How to Watermark Cryptographic Functions *

Ryo Nishimaki
NTT
Secure Platform Laboratories
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. Our watermarking for cryptographic functions is a generalized notion of copyrighted functions introduced by Naccache, Shamir, and Stern (PKC 1999) and our scheme is based on an identity-based encryption scheme whose private keys for identities (i.e., decryption functions) are marked, so our technique can be used to construct black-box traitor tracing schemes.

Keywords: digital watermarking, dual pairing vector space, dual system encryption, vector decomposition problem

*This is the revised full version of "How to Watermark Cryptographic Functions" that appeared in Eurocrypt 2013 [Nis13]. We added the full proof and revised some definitions.

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+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+12]. "General-purpose" means that program watermarking/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.

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.

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

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 [CT02], 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$^+$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) [PW11]. 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) [PW11]. 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* ciphertexts/keys and the second type is called *semi-functional* ciphertexts/keys. 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 ciphertexts/keys are computationally indistinguishable from semi-functional ciphertexts/keys. In most cases, function indices of LTFs consist of *ciphertexts of homomorphic encryption* [FGK$^+$10, HO12, PW11], 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 $\mathbb{B}$ and $\mathbb{B}^*$. 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 $\mathbb{B}/\mathbb{B}^*$ for semi-functional ciphertexts/keys as Okamoto-Takashima and Lewko IBE schemes [Lew12, OT10, OT12]. We denote the subset by $\widehat{\mathbb{B}} \subset \mathbb{B}$, $\widehat{\mathbb{B}}^* \subset \mathbb{B}^*$, respectively. A hidden linear subspace for semi-functional ciphertexts and keys 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 $\mathbb{B}$ (resp, $\mathbb{B}^*$) or $\mathbb{B} \setminus \widehat{\mathbb{B}}$ (resp, $\mathbb{B}^* \setminus \widehat{\mathbb{B}}^*$).

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 ciphertexts/keys into normal ciphertexts/keys 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 signatures [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 use an assumption about outputs of adversaries. The assumption says adversaries are not allowed to change the format of functions. Adversaries cannot use iO by the assumption. Someone think the assumption is strong, but we can say our construction is an alternative approach to achieve watermarking since Nishimaki and Wichs [NW15] and Cohen, Holmgren, and Vaikuntanathan [CHV15] proposed program watermarking based on iO in their independent and concurrent works. Candidate constructions of iO are known [GGH$^+$13b], but their underlying cryptographic tool called multilinear maps [GGH13a, CLT13] were attacked [CHL$^+$15, HJ15]. 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, $y \xleftarrow{\mathsf{R}} D$ denote that y is randomly selected from D according to its distribution. If S is a set, then $x \xleftarrow{\mathsf{U}} S$ denotes that x is uniformly selected from S. $y := z$ denotes that y is set, defined or substituted by z. When b is a fixed value, $A(x) \to b$ (e.g., $A(x) \to 1$) denotes the event that machine (or algorithm) A outputs a on input x. 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 $f < \mathsf{negl}(\lambda)$ to mean that f is negligible in λ. A vector symbol denotes a vector representation over $\mathbb{Z}_p$, e.g., $\vec{x}$ denotes $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$. For two vectors $\vec{x}$ and $\vec{v}$, $\langle \vec{x}, \vec{v} \rangle$ denotes the inner-product $\sum_{i=1}^{n} x_i v_i$. The transpose of matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{X}^\top$. A bold face small letter denotes an element of a vector space $\mathbb{V}$, e.g., $\boldsymbol{x} \in \mathbb{V}$. Set $GL(n, \mathbb{Z}_p)$ denotes the general linear group of degree n over $\mathbb{Z}_p$. 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

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

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,

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

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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ 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 $(p, \mathbb{G}, \mathbb{G}_T, e, g)$.

2.2 Function Family of Lossy Trapdoor Functions

Definition 2.3 (Lossy Trapdoor Functions [PW08, 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 (ek, ik) where ek and ik are an evaluation and an inversion key, respectively.

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

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

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

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

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 $ek' \overset{\mathsf{R}}{\leftarrow} \mathsf{LTF.LGen}(1^\lambda)$, the image set $f_{ek'}(\mathsf{D})$ is of size at most $|\mathsf{D}|/2^\ell$.

We define a function family of LTF, $\mathsf{LTF}_\lambda := \{\mathsf{LTF.Eval}_{ek}(\cdot) | (ek, ik) \overset{\mathsf{R}}{\leftarrow} \mathsf{LTF.Gen}(1^\lambda, b), b \in \{0,1\}\}$ 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 [OT09, OT10]

For $\vec{v}, \vec{w} \in \mathbb{Z}_p^n$, $a \in \mathbb{Z}_p$, and $g \in \mathbb{G}$, we define $g^{\vec{v}} := (g^{v_1}, \ldots, g^{v_n})$, $(g^{\vec{v}})^a := g^{a\vec{v}} = (g^{av_1}, \ldots, g^{av_n})$, and $g^{\vec{v}+\vec{w}} := (g^{v_1+w_n}, \ldots, g^{v_n+w_n})$.

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. Number p is a prime. A cyclic group $\mathbb{G}_T$ of order p comes from pairings.

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 $\boldsymbol{x} := (g^{x_1}, \ldots, g^{x_N})$ ($x_i \in \mathbb{Z}_p$ for $i = 1, \ldots, N$).

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

Pairing operation: A pairing function $\boldsymbol{e} : \mathbb{V} \times \mathbb{V} \to \mathbb{G}_T$ is defined by $\boldsymbol{e}(\boldsymbol{x}, \boldsymbol{y}) := \prod_{i=1}^{N} e(g_i, h_i) \in \mathbb{G}_T$ where $\boldsymbol{x} := (g_1, \ldots, g_N) \in \mathbb{V}$ and $\boldsymbol{y} := (h_1, \ldots, h_N) \in \mathbb{V}$. This is non-degenerate bilinear, i.e., $\boldsymbol{e}(\boldsymbol{x}^s, \boldsymbol{y}^t) = e(\boldsymbol{x}, \boldsymbol{y})^{st}$ and if $\boldsymbol{e}(\boldsymbol{x}, \boldsymbol{y}) = 1$ for all $\boldsymbol{y} \in \mathbb{V}$, then $\boldsymbol{x} = \boldsymbol{1}$. For all i and j, $\boldsymbol{e}(\boldsymbol{a}_i, \boldsymbol{a}_j) = e(g, g)^{\delta_{i,j}}$ 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 $\phi_{i,j}(\boldsymbol{a}_k) = \boldsymbol{1}$ if $k \neq j$, which can be easily achieved by $\phi_{i,j}(\boldsymbol{x}) := (\overbrace{1, \ldots, 1}^{i-1}, g_j, \overbrace{1, \ldots, 1}^{N-i})$ where $\boldsymbol{x} := (g_1, \ldots, g_N)$. We call $\phi_{i,j}$ canonical maps. The DPVS generation algorithm $\mathcal{G}_{\mathsf{dpvs}}$ takes input 1^λ and $N \in \mathbb{N}$, and outputs a description of $\mathsf{param}'_{\mathbb{V}} := (p, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ with security parameter λ and N-dimensional $\mathbb{V}$. It can be constructed using $\mathcal{G}_{\mathsf{bmp}}$.

Lewko [Lew12] defined algorithm $\mathsf{Dual}(\mathbb{Z}_p^n)$ as follows. It chooses $\vec{b}_i, \vec{b}_j^* \in \mathbb{Z}_p^n$ and $\psi \xleftarrow{\cup} \mathbb{Z}_p^*$ such that $\langle \vec{b}_i, \vec{b}_j^* \rangle = 0 \bmod p$ for $i \neq j$, $\langle \vec{b}_i, \vec{b}_i^* \rangle = \psi \bmod p$ for all $i \in [n]$ and outputs $(\mathfrak{B}, \mathfrak{B}^*)$ where $\mathfrak{B} := (\vec{b}_1, \ldots, \vec{b}_n)$ and $\mathfrak{B}^* := (\vec{b}_1^*, \ldots, \vec{b}_n^*)$. We consider $\mathsf{Dual}(\mathbb{Z}_p^n)$ implicitly outputs $\psi \in \mathbb{Z}_p$ though we omit it.

We describe a random dual orthonormal bases generator $\mathcal{G}_{\mathsf{ob}}(1^\lambda, N)$, which is used as a subroutine in the proposed scheme. We use a notation by Lewko [Lew12] instead of that of Okamoto

and Takashima.

$$
\begin{aligned}
&\underline{\mathcal{G}_{\mathsf{ob}}(1^\lambda, N)}\\
&\mathsf{param}'_{\mathbb{V}} := (p, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, N),\\
&(\mathfrak{B}, \mathfrak{B}^*) \xleftarrow{\mathsf{R}} \mathsf{Dual}(\mathbb{Z}_p^n),\\
&g_T := e(g,g)^\psi,\ \mathsf{param}_{\mathbb{V}} := (\mathsf{param}'_{\mathbb{V}}, g_T),\\
&\boldsymbol{b}_i := g^{\vec{b}_i}, \mathbb{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N),\\
&\boldsymbol{b}^*_j := g^{\vec{b}^*_j}, \mathbb{B}^* := (\boldsymbol{b}^*_1, \ldots, \boldsymbol{b}^*_N),\\
&\text{return } (\mathsf{param}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^*).
\end{aligned}
$$

We use the notation $(\mathfrak{B}, \mathfrak{B}^*)$ to express dual orthonormal bases in $\mathbb{Z}_p$ to distinguish from dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ in $\mathbb{V}$.

We briefly explain some important properties of DPVS [OT10, Lew12].

Pairing operation: $\boldsymbol{e}(\boldsymbol{x}, \boldsymbol{y}) := \prod_{i=1}^N e(g^{x_i}, g^{y_i}) = e(g,g)^{\sum_{i=1}^N x_i y_i} = e(g,g)^{\vec{x}\cdot\vec{y}} \in \mathbb{G}_T$, where $\boldsymbol{x} := (g^{x_1}, \ldots, g^{x_N})$ and $\boldsymbol{y} := (g^{y_1}, \ldots, g^{y_N})$. Here, $\boldsymbol{x}$ and $\boldsymbol{y}$ are expressed by coefficient vectors over basis $\mathbb{A}$ such that $(x_1, \ldots, x_N)_{\mathbb{A}} = (\vec{x})_{\mathbb{A}} := \prod_{i=1}^N \boldsymbol{a}_i^{x_i}$ and $(y_1, \ldots, y_N)_{\mathbb{A}} = (\vec{y})_{\mathbb{A}} := \prod_{i=1}^N \boldsymbol{a}_i^{y_i}$.

Base change: Canonical basis $\mathbb{A}$ is changed to *dual orthonormal bases* $\mathbb{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N)$ and $\mathbb{B}^* := (\boldsymbol{b}^*_1, \ldots, \boldsymbol{b}^*_N)$ of $\mathbb{V}$ using a uniformly chosen (regular) linear transformation, $\boldsymbol{X} := (\chi_{i,j}) \xleftarrow{\mathsf{U}} GL(N, \mathbb{Z}_p)$, such that $\boldsymbol{b}_i = \prod_{j=1}^N \boldsymbol{a}_j^{\chi_{i,j}} = (g^{\chi_{i,1}}, \ldots, g^{\chi_{i,N}})$, $\boldsymbol{b}^*_i = \prod_{j=1}^N \boldsymbol{a}_j^{\vartheta_{i,j}} = (g^{\vartheta_{i,1}}, \ldots g^{\vartheta_{i,N}})$ $(i = 1, \ldots, N)$, and $(\vartheta_{i,j}) := \psi(\boldsymbol{X}^\top)^{-1}$ where $\psi \xleftarrow{\mathsf{U}} \mathbb{Z}_p^*$. It holds that $\boldsymbol{e}(\boldsymbol{b}_i, \boldsymbol{b}^*_j) = e(g,g)^{\delta_{i,j}}$ ($\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ if $i \neq j$). Here, $\boldsymbol{x} := \boldsymbol{b}_1^{x_1} \times \cdots \times \boldsymbol{b}_N^{x_N} \in \mathbb{V}$ and $\boldsymbol{y} := (\boldsymbol{b}^*_1)^{y_1} \times \cdots \times (\boldsymbol{b}^*_N)^{y_N} \in \mathbb{V}$ can be expressed by coefficient vectors over basis $\mathbb{B}$ and $\mathbb{B}^*$ such that $(x_1, \ldots, x_N)_{\mathbb{B}} = (\vec{x})_{\mathbb{B}}$ and $(y_1, \ldots, y_N)_{\mathbb{B}^*} = (\vec{y})_{\mathbb{B}^*}$, and $\boldsymbol{e}(\boldsymbol{x}, \boldsymbol{y}) = e(g,g)^{\sum_{i=1}^N x_i y_i} = e(g,g)^{\vec{x}\cdot\vec{y}} \in \mathbb{G}_T$.

