key encapsulation mechanisms. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 3–26, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany.

LQ19. Benoît Libert and Chen Qian. Lossy algebraic filters with short tags. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 34–65, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany.

LSSS17a. Benoît Libert, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 332–364, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

LSSS17b. Benoit Libert, Amin Sakzad, Damien Stehle, and Ron Steinfeld. All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. Cryptology ePrint Archive, Report 2017/876, 2017. `http://eprint.iacr.org/2017/876`.

LSW10. Allison B. Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *2010 IEEE Symposium on Security and Privacy*, pages 273–285, Berkeley/Oakland, CA, USA, May 16–19, 2010. IEEE Computer Society Press.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

MR04. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

PW08. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

Supporting Material

A Proof of Theorem 2

Proof. We prove the evasiveness via a sequence of 2 games. In the initial game of G_0, the adversary proceeds as in the real game. And in the next game G_1, when the adversary proposes tags to the VER oracle, it rejects those generate the same chameleon hash as that answered by the lossy tag oracle $\text{LOSS}(\mathsf{tk}, \cdot)$. We use bad_i to denote the event that $\mathcal{A}$ manages to output a non-injective tag in G_i for $i = 0, 1$. It is obvious that $\varepsilon = \Pr[\mathsf{bad}_0]$.

Lemma 15. *If the* CH *is* (t_1, ε_1)*-collision-resistant, the R-wD3DH2 problem is* (t_2, ε_2)*-hard, then* $|\Pr[\mathsf{bad}_0] - \Pr[\mathsf{bad}_1]| \leq \varepsilon_1 + n\varepsilon_2$.

Proof. We use bad_h to denote the event that $\mathcal{A}$ outputs a tag $t = (t_a, (\{B_i, D_i, E_i\}_{i \in [n]}, \mathsf{R}_{\mathsf{CH}}))$ with a hash τ the same as that produced by the LOSS before. It is straightforward that

$$|\Pr[\mathsf{bad}_0] - \Pr[\mathsf{bad}_1]| \leq \Pr[\mathsf{bad}_h \text{ in } \mathsf{G}_1].$$

As in both G_0 and G_1, the chameleon hash trapdoor $\mathsf{sk}_{\mathsf{CH}}$ is used to answer $\text{LOSS}(\mathsf{tk}, \cdot)$ queries, which makes it difficult to use the collision-resistant property of the chameleon hash to bound $\Pr[\mathsf{bad}_h]$ directly. To solve this problem, we use the "deferred analysis" proof technique [Hof12]. That is, we introduce two intermediate games $\mathsf{G}_{1'}$, $\mathsf{G}_{2'}$ defined as follows:

$\mathsf{G}_{1'}$**:** The same as G_1, except that the VER oracle only checks the freshness of τ computed from the proposed tags.

$\mathsf{G}_{2'}$**:** The same as $\mathsf{G}_{1'}$, except that the LOSS oracle returns random tags instead of lossy tags.

It is obvious that the probability of bad_h is the same in G_1 and $\mathsf{G}_{1'}$. Here as we do not need any secret information to answer VER queries, then we can employ the indistinguishable proof and get

$$|\Pr[\mathsf{bad}_h \text{ in } \mathsf{G}_{1'}] - \Pr[\mathsf{bad}_h \text{ in } \mathsf{G}_{2'}]| \leq n\varepsilon_2.$$

Now in $\mathsf{G}_{2'}$, $\mathsf{sk}_{\mathsf{CH}}$ is no longer used and we can use collision-resistant property to bound $\Pr[\mathsf{bad}_h \text{ in } \mathsf{G}_{2'}] \leq \varepsilon_1$. ☐

Next we bound the bad_1 by the 2-3-CDH assumption.

$$\Pr[\mathsf{bad}_1] \leq O(n \cdot q_{\mathsf{eva}} \cdot q_{\mathsf{ver}} \cdot \sqrt{L})\varepsilon_3.$$

On receiving a 2-3-CDH challenge $(g_1, g_2, g_1^a, g_1^b, g_2^a, g_2^b)$, the reduction $\mathcal{B}$'s task is to produce a pair $(g_2^r, g_2^{r \cdot ab})$ with $r \neq 0$. To do this, $\mathcal{B}$ proceeds as follows:

