How to Watermark Cryptographic Functions

Ryo Nishimaki [1]

[1] Secure Platform Laboratories, NTT Corporation, Japan
{nishimaki.ryo}@lab.ntt.co.jp

Abstract

We introduce a notion of watermarking for cryptographic functions and propose a concrete scheme for watermarking cryptographic functions. Informally speaking, a digital watermarking scheme for cryptographic functions embeds information, called a *mark*, into functions such as one-way functions and decryption functions of public-key encryption. There are two basic requirements for watermarking schemes.

1. A mark-embedded function must be functionally equivalent to the original function.
2. It must be difficult for adversaries to remove the embedded mark without damaging the original functionality.

In spite of its importance and usefulness, there have only been a few theoretical works on watermarking for functions (or programs). Furthermore, we do not have rigorous definitions of watermarking for cryptographic functions and concrete constructions.

To solve the above problem, we introduce a notion of watermarking for cryptographic functions and define its security. Furthermore, we present a lossy trapdoor function (LTF) based on the decisional linear (DLIN) problem and a watermarking scheme for the LTF. Our watermarking scheme is secure under the DLIN assumption in the standard model. We use techniques of dual system encryption and dual pairing vector spaces (DPVS) to construct our watermarking scheme. This is a new application of DPVS.

Keywords: Program Watermarking, Lossy Trapdoor Functions , Dual Pairing Vector Space

Contents

1 Introduction

1.1 Background

Digital watermarking is a technology that enables us to embed information, called a "mark", into digital objects such as images, movies, and audio files. Such marks should be detected by using some procedure. There are two main properties of digital watermarking. The first is that the appearance (or functionality) of marked objects is almost the same as that of the original objects The second is that removing embedded marks without destroying the object is difficult. A main application of watermarking is protecting copyright. We can trace and identify owners of digital content by detecting watermarks. For example, if we find a potentially guilty user and illegally copied digital content, we can detect a watermark and identify the owner who distributed the illegal copy.

Most watermarking methods have been designed for perceptual objects, such as images. Only a few studies have focused on watermarking for non-perceptual objects (e.g., software or programs). Software is quite common digital content and can be easily copied. Software piracy is a serious problem today. If illegally copied software is distributed, profits of software companies decrease. Watermarking for programs is one of tools to solve the problem and has very useful, attractive, and practical applications. However, they are little understood. We briefly explain related studies on program watermarking below.

Naccache, Shamir, and Stern introduced the notion of copyrighted functions and proposed a method for tracking different copies of functionally equivalent algorithms containing a sort of "marks" [NSS99]. A copyrighted function is drawn from a keyed function family (this key plays a role of marks). The security of the protocol guarantees that no adversary can output a functionally equivalent function with a new key even if many keyed functions are given. This is related to watermarking schemes for programs (functions), but their security definition is a bit weak and not sufficient for program watermarking because copyrighted functions do not guarantee that embedded marks are not removed.

Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang considered the notion of software watermarking (program watermarking) from a cryptographic point of view in their seminal work [BGI+01, BGI+12]. They proposed a formalization of software watermarking and its security definition. The definition is simulation-based security and strong. They gave an impossibility result for general-purpose program watermarking by using impossibility results of general-purpose program obfuscation [BGI+01, BGI+12]. "General-purpose" means that program watermarking and obfuscation can be applied to *any* program. Their security requirements cannot be achieved, so they leave positive theoretical results about watermarking (achieving concrete constructions for specific function families by using a game-based security definition) as an open problem.

Yoshida and Fujiwara introduced the notion of *watermarking for cryptographic data* and a concrete scheme for signatures [YF11]. Their idea is very exciting, but they did not propose a formal security definition of watermarking for cryptographic data and their scheme is not provably secure. They claim that the security of their scheme is based on the *vector decomposition (VD)* problem, which was introduced by Yoshida, Mitsunari, and Fujiwara [YMF10], but their proof is heuristic, that is, they did not show a reduction.

Hopper, Molnar, and Wagner proposed a rigorous complexity-theoretic (game-based) definition of security for watermarking schemes. Their definition seems to be very useful, but they focused on watermarking for only *perceptual* objects [HMW07]. They gave no concrete construction that satisfies their security definition.

1.2 Motivations and Applications

As explained in the previous section, there is no watermarking scheme for (cryptographic) functions[1] that is provably secure in a complexity-theoretic definition of security. Copyrighted functions by Naccache *et al.* are provably secure based on the factoring assumption, but their definition of security is weaker than that of watermarking, and their construction can only embed a *bounded* number of distinct marks [NSS99]. Before we introduce our contribution, we present several applications of watermarking to explain motivations.

Traceable cryptographic primitives. One application of watermarking for cryptographic functions (we often call it cryptographic watermarking) is constructing various *traceable cryptographic primitives*. If we have a watermarking scheme for cryptographic functions, for example, trapdoor one-way functions, collision-resistant hash functions (CRHF), and decryption functions, we can construct a variety of traceable primitives or copyrighted cryptographic primitives since private-key encryption, public-key encryption (PKE), digital signatures, and so on are constructed from trapdoor one-way functions and often use CRHFs in their algorithms.

[1]We consider functions as a kind of program.

As pointed out by Naccache *et al.* , watermarked functions have the following applications [NSS99]:

- We can produce software or programs that generates ciphertexts of the Feistel cipher based on a one-way function [LR88], signatures of Rompel's signature scheme [Rom90], or decrypted values of ciphertexts under PKE schemes based on a trapdoor one-way function. By watermarking the underlying one-way function, if a malicious user illegally generate copies of such software and distributes them, then a company that sold the software can trace them and identify the guilty users.
- A company can sell MAC-functions based on watermarked one-way functions to users for a log-in system on the Internet. The company records user IDs and marked functions in a database. Users can use the MAC-functions to log-in to a member web site without revealing their identity since all marked functions are functionally equivalent. However, if a malicious user distributes an illegal copy and it is discovered, then the company can identify the guilty user identity by detecting an embedded mark.

Black-box traitor tracing. Kiayias and Yung proposed a method of constructing black-box traitors tracing schemes from copyrighted PKE functions [KY02]. When we broadcast digital content to a set of legitimate subscribers, we can use broadcast encryption schemes. If some of the subscribers leak partial information about their decryption keys to a pirate, who is a malicious user in broadcast encryption systems, then the pirate may be able to construct a pirate-decoder. That is, the pirate may access to the content though s/he is not a subscriber. Traitor tracing enables us to identify such malicious subscribers called traitor [CFN94]. Our cryptographic watermarking scheme can be seen as a generalized notion of copyrighted functions and our construction is based on identity-based encryption (IBE) schemes whose private keys for identities are marked (these are copyrighted decryption functions of PKE), so our construction technique can be used to construct *black-box traitor tracing* schemes and it has a quite powerful application.

Theoretical treatment of watermarking. There are many heuristic methods for software watermarking but there have only been a few studies that theoretically and rigorously treat the problem in spite of its importance. Functions can be seen as a kind of software (and program) and a large amount of software uses cryptographic functions, especially in a broadcast system, users must use software with decryption functions to view content. We believe that our scheme for watermarking for cryptographic functions is an important step toward constructing practical software watermarking.

1.3 Our Contributions and Construction Ideas

To solve problems explained in Section 1.1, we introduce the notion of watermarking for cryptographic functions, define a game-based security definition of them, and propose a concrete construction. Our watermarking scheme is provably secure under the decisional linear (DLIN) assumption. To the best of our knowledge, this is the first provably secure watermarking scheme for functions (programs) in terms of theoretical cryptography and solves the open problem proposed by Barak *et al.* [BGI$^+$01, BGI$^+$12].

Our security notion is based on the notion of strong watermarking introduced by Hopper *et al.* [HMW07], but details are different since we focus on the definition for cryptographic functions. Their definition takes into account only perceptual objects and they modeled the notion of similarity by a perceptual metric space on objects that measures the distance between objects. Therefore, to construct watermarking schemes for cryptographic functions, we need to modify their definition. We define the similarity by preserving functionality. If, for some inputs, a marked function outputs the same outputs as those of an original function for the inputs, then we say that the marked function is similar to the original function. Watermarking schemes should guarantee that no adversary can generate a function which is similar to a marked function for some inputs but unmarked. That is, no adversary can remove embedded marks without destroying underlying functionality. This is a primary difference from copyrighted functions.

We propose a watermarking scheme for lossy trapdoor functions (LTFs) [PW08]. LTFs are powerful cryptographic functions. They imply standard trapdoor one-way functions, oblivious transfers, CRHFs, and secure PKE schemes against adaptive chosen ciphertext attacks (CCA) [PW08]. The watermarking scheme consists of four algorithms, key generation, mark, detect, and remove algorithms. Marked function indices are functionally equivalent to the original ones, that is, for any input, outputs of marked functions are the same as those of the original function. We call this perfect functionality preserving property. The construction can be used to construct an IBE scheme that can generate marked private keys for identities and marked signatures since our LTFs are based on IBE schemes, as explained in the next paragraph. That is, we can construct decryption algorithms in which watermarks can be embedded.

Key Techniques and Ideas Behind Our Construction. Our construction is based on the dual pairing vector space (DPVS) proposed by Okamoto and Takashima [OT09, OT10, OT12]. We can use the IBE scheme of Okamoto and Takashima [OT12] (which is a special case of their inner-product predicate encryption (IPE) scheme) and that of Lewko [Lew12] to construct LTFs. Loosely speaking, LTFs are constructed from homomorphic encryption schemes, and the IBE schemes of Okamoto-Takashima and Lewko are homomorphic. There are many other homomorphic encryption schemes but we selected Okamoto-Takashima and Lewko IBE schemes because they are constructed by DPVS and the dual system encryption methodology introduced by Waters [Wat09]. The methodology is a key technique to achieve a watermarking scheme. In this paper, we write only about the Lewko IBE scheme.

First, we explain how we use the dual system encryption methodology to construct watermarking schemes. We apply the dual system encryption technique to not only security proofs but also *constructions of cryptographic primitives*. In the dual system encryption, there are two types for ciphertexts and private-keys respectively. The first type is called *normal* ciphertext and key and the second type is called *semi-functional* ciphertext and key, respectively. They have the following properties. Semi-functional ciphertexts can be decrypted using normal keys and normal ciphertext can be decrypted using semi-functional keys. However, semi-functional ciphertexts cannot be decrypted using semi-functional keys. Normal ciphertext and key are computationally indistinguishable from semi-functional ciphertext and key, respectively. In most cases, function indices of LTFs consist of *ciphertexts of homomorphic encryption* [FGK+10, HO12, PW08], so, intuitively speaking, if we can construct a function index by using not only (normal) ciphertexts but also semi-functional keys, then the function index is functionally equivalent to a function index generated by (normal ciphertexts and) normal keys as long as normal ciphertexts are used. Moreover, if we use semi-functional ciphertexts, we can determine whether a function index is generated by semi-functional keys or not since semi-functional ciphertexts cannot be decrypted using a semi-functional key. Thus, a function index that consists of semi-functional keys can be seen as a marked index and semi-functional ciphertexts can be used in a detection algorithm of a watermarking scheme. This is the main idea. Note that our construction technique can be used to construct an IBE scheme whose private keys can be marked because our LTFs are based on such an IBE scheme.

Next, we explain how we construct watermarking scheme by using DPVS. DPVS is linear space defined over bilinear groups and a vector consists of group elements [OT09, OT10]. One of key features of DPVS is that if we conceal (i.e., do not publish) some basis of a subspace then we can set a hidden linear subspace. A pair of dual orthonormal bases over groups are denoted by $\boldsymbol{B}$ and $\boldsymbol{B}^*$. They are generated by a random linear transformation matrix that consists of elements in a finite field. We use a hidden linear subspace spanned by a subset of $\boldsymbol{B}$ and $\boldsymbol{B}^*$ for semi-functional ciphertext and key as Okamoto-Takashima and Lewko IBE schemes [Lew12, OT10, OT12]. We denote the subset by $\widehat{\boldsymbol{B}} \subset \boldsymbol{B}$, $\widehat{\boldsymbol{B}}^* \subset \boldsymbol{B}^*$, respectively. A hidden linear subspace for semi-functional ciphertext and key can be used as a detect key and a mark key of our watermarking scheme, respectively. Thus, we can embed "marks" into the hidden linear subspace and they are indistinguishable from non-marked objects because the decisional subspace problem is believed to be hard [OT08, OT10]. Informally speaking, the decisional subspace problem is determining whether a given vector is spanned by $\boldsymbol{B}$ (resp, $\boldsymbol{B}^*$) or $\boldsymbol{B} \setminus \widehat{\boldsymbol{B}}$ (resp, $\boldsymbol{B}^* \setminus \widehat{\boldsymbol{B}^*}$).

Okamoto and Takashima introduced complexity problems based on the DLIN problem to prove the security of their scheme [OT10, OT12] and these problems are deeply related to the VD problem [YMF10] and the decisional subspace problem. The VD problem says that it is difficult to decompose a vector in DPVS into a vector spanned by bases of a subspace. Lewko also introduced the subspace assumption [Lew12], which is implied by the DLIN assumption and highly related to the decisional subspace assumption introduced by Okamoto and Takashima [OT08] and the VD problem. All assumptions introduced by Okamoto-Takashima [OT10, OT12] and Lewko [Lew12] are implied by the standard DLIN assumption.

If we can decompose a vector in DPVS into each linearly independent vector, then we can convert semi-functional ciphertext and key into normal ciphertext and key by eliminating elements in hidden linear subspaces, that is, we can remove an embedded mark from a marked function index. Galbraith and Verheul and Yoshida, Mitsunari, and Fujiwara argued that the VD problem is related to computational Diffie-Hellman problem [GV08, YMF10]. It is believed that the VD problem is hard. Therefore, no adversary can remove marks of our watermarking scheme (this is a just intuition). However, we do not directly use the VD problem but the DLIN problem to prove the security of our scheme. On the other hand, if we have a linear transformation matrix behind dual orthonormal bases of DPVS, then we can easily solve the VD problem [OT08, OT10], that is, we can remove a mark if we have the matrix. Such an algorithm was proposed by Okamoto and Takashima [OT08].