Intractable problem: A decisional problem in this approach is the decisional subspace problem [OT08]. It is to tell $\boldsymbol{v} = (0, \ldots, 0, v_{N_2+1}, \ldots, v_{N_1})_{\mathbb{B}}$ from $\boldsymbol{u} = (v_1, \ldots, v_{N_1})_{\mathbb{B}}$ when an element in $\mathbb{V}$ (N_1 dimension) is given, where $(v_1, \ldots v_{N_1}) \xleftarrow{\mathsf{U}} \mathbb{Z}_p^{N_1}$ 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 $\boldsymbol{v} := (0, \ldots, 0, v_{N_2+1}, \ldots, v_{N_1})_{\mathbb{B}}$ or $\boldsymbol{u} := (v_1, \ldots, v_{N_1})_{\mathbb{B}}$, we can tell $\boldsymbol{v}$ from $\boldsymbol{u}$ using $\boldsymbol{t}^*$ since $e(\boldsymbol{v}, \boldsymbol{t}^*) = 1$ and $e(\boldsymbol{u}, \boldsymbol{t}^*) \neq 1$ with high probability.

Advantage of this approach: It is easy to decompose $\boldsymbol{a}_i^{x_i} = (1, \ldots, 1, g^{x_i}, 1, \ldots, 1)$ from $\boldsymbol{x} := \boldsymbol{a}_1^{x_1} \times \cdots \times \boldsymbol{a}_N^{x_N} = (x_1, \ldots, x_N)_{\mathbb{A}}$. In contrast, the DPVS approach employs basis $\mathbb{B}$, which is linearly transformed from $\mathbb{A}$ using a secret random matrix $\boldsymbol{X} \in \mathbb{Z}_p^{n \times n}$. It seems hard to decompose $\boldsymbol{b}_i^{x_i}$ from $\boldsymbol{x}' := (x_1, \ldots, x_N)_{\mathbb{B}}$ (and the decisional subspace problem seems

intractable). In addition, the secret matrix $\boldsymbol{X}$ (and the dual orthonormal basis $\mathbb{B}^*$ of $\mathbb{V}$) can be used as trapdoors for the decomposability (and distinguishability for the decisional subspace problem through the pairing operation over $\mathbb{B}$ and $\mathbb{B}^*$).

Parameter Hiding. Let $m \leq n$ be a fixed positive integer and $A \in \mathbb{Z}_p^{m \times m}$ be an invertible matrix. Let $S_m \subseteq [n]$ be a subset of size m. Lewko proposed how to obtain new dual orthonormal bases $(\mathfrak{B}_A, \mathfrak{B}_A^*)$ from $(\mathfrak{B}, \mathfrak{B}^*)$. If B_m is an $n \times m$ matrix over $\mathbb{Z}_p$ whose columns are vectors $\vec{b}_i \in \mathfrak{B}$ such that $i \in S_m$, then $B_m A$ is also an $n \times m$ matrix. Let $\mathfrak{B}_A := (\vec{a}_1, \ldots, \vec{a}_n)$ where $\vec{a}_i := \vec{b}_i$ for all $i \notin S_m$ and $\vec{a}_i := (B_m A)_\ell$ for $i \in S_m$, i is the ℓ-th element of S_m and $(B_m A)_\ell$ denotes the ℓ-th column of $B_m A$. If B_m^* is $n \times m$ matrix over $\mathbb{Z}_p$ whose columns are vectors $\vec{b}_i^* \in \mathfrak{B}^*$ such that $i \in S_m$, then $B_m (A^{-1})^\top$ is also $n \times m$ matrix. Let $\mathfrak{B}_A^* := (\vec{a}_1^*, \ldots, \vec{a}_n^*)$ where $\vec{a}_i^* := \vec{b}_i^*$ for all $i \notin S_m$ and $\vec{a}_i^* := (B_m^* (A^{-1})^\top)_\ell$ for $i \in S_m$, i is the ℓ-th element of S_m and $(B_m (A^{-1})^\top)_\ell$ denotes the ℓ-th column of $B_m (A^{-1})^\top$. 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 $A \in \mathbb{Z}_p^{m \times m}$ and set $S_M \subseteq [n]$ of size m, if $(\mathfrak{B}, \mathfrak{B}^*) \xleftarrow{R} \mathsf{Dual}(\mathbb{Z}_p^n)$, then $(\mathfrak{B}_A, \mathfrak{B}_A^*)$ is also distributed as a random sample from $\mathsf{Dual}(\mathbb{Z}_p^n)$ and its distribution is independent of 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 [OT08]) Let λ be a security parameter and $\mathcal{G}_{\mathsf{ob}}$ be an algorithm that outputs a description of a ℓ_1-dimensional DPVS $(p, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ and $\ell_1 > \ell_2$. Let $\mathcal{A}$ be a PPT machine. For all $\lambda \in \mathbb{N}$, we define the $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ advantage of $\mathcal{A}$ as

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

The $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ assumption: For all 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 $\mathbb{A}$ be a canonical basis of $\mathbb{V}$, and $\mathsf{CVDP}^{\mathbb{A}}_{(\ell_1,\ell_2)}$ be a specific class of $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ in which $\mathbb{B}$ is replaced by $\mathbb{A}$. The canonical maps $\phi_{i,j}$ on $\mathbb{V}$ can solve $\mathsf{CVDP}^{\mathbb{A}}_{(\ell_1,\ell_2)}$ in polynomial time.*

Trapdoor. If we have a trapdoor, linear transformation matrix $\boldsymbol{X}$ behind $\mathbb{B}$, 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 $(\boldsymbol{v}, (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_2}), \boldsymbol{X}, \mathbb{B})$ such that $\boldsymbol{v} := \prod_{i=1}^{\ell_1} \boldsymbol{b}_i^{y_i}$ is a target vector for decomposition, $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_2})$ is a subspace to be decomposed into, $\boldsymbol{X}$ is a trapdoor (matrix), and $\mathbb{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_1})$ is a basis generated by using $\boldsymbol{X}$. Algorithm $\mathsf{Decomp}(\boldsymbol{v}, (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_2}), \boldsymbol{X}, \mathbb{B})$: computes $\boldsymbol{u} := \prod_{i=1}^{\ell_1} \prod_{j=1}^{\ell_2} \prod_{\kappa=1}^{\ell_1} (\phi_{\kappa,i}(\boldsymbol{v}))^{\tau_{i,j}\chi_{j,\kappa}}$ where ϕ is the canonical map in Definition 2.4, $(\chi_{i,j}) = \boldsymbol{X}$ and $(\tau_{i,j}) := (\boldsymbol{X})^{-1}$.

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

2.4 Complexity Assumptions

Definition 2.9 (DLIN Assumption) The DLIN problem is to guess $\beta \in \{0,1\}$, given $(\Gamma, g, f, h, f^{\delta}, h^{\sigma}, Q_\beta) \xleftarrow{\mathsf{R}} \mathcal{G}_\beta^{\mathsf{dlin}}(1^\lambda)$, where $\mathcal{G}_\beta^{\mathsf{dlin}}(1^\lambda)$: $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$, $\xi, \kappa, \delta, \sigma \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, $f := g^\xi, h := g^\kappa$, $Q_0 := g^{\delta+\sigma}$, $Q_1 \xleftarrow{\mathsf{U}} \mathbb{G}$, return $\mathcal{I} := (\Gamma, f, h, f^\delta, h^\sigma, Q_\beta)$. This advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dlin}}(\lambda)$ is defined as follows.

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

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

Definition 2.10 (Subspace Assumption) The subspace problem is to guess $\beta \in \{0,1\}$, given (Γ, D, Q_β), where $\mathcal{G}_\beta^{\mathsf{dss}}(1^\lambda)$: $\Gamma \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$, $(\mathfrak{B}, \mathfrak{B}^*) \xleftarrow{\mathsf{U}} \mathsf{Dual}(\mathbb{Z}_p^n)$, $\eta, \beta, \tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, for $i \in [k]$, $U_i := g^{\mu_1 \vec{b}_i + \mu_2 \vec{b}_{k+i} + \mu_3 \vec{b}_{2k+i}}$ and

$$V_i := g^{\tau_1 \eta \vec{b}_i^* + \tau_2 \beta \vec{b}_{k+i}^*} \qquad W_i := g^{\tau_1 \eta \vec{b}_i^* + \tau_2 \beta \vec{b}_{k+i}^* + \tau_3 \vec{b}_{2k+i}^*}$$
$$Q_0 := (V_1, \ldots, V_k) \qquad Q_1 := (W_1, \ldots, W_k)$$

$D := (g^{\vec{b}_1}, \ldots, g^{\vec{b}_{2k}}, g^{\vec{b}_{3k+1}}, \ldots, g^{\vec{b}_n}, g^{\eta \vec{b}_1^*}, \ldots, g^{\eta \vec{b}_k^*}, g^{\beta \vec{b}_{k+1}^*}, \ldots, g^{\beta \vec{b}_{2k}^*}, g^{\vec{b}_{2k+1}^*}, \ldots, g^{\vec{b}_n^*}, U_1, \ldots, U_k, \mu_3)$, return $\mathcal{I} := (\Gamma, D, Q_\beta)$. This advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dss}}(\lambda)$ is defined as follows.

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

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

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 $(\Gamma, g, g^a, g^b, g^c, Q_\beta) \xleftarrow{\mathsf{R}} \mathcal{G}_\beta^{\mathsf{dbdh}}(1^\lambda)$, where $\mathcal{G}_\beta^{\mathsf{dbdh}}(1^\lambda)$: $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$, $a, b, c \xleftarrow{\mathsf{U}} \mathbb{Z}_p$,

$Q_0 := e(g,g)^{abc}$, $Q_1 \xleftarrow{\cup} \mathbb{G}_T$, return $(\Gamma, g, g^a, g^b, g^c, Q_\beta)$. This advantage $\mathsf{Adv}^{\mathsf{dbdh}}_{\mathcal{A}}(\lambda)$ is defined as follows.

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

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

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* $\mathsf{Adv}^{\mathsf{dbdh}}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{B}}$.

Proof. Given DLIN instance $(\Gamma, g, g^\xi, g^\kappa, g^{\delta\xi}, g^{\sigma\kappa}, Q)$, adversary $\mathcal{B}$ for the DLIN problem gives adversary $\mathcal{A}$ for the DBDH problem tuple $(\Gamma, g, g^\kappa, g^\xi, Q, T := e(g^{\delta\xi}, g^\kappa) \cdot e(g^{\sigma\kappa}, g^\xi))$ as a DBDH instance. $T = e(g,g)^{\kappa\xi(\delta+\sigma)}$, so if $Q = g^{\delta+\sigma}$, then the tuple is the same as $\mathcal{G}^{\mathsf{dbdh}}_0$. It implicitly holds that $a = \kappa$, $b = \xi$, $c = \delta + \sigma$, $abc = \kappa\xi(\delta+\sigma)$. If $Q = g^\zeta$ is a uniformly random element in $\mathbb{G}$, then $T = e(g,g)^{\kappa\xi(\delta+\sigma)}$ is a uniformly random element in $\mathbb{G}_T$ and the tuple is the same as $\mathcal{G}^{\mathsf{dbdh}}_1$ since δ and σ are uniformly random and independent of ζ, ξ, and κ. □

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 $\mathsf{LTF}_\lambda := \{\mathsf{LTF.Eval}_{ek}(\cdot) | (ek, ik) \xleftarrow{R} \mathsf{LTF.Gen}(1^\lambda, b), b \in \{0,1\}\}$.