To generate ek, $\mathcal{B}$ firstly generates $(\mathsf{pk}_{\mathsf{CH}}, \mathsf{sk}_{\mathsf{CH}}) \leftarrow \mathsf{CH.gen}$ as in the real game, then it picks a random $k \in [n]$ and $r_i, s_i \xleftarrow{\$} \mathbb{Z}_q$ and computes $R_i := g_1^{r_i}$, $S_i := g_1^{s_i}$

for $i \in [n] \setminus k$, it also sets $R_k := g_1^a$ and $S_k := g_1^b$, in this way it implicitly sets $r_k := a$ and $s_k := b$. Note that $\mathcal{B}$ can compute U_{ij} for $i \neq j$. Then it picks $\overline{w}_0, ..., \overline{w}_L \xleftarrow{\$} \{-1, 0, 1\}$, $\widetilde{w}_0, ..., \widetilde{w}_L \xleftarrow{\$} \mathbb{Z}_q$, computes $W_{\iota,i} := g_\iota^{a \cdot \overline{w}_i} g_\iota^{\widetilde{w}_i}$ for $\iota = 1, 2, i \in [0, L]$. In this way it implicitly sets $w_i = a\overline{w}_i + \widetilde{w}_i$ for $i \in [0, L]$. It is obvious that ek is distributed exactly as in the real game.

When the adversary proposes the query t_a to the LOSS, the core tag part t_c is answered as follows:

1. $\mathcal{B}$ samples a random τ in the range of CH. For $i \neq k$, $\mathcal{B}$ picks $b_i, \rho_i \xleftarrow{\$} \mathbb{Z}_q$ and sets
$$B_i := g_2^{b_i},\ D_i := g_2^{r_i s_i b_i} \cdot H_{\mathsf{G}_2}(\tau)^{\rho_i},\ E_i := g_2^{\rho_i}.$$
2. For $i = k$, $\mathcal{B}$ computes $\overline{w}_\tau := \overline{w}_0 + \sum_{i=1}^L \overline{w}_i \tau[i]$, $\widetilde{w}_\tau := \widetilde{w}_0 + \sum_{i=1}^L \widetilde{w}_i \tau[i]$ and aborts if $\overline{w}_\tau = 0$. Otherwise, it picks $b_k \xleftarrow{\$} \mathbb{Z}_q$ and $\overline{\rho}_k \xleftarrow{\$} \mathbb{Z}_q$ and sets
$$B_k := g_2^{b_k},\ D_k := (g_2^b)^{-\frac{\widetilde{w}_\tau b_k}{\overline{w}_\tau}} \cdot H_{\mathsf{G}_2}(\tau)^{\overline{\rho}_k},\ E_k := (g_2^b)^{-\frac{b_k}{\overline{w}_\tau}} g_2^{\overline{\rho}_k}.$$
where it implicitly defines $\rho_k = \overline{\rho}_k - \frac{bb_k}{\overline{w}_\tau}$.
3. Finally, $\mathcal{B}$ uses the trapdoor $\mathsf{sk}_{\mathsf{CH}}$ for the chameleon hash to find coins R_{CH} such that $\tau = \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, t_a, \{B_i, D_i, E_i\}_{i=1}^n; \mathsf{R}_{\mathsf{CH}})$.

When the adversary finally outputs q_{ver} tags, $\mathcal{B}$ picks a random one as the non-injective tag $(t_a^*, t_c^* = (\{B_i, D_i, E_i\}_{i\in[n]}, \mathsf{R}_{\mathsf{CH}}))$, $\mathcal{B}$ computes $\tau^* := \mathsf{CH.eval}(\mathsf{pk}_{\mathsf{CH}}, (t_a, \{B_i, D_i, E_i\}_{i\in[n]}); \mathsf{R}_{\mathsf{CH}})$ and $\overline{w}_{\tau^*} := \overline{w}_0 + \sum_{i=1}^L \overline{w}_i \tau^*[i]$. If $\overline{w}_{\tau^*} \neq 0$, $\mathcal{B}$ aborts. Otherwise, with probability $1/n$ it should hold that