Our construction is a new application of DPVS. DPVS has been used to construct fully secure functional encryption, IPE, IBE and attribute-based signature [Lew12, LOS+10, OT09, OT10, OT11, OT12], but to the best of our knowledge, a linear transformation matrix for dual orthonormal bases in DPVS has never been explicitly used for algorithms of cryptographic schemes. This is of independent interest.

On the Impossibility of Watermarking. Barak *et al.* showed that if there exists indistinguishability obfuscation (iO), then there is no program watermarking with *perfect* functionality preserving property [BGI+01, BGI+12]. Roughly speaking, if we apply iO to a marked program, then we can remove the mark since if we still detect the mark from the obfuscated marked program, then we can use it to distinguish an obfuscated marked program from an obfuscated *unmarked* program (indistinguishability holds for functionally equivalent programs). We use a different definition from that of Barak *et al.* , but the impossibility result holds in our setting since our watermarking scheme has perfect functionality preserving property. This does not contradict to our results because we restrict adversaries in a security game. The restriction forces adversaries to output a function in a specified format to win the security game. Adversaries cannot use iO under this restriction. Someone might think the restriction is strong, but we can say our construction is an alternative approach to achieve watermarking since Cohen, Holmgren, Nishimaki, Vaikuntanathan, and Wichs [CHN+16] proposed program watermarking based on iO. Candidate constructions of iO are known [GGH+13b, GMM+16, FRS16], but not only their underlying cryptographic tool called multilinear maps [GGH13a, CLT13, GGH15] but also iO constructions were attacked [CHL+15, CGH+15, CFL+16, HJ16, CLLT16a, MSZ16, CLLT16b, ADGM16, CGH16]. Thus, our construction that does not assume iO is still meaningful. See discussion in Section 3 for more details.

1.4 Organization of This Paper

In Section 2, we introduce some notations and known cryptographic definitions, tools, and techniques. In Section 3, we introduce our definition of watermarking for cryptographic functions. In Section 4, we propose a concrete instantiation of watermarking schemes for lossy trapdoor functions. In Section 5, we list a few concluding remarks and open issues.

2 Preliminaries

Notations and Conventions. For any $n \in \mathbb{N} \setminus \{0\}$, let $[n]$ be the set $\{1, \ldots, n\}$. When D is a random variable or distribution, $y \leftarrow D$ denote that y is randomly selected from D according to its distribution. If S is a set, then $x \leftarrow S$ denotes that x is uniformly selected from S. Let $y := z$ denote that y is set, defined or substituted by z. We say that function $f : \mathbb{N} \to \mathbb{R}$ is negligible in $\lambda \in \mathbb{N}$ if $f(\lambda) = \lambda^{-\omega(1)}$. Hereafter, we use $f \leq \mathsf{negl}(\lambda)$ to mean that f is negligible in λ. Bold face small and capital letters denote a vector and matrix element over $\mathbb{Z}_p$ or a group $\mathbb{G}$, e.g., $\boldsymbol{x}$ and $\boldsymbol{X}$ denote $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ (or $\in \mathbb{G}^n$) and $(x_{i,j})_{i,j \in [n]} \in \mathbb{Z}_p^{n \times n}$ (or $\in \mathbb{G}^{n \times n}$). For two vectors $\boldsymbol{x}$ and $\boldsymbol{v}$, $\langle \boldsymbol{x}, \boldsymbol{v} \rangle$ denotes the inner-product $\sum_{i=1}^{n} x_i v_i$. The transpose of matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{X}^{\mathsf{T}}$. We denote probabilistic polynomial-time by PPT.

Let $\mathcal{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ denote two ensembles of random variables indexed by $\lambda \in \mathbb{N}$. The statistical distance between two random variables X and Y over a countable set S is defined as $\Delta(X, Y) := \frac{1}{2} \sum_{\alpha \in S} |\Pr[X = \alpha] - \Pr[Y = \alpha]|$.

Definition 2.1. *We say that $\mathcal{X}$ and $\mathcal{Y}$ are statistically indistinguishable (We write $\mathcal{X} \overset{\mathsf{s}}{\approx} \mathcal{Y}$ to denote this) if*

$$\Delta(X_\lambda, Y_\lambda) \leq \mathsf{negl}(\lambda).$$

Definition 2.2. *We say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable (We write $\mathcal{X} \overset{\mathsf{c}}{\approx} \mathcal{Y}$ to denote this) if for all non-uniform PPT algorithm D,*

$$|\Pr[D(X_\lambda) = 1] - \Pr[D(Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

2.1 Cryptographic Bilinear Maps (or Pairings)

We consider cyclic groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p. A bilinear map is an efficient mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 = \mathbb{G}_T$ satisfying the following properties.

Bilinearity: For all $g \in \mathbb{G}_1$, $\hat{g} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab}$.

Non-degeneracy: If g and $\hat{g}$ generate $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, then $e(g, \hat{g}) \neq 1$.

If $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$, that is, both groups are the same, we call $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ symmetric pairing groups. Let $\mathcal{G}_{\mathsf{bmp}}$ be a standard parameter generation algorithm that takes as input a security parameter λ and outputs parameters $(p, \mathbb{G}, \mathbb{G}_T, e, g)$.

Bracket Notations. For ease of notation, we borrow a bracket notation introcuded by Lin and Vaikuntanathan [LV16] (and in many other papers). For $a, b, \alpha \in \mathbb{Z}_p$, $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_p^n$, we let

$$\begin{aligned}
[a] &:= g^a, [b]_T := e(g,g)^b, \\
[\boldsymbol{v}] &:= ([v_1], \cdots, [v_n]), \\
[\boldsymbol{v}] \oplus [\boldsymbol{w}] &:= [\boldsymbol{v} + \boldsymbol{w}] = ([v_1 + w_1], \ldots, [v_n + w_n]), \\
[\boldsymbol{v}] \ominus [\boldsymbol{w}] &:= [\boldsymbol{v} - \boldsymbol{w}] = ([v_1 - w_1], \ldots, [v_n - w_n]), \\
\alpha \odot [\boldsymbol{v}] &:= [\boldsymbol{v}]^\alpha := [\alpha \boldsymbol{v}] = ([\alpha v_1], \ldots, [\alpha v_n]), \\
\boldsymbol{e}([\boldsymbol{v}], [\boldsymbol{w}]) &:= \bigoplus_{i=1}^{n} e([v_i], [w_i]) = [\langle \boldsymbol{v}, \boldsymbol{w} \rangle]_T.
\end{aligned}$$

2.2 Function Family of Lossy Trapdoor Functions

Definition 2.3 (Lossy Trapdoor Functions [PW11]). *A lossy trapdoor function* LTF *with domain* D *consists of four polynomial-time algorithms having the following properties.*

Injective Key Generation: $\mathsf{LTF.IGen}$ *outputs* $(\mathsf{ek}, \mathsf{ik})$ *where* ek *and* ik *are an evaluation and an inversion key, respectively.*

Evaluation: *For* $X \in \mathsf{D}$, $\mathsf{LTF.Eval}_{\mathsf{ek}}(X)$ *outputs an image* $Y = f_{\mathsf{ek}}(X)$.

Inversion: $\mathsf{LTF.Invert}_{\mathsf{ik}}(Y)$ *outputs a pre-image* $X = f_{\mathsf{ik}}^{-1}(Y)$.

Lossy Key Generation: $\mathsf{LTF.LGen}$ *outputs* $(\mathsf{ek}', \perp)$ *where* ek' *is an evaluation key.*

Correctness: *For all* $(\mathsf{ek}, \mathsf{ik}) \leftarrow \mathsf{LTF.IGen}(1^\lambda)$, *and* $X \in \mathsf{D}$, *we have* $f_{\mathsf{ik}}^{-1}(f_{\mathsf{ek}}(X)) = X$.

Indistinguishability: *Let* λ *be a security parameter. For all PPT* $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{ltf},\mathcal{A}}(\lambda) := \left|\Pr[\mathcal{A}(1^\lambda, \{\mathsf{LTF.IGen}(1^\lambda)\}_1)] - \Pr[\mathcal{A}(1^\lambda, \{\mathsf{LTF.LGen}(1^\lambda)\}_1)]\right| < \mathsf{negl}(\lambda),$$

where $\{A\}_1$ *is the first output of algorithm* A.

Lossiness: *We say that* LTF *is* ℓ*-lossy if for all* $\mathsf{ek}' \leftarrow \mathsf{LTF.LGen}(1^\lambda)$, *the image set* $f_{\mathsf{ek}'}(\mathsf{D})$ *is of size at most* $|\mathsf{D}|/2^\ell$.

We define a function family of LTF, $\mathsf{LTF}_\lambda := \{\mathsf{LTF.Eval}_{\mathsf{ek}}(\cdot) | (\mathsf{ek}, \mathsf{ik}) \leftarrow \mathsf{LTF.Gen}(1^\lambda, b), b \in \{0,1\}\}$ where $\mathsf{LTF.Gen}(1^\lambda, 0) := \mathsf{LTF.IGen}(1^\lambda)$ and $\mathsf{LTF.Gen}(1^\lambda, 1) := \mathsf{LTF.LGen}(1^\lambda)$.

2.3 Dual Pairing Vector Space

The concept of dual pairing vector space is proposed by Okamoto and Takashima [OT09, OT10].

Definition 2.4. *"Dual pairing vector spaces (DPVS)"* $(p, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, \boldsymbol{e})$ *is constructed from a direct product of symmetric pairing groups* $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ *as follows.*

Vector space $\mathbb{V}$**:** *A vector space consists of* N *groups, i.e.,* $\mathbb{V} := \overbrace{\mathbb{G} \times \cdots \times \mathbb{G}}^{N}$, *whose element is expressed by* N*-dimensional vector* $[\boldsymbol{x}] := ([x_1], \ldots, [x_N])$ *where* $x_i \in \mathbb{Z}_p$ *for all* $i \in [N]$.

Canonical basis $\mathbb{A}$**:** *There is canonical basis* $\mathbb{A} := ([\boldsymbol{a}_1], \ldots, [\boldsymbol{a}_N])$ *of* $\mathbb{V}$, *where* $\boldsymbol{a}_1 := (1, 0, \ldots, 0), \boldsymbol{a}_2 := (0, 1, 0, \ldots, 0), \ldots, \boldsymbol{a}_N := (0, \ldots, 0, 1)$.

Pairing operation: *A pairing function* $\boldsymbol{e} : \mathbb{V} \times \mathbb{V} \to \mathbb{G}_T$ *is defined by* $\boldsymbol{e}([\boldsymbol{x}], [\boldsymbol{y}]) := \bigoplus_{i=1}^{N} e([x_i], [y_i]) \in \mathbb{G}_T$ *where* $\boldsymbol{x} := (x_1, \ldots, x_N) \in \mathbb{Z}_p^N$ *and* $\boldsymbol{y} := (y_1, \ldots, y_N) \in \mathbb{Z}_p^N$*. This is non-degenerate bilinear, i.e.,* $\boldsymbol{e}(s \odot [\boldsymbol{x}], t \odot [\boldsymbol{y}]) = st \odot \boldsymbol{e}(\boldsymbol{x}, \boldsymbol{y})$ *and if* $\boldsymbol{e}([\boldsymbol{x}], [\boldsymbol{y}]) = 1$ *for all* $[\boldsymbol{y}] \in \mathbb{V}$*, then* $\boldsymbol{x} = \boldsymbol{0}$*. For all* i *and* j*,* $\boldsymbol{e}([\boldsymbol{a}_i], [\boldsymbol{a}_j]) = [\delta_{i,j}]_T$ *where* $\delta_{i,j} = 1$ *if* $i = j$*, and* 0 *otherwise.*

DPVS also have linear transformations $\phi_{i,j}$ *on* $\mathbb{V}$ *s.t.* $\phi_{i,j}([\boldsymbol{a}_j]) = [\boldsymbol{a}_i]$ *and* $\phi_{i,j}([\boldsymbol{a}_k]) = [\boldsymbol{0}]$ *if* $k \neq j$*, which can be easily achieved by* $\phi_{i,j}([\boldsymbol{x}]) := (\overbrace{[0], \ldots, [0]}^{i-1}, [x_j], \overbrace{[0], \ldots, [0]}^{N-i})$ *where* $\boldsymbol{x} := (x_1, \ldots, x_N)$*. We call* $\phi_{i,j}$ *canonical maps.*

Dual orthonormal bases. Let $\mathsf{Dual}(\mathbb{Z}_p^n)$ an algorithm for generating dual orthonormal bases as follows.

$$\begin{aligned} \underline{\mathsf{Dual}(\mathbb{Z}_p^n):}\ & \text{chooses } \boldsymbol{b}_i, \boldsymbol{b}_j^* \in \mathbb{Z}_p^n, \psi \leftarrow \mathbb{Z}_p^* \text{ such that} \\ & \langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle = 0 \bmod p \text{ for } i \neq j, \\ & \langle \boldsymbol{b}_i, \boldsymbol{b}_i^* \rangle = \psi \bmod p \text{ for all } i \in [n] \\ & \text{outputs } \boldsymbol{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \text{ and } \boldsymbol{B}^* := (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*). \end{aligned}$$

We describe a parameter generation algorithm $\mathcal{G}_{\mathsf{dpvs}}(1^\lambda, n)$ for DPVS.

$$\begin{aligned} \underline{\mathcal{G}_{\mathsf{dpvs}}(1^\lambda, n):}\ & \text{generates } (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}_{\mathsf{bmp}}(1^\lambda), \\ & \mathsf{params}_{\mathbb{V}} := (p, \mathbb{V}, \mathbb{G}_T, e, g) \\ & (\boldsymbol{B}, \boldsymbol{B}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^n), \\ & [\boldsymbol{B}] := ([\boldsymbol{b}_1], \ldots, [\boldsymbol{b}_n]), \\ & [\boldsymbol{B}^*] := ([\boldsymbol{b}_1^*], \ldots, [\boldsymbol{b}_n^*]), \\ & \text{returns } (\mathsf{params}_{\mathbb{V}}, [\boldsymbol{B}], [\boldsymbol{B}^*]). \end{aligned}$$

Parameters $(\mathsf{params}_{\mathbb{V}}, [\boldsymbol{B}], [\boldsymbol{B}^*])$ defines a DPVS over a bilinear group. We consider $\mathsf{Dual}(\mathbb{Z}_p^n)$ implicitly outputs $\psi \in \mathbb{Z}_p^*$ though we omit it.