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 $F \xleftarrow{R} \mathcal{F}_\lambda$ (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 $F \xleftarrow{R} \mathcal{F}_\lambda$ 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.

- 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.

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

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 $\mathsf{WM}(F)$ 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, $(sk, mk, dk, rk) \xleftarrow{\mathsf{R}} \mathsf{WMGen}(1^{\lambda})$.

$\mathsf{Samp}_{\mathcal{F}}$**:** 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, $\widetilde{F} \xleftarrow{\mathsf{R}} \mathsf{Mark}(mk, F)$.

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

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+12]. We borrow some terms from these studies [BGI+12, HMW07]. 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 $\widetilde{F}$, 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 $f' \cong_\varepsilon f$, if $\Pr_{x \leftarrow \{0,1\}^n}[f'(x) = f(x)] \geq \varepsilon$.

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 $(sk, mk, dk, rk) \xleftarrow{R} \mathsf{WMGen}(1^\lambda)$ and $F \xleftarrow{R} \mathsf{Samp}_{\mathcal{F}}(sk)$, it holds that $\mathsf{Detect}(dk, F) \to \mathsf{unmarked}$.

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

Preserving Functionality: For any input $x \in \{0,1\}^n$ and $F \in \mathcal{F}_\lambda$, it holds that $\mathsf{WM}(F)(x) = F(x)$. 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 $F \in \mathcal{F}_\lambda$, $|\mathsf{WM}(F)| \leq p(|F| + |mk|)$.

Non-Removability: We say that a watermarking scheme satisfies non-removability (or is non-removable) if it holds that $\mathsf{Adv}^{\mathsf{nrmv}}_{\mathcal{F},\mathcal{A}}(1^\lambda, \epsilon) := \Pr[\mathsf{Exp}^{\mathsf{nrmv}}_{\mathcal{F},\mathcal{A}}(\lambda, \epsilon) \to \mathsf{win}] < \mathsf{negl}(\lambda)$ 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.

On the security experiments. In our construction, the detection key dk is public, given to $\mathcal{A}$, and we need not the detection oracle $\mathcal{DO}$. For generality, we write the case that dk is not public.

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, detect, and challenge oracles. The mark oracle returns a marked function for a queried non-marked function. The detect oracle determines whether a queried function is marked or not. 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

[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.

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

Oracle $\mathcal{MO}(F)$

$F' \xleftarrow{R} \mathsf{Mark}(mk,F)$;
$\mathsf{MList} := \mathsf{MList} \cup \{F'\}$;
return F';

Oracle $\mathcal{DO}(F)$

$\mathsf{Detect}(dk,F) \to b$;
return b

Oracle $\mathcal{CO}_{\mathcal{F}_\lambda}()$

$F \xleftarrow{R} \mathcal{F}_\lambda$;
$F' \xleftarrow{R} \mathsf{Mark}(mk,F)$;
$\mathsf{CList} := \mathsf{CList} \cup \{F'\}$;
$\mathsf{MList} := \mathsf{MList} \cup \{F'\}$;
return F'

Procedure $\mathsf{IdealDtc}(F)$

if $(\exists F' \in \mathsf{CList} : F \cong_\epsilon F')$; then return $\{\mathsf{marked}\}$
else if $(\exists F' \in \mathsf{MList} : F \cong_\epsilon F')$ then return $\{\mathsf{marked},\mathsf{unmarked}\}$
else return $\{\mathsf{unmarked}\}$

Figure 1: Experiment for non-removability

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 [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 returns $\mathsf{unmarked}$ but the ideal detect procedure returns $\{\mathsf{marked},\mathsf{unmarked}\}$ since an equivalent function is recorded in MList.

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

Inputs: Two functions f_0 and f_1.

Parameters: $0 < \epsilon < 1$.

Set $\mathsf{cnt} := 0$ and $R := 8\lambda(1/\epsilon)^2$. For $i = 1, \ldots, R$, do

1. Choose $z_i \leftarrow \mathsf{D}$ where 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

to construct watermarking schemes that satisfies the preserving functionality if we assume the existence of iO [BGI$^+$12]. To avoid the impossibility result by Barak et al., we restrict adversaries. When we prove the security of our watermarking scheme, we assume that *adversaries must output a function that follows the format of the original functions*. More concretely, in our scheme, a function consists of group elements and one integer and adversaries must output group elements and one integer as a function description to win the security game. Thus, we can avoid the impossibility results. We call this restriction *the same format assumption* in this paper. If arbitrary strategies are allowed, adversaries can use iO to attack our watermarking scheme since obfuscation plays a role of mark-remover [BGI$^+$01, BGI$^+$12] as we explained in Section 1.3. However, they can not use it under the same format assumption since obfuscated functions by the candidate constructions of iO do not consist of group elements [GGH$^+$13b, BR14, BGK$^+$14, AGIS14, AB15, Zim15].

The same format assumption is a strong assumption, but our watermarking scheme is still meaningful. First, all current candidate constructions of iO are based on graded encoding schemes [GGH13a, CLT13, GGH15, CLT15]. Several serious attacks to graded encoding schemes were found in some candidate constructions [CLT13, CHL$^+$15, CGH$^+$15]. Of course, all iO constructions are *not* attacked so far [CGH$^+$15], but the security of graded encoding schemes 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. Nishimaki and Wichs [NW15] and Cohen, Holmgren, and Vaikuntanathan [CHV15] independently and concurrently proposed watermarking schemes against arbitrary strategies under the assumption that there exists iO. Thus, we can say that our watermarking scheme is an alternative construction in 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.

Experiment $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{F},\mathcal{A}}(\lambda)$:
$(pp, sk, mk, dk, rk) \xleftarrow{\mathsf{R}} \mathsf{WMGen}(1^\lambda)$;
$F \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{MO},\mathcal{DO}}(1^\lambda, pp)$;
$\mathsf{Vrfy}(pp, F) \to b$;
If $b = 1$, then return win else return lose

Oracle $\mathcal{MO}()$	**Oracle** $\mathcal{DO}(F)$
$F \xleftarrow{\mathsf{R}} \mathsf{Samp}_{\mathcal{F}}(sk)$; $F' \xleftarrow{\mathsf{R}} \mathsf{Mark}(mk, F)$; return F';	$\mathsf{Detect}(dk, F) \to b$; return b

Figure 3: Experiment for unforgeability

On the secret sampling of functions. We can consider auxiliary security definitions for watermarking where we need a secret key to sample function indices. Adversaries may forge function indices without using a secret key for sampling. Thus, we introduce a verification algorithm to check validity of function indices. The security definition is a natural analogy of the unforgeability of standard digital signatures.

Definition 3.5 (Verifiable Function Indices) Function indices of a watermarking scheme for function family $\mathcal{F}$ are verifiable if the scheme satisfies the following properties.

Verifiability: There exists an algorithm Vrfy (1-bit output) such that for any $(pp, sk, mk, dk, rk) \xleftarrow{\mathsf{R}} \mathsf{WMGen}(1^\lambda)$ and $F \xleftarrow{\mathsf{R}} \mathsf{Samp}_{\mathcal{F}}(sk)$, $\mathsf{Vrfy}(pp, F) \to 1$. Here, we assume that the algorithm WMGen also outputs a public parameter for verification.

Unforgeability: We say that a watermarking scheme satisfies unforgeability (or is unforgeable) if it holds that $\mathsf{Adv}^{\mathsf{uf}}_{\mathcal{F},\mathcal{A}}(1^\lambda) := \Pr[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{F},\mathcal{A}}(\lambda) \to \mathsf{win}] < \mathsf{negl}(\lambda)$. Experiment $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{F},\mathcal{A}}(\lambda)$ is shown in Figure 3.

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 [PW08, 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 $\mathsf{LTF}_{\mathsf{mult}}$ based on the Lewko IBE is as follows.

$\mathsf{LTF.IGen}(1^\lambda)$ **:** It generates $(\mathfrak{D}, \mathfrak{D}^*) \xleftarrow{\cup} \mathsf{Dual}(\mathbb{Z}_p^8)$, chooses $\alpha, \theta, \sigma \xleftarrow{\cup} \mathbb{Z}_p$, $\boldsymbol{\psi} := (\psi_1, \ldots, \psi_\ell) \xleftarrow{\cup} \mathbb{Z}_p^\ell$, and sets $g_T := e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}_1^*}$ and $g_{T_j} := g_T^{\psi_j}$ for all $j \in [\ell]$ where ℓ is the input length of functions. It chooses an arbitrary $\mathsf{tag} \in \mathbb{Z}_p$ and $s_{1,i}, s_{2,i} \xleftarrow{\cup} \mathbb{Z}_p$ for all $i \in [\ell]$ and generates $u_{i,j} := g_{T_j}^{s_{1,i}} \cdot g_T^{m_{i,j}}$ and $\boldsymbol{v}_i := g^{s_{1,i}\vec{d}_1 + s_{1,i}\mathsf{tag}\vec{d}_2 + s_{2,i}\vec{d}_3 + s_{2,i}\mathsf{tag}\vec{d}_4}$ for all $i, j \in [\ell]$ where $m_{i,i} = 1$ and $m_{i,j} = 0$ (if $i \neq j$). (Matrix $\boldsymbol{M} = \{m_{i,j}\}_{i,j} = \boldsymbol{I}$.) It chooses $r_1, r_2 \xleftarrow{\cup} \mathbb{Z}_p$ and generates $\boldsymbol{k}_{\mathsf{tag}} := g^{(\alpha + r_1\mathsf{tag})\theta\vec{d}_1^* - r_1\theta\vec{d}_2^* + r_2\mathsf{tag}\sigma\vec{d}_3^* - r_2\sigma\vec{d}_4^*}$. It returns $ek := (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag}) := (\{u_{i,j}\}_{i,j=1}^\ell, \{\boldsymbol{v}_i\}_{i=1}^\ell, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag})$, $ik := \boldsymbol{\psi}$. Hereafter, $\{u_{i,j}\}_{i,j}$ and $\{\boldsymbol{v}_i\}_i$ 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 $ik := \perp$. (Matrix $\boldsymbol{M} = \boldsymbol{0}$.)

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

$$
\begin{aligned}
y_j &:= \prod_i u_{i,j}^{x_i} = \prod_i g_{T_j}^{x_i s_{1,i}} \cdot g_T^{x_i m_{i,j}} = g_{T_j}^{\langle \vec{x}, \vec{s}_1 \rangle} g_T^{x_j} \\
y_{\ell+1} &:= \prod_i \boldsymbol{v}_i^{x_i} = \prod_i g^{x_i s_{1,i}\vec{d}_1 + x_i s_{1,i}\mathsf{tag}\vec{d}_2 + x_i s_{2,i}\vec{d}_3 + x_i s_{2,i}\mathsf{tag}\vec{d}_4} \\
&= g^{\langle \vec{x}, \vec{s}_1 \rangle \vec{d}_1 + \langle \vec{x}, \vec{s}_1 \rangle \mathsf{tag}\vec{d}_2 + \langle \vec{x}, \vec{s}_2 \rangle \vec{d}_3 + \langle \vec{x}, \vec{s}_2 \rangle \mathsf{tag}\vec{d}_4}
\end{aligned}
$$

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

$\mathsf{LTF.Invert}(ik, \boldsymbol{y})$**:** For input $\boldsymbol{y}$, it computes $x'_j := y_j/(y'_{\ell+1})^{\psi_j} = g_{T_j}^{\langle \vec{x}, \vec{s}_1 \rangle} g_T^{x_j} / g_T^{\langle \vec{x}, \vec{s}_1 \rangle \cdot \psi_j}$ and let $x_j \in \{0,1\}$ be such that $x'_j = g_T^{x_j}$. It returns $\vec{x} = (x_1, \ldots, x_\ell)$.