$$\begin{aligned} D_k^* &= g_2^{ab\cdot b_k^*} \cdot H_{\mathsf{G}_2}(\tau^*)^{\rho_k^*} \\ &= g_2^{ab\cdot b_k^*} g_2^{(a\overline{w}_{\tau^*} + \widetilde{w}_{\tau^*})\cdot \rho_k^*} \\ &= g_2^{ab\cdot b_k^*} \cdot E_k^{*\,\widetilde{w}_{\tau^*}}, \end{aligned}$$

where $E_k^* = g_2^{\rho_k^*}$. Finally, $\mathcal{B}$ outputs $(B_k^*, \frac{D_k^*}{E_k^{*\,\widetilde{w}_{\tau^*}}})$.

Clearly, if $\mathcal{B}$ does not abort, its output is a valid 2-3-CDH answer. By applying known results on programmable hash functions [HK08], the non-abort probability is lower bounded by $O(q_{\mathsf{eva}} \cdot \sqrt{L})$. Hence $\Pr[\mathsf{bad}_1] \leq O(n \cdot q_{\mathsf{eva}} \cdot q_{\mathsf{ver}} \cdot \sqrt{L})\varepsilon_3$. □

Remark 2. Note that here we can achieve the strong evasiveness property, which means that even for an old t_a, the adversary will not be able to produce a fresh t_c such that $t := (t_a, t_c)$ is lossy.

B The Underlying MAC Construction

In this part, to better illustrate the evasiveness proof, we recall the message authentication code (MAC) used by Libert and Qian [LQ19], which is a variant of a MAC construction due to Blazy, Kiltz and Pan [BKP14]. The MAC of [LQ19] adds a duplicate copy of the secret key and also an extra group element h, it publics a linear combination of secret key, then with the help of $\log_g(h)$, anyone can perform the verification. Then reductions for one-verification query and multi-verification query are the same, so it only needs to guess every bit of the verification query once, thus ensures tight unforgeable property. They also introduced an additional randomizer r, which makes the MAC compatible with the indistinguishability of the constructed ABM-LTF. The MAC is described formally as follows.

$\mathsf{MAC.Gen}$**:** Choose bilinear groups $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_T$ of prime order q with asymmetric pairing $e : \mathsf{G}_1 \times \mathsf{G}_2 \to \mathsf{G}_T$. Choose random elements $g_1 \xleftarrow{\$} \mathsf{G}_1, g_2 \xleftarrow{\$} \mathsf{G}_2$ and denote $g_T := e(g_1, g_2)$.

1. Pick random $\alpha, \beta, \theta, \eta \xleftarrow{\$} \mathbb{Z}_q$ and compute $h_1 := g_1^\eta,\ h_2 := g_2^\eta,\ A := g_1^\theta$ and $R := g_1^{\alpha+\theta\beta}$.
2. For each $\mu \in \{0, 1\}$, choose vectors $\mathbf{x}_\mu := (x_{1,\mu}, \ldots, x_{L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$ and $\mathbf{y}_\mu := (y_{1,\mu}, \ldots, y_{L,\mu}) \xleftarrow{\$} \mathbb{Z}_q^L$, compute $\mathbf{z}_\mu := \mathbf{x}_\mu + \theta\mathbf{y}_\mu$ and $\mathbf{Z}_\mu := g_1^{\mathbf{z}_\mu} = (g_1^{z_{1,\mu}}, \ldots, g_1^{z_{L,\mu}})$.
3. $\mathsf{sk}_{\mathsf{MAC}} := (\alpha, \beta, \mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1, \eta)$. $\mathsf{pp} := (g_1, g_2, h_1, h_2, A, R, \mathbf{Z}_0, \mathbf{Z}_1)$.