We briefly explain some important properties of a DPVS.

Intractable problem: A decisional problem in this approach is the decisional subspace problem [OT08]. It is to tell $[\boldsymbol{v}] = [v_{N_2+1}\boldsymbol{b}_{N_2+1} + \cdots +, v_{N_1}\boldsymbol{b}_{N_1}]$ from $[\boldsymbol{u}] = [v_1\boldsymbol{b}_1 + \cdots + v_{N_1}\boldsymbol{b}_{N_1}]$ when an element in $\mathbb{V}$ (N_1 dimension) is given, where $(v_1, \ldots v_{N_1}) \leftarrow \mathbb{Z}_p^{N_1}$ and $N_2 + 1 < N_1$.

Trapdoor: If we have trapdoor $[\boldsymbol{t}^*] \in \mathsf{span}\left\langle [\boldsymbol{b}_1^*], \ldots, [\boldsymbol{b}_{N_2}^*] \right\rangle$, then we can efficiently solve the decisional subspace problem. Given $[\boldsymbol{v}] = [v_{N_2+1}\boldsymbol{b}_{N_2+1} + \cdots +, v_{N_1}\boldsymbol{b}_{N_1}]$ or $[\boldsymbol{u}] = [v_1\boldsymbol{b}_1 + \cdots + v_{N_1}\boldsymbol{b}_{N_1}]$, we can tell $[\boldsymbol{v}]$ from $[\boldsymbol{u}]$ using $[\boldsymbol{t}^*]$ since $\boldsymbol{e}([\boldsymbol{v}], [\boldsymbol{t}^*]) = 1$ and $e([\boldsymbol{u}], [\boldsymbol{t}^*]) \neq 1$ with high probability.

Advantage of this approach: For canonical basis, it is easy to decompose $[x_i\boldsymbol{a}_i] = ([1], \ldots, [1], [x_i], [1], \ldots, [1])$ from $[\boldsymbol{x}] := [x_1\boldsymbol{a}_1] \oplus \cdots \oplus [x_N\boldsymbol{a}_N] = ([x_1], \ldots, [x_N])$. In contrast, the DPVS approach employs basis $[\boldsymbol{B}]$, which is linearly transformed from $\mathbb{A}$ using a secret random matrix $\boldsymbol{B} \in \mathbb{Z}_p^{n \times n}$. It seems hard to decompose $[x_i\boldsymbol{b}_i]$ from $([x_1\boldsymbol{b}_1 + \cdots + x_N\boldsymbol{b}_N])$ (and the decisional subspace problem seems intractable). In addition, the secret matrix $\boldsymbol{B}$ (and the dual orthonormal basis $\boldsymbol{B}^*$ of $\mathbb{V}$) can be used as trapdoors for the decomposability (and distinguishability for the decisional subspace problem through the pairing operation over $[\boldsymbol{B}]$ and $[\boldsymbol{B}^*]$).

Parameter Hiding. Let $m \leq n$ be a fixed positive integer and $\boldsymbol{A} \in \mathbb{Z}_p^{m \times m}$ be an invertible matrix. Let $S_m \subseteq [n]$ be a subset of size m. Lewko proposed how to obtain new dual orthonormal bases $(\boldsymbol{B}_{\boldsymbol{A}}, \boldsymbol{B}_{\boldsymbol{A}}^*)$ from $(\boldsymbol{B}, \boldsymbol{B}^*)$. If $\boldsymbol{B}_m$ is an $n \times m$ matrix over $\mathbb{Z}_p$ whose columns are vectors $\boldsymbol{b}_i \in \boldsymbol{B}$ such that $i \in S_m$, then $\boldsymbol{B}_m\boldsymbol{A}$ is also an $n \times m$ matrix. Let $\boldsymbol{B}_{\boldsymbol{A}} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ where $\boldsymbol{x}_i := \boldsymbol{b}_i$ for all $i \notin S_m$ and $\boldsymbol{x}_i := (\boldsymbol{B}_m\boldsymbol{A})_\ell$ for $i \in S_m$, i is the ℓ-th element of S_m and $(\boldsymbol{B}_m\boldsymbol{A})_\ell$ denotes the ℓ-th column of $\boldsymbol{B}_m\boldsymbol{A}$. If $\boldsymbol{B}_m^*$ is $n \times m$ matrix over $\mathbb{Z}_p$ whose columns are vectors $\boldsymbol{b}_i^* \in \boldsymbol{B}^*$ such that $i \in S_m$, then $\boldsymbol{B}_m(\boldsymbol{A}^{-1})^{\mathsf{T}}$ is also $n \times m$ matrix. Let $\boldsymbol{B}_{\boldsymbol{A}}^* := (\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_n^*)$ where $\boldsymbol{x}_i^* := \boldsymbol{b}_i^*$ for all $i \notin S_m$ and $\boldsymbol{a}_i^* := (\boldsymbol{B}_m^*(\boldsymbol{A}^{-1})^{\mathsf{T}})_\ell$ for $i \in S_m$, i is the ℓ-th element of S_m and $(\boldsymbol{B}_m(\boldsymbol{A}^{-1})^{\mathsf{T}})_\ell$ denotes the ℓ-th column of $\boldsymbol{B}_m(\boldsymbol{A}^{-1})^{\mathsf{T}}$. Lewko showed that these newly generated bases are also dual orthonormal bases.

Lemma 2.5 ([Lew12]). *For any fixed positive integers $m \leq n$, any fixed invertible $\boldsymbol{A} \in \mathbb{Z}_p^{m\times m}$ and set $S_m \subseteq [n]$ of size m, if $(\boldsymbol{B}, \boldsymbol{B}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^n)$, then $(\boldsymbol{B}_{\boldsymbol{A}}, \boldsymbol{B}^*_{\boldsymbol{A}})$ is also distributed as a random sample from $\mathsf{Dual}(\mathbb{Z}_p^n)$ and its distribution is independent of $\boldsymbol{A}$.*

Vector decomposition problem. The VD problem was originally introduced by Yoshida, Mitsunari, and Fujiwara [YMF10]. We present the definition of a higher dimensional version by Okamoto and Takashima [OT08] to fit the VD problem into DPVS.

Definition 2.6 (CVDP: (ℓ_1, ℓ_2)-Computational Vector Decomposition Problem). *Let λ be a security parameter and $\mathcal{G}_{\mathsf{dpvs}}$ be an algorithm that outputs a description of a ℓ_1-dimensional DPVS $(p, \mathbb{V}, \mathbb{G}_T, e, g)$ and $\ell_1 > \ell_2$. Let $\mathcal{A}$ be a PPT machine. For all $\lambda \in \mathbb{N}$, we define the advantage of $\mathcal{A}$ in (ℓ_1, ℓ_2)-computational vector decomposition problem $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ as*

$$\mathsf{Adv}^{\mathsf{cvdp}}_{\mathcal{A},(\ell_1,\ell_2)}(\lambda) := \Pr\left[[\omega] = \sum_{i=1}^{\ell_2}(x_i \odot [\boldsymbol{b}]_i) \;\middle|\; \begin{array}{l} (\mathsf{params}_{\mathbb{V}}, [\boldsymbol{B}], [\boldsymbol{B}^*]) \leftarrow \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, \ell_1), \\ (x_1, \ldots, x_{\ell_1}) \leftarrow (\mathbb{Z}_p)^{\ell_1}, \\ [\boldsymbol{v}] := \sum_{i=1}^{\ell_1}(x_i \odot [\boldsymbol{b}_i]), \\ [\omega] \leftarrow \mathcal{A}(1^\lambda, \mathsf{params}_{\mathbb{V}}, [\boldsymbol{B}], [\boldsymbol{v}]) \end{array} \right].$$

The $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ assumption: For any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{cvdp}}_{\mathcal{A},(\ell_1,\ell_2)}(\lambda) < \mathsf{negl}(\lambda)$.

A specific class of the CVDP instances that are specified over canonical basis $\mathbb{A}$ are tractable.

Lemma 2.7 (Easy Basis [OT08]). *Let $\boldsymbol{A}$ be a canonical basis of $\mathbb{V}$, and $\mathsf{CVDP}^{\boldsymbol{A}}_{(\ell_1,\ell_2)}$ be a specific class of $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ in which $\boldsymbol{B}$ is replaced by $\boldsymbol{A}$. The canonical maps $\phi_{i,j}$ on $\mathbb{V}$ can solve $\mathsf{CVDP}^{\boldsymbol{A}}_{(\ell_1,\ell_2)}$ in polynomial time.*

Trapdoor. If we have a trapdoor, linear transformation matrix $\boldsymbol{B}$ (or $\boldsymbol{B}^*$), then we can efficiently decompose vectors in DPVS, i.e., solve $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ by using the efficient algorithm Decomp given by Okamoto and Takashima [OT08].

The input is $([\boldsymbol{v}], ([\boldsymbol{b}]_1, \ldots, [\boldsymbol{b}]_{\ell_2}), \boldsymbol{B})$ such that $[\boldsymbol{v}] := \sum_{i=1}^{\ell_1}(y_i \odot [\boldsymbol{b}_i])$ is a target vector for decomposition, $([\boldsymbol{b}_1], \ldots, [\boldsymbol{b}_{\ell_2}])$ is a subspace to be decomposed into, and $\boldsymbol{B}$ is a trapdoor (matrix).

$$\underline{\mathsf{Decomp}([\boldsymbol{v}], ([\boldsymbol{b}]_1, \ldots, [\boldsymbol{b}]_{\ell_2}), \boldsymbol{B}):} \text{ outputs } [\boldsymbol{u}] := \sum_{i=1}^{\ell_1}\sum_{j=1}^{\ell_2}\sum_{\kappa=1}^{\ell_1}(\tau_{i,j}\chi_{j,\kappa} \odot \phi_{\kappa,i}([\boldsymbol{v}]))$$

where ϕ is the canonical map in Definition 2.4

$$(\chi_{i,j}) = \boldsymbol{B},\; (\tau_{i,j}) := (\boldsymbol{B})^{-1}.$$

Lemma 2.8 ([OT08]). *Algorithm Decomp solves $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ by using $\boldsymbol{B} := (\chi_{i,j})$ such that $[\boldsymbol{b}_i] := \sum_{j=1}^{\ell_1}(\chi_{i,j} \odot [\boldsymbol{a}_j])$.*

2.4 Complexity Assumptions

Definition 2.9 (DLIN Assumption). *The DLIN problem is to guess $\beta \in \{0,1\}$, given $(\Gamma, [1], [a], [b], [ax], [by], Q_\beta) \leftarrow \mathcal{G}^{\mathsf{dlin}}_\beta(1^\lambda)$, where*

$$\begin{aligned} \underline{\mathcal{G}^{\mathsf{dlin}}_\beta(1^\lambda):}\ & \textit{generates } \Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}_{\mathsf{bmp}}(1^\lambda), \\ & a, b, x, y \leftarrow \mathbb{Z}_p, \\ & Q_0 := [x + y],\; Q_1 \leftarrow \mathbb{G}, \\ \textit{returns } & \mathcal{I} := (\Gamma, [1], [a], [b], [ax], [by], Q_\beta). \end{aligned}$$

This advantage $\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{A}}(\lambda)$ is defined as follows.

$$\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{A}}(\lambda) := \left|\Pr\left[\mathcal{A}(\mathcal{I}) = 1 \mid \mathcal{I} \leftarrow \mathcal{G}^{\mathsf{dlin}}_0(1^\lambda)\right] - \Pr\left[\mathcal{A}(\mathcal{I}) = 1 \mid \mathcal{I} \leftarrow \mathcal{G}^{\mathsf{dlin}}_1(1^\lambda)\right]\right|.$$

We say that the DLIN assumption holds if for all PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Definition 2.10 (Subspace Assumption). *First, we define an instance generation algorithm of the subspace problem as follow.*

$$\begin{aligned}
&\underline{\mathcal{G}_b^{\mathsf{dss}}(1^\lambda):} \ \textit{generates}\ \Gamma \leftarrow \mathcal{G}_{\mathsf{bmp}}(1^\lambda), (\boldsymbol{B}, \boldsymbol{B}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^n),\\
&\qquad \eta, \beta, \tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3 \leftarrow \mathbb{Z}_p,\\
&\textit{for } i \in [k], U_i := [\mu_1 \boldsymbol{b}_i + \mu_2 \boldsymbol{b}_{k+i} + \mu_3 \boldsymbol{b}_{2k+i}],\\
&\qquad V_i := [\tau_1 \eta \boldsymbol{b}_i^* + \tau_2 \beta \boldsymbol{b}_{k+i}^*],\\
&\qquad W_i := [\tau_1 \eta \boldsymbol{b}_i^* + \tau_2 \beta \boldsymbol{b}_{k+i}^* + \tau_3 \boldsymbol{b}_{2k+i}^*],\\
&\qquad Q_0 := (V_1, \ldots, V_k), Q_1 := (W_1, \ldots, W_k),\\
&\qquad D := ([\boldsymbol{b}_1], \ldots, [\boldsymbol{b}_{2k}], [\boldsymbol{b}_{3k+1}], \ldots, [\boldsymbol{b}_n], [\eta \boldsymbol{b}_1^*], \ldots, [\eta \boldsymbol{b}_k^*], [\beta \boldsymbol{b}_{k+1}^*], \ldots, [\beta \boldsymbol{b}_{2k}^*],\\
&\qquad\qquad [\boldsymbol{b}_{2k+1}^*], \ldots, [\boldsymbol{b}_n^*], U_1, \ldots, U_k, \mu_3),\\
&\quad \textit{returns}\ \mathcal{I} := (\Gamma, D, Q_b).
\end{aligned}$$

The subspace problem is to guess $b \in \{0,1\}$, given (Γ, D, Q_b). This advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dss}}(\lambda)$ is defined as follows.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dss}}(\lambda) := \left|\Pr\left[\mathcal{A}(\mathcal{I}) = 1 \mid \mathcal{I} \leftarrow \mathcal{G}_0^{\mathsf{dss}}(1^\lambda)\right] - \Pr\left[\mathcal{A}(\mathcal{I}) = 1 \mid \mathcal{I} \leftarrow \mathcal{G}_1^{\mathsf{dss}}(1^\lambda)\right]\right|$$

We say that the subspace assumption holds if for all PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dss}}(\lambda) \le \mathsf{negl}(\lambda)$.