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 $\boldsymbol{y} = (g_T^{s'_1 \boldsymbol{\psi}}, e(g,g)^{s'_1}) = (g_{T_1}^{s'_1}, \ldots, g_{T_\ell}^{s'_1}, e(g,g)^{s'_1})$ where $s'_1 = \langle \vec{x}, \vec{s}_1 \rangle \in \mathbb{Z}_p$. 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 $\langle \vec{x}, \vec{s}_1 \rangle$ 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 $g_T^{\boldsymbol{Y}} = (g_T^{y_{i,j}}) \in \mathbb{G}_T^{h \times w}$. Algorithm $\mathsf{GenConceal}(h, w)$ which was introduced by Peikert and Waters is as follows.

1. Choose $\boldsymbol{\zeta} := (\zeta_1, \ldots, \zeta_h) \xleftarrow{\cup} \mathbb{Z}_p^h$ and $\boldsymbol{\psi} := (\psi_1, \ldots, \psi_w, 1) \xleftarrow{\cup} \mathbb{Z}_p^w \times \{1\}$.
2. Let $\boldsymbol{V} := \boldsymbol{\zeta} \otimes \boldsymbol{\psi} = \boldsymbol{\zeta}^\top \boldsymbol{\psi} \in \mathbb{Z}_p^{h \times (w+1)}$ be the outer product of $\boldsymbol{\zeta}$ and $\boldsymbol{\psi}$.
3. Output $\boldsymbol{C} := g_T^{\boldsymbol{V}} \in \mathbb{G}_T^{h \times (w+1)}$ 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. For $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_\ell, 1)$, $g_T^{\zeta \boldsymbol{\psi}}$ denotes $(g_T^{\zeta \psi_1}, \ldots, g_T^{\zeta \psi_\ell}, g_T^{\zeta})$. We need three steps to show the lemma. First, we will show that if the DBDH assumption holds, then $(g_T^{\boldsymbol{\psi}}, \boldsymbol{y} = g_T^{\zeta \boldsymbol{\psi}})$ is computationally indistinguishable from $(g_T^{\boldsymbol{\psi}}, \boldsymbol{y} = g_T^{\boldsymbol{t}})$ where $\zeta \xleftarrow{\cup} \mathbb{Z}_p$, $\boldsymbol{\psi} \xleftarrow{\cup} \mathbb{Z}_p^\ell \times \{1\}$, and $\boldsymbol{t} \xleftarrow{\cup} \mathbb{Z}_p^{\ell+1}$. 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 $\alpha_0, \zeta \xleftarrow{\cup} \mathbb{Z}_p$ and $\boldsymbol{\psi} \xleftarrow{\cup} \mathbb{Z}_p^\ell \times \{1\}$ and sets $g_T := e(g,g)^{\alpha_0}$ and $\boldsymbol{y} := (g_T^{\zeta \psi_1}, \ldots, g_T^{\zeta \psi_j}, y_{j+1}, \ldots, y_\ell, g_T^{\zeta})$ where $y_k \xleftarrow{\cup} \mathbb{G}_T$ for $k > j$. That is, y_k is uniformly random element for $k > j$. The output is $(g^{\boldsymbol{\psi}}, \boldsymbol{y})$.

We note that $\mathsf{HYB}_0 = (g_T^{\boldsymbol{\psi}}, g_T^{\boldsymbol{t}})$ and $\mathsf{HYB}_\ell = (g_T^{\boldsymbol{\psi}}, g_T^{\zeta \boldsymbol{\psi}})$. 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 uses distinguisher $\mathcal{D}$ for HYB_j and HYB_{j-1}. $\mathcal{B}$ is given input $(\mathsf{param}_{\mathbb{G}}, g, g^a, g^b, g^c, Q)$ and computes $(\boldsymbol{\tau}, \boldsymbol{y}) \in \mathbb{G}_T^{\ell+1} \times \mathbb{G}_T^{\ell+1}$ as follows. $\mathcal{B}$ sets $\tau_{\ell+1} := g_T := e(g,g)^{\alpha_0}$ for $\alpha_0 \xleftarrow{\cup} \mathbb{Z}_p$, $y_{\ell+1} := e(g, g^c)^{\alpha_0} = g_T^c$. It implicitly holds $\zeta := c$.

- For $k \in [j-1]$, $\mathcal{B}$ chooses ψ_k and sets $\tau_k := g_T^{\psi_k}$, $y_k := e(g, g^c)^{\alpha_0 \psi_k} = (g_T^c)^{\psi_k}$.
- For $k = j+1, \ldots, \ell$, $\mathcal{B}$ chooses ψ_k and sets $\tau_k := g_T^{\psi_k}$, $y_k \xleftarrow{\mathsf{U}} \mathbb{G}_T$.

Finally, $\mathcal{B}$ embeds the instance, that is, sets $\tau_j := e(g^a, g^b)^{\alpha_0} = g_T^{ab}$, $y_j := Q^{\alpha_0}$ (implicitly $\psi_j := ab$). If $Q = e(g,g)^{abc}$, then $y_j = g_T^{\psi_j \zeta}$ and $(\boldsymbol{\tau}, \boldsymbol{y}) = \mathsf{HYB}_j$. If $Q \xleftarrow{\mathsf{U}} \mathbb{G}_T$, then $y_j \xleftarrow{\mathsf{U}} g_T^{t_j}$ where $t_j \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and $(\boldsymbol{\tau}, \boldsymbol{y}) = \mathsf{HYB}_{j-1}$. 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 uses distinguisher $\mathcal{D}'$ for HYB'_i and HYB'_{i-1}. $\mathcal{D}$ is given instance $(g_T^{\boldsymbol{\psi}}, \boldsymbol{y} \in \mathbb{G}_T^{\ell+1})$, $\mathcal{D}$ generates matrix $\boldsymbol{C}$ as follows.

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

If $\boldsymbol{y} = g_T^{\zeta \boldsymbol{\psi}}$, then the distribution is the same as HYB'_i, else if $\boldsymbol{y}$ is uniformly random, then the distribution is the same as HYB'_{i-1}. Therefore, $\mathsf{HYB}'_0 \overset{\mathsf{C}}{\approx} \mathsf{HYB}'_{\ell}$.

Finally, we prove the lemma. Our final goal is to show $(\{u_{i,j}\}_{i,j}, \{\boldsymbol{v}_i\}_i, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag})$ for $\boldsymbol{M} = \boldsymbol{I}$ is indistinguishable from $(\{u_{i,j}\}_{i,j}, \{\boldsymbol{v}_i\}_i, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag})$ for $\boldsymbol{M} = \boldsymbol{0}$. In the above simulation, we only consider $\{u_{i,j}\}_{i,j}$ and g_T^{ζ} instead of $\left\{ \boldsymbol{v}_i = g^{s_{1,i}\vec{d}_1 + s_{1,i}\mathsf{tag}\vec{d}_2 + s_{2,i}\vec{d}_3 + s_{2,i}\mathsf{tag}\vec{d}_4} \right\}_i$. The key point is that we can replace $(\ell+1)$-th column element g_T^{ζ} with $g^{\zeta\vec{d}_1 + \zeta\mathsf{tag}\vec{d}_2 + s_2\vec{d}_3 + s_2\mathsf{tag}\vec{d}_4}$ where $(\mathfrak{D}, \mathfrak{D}^*) \xleftarrow{\mathsf{R}} \mathsf{Dual}(\mathbb{Z}_p^8)$ in the above simulation since the simulator can generate bases $(\mathfrak{D}, \mathfrak{D}^*)$ and choose $\mathsf{tag} \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ by itself. We must simulate $\boldsymbol{V} = g^{\zeta\vec{d}_1 + \zeta\mathsf{tag}\vec{d}_2 + s_2\vec{d}_3 + s_2\mathsf{tag}\vec{d}_4}$ and $\boldsymbol{k}_{\mathsf{tag}} = g^{(\alpha + r_1\mathsf{tag})\theta\vec{d}_1^* - r_1\theta\vec{d}_2^* r_2\mathsf{tag}\sigma\vec{d}_3^* - r_2\sigma\vec{d}_4^*}$ for $s_2 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$. If we have $g^{\zeta} = g^c$ and matrix $\mathfrak{D} = (\vec{d}_1, \ldots, \vec{d}_8)$, then we can compute $g^{\zeta\vec{d}_1 + \zeta\mathsf{tag}\vec{d}_2} = (g^{\zeta})^{\vec{d}_1 + \mathsf{tag}\vec{d}_2}$ without knowing ζ since simulator $\mathcal{B}$ generates $(\mathfrak{D}, \mathfrak{D}^*) \xleftarrow{\mathsf{R}} \mathsf{Dual}(\mathbb{Z}_p^8)$ by itself. $\mathcal{B}$ can also generate $\boldsymbol{k}_{\mathsf{tag}}$ since it has $\mathfrak{D}^*$. Therefore, in the above simulation we can replace $(\ell+1)$-th column element g_T^{ζ} with $g^{\zeta\vec{d}_1 + \zeta\mathsf{tag}\vec{d}_2 + s_2\vec{d}_3 + s_2\mathsf{tag}\vec{d}_4}$ and add $\boldsymbol{k}_{\mathsf{tag}} = g^{(\alpha + r_1\mathsf{tag})\theta\vec{d}_1^* - r_1\theta\vec{d}_2^* r_2\mathsf{tag}\sigma\vec{d}_3^* - r_2\sigma\vec{d}_4^*}$ (We can set $\alpha_0 := \alpha\theta\vec{d}_1 \cdot \vec{d}_1^*$). Therefore, $\mathsf{LTF}_{\mathsf{mult}}$ satisfies indistinguishability. □

4.2 Watermarking Scheme for $\mathsf{LTF}_{\mathsf{mult}}$

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

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

$\mathsf{WMGen}(1^\lambda)$**:** It generates $(\mathfrak{D}, \mathfrak{D}^*) \xleftarrow{\cup} \mathsf{Dual}(\mathbb{Z}_p^8)$, chooses $\alpha, \theta, \sigma \xleftarrow{\cup} \mathbb{Z}_p$ and $u_7, u_8 \xleftarrow{\cup} \mathbb{Z}_p^*$, and sets $g_T := e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}_1^*}$, $pp := \widehat{\mathbb{D}} := (\mathsf{param}_{\mathbb{V}}, g_T, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4})$, $sk := (\widehat{\mathbb{D}}, g^{\alpha\theta\vec{d}_1^*}, g^{\theta\vec{d}_1^*}, g^{\theta\vec{d}_2^*}, g^{\sigma\vec{d}_3^*}, g^{\sigma\vec{d}_4^*})$, $mk := (g^{\vec{d}_7^*}, g^{\vec{d}_8^*})$, $dk := (\widehat{\mathbb{D}}, \boldsymbol{c} := g^{u_7\vec{d}_7+u_8\vec{d}_8})$, and $rk := (\mathfrak{D}, \mathfrak{D}^*)$. 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 $\boldsymbol{\psi} \xleftarrow{\cup} \mathbb{Z}_p^\ell$, $\vec{s}_1, \vec{s}_2 \xleftarrow{\cup} \mathbb{Z}_p^\ell$, and generates $(ek, ik) := ((\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag}), \boldsymbol{\psi})$ as $\mathsf{LTF.IGen}$. It computes $\boldsymbol{k}_{\mathsf{tag}} := g^{(\alpha+r_1\mathsf{tag})\theta\vec{d}_1^*-r_1\theta\vec{d}_2^*+r_2\mathsf{tag}\sigma\vec{d}_3^*-r_2\sigma\vec{d}_4^*}$.