$\mathsf{MAC.Tag}(\mathsf{pp}, \mathsf{sk}_{\mathsf{MAC}}, \mathsf{m})$**:** To compute the MAC value $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ for a message $\mathsf{m} \in \{0, 1\}^L$: pick $b, \rho \xleftarrow{\$} \mathbb{Z}_q$, compute $\sigma_1 := g_2^b$, $\sigma_2 := h_2^{\alpha b} g_2^{\rho x_{\mathsf{m}}}$, $\sigma_3 := h_2^{\beta b} g_2^{\rho y_{\mathsf{m}}}$ and $\sigma_4 := g_2^\rho$, where $x_{\mathsf{m}} := \sum_{k=1}^L x_{k,\mathsf{m}_k}$, $y_{\mathsf{m}} := \sum_{k=1}^L y_{k,\mathsf{m}_k}$.

$\mathsf{MAC.Ver}$**:** Accept if the following equality holds, and reject otherwise.

$$e(g_1, \sigma_2) \cdot e(A, \sigma_3) = e(R, \sigma_1)^\eta \cdot e(Z_{\mathsf{m}}, \sigma_4),$$

where $Z_{\mathsf{m}} := \Pi_{k=1}^L g_1^{z_{k,\mathsf{m}_k}}$.

Lemma 16 (Lemma 4 in [LQ19][6]). *If the DDH1 problem is (t_1, ε_1)-hard, the DDH2 problem is (t_2, ε_2)-hard, then the above MAC is $(q_{\mathsf{ver}}, t_{\mathcal{A}}, \varepsilon)$-unforgeable, where $t_1 \approx t_2 \approx t_{\mathcal{A}} + \mathsf{poly}(\lambda)$, and $\varepsilon \le 2L \cdot \varepsilon_1 + \varepsilon_2$.*

C Proof of Theorem 5

Proof. For any adversary $\mathcal{A}$ runs in the real world, we construct a simulator $\mathcal{S}$ that runs in the ideal world, interacts with $\mathcal{A}$ as shown in the following and outputs $\mathcal{A}$s output, then we prove the outputs are indistinguishable.

- To initialize the game, $\mathcal{S}$ invokes the $\mathsf{LKeygen}$ algorithm and returns pk_l to the adversary $\mathcal{A}$.
- When $\mathcal{A}$ issues the encryption query with dist that indicates a distribution of N related messages, $\mathcal{S}$ transfers this query to its challenger. Then $\mathcal{S}$ picks $(\mathsf{m}_1, \ldots, \mathsf{m}_N) \xleftarrow{\$} \{0, 1\}^{\ell \times N}$, chooses $(x_1, \ldots, x_N) \leftarrow D_x^N$, and for $\xi \in [N]$, it runs the Lenc algorithm to generate $\mathsf{c}^\xi \leftarrow \mathsf{Lenc}(\mathsf{pk}_l, \mathsf{tk}, \mathsf{m}_\xi)$ with x_ξ.

[6] Note that the unforgeable property holds even when part of the secret key η is given to the adversary. We will use this property in the later proof.

– When $\mathcal{A}$ makes decryption queries with $\mathsf{c} = (y_1, y_2, t_c, y_3, y_4)$, $\mathcal{S}$ proceeds as follows:
  1. Test whether $t = (t_a, t_c)$ is injective with ik_2 generated by $\mathsf{LKeygen}$, and abort if not.
  2. Use ik_2 to invert y_2 to get x, compute $(k_1, k_2) := h(x)$.
  3. Verify if $y_1 = \mathsf{LTF.Eval}(\mathsf{ek}_1, x)$, $y_2 = \mathsf{ABM.Eval}(\mathsf{ek}_2, t, x)$ and $\mathsf{MAC.ver}(k_2, (y_2, y_3), y_4) = 1$, abort if any of the equalities does not hold or $x \notin \mathsf{DomD}$.
  4. Compute $\mathsf{m} := y_3 \oplus k_1$, and return m to $\mathcal{A}$.

– When $\mathcal{A}$ issues corruption queries with a set $I \subset [N]$, $\mathcal{S}$ transfers this query to its challenger and receives $(\mathsf{m}'_\xi)_{\xi \in I}$. Then $\mathcal{S}$ invokes the Opener algorithm to explain c^ξ to m'_ξ with randomness x'_ξ. Since t_c^ξ is pseudorandom, one can also explain t_c^ξ as a random tag efficiently.