Theorem 2.11 ([Lew12]). *The DLIN assumption implies the subspace assumption.*

Definition 2.12 (DBDH assumption). *The DBDH problem is to guess $\beta \in \{0,1\}$, given $(\Gamma, [1], [a], [b], [c], Q_\beta) \leftarrow \mathcal{G}_\beta^{\mathsf{dbdh}}(1^\lambda)$, where $\mathcal{G}_\beta^{\mathsf{dbdh}}(1^\lambda)$: $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$, $a, b, c \leftarrow \mathbb{Z}_p$, $Q_0 := [abc]_T$, $Q_1 \leftarrow \mathbb{G}_T$, return $(\Gamma, [1], [a], [b], [c], Q_\beta)$. This advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dbdh}}(\lambda)$ is defined as follows.*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dbdh}}(\lambda) := \left| \ \Pr\left[\mathcal{A}(\mathcal{I}) = 1 \,\middle|\, \mathcal{I} \leftarrow \mathcal{G}_0^{\mathsf{dbdh}}(1^\lambda)\ \right] - \Pr\left[\mathcal{A}(\mathcal{I}) = 1 \,\middle|\, \mathcal{I} \leftarrow \mathcal{G}_1^{\mathsf{dbdh}}(1^\lambda)\ \right] \right|$$

We say that the DBDH assumption holds if for all PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dbdh}}(\lambda) < \mathsf{negl}(\lambda)$.

Boyen and Waters pointed out that the following theorem trivially holds [BW06], but for confirmation we write a proof[2].

Theorem 2.13. *For any PPT adversary $\mathcal{A}$, there exists PPT algorithm $\mathcal{B}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dbdh}} \le \mathsf{Adv}_{\mathcal{B}}^{\mathsf{dlin}}$.*

Proof. Given DLIN instance $(\Gamma, [1], [a], [b], [xa], [yb], Q)$, adversary $\mathcal{B}$ for the DLIN problem gives adversary $\mathcal{A}$ for the DBDH problem tuple $(\Gamma, [1], [a], [b], Q, T := e([xa], [b]) \odot e([yb], [a]))$ as a DBDH instance. $T = [ab(x+y)]_T$, so if $Q = [x+y]$, then the tuple is the same as $\mathcal{G}_0^{\mathsf{dbdh}}$. It implicitly holds that $a' = a$, $b' = b$, $c' = x + y$, $a'b'c' = ab(x+y)$. If $Q = [z]$ is a uniformly random element in $\mathbb{G}$, then $T = [ba(x+y)]_T$ is a uniformly random element in $\mathbb{G}_T$ and the tuple is the same as $\mathcal{G}_1^{\mathsf{dbdh}}$ since x and y are uniformly random and independent of z, a, and b. ■

3 Definitions of Cryptographic Watermarking

We define watermarking schemes for cryptographic functions (one-way functions, hash functions, etc.). Our definition of watermarking schemes can be extended to treat cryptographic data introduced by Yoshida and Fujiwara [YF11]. In this paper, we focus on a family of functions $\mathcal{F} := \{\mathcal{F}_\lambda\}_\lambda$. For example, LTFs are cryptographic functions. Function F is sampled from family $\mathsf{LTF}_\lambda := \{\mathsf{LTF.Eval}_{\mathsf{ek}}(\cdot) | (\mathsf{ek}, \mathsf{ik}) \overset{\mathsf{R}}{\leftarrow} \mathsf{LTF.Gen}(1^\lambda, b), b \in \{0,1\}\}$.

A watermarking key generation algorithm for a family $\mathcal{F}$ takes as inputs security parameter λ and outputs secret key sk, marking key mk, detection key dk, and removing key rk. Our watermarking schemes are *public detection* watermarking schemes, that is, dk is public. The secret key is used in a sampling algorithm $\mathsf{Samp}_{\mathcal{F}}$, which outputs a function $F \overset{\mathsf{R}}{\leftarrow} \mathcal{F}_\lambda$ (The sampling algorithm takes sk as an input). *Note that the description of* $\mathsf{Samp}_{\mathcal{F}}$ *does not include* sk. Our cryptographic watermarking schemes for cryptographic functions $\mathcal{F}$ use secret key sk to choose a function $F \overset{\mathsf{R}}{\leftarrow} \mathcal{F}_\lambda$ from the function family. It seems to be a restriction because adversaries cannot generate functions in the family by themselves and use them for attacks. Function families where we can publicly sample a function is more general. However, it is very reasonable in our setting due to the following reason.

[2]This proof is based on personal communication with Keita Xagawa

- In a realistic setting, only authorized entities can generate marked functions from an original non-marked function which is privately generated. They do not distribute non-marked functions.

A marking key allows us to embed a mark in function F. A marked function F' must be similar to original function F. A detection and removing key allow us to detect and remove a mark in marked function F', respectively. We sometimes use notation $\mathcal{M}(F)$ to denote a marked function of F.

Definition 3.1 (Watermarking Scheme for Functions). *A watermarking scheme for family $\mathcal{F}$ is a tuple of algorithms* $\mathsf{CWM}_{\mathcal{F}} := \{\mathsf{WMGen}, \mathsf{Samp}_{\mathcal{F}}, \mathsf{Mark}, \mathsf{Detect}, \mathsf{Remove}\}$ *as follows.*

WMGen**:** *The key generation algorithm takes as an input security parameter λ and outputs secret key sk, marking key mk, detection key dk, and removing key rk. That is, $(sk, mk, dk, rk) \overset{R}{\leftarrow} \mathsf{WMGen}(1^\lambda)$.*

$\mathsf{Samp}_{\mathcal{F}}$**:** *The sampling algorithm takes as an input sk and outputs a function in the family $\mathcal{F}$.*

Mark**:** *The marking algorithm takes as inputs mk and unmarked function F and outputs marked function $\widetilde{F}$. That is, $\widetilde{F} \overset{R}{\leftarrow} \mathsf{Mark}(mk, F)$.*

Detect**:** *The detection algorithm takes as inputs dk and function F' and outputs* marked *(detect a mark) or* $\mathsf{unmarked}$ *(no mark), that is,* $\mathsf{Detect}(dk, F') = \mathsf{marked}/\mathsf{unmarked}$.

Remove**:** *The removing algorithm takes as inputs rk and marked function $\widetilde{F}$ and outputs unmarked function $F := \mathsf{Remove}(rk, \widetilde{F})$.*

As Hopper *et al.* noted [HMW07], we do not allow any online communication between the Detect and Mark procedures.

We define the security of cryptographic watermarking based on the definition of strong watermarking with respect to the metric space proposed by Hopper *et al.* [HMW07] and software watermarking proposed by Barak *et al.* [BGI+01, BGI+12]. We borrow some terms from these studies [HMW07, BGI+01, BGI+12]. Hopper *et al.* defined a metric space equipped with distance function d and say that object O_1 and O_2 are similar if $d(O_1, O_2) \leq \delta$ for some δ. However, we do not directly use it since we focus on function families (not perceptual objects).

Basically, the following properties should be satisfied. Most objects $F \in \mathcal{F}_\lambda$ sampled by the sampling algorithm must be unmarked. We define similarity by ϵ-approximation. That is, if for randomly chosen input x, output $F(x)$ is equal to $F'(x)$ with probability ϵ, then we say F' ϵ-approximates F. Given marked function F', an adversary should not be able to construct a new function F^*, which ϵ-approximates F' but is unmarked without removing key rk.

Our definition of the non-removability below is a game-based definition and based on the notion of strong watermarking by Hopper *et al.* [HMW07]. Our definitions are specialized to focus on cryptographic functions (do not use metric spaces). The non-removability states that even if the adversary is given marked functions, it cannot find a function that is similar to a marked function but does not contain any mark. This is based on the security against removal introduced by Hopper *et al.* [HMW07].

Before we introduce our definitions, we introduce the notion of ϵ-approximation of functions.

Definition 3.2 (ϵ-Approximating a Function). *A function f' is said to ϵ-approximate a function $f : \{0,1\}^n \to \{0,1\}^*$, denoted by $f' \cong_\epsilon f$, if $\Pr_{x \leftarrow \{0,1\}^n}[f'(x) = f(x)] \geq \epsilon$.*

Definition 3.3 (Secure Watermarking for Functions). *A watermarking scheme for function family $\mathcal{F}$ is secure if it satisfies the following properties.*

Meaningfulness: *For any $(sk, mk, dk, rk) \overset{R}{\leftarrow} \mathsf{WMGen}(1^\lambda)$ and $F \overset{R}{\leftarrow} \mathsf{Samp}_{\mathcal{F}}(sk)$, it holds that* $\mathsf{Detect}(dk, F) = \mathsf{unmarked}$.

Correctness: *For any $(sk, mk, dk, rk) \overset{R}{\leftarrow} \mathsf{WMGen}(1^\lambda)$, $F \overset{R}{\leftarrow} \mathsf{Samp}_{\mathcal{F}}(sk)$, and $\mathcal{M}(F) \overset{R}{\leftarrow} \mathsf{Mark}(mk, F)$, it holds that* $\mathsf{Detect}(dk, \mathcal{M}(F)) = \mathsf{marked}$ *and* $\mathsf{Detect}(dk, \mathsf{Remove}(rk, \mathcal{M}(F))) = \mathsf{unmarked}$.

Preserving Functionality: *For any input $x \in \{0,1\}^n$ and $F \in \mathcal{F}_\lambda$, it holds that $\mathcal{M}(F)(x) = F(x)$. If function F' preserves the functionality of function F, then we write $F \equiv F'$.*

Polynomial Blowup: *There exists a polynomial p such that for any $F \in \mathcal{F}_\lambda$, $|\mathcal{M}(F)| \leq p(|F| + |mk|)$.*

Non-Removability: *We say that a watermarking scheme satisfies non-removability (or is non-removable) if it holds that $\mathsf{Adv}^{\mathsf{nrmv}}_{\mathcal{F},\mathcal{A}}(1^\lambda, \epsilon) := \Pr[\mathsf{Exp}^{\mathsf{nrmv}}_{\mathcal{F},\mathcal{A}}(\lambda, \epsilon) = \mathsf{win}] \leq \mathsf{negl}(\lambda)$ where ϵ is a parameter for ϵ-approximation of functions. Experiment $\mathsf{Exp}^{\mathsf{nrmv}}_{\mathcal{F},\mathcal{A}}(\lambda, \epsilon)$ is shown in Figure 1.*

```
Experiment $\mathsf{Exp}^{\mathsf{nrmv}}_{\mathcal{F},\mathcal{A}}(\lambda,\epsilon)$:
$(sk, mk, dk, rk) \xleftarrow{R} \mathsf{WMGen}(1^\lambda)$; $\mathsf{CList} := \emptyset$;
$F^* \xleftarrow{R} \mathcal{A}^{\mathcal{MO},\mathcal{CO}}(1^\lambda, dk)$;
$\mathsf{Detect}(dk, F^*) = b$; $\mathsf{IdealDtc}(F^*) = B'$;
If $b = \mathsf{unmarked}$ and $B' = \{\mathsf{marked}\}$; then return win else return lose
```

```
Oracle $\mathcal{MO}(F)$
$F' \xleftarrow{R} \mathsf{Mark}(mk, F)$;
return $F'$;
```

```
Oracle $\mathcal{CO}_{\mathcal{F}_\lambda}()$
$F \xleftarrow{R} \mathsf{Samp}_{\mathcal{F}}(sk)$;
$F' \xleftarrow{R} \mathsf{Mark}(mk, F)$;
$\mathsf{CList} := \mathsf{CList} \cup \{F'\}$;
return $F'$
```

```
Procedure $\mathsf{IdealDtc}(F^*)$
if $(\exists F' \in \mathsf{CList} :\ F^* \cong_\epsilon F')$; then return $\{\mathsf{marked}\}$
else return $\{\mathsf{unmarked}\}$
```

Figure 1: Experiment for non-removability

On the security experiments. In our construction, the detection key dk is public and given to $\mathcal{A}$. Thus, we need not consider an oracle that receives a function F' and returns $\mathsf{Detect}(dk, F')$. Such an oracle is called the detect oracle and denoted by $\mathcal{DO}$. If we consider private-detection-key setting, adversaries are given access to $\mathcal{DO}$.

The adversary tries to find a function such that the outputs of the actual detection algorithm and the *ideal detection procedure* are different. The ideal detection procedure searches a database and outputs a decision by using online communication to the marking algorithm. The parameter ϵ is called the approximation factor. The adversary has access to oracles, i.e., the mark and challenge oracles. The mark oracle returns a marked function for a queried non-marked function. The challenge oracle generates a new (non-marked) function, embeds a mark in the new function, and returns the marked function (the original non-marked function is hidden).

Eventually, the adversary outputs function F. It means that the adversary claims that it succeeded in removing a mark from some marked function F' without the remove key. The function F should be similar to (ϵ-approximate) the function F'.[3] This is for security against removal.

In the ideal detection procedure, we need check function F ϵ-approximates F'. We can achieve this by comparing outputs of F and F' for uniformly random input x in many times. We can use Chernoff bound to analyze it. This algorithm is very standard one and similar analyses were shown in many papers. In this paper, we refer to the analysis by Nishimaki and Wichs [NW15]. The check algorithm is in Figure 2. See the work by Nishimaki and Wichs [NW15] for the detail of the analysis.