$\mathsf{Mark}(mk, ek)$**:** It parses $ek = (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag}) = (\boldsymbol{U}, \boldsymbol{V}, g^{(\alpha+r_1\mathsf{tag})\theta\vec{d}_1^*-r_1\theta\vec{d}_2^*+r_2\mathsf{tag}\sigma\vec{d}_3^*-r_2\sigma\vec{d}_4^*}, \mathsf{tag})$, chooses $t_7, t_8 \xleftarrow{\cup} \mathbb{Z}_p$, and computes $\widetilde{\boldsymbol{k}}_{\mathsf{tag}} := \boldsymbol{k}_{\mathsf{tag}} \cdot g^{t_7\vec{d}_7^*+t_8\vec{d}_8^*}$ by using $g^{\vec{d}_7^*}$ and $g^{\vec{d}_8^*}$. It outputs marked function index $\mathsf{WM}(ek) = (\boldsymbol{U}, \boldsymbol{V}, \widetilde{\boldsymbol{k}}_{\mathsf{tag}}, \mathsf{tag})$.

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

$\mathsf{Vrfy}(pp, ek)$**:** It parses $ek = (\{u_{i,j}\}_{i,j}, \{\boldsymbol{v}_i\}_i, \boldsymbol{k}_{\mathsf{tag}}, \mathsf{tag})$, chooses $s_1, s_2 \xleftarrow{\cup} \mathbb{Z}_p$, computes $u' := g_T^{s_1}$ and $\boldsymbol{v}' := g^{s_1\vec{d}_1+s_1\mathsf{tag}\vec{d}_2+s_2\vec{d}_3+s_2\mathsf{tag}\vec{d}_4}$, and checks $e(\boldsymbol{v}', \boldsymbol{k}_{\mathsf{tag}}) = u'$. The equation holds, then outputs 1. Else 0.

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 $(g^{\vec{d}_7^*}, g^{\vec{d}_8^*})$. Thus, meaningfulness hold with probability 1.

Preserving functionality holds since elements in $\boldsymbol{U}$ and $\boldsymbol{V}$ do not include vectors $g^{\vec{d}_7}$ and $g^{\vec{d}_8}$ and vector $g^{t_1\vec{d}_7^*+t_2\vec{d}_8^*}$ does not interfere the computation of $\mathsf{LTF.Eval}$.

We can also verify that verifiability holds. The procedure can be seen as verification of signatures for tags (IBE schemes are transformed into signature schemes by the Naor's transformation).

Note that if we do not have secret key $(g^{\vec{d}_1^*}, \ldots, g^{\vec{d}_4^*})$, 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 Proofs for $\mathsf{CWM}_{\mathsf{mult}}$

Definition 4.4 (The same format assumption) Adversaries in the experiment $\mathsf{Exp}^{\mathsf{nrmv}}_{\mathcal{F},\mathcal{A}}(\lambda,\epsilon)$ and $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{F},\mathcal{A}}(\lambda)$ for $\mathsf{CWM}_{\mathsf{mult}}$ output group elements and one integer.

Theorem 4.5 *Our watermarking scheme* $\mathsf{CWM}_{\mathsf{mult}}$ *for* $\epsilon = 1/\mathrm{poly}(\lambda)$ *is secure under the DLIN and same format assumptions.*

We prove the theorem by proving Theorems 4.6.

Theorem 4.6 (Non-Removability) *Our scheme* $\mathsf{CWM}_{\mathsf{mult}}$ *satisfies non-removability under the subspace and same format assumptions.*

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

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

and can generate $(e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}_1^*}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}) := (e(g^{\vec{b}_4^*}, g^{\vec{b}_4})^{\alpha'\mu_3\theta}, g^{\vec{b}_3^*}, g^{\vec{b}_4^*}, g^{\vec{b}_5^*}, g^{\vec{b}_6^*})$ and $mk = (g^{\vec{d}_7^*}, g^{\vec{d}_8^*}) := (g^{\vec{b}_1}, g^{\vec{b}_2})$. $\mathcal{B}$ can compute a detection key since $g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}$ are given and $\mathcal{B}$ can compute $(g^{\eta\vec{b}_1^*})^{u'_1}(g^{\beta\vec{b}_2^*})^{u'_2}$. $\mathcal{B}$ has $(g^{\vec{d}_2^*}, \ldots, g^{\vec{d}_8^*})$ but does not have $g^{\vec{d}_1^*}$ since $g^{\vec{b}_3}$ 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, $sk = (\perp, \perp, g^{\theta\vec{d}_2^*}, g^{\sigma\vec{d}_3^*}, g^{\sigma\vec{d}_4^*}) := (\perp, \perp, g^{\theta\vec{b}_4}, g^{\sigma\vec{b}_5}, g^{\sigma\vec{b}_6})$.

It implicitly holds $\alpha = \alpha'\mu_3$. To simulate the challenge oracle without the complete sk, for tag, $\mathcal{B}$ chooses $r'_1, r_2, t_7, t_8 \xleftarrow{\cup} \mathbb{Z}_p$ and computes

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

We set $r_1 := \mu_3 r'_1$. This is a valid marked index. If $\mathcal{A}$ outputs valid unmarked index $ek^* = (\boldsymbol{U}^*, \boldsymbol{V}^*, \boldsymbol{k}^*_{\mathsf{tag}^*}, \mathsf{tag}^*)$, 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 $g^{\vec{d}^*_1}$, that is, for some $s_1 \in \mathbb{Z}_p$, it may not hold $e(g^{s_1\vec{d}_1}, \boldsymbol{k}^*_{\mathsf{tag}^*}) = g_T^{s_1}$. 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 $e(g^{s_1\vec{d}_1 + s_1\mathsf{tag}^*\vec{d}_2 + s_2\vec{d}_3 + s_2\mathsf{tag}^*\vec{d}_4}, \boldsymbol{k}^*_{\mathsf{tag}^*}) = g_T^{s_1}$ where $g_T = e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}$.

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

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

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

Before we prove the unforgeability, we give a few remarks. First, we prove the unforgeability in a model where the adversary is not allowed to output a function index whose tag is equal to one of tags of indices generated by the mark oracle. Of course, it may be possible that the adversary outputs a function index whose tag is the same as a given tag and functionality is different from given functions by the oracles.

We can convert this weaker security into the strong security where the adversary is allowed to output a function index whose tag is equivalent to a given tag by the mark oracle because our function indices can be seen as signatures for tags (Naor's transformation) and there is a generic transformation as follows. We can extend the unforgeability to stronger one by using known techniques that convert standard unforgeable signature schemes into *strongly unforgeable* signature schemes. We now define algorithm $\mathsf{Xtr}(pk, sk, \mathsf{tag})$. It chooses $r_1, r_2 \xleftarrow{\cup} \mathbb{Z}_p$ and outputs $\boldsymbol{k}_{\mathsf{tag}} := g^{(\alpha+r_1\mathsf{tag})\theta\vec{d}^*_1 - r_1\theta\vec{d}^*_2 + r_2\mathsf{tag}\sigma\vec{d}^*_3 - r_2\sigma\vec{d}^*_4}$. We can consider $\boldsymbol{k}_{\mathsf{tag}}$ be a signature for tag. Naor pointed out that signature schemes can be derived from IBE schemes [BF03]. Thus, we can prove

the unforgeability of our watermarking schemes by using the unforgeability of signature schemes derived from IBE schemes of Lewko.

Huang, Wong, and Zhao proposed a generic transformation technique for converting unforgeable signature schemes into strongly unforgeable ones [HWZ07]. Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be strong one-time signature scheme. The conversion is as follows: generates $(vk_{\mathsf{ot}}, sk_{\mathsf{ot}}) \xleftarrow{\mathsf{R}} \mathsf{Gen}(1^\lambda)$, $\boldsymbol{k}_{vk_{\mathsf{ot}}} \xleftarrow{\mathsf{R}} \mathsf{Xtr}(pk, sk, vk_{\mathsf{ot}})$, that is, tag is replaced by vk_{ot}, and $\mathsf{sig} := \mathsf{Sign}(sk_{\mathsf{ot}}, \mathsf{tag} \parallel \boldsymbol{k}_{vk_{\mathsf{ot}}})$ and outputs $(\boldsymbol{k}_{vk_{\mathsf{ot}}}, \mathsf{sig}, vk_{\mathsf{ot}})$ as a signature. If we show that the adversary cannot forge a marked index for tag which is not queried to the oracle and $\boldsymbol{k}_{\mathsf{tag}}$ in our scheme is replaced with $(\boldsymbol{k}_{vk_{\mathsf{ot}}}, \mathsf{sig}.vk_{\mathsf{ot}})$ (of course, ek includes tag though we omit it), then our watermarking scheme satisfies strong unforgeability (what we defined in Section 3) by the strongly unforgeable property. That is, we can show that the adversary cannot output a function index with a given tag. In this paper, we prove the standard unforgeability for simplicity.

Next, we prove unforgeability.

Theorem 4.7 *Our scheme* $\mathsf{CWM}_{\mathsf{mult}}$ *satisfies unforgeability under the subspace and same format assumptions.*

Proof. Let q_{M} be the number of queries to the mark oracle. There are two types of element $\boldsymbol{k}_{\mathsf{tag}}$ (we call this element key hereafter) in our scheme [Wat09].

Normal key: $\boldsymbol{k}_{\mathsf{tag}} = g^{(\alpha + r_1 \mathsf{tag})\theta \vec{d}_1^* - r_1 \theta \vec{d}_2^* + r_2 \mathsf{tag} \sigma \vec{d}_3^* - r_2 \sigma \vec{d}_4^* + t_7 \vec{d}_7^* + t_8 \vec{d}_8^*}$

Semi-functional key: $\boldsymbol{k}_{\mathsf{tag}} = g^{(\alpha + r_1 \mathsf{tag})\theta \vec{d}_1^* - r_1 \theta \vec{d}_2^* + r_2 \mathsf{tag} \sigma \vec{d}_3^* - r_2 \sigma \vec{d}_4^* + t_5 \vec{d}_5^* + t_6 \vec{d}_6^* + t_7 \vec{d}_7^* + t_8 \vec{d}_8^*}$. We can generate semi-functional keys if we have the secret key and $(g^{\vec{d}_5^*}, g^{\vec{d}_6^*})$.

Both types of keys give a correct output (we can check this by simple calculation). By the terminology of Gerbush, Lewko, O'neill, and Waters [GLOW12], we can define forgery classes, Type I and Type II forgery as follows.

$$\mathcal{V}_I := \{(\mathsf{tag}^*, \boldsymbol{k}^*_{\mathsf{tag}^*}) \in \mathcal{V} \mid t_5, t_6 \in \mathbb{Z}_p^*,\ \boldsymbol{e}(g^{t_5 \vec{d}_5 + t_6 \vec{d}_6}, \boldsymbol{k}^*_{\mathsf{tag}^*}) = 1\}$$
$$\mathcal{V}_{II} := \{(\mathsf{tag}^*, \boldsymbol{k}^*_{\mathsf{tag}^*}) \in \mathcal{V} \mid t_5, t_6 \in \mathbb{Z}_p^*,\ \boldsymbol{e}(g^{t_5 \vec{d}_5 + t_6 \vec{d}_6}, \boldsymbol{k}^*_{\mathsf{tag}^*}) \neq 1\}$$
$$\mathcal{V} := \{(\mathsf{tag}^*, \boldsymbol{k}^*_{\mathsf{tag}^*}) \mid s_1, s_2 \in \mathbb{Z}_p^*,\ e(g^{s_1 \vec{d}_1 + s_1 \mathsf{tag}^* \vec{d}_2 + s_2 \vec{d}_3 + s_2 \mathsf{tag}^* \vec{d}_4}, \boldsymbol{k}^*_{\mathsf{tag}^*}) = g_T^{s_1}\}.$$

Set $\mathcal{V}$ is a set of keys that passes the verification algorithm. A Type I and II forgery can be seen as a normal and simulation key, respectively and it holds $\mathcal{V} = \mathcal{V}_I \cup \mathcal{V}_{II}$ and $\mathcal{V}_I \cap \mathcal{V}_{II} \neq \emptyset$.