To prove the indistinguishability of $\mathcal{S}$'s output and $\mathcal{A}$'s output, we proceed via a sequence of games.

G_0**:** The real $\mathsf{SIM\text{-}SO\text{-}CCA}$ security game.

G_1**:** Modify the generation of the challenge ciphertext. Instead of choosing random $t_c^\xi \xleftarrow{\$} \mathcal{T}_c$, generate $t_c^\xi \xleftarrow{\$} \mathsf{ABM.LGen}(\mathsf{tk}, t_a^\xi)$. And in the corruption phase, explain t_c^ξ as a random tag with the Resam algorithm. The difference of G_0 and G_1 can be bounded by the indistinguishability of the $\mathsf{ABM\text{-}LTF}$.

G_2**:** Modify the decryption oracle, reject queries with non-injective tags. We use event E to denote the event that a decryption query $\mathsf{c} = (y_1, y_2, t_c, y_3, y_4)$ corresponds to a non-injective tag. It is obvious that G_1 and G_2 are the same as long as E does not happen. We divide E in the following three cases:

E_1: $(y_1, t_c) \notin \{(y_1^1, t_c^1), \ldots, (y_1^N, t_c^N)\}$. The probability of this event can be bounded according to the evasiveness property of $\mathsf{ABM\text{-}LTF}$.

E_2: $(y_1, t_c) \in \{(y_1^1, t_c^1), \ldots, (y_1^N, t_c^N)\}$ but $(y_1, t_c, y_2) \notin \{(y_1^1, t_c^1, y_2^1), \ldots, (y_1^N, t_c^N, y_2^N)\}$. In this case, it indicates that $(x := \mathsf{LTF.Invert}(\mathsf{ik}_1, y_1), t_c) \in \{(x^{(1)}, t_c^{(1)}), \ldots, (x^{(N)}, t_c^{(N)})\}$ except with negligible probability. Since $\mathsf{ABM.Eval}$ is a deterministic function on x and (t_a, t_c), a modified y_2 will certainly be rejected. Hence the probability of E_2 happens is 0.

E_3: $(y_1, t_c, y_2) \in \{(y_1^1, t_c^1, y_2^1), \ldots, (y_1^N, t_c^N, y_2^N)\}$. In this case, y_3 must be different, since $\mathsf{MAC.eval}$ is a deterministic algorithm on k_1, y_2, y_3. And when y_3 is changed, the probability of E_3 can be bounded by $N\varepsilon_4$.

G_3**:** Instead of using the trapdoor for LTF to answer decryption queries, use the inversion key of $\mathsf{ABM\text{-}LTF}$ to answer decryption queries as in the simulated game. Since in G_2 all decryption queries correspond to injective tags, the inversion result with the $\mathsf{ABM.Inver}$ will be the same as that with the $\mathsf{LTF.Invert}$ algorithm.

G_4**:** Modify the generation of ek_1 to $\mathsf{ek}_1 \xleftarrow{\$} \mathsf{LTF.Lgen}(\lambda)$. Since here we do not need the inversion key of LTF any more, the difference of G_3 and G_4 is bounded by the indistinguishability of LTF.

G_5**:** Modify the key generation phase to the $\mathsf{LKeygen}$ algorithm. According to the requirement of $\mathsf{LKeygen}$ algorithm, G_4 and G_5 have the same distribution.

G_6**:** Modify the generation of the challenge ciphertext. Instead of firstly picking $(\mathsf{m}_1, \ldots, \mathsf{m}_N) \leftarrow \mathsf{dist}$ and encrypting these messages, pick random $(\mathsf{m}_1, \ldots, \mathsf{m}_N) \xleftarrow{\$} \{0,1\}^{\ell \times N}$, and encrypt these messages as in the simulated game. In the corruption phase, pick $(\mathsf{m}_1, \ldots, \mathsf{m}_N) \leftarrow \mathsf{dist}$ and answer according to the Opener algorithm. From the requirement of the Opener, G_5 and G_6 proceeds statistically close. And it is obvious that G_6 proceeds exactly as the simulator does in the ideal world.

$\square$