Theorem 3.4 ([NW15]). *If F ϵ-approximates F', then the algorithm* $\mathsf{Test}(F, F')$ *outputs* 1 *except with negligible probability.*

As Hopper *et al.* explained [HMW07], we must introduce the challenge oracle because if it does not exist, then we cannot define a meaningful security experiment for non-removability. Adversaries should try to remove a mark in a marked function whose original unmarked function is unknown to adversaries. Thus, we need an entity that gives adversaries freshly sampled unmarked-functions.

We can consider the following trivial attack scenario, but it is not a valid attack. If the adversary samples an unmarked function $F \in \mathcal{F}_\lambda$, queries it to the mark oracle, and finally outputs them as solutions. The actual detect algorithm apparently returns $\mathsf{unmarked}$ for F and the ideal detect procedure does not return $\{\mathsf{marked}\}$ for F since an equivalent function is not recorded in CList. Thus, the adversary cannot win by this strategy.

Discussion on the definition. We require that legal marked functions output by the marking algorithm satisfy the preserving functionality. As we introduced in the introduction, it is impossible to construct watermarking

[3]In the previous version of this paper, we used the pefect functionality preserving condition for the adversary. However, we revised the definition based on the definition by Nishimaki and Wichs [NW15] since we cannot check the perfect functionality preserving condition in polynomial time.

```
Inputs: Two functions $f_0$ and $f_1$.

Parameters: $0 < \epsilon < 1$.

Set $\mathsf{cnt} := 0$ and $R := 8\lambda(1/\epsilon)^2$. For $i = 1, \dots, R$, do
    1. Choose $z_i \leftarrow \mathsf{D}$ where $\mathsf{D}$ is the domain of functions.
    2. If $f_0(z_i) = f_1(z_i)$ then set $\mathsf{cnt} := \mathsf{cnt} + 1$
  Let $\widehat{\epsilon} := \mathsf{cnt}/R \in [0,1]$ be the fraction of trials in which $f_0(z_i) = f_1(z_i)$.
  If $\widehat{\epsilon} < \frac{3}{4}\epsilon$, then output $0$, else $1$.
```

Figure 2: Test algorithm Test for approximation of functions

schemes that satisfies the preserving functionality if we assume the existence of iO [BGI+12]. To avoid the impossibility result by Barak *et al.* , we restrict adversaries in the non-removability game. When we prove the security of our watermarking scheme, *adversaries must output a function that follows a specified format* in function family $\mathcal{F}$ to win the security game. More concretely, in our scheme, a function consists of ℓ^2 group elements in $\mathbb{G}_T$, $8(\ell+1)$ group elements in $\mathbb{G}$, and one integer in $\mathbb{Z}_p$ where ℓ is the input length of a function and adversaries must output ℓ^2 elements in $\mathbb{G}_T$, $8(\ell+1)$ elements in $\mathbb{G}$, and one integer in $\mathbb{Z}_p$ as a non-marked function to win the security game. Thus, we can avoid the impossibility results. We call this restricted game *non-removability under the same format restriction* in this paper. If arbitrary strategies are allowed, adversaries can use iO to attack our watermarking scheme since obfuscation plays a role of a mark-remover [BGI+01, BGI+12] as we explained in Section 1.3. However, they can not use it in the security game of non-removability under the same format restriction since obfuscated functions by the candidate constructions of iO do not consist of group elements [GGH+13b, BR14, BGK+14, AGIS14, AB15, Zim15]. Even if obfuscated circuits are encoded into group elements (ex. using only $[0]$ and $[1]$ for bits 0 and 1, respectively), the number of group elements does not match that of an original format.

The restriction on adversaries is too strong, but our watermarking scheme is still meaningful. First, all current candidate constructions of iO are based on graded encoding schemes [GGH13a, CLT13, GGH15]. Several serious attacks on not only graded encoding schemes but also iO were found in some candidate constructions [CHL+15, CGH+15, HJ16, CFL+16, MSZ16, ADGM16, CLLT16b, CGH16]. Of course, a few iO constructions are *not* attacked so far [GMM+16, FRS16], but the security of graded encoding schemes and iO have not been well-studied yet. Constructing watermarking schemes under the assumption that there is no iO is meaningful though it may be a bad news. Cohen, Holmgren, Nishimaki, Vaikuntanathan, and Wichs proposed watermarking schemes against arbitrary strategies under the assumption that there exists iO (and one-way function). Thus, we can say that our watermarking scheme is an alternative construction in the case that there is no iO. Second, in realistic scenarios, potentially illegal users may be required to reveal illegal copy of marked functions and if they reveal functions whose format is different from that of original functions, then we can decide that they are suspicious users. Of course, in this case, we cannot achieve the black-box type tracing, but it is still meaningful. Lastly, our construction of watermarking scheme itself is interesting because hidden subspaces of DPVS can be used to achieve watermarking.

4 Proposed Watermarking Scheme based on Lewko's scheme

We present LTFs and watermarking schemes for the LTFs that are secure under the DLIN assumption in this section.

Generally speaking, LTFs can be constructed from homomorphic encryption schemes as observed in many papers [FGK+10, HO12, PW11]. Lewko and Okamoto-Takashima proposed an IBE and IPE scheme based on DPVS which is homomorphic and secure under the DLIN assumption, respectively (See Appendix A for descriptions of their schemes). We can easily construct a LTF from the IBE scheme by applying the matrix encryption technique introduced by Peikert and Waters [PW11]. Note that IBE schemes are obtained from IPE schemes where the predicate is the equality test.

In this section, we present a scheme based on the Lewko IBE scheme. Basically, previous works used homomorphic *PKE* schemes to construct LTFs. However, we use homomorphic *IBE* schemes to achieve a watermarking scheme because we would like to use the *dual system encryption* technique. We assign a tag for each function index and use the tag as an identity of IBE. To construct LTFs based on IBE schemes, we use not only ciphertexts under some identity but also a private key for the identity. If there is no private key for identities, then we cannot obtain valid outputs that can be inverted by an inversion key of the LTF. Note that the private key for an identity of IBE is *not a trapdoor inversion key for the LTF*.

4.1 LTF based on Lewko's IBE scheme

Our LTF LTF based on the Lewko IBE is as follows.

$\mathsf{LTF.IGen}(1^\lambda)$ **:** It generates $(\boldsymbol{D}, \boldsymbol{D}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^8)$, chooses $\alpha, \theta, \sigma \leftarrow \mathbb{Z}_p$, $\boldsymbol{\psi} := (\psi_1, \ldots, \psi_\ell) \leftarrow \mathbb{Z}_p^\ell$, and sets $g_T := [\alpha\theta \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*]_T$ and $g_{T_j} := \psi_j \odot g_T = [\psi_j \alpha\theta \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*]_T$ for all $j \in [\ell]$ where ℓ is the input length of functions. It chooses an arbitrary $\mathsf{tag} \in \mathbb{Z}_p$ and $s_{1,i}, s_{2,i} \leftarrow \mathbb{Z}_p$ for all $i \in [\ell]$ and generates

$$\begin{aligned} u_{i,j} &:= (s_{1,i} \odot g_{T_j}) \oplus (m_{i,j} \odot g_T) \\ &= [(s_{1,i}\psi_j + m_{i,j})\alpha\theta \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*]_T, \\ \boldsymbol{v}_i &:= [s_{1,i}\boldsymbol{d}_1 + s_{1,i}\mathsf{tag}\boldsymbol{d}_2 + s_{2,i}\boldsymbol{d}_3 + s_{2,i}\mathsf{tag}\boldsymbol{d}_4], \end{aligned}$$

for all $i, j \in [\ell]$ where $m_{i,i} = 1$ and $m_{i,j} = 0$ (if $i \neq j$). It defines a matrix $\boldsymbol{M} := \{m_{i,j}\}_{i,j} = \boldsymbol{I}$. It chooses $r_1, r_2 \leftarrow \mathbb{Z}_p$ and generates

$$\boldsymbol{k}_{\mathsf{tag}} := [(\alpha + r_1\mathsf{tag})\theta\boldsymbol{d}_1^* - r_1\theta\boldsymbol{d}_2^* + r_2\mathsf{tag}\sigma\boldsymbol{d}_3^* - r_2\sigma\boldsymbol{d}_4^*].$$

It returns $\mathsf{ek} := (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag}) := (\{u_{i,j}\}_{i,j=1}^\ell, \{\boldsymbol{v}_i\}_{i=1}^\ell, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag})$, $\mathsf{ik} := \boldsymbol{\psi}$. Hereafter, $\{u_{i,j}\}$ and $\{\boldsymbol{v}_i\}$ denote $\{u_{i,j}\}_{i,j=1}^\ell$ and $\{\boldsymbol{v}_i\}_{i=1}^\ell$, respectively if it is clear from the context.

$\mathsf{LTF.LGen}(1^\lambda)$ **:** This is the same as $\mathsf{LTF.IGen}$ except that for all $i, j \in [\ell]$, $m_{i,j} = 0$ and $\mathsf{ik} := \perp$. (Matrix $\boldsymbol{M} = \boldsymbol{0}$.)

$\mathsf{LTF.Eval}(\mathsf{ek}, \boldsymbol{x})$**:** First, it parses $ek = (\{u_{i,j}\}, \{\boldsymbol{v}_i\}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag})$. For input $\boldsymbol{x} \in \{0,1\}^\ell$, it computes

$$\begin{aligned} y_j &:= \sum_i (x_i \odot u_{i,j}) &&= \sum_i (x_i s_{1,i} \odot g_{T_j}) \oplus (x_i m_{i,j} \odot g_T) \\ & &&= (\langle \boldsymbol{x}, \boldsymbol{s}_1\rangle \odot g_{T_j}) \oplus (x_j m_{j,j} \odot g_T) \\ y_{\ell+1} &:= \sum_i (x_i \odot \boldsymbol{v}_i) &&= \sum_i [x_i s_{1,i}\boldsymbol{d}_1 + x_i s_{1,i}\mathsf{tag}\boldsymbol{d}_2 + x_i s_{2,i}\boldsymbol{d}_3 + x_i s_{2,i}\mathsf{tag}\boldsymbol{d}_4] \\ & &&= [\langle \boldsymbol{x}, \boldsymbol{s}_1\rangle\boldsymbol{d}_1 + \langle \boldsymbol{x}, \boldsymbol{s}_1\rangle\mathsf{tag}\boldsymbol{d}_2 + \langle \boldsymbol{x}, \boldsymbol{s}_2\rangle\boldsymbol{d}_3 + \langle \boldsymbol{x}, \boldsymbol{s}_2\rangle\mathsf{tag}\boldsymbol{d}_4] \end{aligned}$$

where $\boldsymbol{s}_1 := (s_{1,1}, \ldots, s_{1,\ell})$, $\boldsymbol{s}_2 := (s_{2,1}, \ldots, s_{2,\ell})$, and $y'_{\ell+1} := \boldsymbol{e}(y_{\ell+1}, \boldsymbol{k}_{\mathsf{tag}}) = [\alpha\theta\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* \langle \boldsymbol{x}, \boldsymbol{s}_1\rangle]_T = \langle \boldsymbol{x}, \boldsymbol{s}_1\rangle \odot g_T$ and returns output $\boldsymbol{y} := (y_1, \ldots, y_\ell, y'_{\ell+1})$.

$\mathsf{LTF.Invert}(\mathsf{ik}, \boldsymbol{y})$**:** For input $\boldsymbol{y}$, it computes

$$\begin{aligned} x'_j &:= y_j \ominus (\psi_j \odot y'_{\ell+1}) \\ &= (\langle \boldsymbol{x}, \boldsymbol{s}_1\rangle \odot g_{T_j}) \oplus (x_j \odot g_T) \ominus (\langle \boldsymbol{x}, \boldsymbol{s}_1\rangle\psi_j \odot g_T) \\ &= x_j \odot g_T \end{aligned}$$

and let $x_j \in \{0,1\}$ be such that $x'_j = x_j \odot g_T$. It returns $\boldsymbol{x} = (x_1, \ldots, x_\ell)$.

Theorem 4.1. $\mathsf{LTF}_{\mathsf{mult}}$ *is a lossy trapdoor function if the DBDH assumption holds.*

Lemma 4.2 (Lossiness of $\mathsf{LTF}_{\mathsf{mult}}$). $\mathsf{LTF}_{\mathsf{mult}}$ *is $(\ell - \log p)$-lossy.*

Proof. We compute lossiness ℓ'. For a lossy function index generated by $\mathsf{LTF.LGen}$, an output is $\boldsymbol{y} = (s'_1\boldsymbol{\psi} \odot g_T, [s'_1]_T) = (s'_1 \odot g_{T_1}, \ldots, s'_1 \odot g_{T_\ell}, [s'_1]_T)$ where $s'_1 = \langle \boldsymbol{x}, \boldsymbol{s}_1\rangle \in \mathbb{Z}_p$ since $x_{j,j} = 0$ in the lossy mode. Here, secret trapdoor $\boldsymbol{\psi}$ is fixed by the function index. This means that for any given image $\boldsymbol{y}$, there are at most p possible values for $\langle \boldsymbol{x}, \boldsymbol{s}_1\rangle$ and pre-images. Therefore, equation $|\mathsf{D}|/2^{\ell'} = p$ holds by the definition of the lossiness. By this equation, we can derive equation $\ell' = \ell - \log p$ since $|\mathsf{D}| = 2^\ell$. ■

We introduce some notations before we show the indistinguishability. We borrow the notation introduced by Peikert and Waters [PW11]. For matrix $\boldsymbol{Y} = (y_{i,j}) \in \mathbb{Z}_p^{h \times w}$, we define $[\boldsymbol{Y}]_T = ([y_{i,j}]_T) \in \mathbb{G}_T^{h \times w}$. Algorithm $\mathsf{GenConceal}(h, w)$ which was introduced by Peikert and Waters is as follows.

1. Choose $\boldsymbol{\zeta} := (\zeta_1, \ldots, \zeta_h) \leftarrow \mathbb{Z}_p^h$ and $\boldsymbol{\psi} := (\psi_1, \ldots, \psi_w, 1) \leftarrow \mathbb{Z}_p^w \times \{1\}$.