To show that our scheme satisfies unforgeability, we introduce the following games. We consider game Game-i where the mark oracle generates semi-functional keys for the first $i \in [q_{\mathsf{M}}]$ queries and normal keys for the remaining $(q_{\mathsf{M}} - i)$ queries. Note that the marking oracle does not give unmarked function indices. It generates function indices, embeds marks to them, and gives marked indices. Let $\mathsf{Adv}_i^{\mathsf{forge\text{-}N}}$ (resp. $\mathsf{Adv}_i^{\mathsf{forge\text{-}S}}$) denote the advantage of the adversary in Game-(i) for outputting a Type I (resp. Type II) forgery key for a non-given tag.

1. In Game-(0), the challenge oracles return normal keys. First, We can show Lemma 4.8: If $\mathcal{A}$ outputs a Type II forgery key, then we can construct algorithm $\mathcal{B}_1$ (simulator for $\mathcal{A}$) that solves the subspace problem

2. Next, we consider Game-(i). We can show Lemma 4.9: If $\mathcal{A}$ detects the change of answers by the mark oracle (from normal key answer to semi-functional key answer), we can construct algorithm $\mathcal{B}$ (simulator for $\mathcal{A}$) which solves the subspace problem.

3. Last, we consider Game-(q_{M}), where all answers of the mark oracle to $\mathcal{A}$ are semi-functional keys. We can show Lemma 4.10: If adversary $\mathcal{A}$ outputs a Type I forgery key, then we can construct algorithm $\mathcal{B}_2$ which solves the subspace problem.

By Lemma 4.8, 4.9, and 4.10, we can show the following:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{Forge}}_{\mathcal{A}}(\lambda) &= \mathsf{Adv}^{\mathsf{forge\text{-}N}}_{0} + \mathsf{Adv}^{\mathsf{forge\text{-}S}}_{0} \\
&< \mathsf{Adv}^{\mathsf{forge\text{-}N}}_{0} + \mathsf{Adv}^{\mathsf{dss}}_{\mathcal{B}_1} \\
&< \mathsf{Adv}^{\mathsf{forge\text{-}N}}_{q_{\mathsf{M}}} + q_{\mathsf{M}}\mathsf{Adv}^{\mathsf{dss}}_{\mathcal{B}} + \mathsf{Adv}^{\mathsf{dss}}_{\mathcal{B}_1} \\
&< \mathsf{Adv}^{\mathsf{dss}}_{\mathcal{B}_2} + q_{\mathsf{M}}\mathsf{Adv}^{\mathsf{dss}}_{\mathcal{B}} + \mathsf{Adv}^{\mathsf{dss}}_{\mathcal{B}_1}
\end{aligned}$$

This proof strategy is based on the dual system encryption methodology by Waters [Wat09] or the dual form signature methodology by Gerbush, Lewko, O'neill, and Waters [GLOW12].

Lemma 4.8 *We assume the same format assumption. If $\mathcal{A}$ outputs a marked index that include Type II forgery key in* Game-(0)*, then we can construct an algorithm that break the subspace assumption with $k = 2$ and $n = 8$.*

Lemma 4.9 *We assume the same format assumption. If there exists $\mathcal{A}$ that distinguishes* Game-$(i-1)$ *from* Game-(i)*, then we can construct an algorithm that break the subspace assumption with $k = 2$ and $n = 8$.*

Lemma 4.10 *We assume the same format assumption. If $\mathcal{A}$ outputs a marked index that include Type I forgery key in* Game-(q_{M})*, then we can construct an algorithm that break the subspace assumption with $k = 1$ and $n = 8$.*

The theorem follows from these lemmas (the DLIN assumption implies the subspace assumption).

First, we give a proof of Lemma 4.8.

Proof of lemma. If $\mathcal{A}$ outputs ek^* where $\mathsf{Vrfy}(pp, ek^*) \to 1$, then we construct algorithm $\mathcal{B}$ which solves the subspace problem $k = 2$ and $n = 8$. $\mathcal{B}$ is given Γ, $D = (g^{\vec{b}_1}, \ldots, g^{\vec{b}_4}, g^{\vec{b}_7}, g^{\vec{b}_8}, g^{\eta\vec{b}^*_1}, g^{\eta\vec{b}^*_2}, g^{\beta\vec{b}^*_3}, g^{\beta\vec{b}^*_4}, g^{\vec{b}^*_5}, \ldots, g^{\vec{b}^*_8}, U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_3+\mu_3\vec{b}_5}, U_2 = g^{\mu_1\vec{b}_2+\mu_2\vec{b}_4+\mu_3\vec{b}_6}, \mu_3)$, and Q_{b} where Q_{b} is

$(V_1, V_2) = (g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_3^*}, g^{\tau_1\eta\vec{b}_2^*+\tau_2\beta\vec{b}_4^*})$ or $(W_1, W_2) = (g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_3^*+\tau_3\vec{b}_5^*}, g^{\tau_1\eta\vec{b}_2^*+\tau_2\beta\vec{b}_4^*+\tau_3\vec{b}_6^*})$. For dual orthonormal bases $(\mathfrak{D}, \mathfrak{D}^*)$, we first consider dual orthonormal bases $(\mathfrak{F}, \mathfrak{F}^*)$ as follows:

$$\vec{f}_1 := \eta\vec{b}_1^* \quad \vec{f}_2 := \eta\vec{b}_2^* \quad \vec{f}_3 := \beta\vec{b}_3^* \quad \vec{f}_4 := \beta\vec{b}_4^* \quad \vec{f}_5 := \vec{b}_5^* \quad \vec{f}_6 := \vec{b}_6^* \quad \vec{f}_7 := \vec{b}_7^* \quad \vec{f}_8 := \vec{b}_8^*$$
$$\vec{f}_1^* := \eta^{-1}\vec{b}_1 \quad \vec{f}_2^* := \eta^{-1}\vec{b}_2 \quad \vec{f}_3^* := \beta^{-1}\vec{b}_3 \quad \vec{f}_4^* := \beta^{-1}\vec{b}_4 \quad \vec{f}_5^* := \vec{b}_5 \quad \vec{f}_6^* := \vec{b}_6 \quad \vec{f}_7^* := \vec{b}_7 \quad \vec{f}_8^* := \vec{b}_8.$$

$\mathcal{B}$ chooses random matrix $A \in \mathbb{Z}_p^{2\times 2}$, which is invertible except negligible probability. Matrix $A \in \mathbb{Z}_p^{2\times 2}$ and $(A^{-1})^\top$ are applied to $\vec{f}_5, \vec{f}_6$ and $\vec{f}_5^*, \vec{f}_6^*$ as changes of basis matrix, respectively. That is, $\mathcal{B}$ sets $\vec{d}_i = \vec{f}_i$ and $\vec{d}_i^* = \vec{f}_i^*$ for $i = 1, \ldots, 4, 7, 8$ and it implicitly sets $\mathfrak{D} = \mathfrak{F}_A$, $\mathfrak{D}^* = \mathfrak{F}_A^*$. By Lemma 2.5, $(\mathfrak{D}, \mathfrak{D}^*)$ are correct dual orthonormal bases.

In order to set $\theta := \theta'\eta$, $\sigma := \sigma'\beta$ implicitly, $\mathcal{B}$ chooses $\alpha, \theta', \sigma' \xleftarrow{\cup} \mathbb{Z}_p$, and computes $\widehat{\mathbb{D}} := (e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}_1^*}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}) := (e(g^{\vec{b}_1}, g^{\eta\vec{b}_1^*})^{\alpha\theta'}, g^{\eta\vec{b}_1^*}, g^{\eta\vec{b}_2^*}, g^{\beta\vec{b}_3^*}, g^{\beta\vec{b}_4^*})$ and $sk = (g^{\alpha\theta\vec{d}_1^*}, g^{\theta\vec{d}_1^*}, g^{\theta\vec{d}_2^*}, g^{\sigma\vec{d}_3^*}, g^{\sigma\vec{d}_4^*}) := (g^{\alpha\theta'\vec{b}_1}, g^{\theta'\vec{b}_1}, g^{\theta'\vec{b}_2}, g^{\sigma'\vec{b}_3}, g^{\sigma'\vec{b}_4})$ since it has $(g^{\eta\vec{b}_1^*}, g^{\eta\vec{b}_2^*}, g^{\beta\vec{b}_3^*}, g^{\beta\vec{b}_4^*})$, $(g^{\vec{b}_1}, \ldots, g^{\vec{b}_4})$, θ', and σ'. $\mathcal{B}$ has $dk = (g^{u_7 g^{\vec{d}_7} + u_8 g^{\vec{d}_8}})$, which is calculated by A and $(g^{\vec{b}_7^*}, g^{\vec{b}_8^*})$ and mark key $mk = (g^{\vec{d}_7^*}, g^{\vec{d}_8^*})$, which is calculated by A and $(g^{\vec{b}_7}, g^{\vec{b}_8})$ and can simulate the mark oracle since it has $(g^{\vec{b}_7}, g^{\vec{b}_8})$ and $(g^{\vec{b}_7^*}, g^{\vec{b}_8^*})$. Note that $\mathcal{B}$ does not have $g^{\vec{d}_5^*}$ and $g^{\vec{d}_6^*}$ since $g^{\vec{b}_5}$ and $g^{\vec{b}_6}$ are not given.

In order to simulate the mark oracle, for tag, $\mathcal{B}$ chooses $r_1, r_2, t_7, t_8, \mathsf{tag} \xleftarrow{\cup} \mathbb{Z}_p$ and computes $\widetilde{\boldsymbol{k}}_{\mathsf{tag}} := g^{(\alpha+r_1\mathsf{tag})\theta\vec{d}_1^* - r_1\theta\vec{d}_2^* + r_2\mathsf{tag}\sigma\vec{d}_3^* - r_2\sigma\vec{d}_4^* + t_7\vec{d}_7^* + t_8\vec{d}_8^*}$. This is a valid marked index. Let $Q_{\mathsf{b}} = (T_1, T_2)$. If $\mathcal{A}$ outputs a valid $ek^* = (\boldsymbol{U}^*, \boldsymbol{V}^*, \boldsymbol{k}_{\mathsf{tag}^*}^*, \mathsf{tag}^*)$ which include a Type II forgery key $\boldsymbol{k}_{\mathsf{tag}^*}^*$ where $\boldsymbol{k}_{\mathsf{tag}^*}^* \in \mathcal{V}_{II}$, that is, for $s_1, s_2, t_5, t_6 \xleftarrow{\cup} \mathbb{Z}_p^*$

$$e(g^{s_1\vec{d}_1 + s_1\mathsf{tag}^*\vec{d}_2 + s_2\vec{d}_3 + s_2\mathsf{tag}^*\vec{d}_4}, \boldsymbol{k}_{\mathsf{tag}^*}^*) = g_T^{s_1} \wedge \boldsymbol{e}(g^{t_5\vec{d}_5 + t_6\vec{d}_6}, \boldsymbol{k}_{\mathsf{tag}^*}^*) \neq 1,$$

then computes $C_0 := \boldsymbol{e}(T_1, g^{\vec{b}_1})^{\theta'\alpha} = e(g,g)^{\alpha\theta\tau_1\vec{d}_1\cdot\vec{d}_1^*}$, $C := T_1(T_2)^{\mathsf{tag}^*}$, and $\Delta := \boldsymbol{e}(C, \boldsymbol{k}_{\mathsf{tag}^*})$. In this game, we assume that $\mathcal{A}$ outputs an index that includes a Type II forgery key, so coefficients of $\vec{d}_5^*, \vec{d}_6^*$ in $\boldsymbol{k}_{\mathsf{tag}^*}$ are not 0.

- If $T_1 = g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_3^*} = g^{\tau_1\vec{d}_1+\tau_2\vec{d}_3}$ and $T_2 = g^{\tau_1\eta\vec{b}_2^*+\tau_2\beta\vec{b}_4^*} = g^{\tau_1\vec{d}_2+\tau_2\vec{d}_4}$, then it holds that $C = g^{\tau_1\vec{d}_1+\tau_1\mathsf{tag}\vec{d}_2+\tau_2\vec{d}_3+\tau_2\mathsf{tag}\vec{d}_4}$ and $\Delta := \boldsymbol{e}(C, \boldsymbol{k}_{\mathsf{tag}^*}^*) = e(g,g)^{\alpha\theta\tau_1\vec{d}_1\cdot\vec{d}_1^*}$ due to the verification condition. We can consider τ_1 as s_1.
- If $T_1 = g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_3^*+\tau_3\vec{b}_5^*} = g^{\tau_1\vec{d}_1+\tau_2\vec{d}_3+\tau_3\vec{b}_5^*}$ and $T_2 = g^{\tau_1\eta\vec{b}_2^*+\tau_2\beta\vec{b}_4^*+\tau_3\vec{b}_6^*} = g^{\tau_2\vec{d}_1+\tau_2\vec{d}_4+\tau_3\vec{b}_6^*}$, then $C = g^{\tau_1\vec{d}_1+\tau_1\mathsf{tag}\vec{d}_2+\tau_2\vec{d}_3+\tau_2\mathsf{tag}\vec{d}_4+\tau_3\vec{b}_5^*+\tau_3\mathsf{tag}\vec{b}_6^*}$ and $\Delta := \boldsymbol{e}(C, \boldsymbol{k}_{\mathsf{tag}^*}^*) = e(g,g)^{\alpha\theta\tau_1\vec{d}_1\cdot\vec{d}_1^*+\gamma^*}$ where $\gamma^* \neq 0$ due to the verification condition.

In the latter case, the coefficient vector of $(\vec{b}_5^*, \vec{b}_6^*)$ is $(\tau_3, \mathsf{tag}\tau_3)$, thus the coefficient vector of $(\vec{d}_5, \vec{d}_6)$ is $\tau_3 A^{-1}(1, \mathsf{tag}^*)^\top$ and $\gamma = (t_5^*, t_6^*)\tau_3 A^{-1}(1, \mathsf{tag}^*)^\top$. These coefficients are uniformly random since $\mathcal{B}$ chose uniformly random A. If $\Delta/C_0 = 1$, then $\mathcal{B}$ outputs 0 ($\mathsf{b} = 0$). Else if $\Delta/C_0 \neq 1$, then it outputs 1. Thus, $\mathcal{B}$ can break the assumption. ■

Next, we prove Lemma 4.9.

Proof of lemma. If $\mathcal{A}$ outputs ek^* where $\mathsf{Vrfy}(pp, ek^*) \to 1$, then we construct algorithm $\mathcal{B}$ which solves the subspace problem $k = 2$ and $n = 8$. $\mathcal{B}$ is given Γ, $D = (g^{\vec{b}_1}, \ldots, g^{\vec{b}_4}, g^{\vec{b}_7}, g^{\vec{b}_8}, g^{\eta\vec{b}^*_1}, g^{\eta\vec{b}^*_2}, g^{\beta\vec{b}^*_3}, g^{\beta\vec{b}^*_4}, g^{\vec{b}^*_5}, \ldots, g^{\vec{b}^*_8}, U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_3+\mu_3\vec{b}_5}, U_2 = g^{\mu_1\vec{b}_2+\mu_2\vec{b}_4+\mu_3\vec{b}_6}, \mu_3)$, and Q_{b} where Q_{b} is $(V_1, V_2) = (g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4})$ or $(W_1, W_2) = (g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3+\tau_3\vec{b}^*_5}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4+\tau_3\vec{b}^*_6})$. For dual orthonormal bases $(\mathfrak{D}, \mathfrak{D}^*)$, we set

$$\begin{array}{llllll} \vec{d}_1 := \vec{b}_1 & \vec{d}_2 := \vec{b}_2 & \vec{d}_3 := \vec{b}_3 & \vec{d}_4 := \vec{b}_4 & \vec{d}_7 := \vec{b}_7 & \vec{d}_8 := \vec{b}_8 \\ \vec{d}^*_1 := \vec{b}^*_1 & \vec{d}^*_2 := \vec{b}^*_2 & \vec{d}^*_3 := \vec{b}^*_3 & \vec{d}^*_4 := \vec{b}^*_4 & \vec{d}^*_7 := \vec{b}^*_7 & \vec{d}^*_8 := \vec{b}^*_8 \end{array}$$

$\mathcal{B}$ chooses random matrix $A \in \mathbb{Z}_p^{2\times 2}$, which is invertible except negligible probability. Matrix A and $(A^{-1})^\top$ are applied as changes of basis matrix to $\vec{f}_5, \vec{f}_6$ and $\vec{f}^*_5, \vec{f}^*_6$, respectively, that is, it implicitly holds that $\mathfrak{D} := \mathfrak{B}_A$ and $\mathfrak{D}^* := \mathfrak{B}^*_A$ and they are correct distribution and reveal no information about A by Lemma 2.5. In order to set $\theta := \eta$ and $\sigma := \beta$ implicitly, $\mathcal{B}$ chooses $\alpha \xleftarrow{\cup} \mathbb{Z}_p$ and compute $e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1} := e(g^{\vec{b}_1}, g^{\eta\vec{b}^*_1})^\alpha$. Here it holds that $g^{\theta\vec{d}^*_1} = g^{\eta\vec{b}^*_1}$ and $g^{\sigma\vec{d}^*_3} = g^{\beta\vec{b}^*_3}$.

$\mathcal{B}$ can generate $\widehat{\mathbb{D}} := (e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}) := (g^{\vec{b}_1}, \ldots, g^{\vec{b}_4})$ and $sk = (g^{\alpha\theta\vec{d}^*_1}, g^{\theta\vec{d}^*_1}, g^{\theta\vec{d}^*_2}, g^{\sigma\vec{d}^*_3}, g^{\sigma\vec{d}^*_4}) := ((g^{\eta\vec{b}^*_1})^\alpha, g^{\eta\vec{b}^*_1}, g^{\eta\vec{b}^*_2}, g^{\beta\vec{b}^*_3}, g^{\beta\vec{b}^*_4})$. $\mathcal{B}$ has $mk = (g^{\vec{d}^*_7}, g^{\vec{d}^*_8})$, which is calculated by A and $(g^{\vec{b}^*_7}, g^{\vec{b}^*_8})$ and $dk = (g^{u_7 g^{\vec{d}_7} + u_8 g^{\vec{d}_8}})$, which is calculated by A and $((g^{\vec{b}_7}, g^{\vec{b}_8}))$ and can simulate the mark oracle since it has $(g^{\vec{b}^*_7}, g^{\vec{b}^*_8})$, $(g^{\vec{b}_7}, g^{\vec{b}_8})$, and α.

$\mathcal{B}$ can generate normal keys by using sk and mk, so it returns normal keys for j-th query where $i < j$. It can also compute semi-functional keys since it has $(g^{\vec{b}^*_5}, g^{\vec{b}^*_6})$ and can compute random linear combination of them, that is, $g^{t_5\vec{b}^*_5+t_6\vec{b}^*_6}$. It is equivalent to a vector spanned by $g^{\vec{d}^*_5}$ and $g^{\vec{d}^*_6}$. Thus, $\mathcal{B}$ returns semi-functional keys for j-th query where $j < i$. Let $Q_{\mathsf{b}} = (T_1, T_2)$. In order to simulate i-th query, $\mathcal{B}$ chooses $t_7, t_8, \mathsf{tag}_i \xleftarrow{\cup} \mathbb{Z}_p$ and computes $\widetilde{\boldsymbol{k}}_{\mathsf{tag}_i} := (g^{\eta\vec{b}^*_1})^\alpha T_1^{\mathsf{tag}_i}(T_2)^{-1} g^{t_7\vec{d}^*_7+t_8\vec{d}^*_8}$. Here, we can consider as $r_1 := \tau_1$, $r_2 := \tau_2$.

- If $T_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3}$ and $T_2 = g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4}$, then

$$\widetilde{\boldsymbol{k}}_{\mathsf{tag}_i} = g^{(\alpha+\tau_1\mathsf{tag}_i)\theta\vec{d}^*_1-\tau_1\theta\vec{d}^*_2+\tau_2\sigma\mathsf{tag}_i\vec{d}^*_3-\tau_2\sigma\vec{d}^*_4+t_7\vec{d}^*_7+t_8\vec{d}^*_8}.$$

  This is exactly Game-$(i-1)$.

- If $T_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3+\tau_3\vec{b}^*_5}$ and $T_2 = g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4+\tau_3\vec{b}^*_6}$, then

$$\widetilde{\boldsymbol{k}}_{\mathsf{tag}_i} = g^{(\alpha+\tau_1\mathsf{tag}_i)\theta\vec{d}^*_1-\tau_1\theta\vec{d}^*_2+\tau_2\sigma\mathsf{tag}_i\vec{d}^*_3-\tau_2\sigma\vec{d}^*_4+\mathsf{tag}_i\tau_3\vec{b}^*_5-\tau_3\vec{b}^*_6+t_7\vec{d}^*_7+t_8\vec{d}^*_8}.$$

  This is exactly Game-(i).

In the latter case, the coefficient vector of $\vec{b}^*_5$ and $\vec{b}^*_6$ is $\vec{X} := (\mathsf{tag}_i\tau_3, -\tau_3)$. If $\mathcal{A}$ outputs valid $ek^* = (\boldsymbol{U}^*, \boldsymbol{V}^*, \boldsymbol{k}^*_{\mathsf{tag}^*})$ where $\boldsymbol{k}^*_{\mathsf{tag}^*} \in \mathcal{V}$ (maybe $\boldsymbol{k}^*_{\mathsf{tag}^*}$ includes $g^{\vec{d}^*_5}$ or $g^{\vec{d}^*_6}$), then $\mathcal{B}$ computes

$C := U_1(U_2)^{\mathsf{tag}^*} = g^{\mu_1\vec{d}_1+\mu_1\mathsf{tag}^*\vec{d}_2+\mu_2\vec{d}_3+\mu_2\mathsf{tag}^*\vec{d}_4+\mu_3\vec{b}_5+\mu_3\mathsf{tag}^*\vec{b}_6}$ and $C_0 := (\boldsymbol{e}(g^{\eta\vec{b}^*_1}, U_1))^{\alpha} = (e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1})^{\mu_1}$. The coefficient vector of $(\vec{b}_5, \vec{b}_6)$ is $\vec{Y} := (\mu_3, \mathsf{tag}^*\mu_3)$. The adversary output a function index whose tag is new, so tag^* is different from all tag_i which are given from the mark oracle. The coefficient vector of $(\vec{d}^*_5, \vec{d}^*_6)$ and $(\vec{d}_5, \vec{d}_6)$ are $\vec{X}' := \tau_3 A^{\top}(\mathsf{tag}_i, -1)^{\top}$ and $\vec{Y}' := \mu_3 A^{-1}(1, \mathsf{tag}^*)$, respectively. The distribution of all values except $\widetilde{\boldsymbol{k}}_{\mathsf{tag}_i}$ and (C_0, C) is independent of transformation matrix A and $\mathsf{tag}^* \neq \mathsf{tag}_i$, so coefficient $\vec{X}'$ and $\vec{Y}'$ are uniformly random by Lemma 2.5. $\mathcal{B}$ computes $\Delta := \boldsymbol{e}(C, \boldsymbol{k}_{\mathsf{tag}^*})$.