2. Let $\boldsymbol{V} := \boldsymbol{\zeta} \otimes \boldsymbol{\psi} = \boldsymbol{\zeta}^{\mathsf{T}} \boldsymbol{\psi} \in \mathbb{Z}_p^{h \times (w+1)}$ be the outer product of $\boldsymbol{\zeta}$ and $\boldsymbol{\psi}$.

3. Output $\boldsymbol{C} := [\boldsymbol{V}]_T \in \mathbb{G}_T^{h \times (w+1)}$ as the concealer matrix and $\boldsymbol{\psi}$ as the trapdoor.

The original concealer matrix by Peikert and Waters is over $\mathbb{G}^{h \times w}$, but we use a matrix over $\mathbb{G}_T$ since we use bilinear maps.

Lemma 4.3 (Indistinguishability of $\mathsf{LTF}_{\mathsf{mult}}$). *If the DBDH assumption holds, then* $\mathsf{LTF}_{\mathsf{mult}}$ *satisfies indistinguishability.*

Proof. Let $\alpha_0 \leftarrow \mathbb{Z}_p$ and $g_T := [\alpha_0]_T$. For $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_\ell, 1)$, $[\alpha_0 \zeta \boldsymbol{\psi}]_T$ denotes $([\alpha_0 \zeta \psi_1]_T, \ldots, [\alpha_0 \zeta \psi_\ell]_T, [\alpha_0 \zeta]_T)$. We need three steps to show the lemma.

First, we will show that if the DBDH assumption holds, then $([\alpha_0 \boldsymbol{\psi}]_T, \boldsymbol{\gamma} = [\alpha_0 \zeta \boldsymbol{\psi}]_T)$ is computationally indistinguishable from $([\alpha_0 \boldsymbol{\psi}]_T, \boldsymbol{\gamma} = [\alpha_0 \boldsymbol{t}]_T)$ where $\zeta \leftarrow \mathbb{Z}_p$, $\boldsymbol{\psi} \leftarrow \mathbb{Z}_p^\ell \times \{1\}$, and $\boldsymbol{t} \leftarrow \mathbb{Z}_p^{\ell+1}$. Note that the $(\ell+1)$-th element of $\boldsymbol{\psi}$ is fixed to 1. To show the indistinguishability, we define hybrid distribution

Hyb_j**:** We chooses $\alpha_0, \zeta \leftarrow \mathbb{Z}_p$ and $\boldsymbol{\psi} \leftarrow \mathbb{Z}_p^\ell \times \{1\}$ and sets $\boldsymbol{\gamma} := ([\alpha_0 \zeta \psi_1]_T, \ldots, [\alpha_0 \zeta \psi_j]_T, [\alpha_0 y_{j+1}]_T, \ldots, [\alpha_0 y_\ell]_T, [\alpha_0 \zeta]_T)$ where $y_k \leftarrow \mathbb{Z}_p$ for $k > j$. That is, $[\alpha_0 y_k]_T$ is uniformly random element for $k > j$. The output is $([\alpha_0 \boldsymbol{\psi}], \boldsymbol{\gamma})$.

We note that $\mathsf{Hyb}_0 = ([\alpha_0 \boldsymbol{\psi}]_T, [\alpha_0 \boldsymbol{t}]_T)$ and $\mathsf{Hyb}_\ell = ([\alpha_0 \boldsymbol{\psi}]_T, [\alpha_0 \zeta \boldsymbol{\psi}]_T)$. We show that for each $j \in [\ell]$, Hyb_j and Hyb_{j-1} are computationally indistinguishable under the DBDH assumption.

We construct PPT algorithm $\mathcal{B}$ that solves the DBDH problem by using distinguisher $\mathcal{D}$ for Hyb_j and Hyb_{j-1}. $\mathcal{B}$ is given input $(\mathsf{param}_{\mathbb{G}}, [1], [a], [b], [c], Q)$ and computes $(\boldsymbol{\tau}, \boldsymbol{\gamma}) \in \mathbb{G}_T^{\ell+1} \times \mathbb{G}_T^{\ell+1}$ as follows. $\mathcal{B}$ sets $\tau_{\ell+1} := g_T := [\alpha_0]_T$ for $\alpha_0 \leftarrow \mathbb{Z}_p$, $y_{\ell+1} := \alpha_0 \odot e([1], [c]) = c \odot g_T$. It implicitly holds $\zeta := c$.

- For $k \in [j-1]$, $\mathcal{B}$ chooses ψ_k and sets $\tau_k := \psi_k \odot g_T$, $\gamma_k := \alpha_0 \psi_k \odot e([1], [c]) = c\psi_k \odot g_T$.
- For $k = j+1, \ldots, \ell$, $\mathcal{B}$ chooses ψ_k and sets $\tau_k := \psi_k \odot g_T$, $\gamma_k := y_k \odot g_T$ where $y_k \leftarrow \mathbb{Z}_p$.

Finally, $\mathcal{B}$ embeds the instance, that is, sets $\tau_j := \alpha_0 \odot e([a], [b]) = ab \odot g_T$, $\gamma_j := \alpha_0 \odot Q$ (implicitly $\psi_j := ab$). If $Q = [abc]_T$, then $\gamma_j = \psi_j \zeta \odot g_T$ and $(\boldsymbol{\tau}, \boldsymbol{\gamma}) = \mathsf{Hyb}_j$. If $Q \leftarrow \mathbb{G}_T$, then $\gamma_j := y_j \odot g_T$ where $y_j \leftarrow \mathbb{Z}_p$ ($Q = [y_j]_T$) and $(\boldsymbol{\tau}, \boldsymbol{\gamma}) = \mathsf{Hyb}_{j-1}$. Therefore, $\mathsf{Hyb}_j \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_{j-1}$. As a corollary, $\mathsf{Hyb}_0 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_\ell$.

Second, We define new hybrid distributions $\mathsf{Hyb}'_0, \ldots, \mathsf{Hyb}'_{\ell'}$ over matrices $\boldsymbol{C} \in \mathbb{G}_T^{\ell' \times (\ell+1)}$. In Hyb'_i, elements in the first i rows of $\boldsymbol{C}$ are computed as in $\mathsf{GenConceal}$. On the other hand, elements in the last $(\ell' - i)$ rows are uniformly random over $\mathbb{G}_T^{\ell+1}$. $\mathsf{Hyb}'_{\ell'}$ is the same as $\mathsf{GenConceal}$ and Hyb'_0 is the uniform distribution over $\mathbb{G}_T^{\ell' \times (\ell+1)}$. We show that for each $i \in [\ell']$, $\mathsf{Hyb}'_i \overset{\mathsf{c}}{\approx} \mathsf{Hyb}'_{i-1}$ if the DBDH assumption holds. We construct PPT algorithm $\mathcal{D}$ that distinguishes Hyb_0 from Hyb_ℓ by using distinguisher $\mathcal{D}'$ for Hyb'_i and Hyb'_{i-1}. $\mathcal{D}$ is given instance $([\alpha_0 \boldsymbol{\psi}]_T, \boldsymbol{\gamma} \in \mathbb{G}_T^{\ell+1})$, $\mathcal{D}$ generates matrix $\boldsymbol{C}$ as follows.

- For each $k \leq (i-1)$, chooses $\zeta_k \leftarrow \mathbb{Z}_p$ and set the k-th row of $\boldsymbol{C}$ be $\boldsymbol{c}_k := \zeta_k \odot ([\alpha_0 \boldsymbol{\psi}]_T) = [\alpha_0 \zeta_k \boldsymbol{\psi}]_T$.
- For $k = i$, sets the i-th row of $\boldsymbol{C}$ be $\boldsymbol{c}_i := \boldsymbol{\gamma}$. That is, $\mathcal{D}$ embeds the instance.
- For the other rows, sets uniformly random elements over $\mathbb{G}_T^{(\ell+1)}$.

If $\boldsymbol{\gamma} = [\alpha_0 \zeta \boldsymbol{\psi}]_T$, then the distribution is the same as Hyb'_i, else if $\boldsymbol{\gamma}$ is uniformly random, then the distribution is the same as Hyb'_{i-1}. That is, we show that $\mathsf{Hyb}_0 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}^0_\ell$ implies $\mathsf{Hyb}'_{i-1} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}'_i$. Therefore, $\mathsf{Hyb}'_0 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}'_\ell$ if the DBDH assumption holds.

Finally, we prove the lemma. Our final goal is to show $(\{u_{i,j}\}, \{\boldsymbol{v}_i\}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag})$ for $\boldsymbol{M} = \boldsymbol{I}$ is indistinguishable from $(\{u_{i,j}\}, \{\boldsymbol{v}_i\}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag})$ for $\boldsymbol{M} = \boldsymbol{0}$. In the above simulation, we only consider $\{u_{i,j}\}$ and $\zeta \odot g_T$ instead of $\{u_{i,j}\}$, $\{\boldsymbol{v}_i = [s_{1,i}\boldsymbol{d}_1 + s_{1,i}\mathsf{tag}\boldsymbol{d}_2 + s_{2,i}\boldsymbol{d}_3 + s_{2,i}\mathsf{tag}\boldsymbol{d}_4]\}$, $\boldsymbol{k}_{\mathsf{tag}}$, and tag. The key point is that we can replace $(\ell+1)$-th column element $\zeta \odot g_T$ in $\boldsymbol{C}$ with $[\zeta \boldsymbol{d}_1 + \zeta \mathsf{tag}\boldsymbol{d}_2 + s_2\boldsymbol{d}_3 + s_2\mathsf{tag}\boldsymbol{d}_4]$ where $(\boldsymbol{D}, \boldsymbol{D}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^8)$ in the above simulation since the simulator can generate bases $(\boldsymbol{D}, \boldsymbol{D}^*)$ and choose $\mathsf{tag} \leftarrow \mathbb{Z}_p$ by itself. We must simulate

$\boldsymbol{V} = [\zeta \boldsymbol{d}_1 + \zeta \mathsf{tag} \boldsymbol{d}_2 + s_2 \boldsymbol{d}_3 + s_2 \mathsf{tag} \boldsymbol{d}_4]$ and $\boldsymbol{k}_{\mathsf{tag}} = [(\alpha + r_1 \mathsf{tag})\theta \boldsymbol{d}_1^* - r_1 \theta \boldsymbol{d}_2^* r_2 \mathsf{tag} \sigma \boldsymbol{d}_3^* - r_2 \sigma \boldsymbol{d}_4^*]$ for $s_2 \leftarrow \mathbb{Z}_p$. If we have $[\zeta] = [c]$ and matrix $\boldsymbol{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_8)$, then we can compute $[\zeta \boldsymbol{d}_1 + \zeta \mathsf{tag} \boldsymbol{d}_2] = (\boldsymbol{d}_1 + \mathsf{tag} \boldsymbol{d}_2) \odot [\zeta]$ without knowing ζ since simulator $\mathcal{B}$ generates $(\boldsymbol{D}, \boldsymbol{D}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^8)$ by itself. $\mathcal{B}$ can also generate $\boldsymbol{k}_{\mathsf{tag}}$ since it has $\boldsymbol{D}^*$. Therefore, in the above simulation we can replace $(\ell + 1)$-th column element $\zeta \odot g_T$ with $[\zeta \boldsymbol{d}_1 + \zeta \mathsf{tag} \boldsymbol{d}_2 + s_2 \boldsymbol{d}_3 + s_2 \mathsf{tag} \boldsymbol{d}_4]$ and add $\boldsymbol{k}_{\mathsf{tag}} = [(\alpha + r_1 \mathsf{tag})\theta \boldsymbol{d}_1^* - r_1 \theta \boldsymbol{d}_2^* + r_2 \mathsf{tag} \sigma \boldsymbol{d}_3^* - r_2 \sigma \boldsymbol{d}_4^*]$ (We can set $\alpha_0 := \alpha \theta \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*$). Therefore, $\mathsf{LTF}_{\mathsf{mult}}$ satisfies indistinguishability. ■

4.2 Watermarking Scheme for Our LTF

$\mathsf{LTF}_{\mathsf{mult}}$ In this section, we present our watermarking scheme. First, we give an overview of our construction.

We added extra two dimensions of DPVS to the original Lewko IBE scheme since we use the extra dimensions to embed watermarks. Even if we add a vector spanned by $\boldsymbol{d}_7^*$ and $\boldsymbol{d}_8^*$ to an element $\boldsymbol{k}_{\mathsf{tag}}$ in a function index, which is spanned by $\boldsymbol{d}_1^*, \ldots, \boldsymbol{d}_4^*$, it is indistinguishable from the original one since vectors $\boldsymbol{d}_7, \boldsymbol{d}_8, \boldsymbol{d}_7^*, \boldsymbol{d}_8^*$ are hidden. Moreover, the marked index works as the original non-marked index since elements in function index $\boldsymbol{V}$ are spanned by $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_4$ and components $\boldsymbol{d}_7^*, \boldsymbol{d}_8^*$ are canceled. However, if we have a vector which is spanned by $\boldsymbol{d}_7, \boldsymbol{d}_8$, then we can detect the mark which is generated by $\boldsymbol{d}_7^*, \boldsymbol{d}_8^*$. If we have complete dual orthonormal bases $(\boldsymbol{D}, \boldsymbol{D}^*)$, then we can use the vector decomposition algorithm introduced in Section 2.3 and eliminate the vector spanned by $\boldsymbol{d}_7^*, \boldsymbol{d}_8^*$, i.e., watermarks.