If $\boldsymbol{k}^*_{\mathsf{tag}^*} \in \mathcal{V}_I$ and $\Delta/C_0 = 1$ holds due to the verification condition, then $\mathcal{B}$ outputs 0. Note that $g^{\vec{b}_5}$ and $g^{\vec{b}_6}$ in C does not affect this condition since $\boldsymbol{k}^*_{\mathsf{tag}^*} \in \mathcal{V}_I$.

If $\boldsymbol{k}^*_{\mathsf{tag}^*} \in \mathcal{V}_{II}$ and $\Delta/C_0 \neq 1$ holds, then $\mathcal{B}$ outputs 1.

Thus, if $\mathcal{A}$ detects the change, that is, if there is non-negligible difference between that $\mathcal{A}$ outputs a normal conversion key and a semi-functional conversion key, then $\mathcal{B}$ can break the subspace assumption. ■

Last, we prove Lemma 4.10.

Proof of lemma. If $\mathcal{A}$ outputs ek^* where $\mathsf{Vrfy}(pp, ek^*) \to 1$, then we construct algorithm $\mathcal{B}$ which solves Subspace problem $k = 1$ and $n = 8$. $\mathcal{B}$ is given Γ, $D = (g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_4}, \ldots, g^{\vec{b}_8}, g^{\eta\vec{b}^*_1}, g^{\beta\vec{b}^*_2}, g^{\vec{b}^*_3}, \ldots, g^{\vec{b}^*_8}, U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_2+\mu_3\vec{b}_3}, \mu_3)$, and Q_{b} where Q_{b} is $V_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$ or $W_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2+\tau_3\vec{b}^*_3}$. For dual orthonormal bases $(\mathfrak{D}, \mathfrak{D}^*)$, we set

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

$\mathcal{B}$ chooses $\theta, \alpha', \sigma \xleftarrow{\cup} \mathbb{Z}_p$, and can compute $\widehat{\mathbb{D}} := (e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}) := (\boldsymbol{e}(g^{\vec{b}^*_4}, g^{\vec{b}_4})^{\alpha'\mu_3\theta}, g^{\vec{b}^*_3}, g^{\vec{b}^*_4}, g^{\vec{b}^*_5}, g^{\vec{b}^*_6})$ where $\alpha := \alpha'\mu_3$, $dk = (g^{u_7 g^{\vec{d}_7} + u_8 g^{\vec{d}_8}}) := (g^{u_7 g^{\vec{b}^*_7} + u_8 g^{\vec{b}^*_8}})$ for $u_7, u_8 \xleftarrow{\cup} \mathbb{Z}^*_p$ and $mk = (g^{\vec{d}^*_7}, g^{\vec{d}^*_8}) := (g^{\vec{b}_7}, g^{\vec{b}_8})$ (i.e., can simulate the mark algorithm), but does not have $g^{\vec{d}^*_1}$ since $g^{\vec{b}_3}$ is not given. That is, it has incomplete $sk = (g^{\alpha\theta\vec{d}^*_1}, g^{\theta\vec{d}^*_1}, g^{\theta\vec{d}^*_2}, g^{\sigma\vec{d}^*_3}, g^{\sigma\vec{d}^*_4}) := (\perp, \perp, g^{\theta\vec{b}_4}, g^{\sigma\vec{b}_5}, g^{\sigma\vec{b}_6})$. $\mathcal{B}$ also has $(g^{\vec{d}^*_5}, g^{\vec{d}^*_6}) = (g^{\vec{b}_1}, g^{\vec{b}_2})$. Here, $U_1 = g^{\mu_1\vec{d}^*_5+\mu_2\vec{d}^*_6+\mu_3\vec{d}^*_1}$ and we use it to simulate keys ($\boldsymbol{k}_{\mathsf{tag}}$) without the complete secret key. In order to simulate the mark oracle, $\mathcal{B}$ chooses $r'_1, r_2, t'_5, t'_6, t_7, t_8, \mathsf{tag} \xleftarrow{\cup} \mathbb{Z}_p$ and computes

$$\begin{aligned}
\widetilde{\boldsymbol{k}}_{\mathsf{tag}} &:= (U_1)^{(\alpha'-r'_1\mathsf{tag})\theta} g^{-r'_1\mu_3\theta\vec{d}^*_2+r_2\mathsf{tag}\sigma\vec{d}^*_3-r_2\sigma\vec{d}^*_4+t'_5\vec{d}^*_5+t'_6\vec{d}^*_6+t_7\vec{d}^*_7+t_8\vec{d}^*_8} \\
&= g^{(\alpha+r_1\mathsf{tag})\theta\vec{d}^*_1-r_1\theta\vec{d}^*_2+r_2\mathsf{tag}\sigma\vec{d}^*_3-r_2\sigma\vec{d}^*_4+(t'_5-\theta(\alpha'+r'_1\mathsf{tag})\mu_1)\vec{d}^*_5+(t'_6-\theta(\alpha'+r'_1\mathsf{tag})\mu_2)\vec{d}^*_6+t_7\vec{d}^*_7+t_8\vec{d}^*_8}.
\end{aligned}$$

It implicitly holds $r_1 := \mu_3 r'_1$. This is a valid marked semi-functional key. If $\mathcal{A}$ outputs normal key $ek^* = (\boldsymbol{U}^*, \boldsymbol{V}^*, \boldsymbol{k}^*_{\mathsf{tag}^*}, \mathsf{tag}^*)$ where

$$\boldsymbol{k}^*_{\mathsf{tag}^*} \in \mathcal{V}_I,$$

then $\mathcal{B}$ chooses $s_1, s_2 \xleftarrow{\cup} \mathbb{Z}_p$ and computes $C := g^{s_1\vec{d}_1+s_1\mathsf{tag}^*\vec{d}_2+s_2\vec{d}_3+s_2\mathsf{tag}^*\vec{d}_4}$, $C_0 := e(g,g)^{s_1\alpha\theta\vec{b}_4\cdot\vec{b}^*_4}$, and $\Delta := \boldsymbol{e}(C \cdot Q_{\mathsf{b}}, \boldsymbol{k}^*_{\mathsf{tag}^*})$. If $Q_{\mathsf{b}} = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2} = g^{\tau_1\eta\vec{d}_5+\tau_2\beta\vec{d}_6}$, then $\Delta/C_0 = 1$ due to the

verification condition and $\boldsymbol{k}^*_{\mathsf{tag}^*} \in \mathcal{V}_I$. If $Q_{\mathsf{b}} = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2+\tau_3\vec{b}^*_3} = g^{\tau_1\eta\vec{d}_5+\tau_2\beta\vec{d}_6+\tau_3\vec{d}_1}$, then $\Delta/C_0 = e(g,g)^{\tau_3(\alpha+r^*_1\mathsf{tag}^*)\theta\vec{d}_1\cdot\vec{d}^*_1} \neq 1$. Thus, $\mathcal{B}$ can break the subspace assumption. ■

Thus, we finished the proof of Theorem 4.7 □

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 under the DLIN assumption and same format assumption in the standard model. 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$^+$12]. 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 *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 528–556, 2015. Full version available from http://eprint.iacr.org/2015/025.

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 646–658, 2014. Full version available from http://eprint.iacr.org/2014/222.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[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 *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 1–18, 2001.

[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. *J. ACM*, 59(2):6, 2012.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 221–238, 2014. Full version available from http://eprint.iacr.org/2013/631.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 1–25, 2014. Full version available from http://eprint.iacr.org/2013/563.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1994.

[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 zeros: New mmap attacks and their limitations. *IACR Cryptology ePrint Archive*, 2015:596, 2015. To appear in CRYPTO'15.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 3–12, 2015.

[CHV15] Aloni Cohen, Justin Holmgren, and Vinod Vaikuntanathan. Publicly verifiable software watermarking. *IACR Cryptology ePrint Archive*, 2015:373, 2015.

[CLL+12] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter ibe and signatures via asymmetric pairings. In *Pairing*, volume 7708 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 2012.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2015:162, 2015.

[CT02] Christian S. Collberg and Clark D. Thomborson. Watermarking, tamper-proofing, and obfuscation-tools for software protection. *IEEE Trans. Software Eng.*, 28(8):735–746, 2002.

[FGK+10] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In *Public Key*

Cryptography, volume 6056 of *Lecture Notes in Computer Science*, pages 279–295. Springer, 2010.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

[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 Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 2013. Full version available from http://eprint.iacr.org/2013/451.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 498–527, 2015.

[GLOW12] Michael Gerbush, Allison B. Lewko, Adam O'Neill, and Brent Waters. Dual form signatures: An approach for proving security from static assumptions. In *ASIACRYPT*, volume 7658 of *LNCS*, pages 25–42. Springer, 2012.

[GV08] Steven D. Galbraith and Eric R. Verheul. An analysis of the vector decomposition problem. In *Public Key Cryptography*, volume 4939 of *LNCS*, pages 308–327. Springer, 2008.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. *IACR Cryptology ePrint Archive*, 2015:301, 2015.

[HMW07] Nicholas Hopper, David Molnar, and David Wagner. From weak to strong watermarking. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 362–382, 2007.

[HO12] Brett Hemenway and Rafail Ostrovsky. Extended-DDH and lossy trapdoor functions. In *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 627–643. Springer, 2012.

[HWZ07] Qiong Huang, Duncan S. Wong, and Yiming Zhao. Generic transformation to strongly unforgeable signatures. In *ACNS*, volume 4521 of *LNCS*, pages 1–17. Springer, 2007.

[KY02] Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In *EUROCRYPT*, volume 2332 of *LNCS*, pages 450–465. Springer, 2002.

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 318–335. Springer, 2012.

[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 *EUROCRYPT*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Computing*, 17(2):373–386, 1988.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, 2010.

[Nis13] Ryo Nishimaki. How to watermark cryptographic functions. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 111–125, 2013. Full version available from http://eprint.iacr.org/2014/472.

[NSS99] David Naccache, Adi Shamir, and Julien P. Stern. How to copyright a function? In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, pages 188–196, 1999.

[NW15] Ryo Nishimaki and Daniel Wichs. Watermarking cryptographic programs against arbitrary removal strategies. *IACR Cryptology ePrint Archive*, 2015:344, 2015.

[OT08] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing*, volume 5209 of *LNCS*, pages 57–74. Springer, 2008.

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, volume 5912 of *LNCS*, pages 214–231. Springer, 2009.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, volume 6223 of *LNCS*, pages 191–208. Springer, 2010.

[OT11] Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In *Public Key Cryptography*, volume 6571 of *LNCS*, pages 35–52. Springer, 2011.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 591–608. Springer, 2012.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC'08*, pages 187–196. ACM, 2008.

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

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394. ACM, 1990.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO'09*, volume 5677 of *LNCS*, pages 619–636. Springer, 2009. full version available from http://eprint.iacr.org/2009/385.

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

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

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 439–467, 2015. Full version available from http://eprint.iacr.org/2014/776.

A Lewko IBE Scheme

We present Lewko IBE scheme and Okamoto-Takashima IPE scheme for reference.

A.1 Identity-Based Encryption

We review Lewko IBE scheme, $\mathsf{IBE}_{\mathsf{L}}$ [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 $(\mathfrak{D}, \mathfrak{D}^*) \xleftarrow{\mathsf{U}} \mathsf{Dual}(\mathbb{Z}_p^6)$, chooses $\alpha, \theta, \sigma \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, and sets $pk := (\Lambda, e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}_1^*}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4})$, $msk := (g^{\theta\vec{d}_1^*}, g^{\alpha\vec{d}_1^*}, g^{\theta\vec{d}_2^*}, g^{\sigma\vec{d}_3^*}, g^{\sigma\vec{d}_4^*})$. It outputs (pk, msk).

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

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

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

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