Our watermarking scheme $\mathsf{CWM}_{\mathsf{mult}}$ for $\mathsf{LTF}_{\mathsf{mult}}$ is as follows:

$\mathsf{WMGen}(1^\lambda)$**:** It generates $(\boldsymbol{D}, \boldsymbol{D}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^8)$, chooses $\alpha, \theta, \sigma \leftarrow \mathbb{Z}_p$ and $u_7, u_8 \leftarrow \mathbb{Z}_p^*$, and sets $g_T := [\alpha \theta \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*]_T$, $pp := \widehat{\mathbb{D}} := (\mathsf{param}_{\mathbb{V}}, g_T, [\boldsymbol{d}_1], \ldots, [\boldsymbol{d}_4])$, $sk := (\widehat{\mathbb{D}}, [\alpha \theta \boldsymbol{d}_1^*], [\theta \boldsymbol{d}_1^*], [\theta \boldsymbol{d}_2^*], [\sigma \boldsymbol{d}_3^*], [\sigma \boldsymbol{d}_4^*])$, $mk := ([\boldsymbol{d}_7^*], [\boldsymbol{d}_8^*])$, $dk := (\widehat{\mathbb{D}}, \boldsymbol{c} := [u_7 \boldsymbol{d}_7 + u_8 \boldsymbol{d}_8])$, and $rk := (\boldsymbol{D}, \boldsymbol{D}^*)$. Keys sk, mk, and rk are secret. Parameter pp and key dk are public.

$\mathsf{Samp}(sk)$**:** The sampling algorithm chooses $\mathsf{tag} \in \mathbb{Z}_p$ and $\boldsymbol{\psi} \leftarrow \mathbb{Z}_p^\ell$, $\boldsymbol{s}_1, \boldsymbol{s}_2 \leftarrow \mathbb{Z}_p^\ell$, and generates $(\mathsf{ek}, \mathsf{ik}) := ((\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag}), \boldsymbol{\psi})$ as $\mathsf{LTF.IGen}$. It computes $\boldsymbol{k}_{\mathsf{tag}} := [(\alpha + r_1 \mathsf{tag})\theta \boldsymbol{d}_1^* - r_1 \theta \boldsymbol{d}_2^* + r_2 \sigma \mathsf{tag} \boldsymbol{d}_3^* - r_2 \sigma \boldsymbol{d}_4^*]$.

$\mathsf{Mark}(mk, \mathsf{ek})$**:** It parses $\mathsf{ek} = (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag}) \in \mathbb{G}_T^{\ell^2} \times \mathbb{G}^{8\ell} \times \mathbb{G}^8 \times \mathbb{Z}_p$, chooses $t_7, t_8 \leftarrow \mathbb{Z}_p$, and computes $\widetilde{\boldsymbol{k}}_{\mathsf{tag}} := \boldsymbol{k}_{\mathsf{tag}} \cdot [t_7 \boldsymbol{d}_7^* + t_8 \boldsymbol{d}_8^*]$ by using $[\boldsymbol{d}_7^*]$ and $[\boldsymbol{d}_8^*]$. It outputs marked function index $\mathcal{M}(\mathsf{ek}) = (\boldsymbol{U}, \boldsymbol{V}, \widetilde{\boldsymbol{k}}_{\mathsf{tag}}, \mathsf{tag})$.

$\mathsf{Detect}(dk, \widetilde{\mathsf{ek}})$**:** It parses $\widetilde{ek} = (\boldsymbol{U}, \boldsymbol{V}, \widetilde{\boldsymbol{k}}_{\mathsf{tag}}, \mathsf{tag})$ and $dk = (\widehat{\mathbb{D}}, \boldsymbol{c})$. Next, it computes $\Delta := \boldsymbol{e}(\boldsymbol{c}, \widetilde{\boldsymbol{k}}_{\mathsf{tag}})$. If the following condition holds, then it outputs marked, otherwise outputs unmarked.

- $\Delta = \boldsymbol{e}(\boldsymbol{c}, \widetilde{\boldsymbol{k}}_{\mathsf{tag}}) \neq 1$

$\mathsf{Remove}(rk, \widetilde{ek})$**:** It parses $rk = (\boldsymbol{D}, \boldsymbol{D}^*)$ and $\widetilde{ek} = (\boldsymbol{U}, \boldsymbol{V}, \widetilde{\boldsymbol{k}}_{\mathsf{tag}}, \mathsf{tag})$, runs the decomposition algorithm. That is, it computes $\sum_{j=1}^{m} [z_j \boldsymbol{d}_j^*] = \mathsf{Decomp}(\widetilde{\boldsymbol{k}}_{\mathsf{tag}}, ([\boldsymbol{d}_1^*], \ldots, [\boldsymbol{d}_m^*]), \boldsymbol{D}^*, ([\boldsymbol{d}_1], \ldots, [\boldsymbol{d}_8]))$ for all $m < 8$, where $z_j \in \mathbb{Z}_p$ and obtains $[z_j \boldsymbol{d}_j^*]$ for $j = 1, \ldots, 8$. It holds $\widetilde{\boldsymbol{k}}_{\mathsf{tag}} = [z_1 \boldsymbol{d}_1^* + \cdots + z_8 \boldsymbol{d}_8^*]$. It computes $\boldsymbol{k}'_{\mathsf{tag}} := \widetilde{\boldsymbol{k}}_{\mathsf{tag}} \ominus [z_7 \boldsymbol{d}_7^* + z_8 \boldsymbol{d}_8^*]$ and outputs $(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}'_{\mathsf{tag}}, \mathsf{tag})$ as an unmarked index.

We can easily verify that meaningfulness, correctness, and polynomial blowup hold. Note that the sampling algorithm uses sk to sample a function index and sk does *not* include $([\boldsymbol{d}_7^*], [\boldsymbol{d}_8^*])$. Thus, meaningfulness hold with probability 1.

Preserving functionality holds since elements in $\boldsymbol{U}$ and $\boldsymbol{V}$ do not include vectors $[\boldsymbol{d}_7]$ and $[\boldsymbol{d}_8]$ and vector $[t_1 \boldsymbol{d}_7^* + t_2 \boldsymbol{d}_8^*]$ does not interfere the computation of $\mathsf{LTF.Eval}$. Note that if we do not have secret key $([\boldsymbol{d}_1^*], \ldots, [\boldsymbol{d}_4^*])$, then we cannot compute a complete function index, that is, we cannot compute an element $\boldsymbol{k}_{\mathsf{tag}}$. This seems to be a restriction, but in the scenario of watermarking schemes, this is acceptable by following reasons. We use watermarking schemes to authorize objects and such objects are privately generated by authors. For example, movies, music files, and software are generated by some companies and they do not distribute unauthorized (unmarked) objects. Moreover, in the experiment on security, the adversary is given a oracle which gives marked function indices. Thus, it is reasonable that unauthorized parties cannot efficiently sample functions by themselves.

4.3 Security Proof

Definition 4.4 (Non-removability under the same format restriction). *We say that a watermarking scheme satisfies non-removability under the same format restriction if it is non-removable against adversaries in the experiment* $\mathsf{Exp}^{\mathsf{nrmv}}_{\mathcal{F},\mathcal{A}}(\lambda, \epsilon)$ *for* $\mathsf{CWM}_{\mathsf{mult}}$ *that must output* ℓ^2 *elements in* $\mathbb{G}_T$*,* $8(\ell+1)$ *elements in* $\mathbb{G}$*, and one integer in* $\mathbb{Z}_p$ *as a non-marked function to win the game.*

Definition 4.5 (Secure watermarking under the same format restriction). *We say that a watermarking scheme is secure under the same format restriction if it is a secure watermarking scheme that satisfies non-removability under the same format restriction instead of standard non-removability.*

Theorem 4.6. *Our watermarking scheme* $\mathsf{CWM}_{\mathsf{mult}}$ *for* $\epsilon = 1/\mathrm{poly}(\lambda)$ *is secure with same format under the DLIN.*

We prove the theorem by proving Theorems 4.7.

Theorem 4.7 (Non-Removability under the same format restriction). *Our scheme* $\mathsf{CWM}_{\mathsf{mult}}$ *satisfies non-removability with same format under the subspace assumption.*

Proof. If $\mathcal{A}$ outputs ek^*, where $\mathsf{Detect}(dk, ek^*) = \mathsf{unmarked}$ and $\mathsf{IdealDtc}(ek^*) = \mathsf{marked}$, then we construct algorithm $\mathcal{B}$, which solves the subspace problem with $k = 1$ and $n = 8$. $\mathcal{B}$ is given Γ, $D = ([\boldsymbol{b}_1], [\boldsymbol{b}_2], [\boldsymbol{b}_4], \ldots, [\boldsymbol{b}_8], [\eta\boldsymbol{b}_1^*], [\beta\boldsymbol{b}_2^*], [\boldsymbol{b}_3^*], \ldots, [\boldsymbol{b}_8^*], U_1 = [\mu_1\boldsymbol{b}_1 + \mu_2\boldsymbol{b}_2 + \mu_3\boldsymbol{b}_3], \mu_3)$, and Q_{b} where Q_{b} is $V_1 = [\tau_1\eta\boldsymbol{b}_1^* + \tau_2\beta\boldsymbol{b}_2^*]$ or $W_1 = [\tau_1\eta\boldsymbol{b}_1^* + \tau_2\beta\boldsymbol{b}_2^* + \tau_3\boldsymbol{b}_3^*]$. $\mathcal{B}$ chooses $\theta, \alpha', \sigma \leftarrow \mathbb{Z}_p$, sets

$$\begin{array}{llllllll} \boldsymbol{d}_1 := \boldsymbol{b}_3^* & \boldsymbol{d}_2 := \boldsymbol{b}_4^* & \boldsymbol{d}_3 := \boldsymbol{b}_5^* & \boldsymbol{d}_4 := \boldsymbol{b}_6^* & \boldsymbol{d}_5 := \boldsymbol{b}_7^* & \boldsymbol{d}_6 := \boldsymbol{b}_8^* & \boldsymbol{d}_7 := \boldsymbol{b}_1^* & \boldsymbol{d}_8 := \boldsymbol{b}_2^* \\ \boldsymbol{d}_1^* := \boldsymbol{b}_3 & \boldsymbol{d}_2^* := \boldsymbol{b}_4 & \boldsymbol{d}_3^* := \boldsymbol{b}_5 & \boldsymbol{d}_4^* := \boldsymbol{b}_6 & \boldsymbol{d}_5^* := \boldsymbol{b}_7 & \boldsymbol{d}_6^* := \boldsymbol{b}_8 & \boldsymbol{d}_7^* := \boldsymbol{b}_1 & \boldsymbol{d}_8^* := \boldsymbol{b}_2, \end{array}$$

and can generate $([\alpha\theta\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*]_T, [\boldsymbol{d}_1], \ldots, [\boldsymbol{d}_4]) := \alpha'\mu_3\theta \odot e([\boldsymbol{b}_4^*], [\boldsymbol{b}_4]), [\boldsymbol{b}_3^*], [\boldsymbol{b}_4^*], [\boldsymbol{b}_5^*], [\boldsymbol{b}_6^*])$ and $mk = ([\boldsymbol{d}_7], [\boldsymbol{d}_8]) := ([\boldsymbol{b}_1], [\boldsymbol{b}_2])$. $\mathcal{B}$ can compute a detection key since $[\eta\boldsymbol{b}_1^*], [\beta\boldsymbol{b}_2^*]$ are given and $\mathcal{B}$ can compute $([\eta\boldsymbol{b}_1^*])^{u_1'}([\beta\boldsymbol{b}_2^*])^{u_2'}$. $\mathcal{B}$ has $([\boldsymbol{d}_2^*], \ldots, [\boldsymbol{d}_8^*])$ but does not have $[\boldsymbol{d}_1^*]$ since $[\boldsymbol{b}_3]$ is not given. That is, $\mathcal{B}$ has the mark key and perfectly simulates the mark oracle. On the other hand, the secret key is incomplete as follows, $sk = (\perp, \perp, [\theta\boldsymbol{d}_2^*], [\sigma\boldsymbol{d}_3^*], [\sigma\boldsymbol{d}_4^*]) := (\perp, \perp, [\theta\boldsymbol{b}_4], [\sigma\boldsymbol{b}_5], [\sigma\boldsymbol{b}_6])$.

It implicitly holds $\alpha = \alpha'\mu_3$. To simulate the challenge oracle without the complete sk, for tag, $\mathcal{B}$ chooses $r_1', r_2, t_7, t_8 \leftarrow \mathbb{Z}_p$ and computes

$$\begin{aligned} \widetilde{\boldsymbol{k}}_{\mathsf{tag}} &:= ((\alpha' + r_1'\mathsf{tag})\theta \odot U_1) \oplus [-r_1'\mu_3\theta\boldsymbol{d}_2^* + r_2\mathsf{tag}\sigma\boldsymbol{d}_3^* - r_2\sigma\boldsymbol{d}_4^* + t_7\boldsymbol{d}_7^* + t_8\boldsymbol{d}_8^*] \\ &= [(\alpha + r_1\mathsf{tag})\theta\boldsymbol{d}_1^* - r_1\theta\boldsymbol{d}_2^* + r_2\mathsf{tag}\sigma\boldsymbol{d}_3^* - r_2\sigma\boldsymbol{d}_4^* + (t_7 - \theta(\alpha' + r_1'\mathsf{tag})\mu_1)\boldsymbol{d}_7^* + (t_8 - \theta(\alpha' + r_1'\mathsf{tag})\mu_2)\boldsymbol{d}_8^*]. \end{aligned}$$

We set $r_1 := \mu_3 r_1'$. This is a valid marked index. If $\mathcal{A}$ outputs valid unmarked index $ek^* = (\boldsymbol{U}^*, \boldsymbol{V}^*, \boldsymbol{k}^*_{\mathsf{tag}^*}, \mathsf{tag}^*) \in \mathbb{G}_T^{\ell^2} \times \mathbb{G}^{8\ell} \times \mathbb{G}^8 \times \mathbb{Z}_p$, then $\mathcal{B}$ computes $\Delta := \boldsymbol{e}(Q_{\mathsf{b}}, \boldsymbol{k}^*_{\mathsf{tag}^*})$.

There is a possibility that $\boldsymbol{k}^*_{\mathsf{tag}^*}$ does not include $[\boldsymbol{d}_1^*]$, that is, for some $s_1 \in \mathbb{Z}_p$, it may not hold $e([s_1\boldsymbol{d}_1], \boldsymbol{k}^*_{\mathsf{tag}^*}) = g_T^{s_1}$. However, this case does not happen with non-negligible probability due to the ϵ-approximation condition and the property of algorithm Test. Correctly computing $(\ell+1)$-th element of the output of the function for random input $\vec{x}$ is equivalent to that for some s_1 it holds $e([s_1\boldsymbol{d}_1 + s_1\mathsf{tag}^*\boldsymbol{d}_2 + s_2\boldsymbol{d}_3 + s_2\mathsf{tag}^*\boldsymbol{d}_4], \boldsymbol{k}^*_{\mathsf{tag}^*}) = s_1 \odot g_T$ where $g_T = [\alpha\theta\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*]_T$.

For randomly chosen x, it holds that $y_{\ell+1} := \sum_i x_i \odot (\boldsymbol{v}_i^*) = \sum_i [x_i s^*_{1,i}\boldsymbol{d}_1 + x_i s^*_{1,i}\mathsf{tag}\boldsymbol{d}_2 + x_i s^*_{2,i}\boldsymbol{d}_3 + x_i s^*_{2,i}\mathsf{tag}\boldsymbol{d}_4]$ and $\boldsymbol{e}(y_{\ell+1}, \boldsymbol{k}^*_{\mathsf{tag}^*}) = [\alpha\theta\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*\langle \boldsymbol{x}, \boldsymbol{s}_1^*\rangle]_T$ for some $\boldsymbol{s}_1^*, \boldsymbol{s}_2^* \in \mathbb{Z}_p^\ell$ with probability ϵ. We can consider $\langle \boldsymbol{x}, \boldsymbol{s}_1^*\rangle = s_1$.

If $\Delta = 1$, then $\mathcal{B}$ outputs 0 ($\mathsf{b} = 0$), otherwise, it outputs 1. $\mathcal{B}$ can output correct b. Analysis is as follows.

- If $Q_0 = [\tau_1\eta\boldsymbol{b}_1^* + \tau_2\beta\boldsymbol{b}_2^*] = [\tau_1\eta\boldsymbol{d}_7 + \tau_2\beta\boldsymbol{d}_8]$ is given, then $\Delta = 1$ since $\mathcal{A}$ succeeds removing the mark and $\boldsymbol{k}^*_{\mathsf{tag}^*}$ does not include vectors $\boldsymbol{d}_7^*$ and $\boldsymbol{d}_8^*$.
- If $Q_1 = [\tau_1\eta\boldsymbol{b}_1^* + \tau_2\beta\boldsymbol{b}_2^* + \tau_3\boldsymbol{b}_3^*] = [\tau_1\eta\boldsymbol{d}_7 + \tau_2\beta\boldsymbol{d}_8 + \tau_3\boldsymbol{d}_1]$ is given, then $\Delta \neq 1$ since $\boldsymbol{k}^*_{\mathsf{tag}^*}$ includes $[\boldsymbol{d}_1^*]$ with non-negligible probability due to the ϵ-approximation condition.

Thus, $\mathcal{B}$ breaks the problem. ■

5 Concluding Remarks

We introduced the notion of cryptographic watermarking schemes, defined its security notion, and proposed a concrete construction by using DPVS. It is secure with same format, which is a restriced security model, under the DLIN assumption. This gives us the first positive result about provably secure watermarking schemes. We can construct a similar scheme by using Okamoto-Takashima IPE scheme. We list a few remarks.

Constructions Based on the Symmetric External Diffie-Hellman Assumption. Chen, Lim, Ling, Wang, and Wee proposed an IBE scheme by using the subspace assumption based on the symmetric external Diffie-Hellman (SXDH) assumption, where the decisional Diffie-Hellman assumption holds in both groups of an asymmetric pairing group [CLL+13]. Their IBE scheme is similar to Lewko's scheme and we can apply our technique to their scheme. Thus, we can construct a more efficient watermarking scheme based on the SXDH assumption.

Constructions Based on Composite-Order Pairing Groups. We use the canceling property of DPVS and sub-group decision type assumption to prove the security. Composite-order pairing groups also have such properties [LW10, LOS+10]. Therefore, we can construct watermarking schemes based on composite-order pairing groups. However, we do not give concrete constructions in this paper since, generally speaking, schemes based on composite-order groups are less efficient than schemes based on prime-order groups due to large composites. One may think that we do not have remove algorithms if we use composite-order groups since we do not have trapdoor matrices of DPVS and the decomposition algorithm by Okamoto and Takashima. However, we note that if we use prime factors of composites as trapdoors, then we can also achieve remove algorithms in the composite-order group setting.

Open Issues. Our watermarking schemes are called the detection-type watermarking scheme, in which we can verify just one-bit information, embedded or not. We can consider a stronger variant called the extraction-type watermarking scheme, in which we can embed a message as a mark and extract it. In fact, our schemes can be modified into extraction-type schemes by adding extra $(2\mu - 2)$-dimension to our schemes for μ-bit messages since we can embed a one-bit message for each 2-dimension. However, this is quite inefficient. Thus, it is an open problem to construct more efficient extraction-type watermarking schemes.

Acknowledgments.

The author is grateful to Keita Xagawa for illuminating discussions and insightful comments about constructions of watermarking schemes. The author would like to thank Kazumaro Aoki and Atsushi Fujioka for helpful comments about applications of watermarking and the presentation of the introduction. The author would like to thank Shota Yamada for pointing out that we can use prime factors of a composite to construct a remove algorithm in the composite-order group setting. The author also would like to thank Masayuki Abe, Angelo De Caro, Akinori Kawachi, Katsuyuki Takashima, Maki Yoshida and anonymous reviewers for helpful comments.

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Dodis and Nielsen [DN15], pages 528–556. (Cited on page 11.)

[ADGM16] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over GGH13. *IACR Cryptology ePrint Archive*, 2016:1003, 2016. (Cited on page 4, 11.)

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 646–658. ACM Press, November 2014. (Cited on page 11.)

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Kilian [Kil01], pages 1–18. (Cited on page 1, 2, 4, 9, 11.)

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6, 2012. (Cited on page 1, 2, 4, 9, 11.)

[BGK$^+$14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, Heidelberg, May 2014. (Cited on page 11.)

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 1–25. Springer, Heidelberg, February 2014. (Cited on page 11.)

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444. Springer, Heidelberg, May / June 2006. (Cited on page 8.)

[CFL$^+$16] Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new CLT multilinear map over the integers. In Fischlin and Coron [FC16], pages 509–536. (Cited on page 4, 11.)

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Heidelberg, August 1994. (Cited on page 2.)

[CGH$^+$15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 247–266. Springer, Heidelberg, August 2015. (Cited on page 4, 11.)

[CGH16] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. Cryptology ePrint Archive, Report 2016/998, 2016. `http://eprint.iacr.org/2016/998`. (Cited on page 4, 11.)

[CHL$^+$15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Heidelberg, April 2015. (Cited on page 4, 11.)

[CHN$^+$16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1115–1127. ACM Press, June 2016. (Cited on page 4.)

[CLL$^+$13] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter IBE and signatures via asymmetric pairings. In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012*, volume 7708 of *LNCS*, pages 122–140. Springer, Heidelberg, May 2013. (Cited on page 16.)

[CLLT16a] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In Robshaw and Katz [RK16], pages 607–628. (Cited on page 4.)

[CLLT16b] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. *IACR Cryptology ePrint Archive*, 2016:1011, 2016. (Cited on page 4, 11.)

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, Heidelberg, August 2013. (Cited on page 4, 11.)

[DN15] Yevgeniy Dodis and Jesper Buus Nielsen, editors. *TCC 2015, Part II*, volume 9015 of *LNCS*. Springer, Heidelberg, March 2015. (Cited on page 16, 18.)

[FC16] Marc Fischlin and Jean-Sébastien Coron, editors. *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*. Springer, Heidelberg, May 2016. (Cited on page 17, 18.)

[FGK$^+$10] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 279–295. Springer, Heidelberg, May 2010. (Cited on page 3, 11.)

[FRS16] Rex Fernando, Peter M. R. Rasmussen, and Amit Sahai. Preventing clt zeroizing attacks on obfuscation. *IACR Cryptology ePrint Archive*, 2016:1070, 2016. (Cited on page 4, 11.)

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013. (Cited on page 4, 11.)

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Cited on page 4, 11.)

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Dodis and Nielsen [DN15], pages 498–527. (Cited on page 4, 11.)

[GMM$^+$16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. volume 9986 of *Lecture Notes in Computer Science*, pages 241–268, 2016. (Cited on page 4, 11.)

[GV08] Steven D. Galbraith and Eric R. Verheul. An analysis of the vector decomposition problem. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 308–327. Springer, Heidelberg, March 2008. (Cited on page 3.)

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In Fischlin and Coron [FC16], pages 537–565. (Cited on page 4, 11.)

[HMW07] Nicholas Hopper, David Molnar, and David Wagner. From weak to strong watermarking. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 362–382. Springer, Heidelberg, February 2007. (Cited on page 1, 2, 9, 10.)

[HO12] Brett Hemenway and Rafail Ostrovsky. Extended-DDH and lossy trapdoor functions. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 627–643. Springer, Heidelberg, May 2012. (Cited on page 3, 11.)

[Kil01] Joe Kilian, editor. *CRYPTO 2001*, volume 2139 of *LNCS*. Springer, Heidelberg, August 2001. (Cited on page 16.)

[KY02] Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 450–465. Springer, Heidelberg, April / May 2002. (Cited on page 2.)

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In Pointcheval and Johansson [PJ12], pages 318–335. (Cited on page 3, 4, 7, 8, 20.)

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May 2010. (Cited on page 4, 16.)

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2), 1988. (Cited on page 2.)

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. Cryptology ePrint Archive, Report 2016/795, 2016. `http://eprint.iacr.org/2016/795`. (Cited on page 5.)

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, February 2010. (Cited on page 16.)

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Robshaw and Katz [RK16], pages 629–658. (Cited on page 4, 11.)

[NSS99] David Naccache, Adi Shamir, and Julien P. Stern. How to copyright a function? In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 188–196. Springer, Heidelberg, March 1999. (Cited on page 1, 2.)

[NW15] Ryo Nishimaki and Daniel Wichs. Watermarking cryptographic programs against arbitrary removal strategies. Cryptology ePrint Archive, Report 2015/344, 2015. http://eprint.iacr.org/2015/344. (Cited on page 10.)

[OT08] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 57–74. Springer, Heidelberg, September 2008. (Cited on page 3, 6, 7.)

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231. Springer, Heidelberg, December 2009. (Cited on page 3, 4, 5.)

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010. (Cited on page 3, 4, 5.)

[OT11] Tatsuaki Okamoto and Katsuyuki Takashima. Decentralized attribute-based signatures. Cryptology ePrint Archive, Report 2011/701, 2011. http://eprint.iacr.org/2011/701. (Cited on page 4.)

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In Pointcheval and Johansson [PJ12], pages 591–608. (Cited on page 3, 4.)

[PJ12] David Pointcheval and Thomas Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*. Springer, Heidelberg, April 2012. (Cited on page 18, 19.)

[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. ACM Press, May 2008. (Cited on page 2, 3.)

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM J. Comput.*, 40(6):1803–1844, 2011. (Cited on page 5, 11, 13.)

[RK16] Matthew Robshaw and Jonathan Katz, editors. *CRYPTO 2016, Part II*, volume 9815 of *LNCS*. Springer, Heidelberg, August 2016. (Cited on page 17, 19.)

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990. (Cited on page 2.)

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009. (Cited on page 3.)

[YF11] Maki Yoshida and Toru Fujiwara. Toward digital watermarking for cryptographic data. *IEICE Transactions*, 94-A(1):270–272, 2011. (Cited on page 1, 8.)

[YMF10] Maki Yoshida, Shigeo Mitsunari, and Toru Fujiwara. The vector decomposition problem. *IEICE Transactions*, 93-A(1):188–193, 2010. (Cited on page 1, 3, 7.)

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 439–467. Springer, Heidelberg, April 2015. (Cited on page 11.)

A Lewko IBE Scheme

We review Lewko IBE scheme, $\mathsf{IBE_L}$ [Lew12] in this section.

$\mathsf{Setup}(1^\lambda)$**:** It generates $\Lambda := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$ and $(\boldsymbol{D}, \boldsymbol{D}^*) \xleftarrow{\mathsf{U}} \mathsf{Dual}(\mathbb{Z}_p^6)$, chooses $\alpha, \theta, \sigma \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, and sets $pk := (\Lambda, [\alpha\theta\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*]_T, [\boldsymbol{d}_1], \ldots, [\boldsymbol{d}_4])$, $msk := ([\theta\boldsymbol{d}_1^*], [\alpha\boldsymbol{d}_1^*], [\theta\boldsymbol{d}_2^*], [\sigma\boldsymbol{d}_3^*], [\sigma\boldsymbol{d}_4^*])$. It outputs (pk, msk).

$\mathsf{Gen}(msk, \mathsf{id})$**:** It chooses $r_1, r_2 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and generates $sk_{\mathsf{id}} := [(\alpha + r_1\mathsf{id})\theta\boldsymbol{d}_1^* - r_1\theta\boldsymbol{d}_2^* + r_2\mathsf{id}\sigma\boldsymbol{d}_3^* - r_2\sigma\boldsymbol{d}_4^*]$.

$\mathsf{Enc}(pk, \mathsf{id}, M)$**:** It chooses $s_1, s_2 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and generates $C_0 := M \oplus ([\alpha\theta\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*]_T)^{s_1}$ and $C := [s_1\boldsymbol{d}_1 + s_1\mathsf{id}\boldsymbol{d}_2 + s_2\boldsymbol{d}_3 + s_2\mathsf{id}\boldsymbol{d}_4]$. It outputs ciphertext $\mathsf{ct} := (C_0, C)$.

$\mathsf{Dec}(sk_{\mathsf{id}}, \mathsf{ct})$**:** It outputs $M := C_0/\boldsymbol{e}(sk_{\mathsf{id}}, C)$.

Theorem A.1 ([Lew12]). *If the DLIN assumption holds, then* $\mathsf{IBE_L}$ *is adaptively secure against chosen plaintext attacks.*