Lossy Algebraic Filters With Short Tags

Benoît Libert[1,2] and Chen Qian[3]

[1] CNRS, Laboratoire LIP, France
[2] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France
[3] Univ Rennes, CNRS, IRISA (France)

Abstract. Lossy algebraic filters (LAFs) are function families where each function is parametrized by a tag, which determines if the function is injective or lossy. While initially introduced by Hofheinz (Eurocrypt 2013) as a technical tool to build encryption schemes with key-dependent message chosen-ciphertext (KDM-CCA) security, they also find applications in the design of robustly reusable fuzzy extractors. So far, the only known LAF family requires tags comprised of $\Theta(n^2)$ group elements for functions with input space $\mathbb{Z}_p^n$, where p is the group order. In this paper, we describe a new LAF family where the tag size is only linear in n and prove it secure under simple assumptions in asymmetric bilinear groups. Our construction can be used as a drop-in replacement in all applications of the initial LAF system. In particular, it can shorten the ciphertexts of Hofheinz's KDM-CCA-secure public-key encryption scheme by 19 group elements. It also allows substantial space improvements in a recent fuzzy extractor proposed by Wen and Liu (Asiacrypt 2018). As a second contribution, we show how to modify our scheme so as to prove it (almost) tightly secure, meaning that security reductions are not affected by a concrete security loss proportional to the number of adversarial queries.

Keywords. Lossy algebraic filters, efficiency, tight security, standard assumptions.

1 Introduction

As introduced by Peikert and Waters a decade ago [40], lossy trapdoor functions (LTFs) are function families where injective functions – which are efficiently invertible using a trapdoor - are computationally indistinguishable from many-to-one functions, wherein the image is drastically smaller than the domain. Since their introduction, they drew a lot of attention [19,23,25,44,46] and revealed powerful enough to imply chosen-ciphertext (IND-CCA2) security [40], deterministic public-key encryption in the standard model [9,15,42], as well as encryption schemes achieving the best possible security against selective-opening (SO) adversaries [2,5] or using imperfect randomness [1].

Lossy Algebraic Filters. Lossy algebraic filters (LAFs) are a variant LTFs introduced by Hofheinz [26] as a tool enabling the design of chosen-ciphertext-secure encryption schemes with key-dependent message (KDM-CCA) security [6]. Recently, they were also used by Wen and Liu [45] in the construction of robustly reusable fuzzy extractors. In LAF families, each function takes as arguments an input x and a tag t, which determines if the function behaves as a lossy or an injective function. More specifically, each tag $t = (t_\mathsf{c}, t_\mathsf{a})$ is comprised of an auxiliary component t_a, which may consist of any public data, and a core component t_c. For any auxiliary component t_a, there should exist at least one t_c such that $t = (t_\mathsf{c}, t_\mathsf{a})$ induces a lossy function $f_\mathsf{LAF}(t, \cdot)$. LAFs strengthen the requirements of lossy trapdoor functions in that, for any lossy tag t, the function $f_\mathsf{LAF}(t, x)$ always reveals the same information about the input x, regardless of which tag is used. In particular, for a given evaluation key ek, multiple evaluations $f_\mathsf{LAF}(t_1, x), \ldots, f_\mathsf{LAF}(t_n, x)$ for distinct lossy tags do not reveal any more information about x than a single evaluation. On the other hand, LAFs depart from lossy trapdoor functions in that they need not be efficiently invertible using a trapdoor. For their applications to KDM-CCA security [26] and fuzzy extractors [45], lossy algebraic filters are expected to satisfy two security properties. The first one, called *indistinguishability*, requires that lossy tags be indistinguishable from random tags. The second one, named *evasiveness*, captures that lossy tags should be hard to come by without a trapdoor.

So far, the only known LAF realization is a candidate, suggested by Hofheinz [26], which relies on the Decision Linear assumption (DLIN) [12] in groups with a bilinear map. While efficient and based on a standard assumption, it incurs relatively large tags comprised of a quadratic number of group elements in the number of input symbols. More precisely, for functions admitting inputs $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{Z}_p^n$, where p is the order of a pairing-friendly $\mathbb{G}$, the core components t_c contain $\Theta(n^2)$ elements of $\mathbb{G}$. For the application to KDM-CCA security [26] (where t_c should be part of ciphertexts), quadratic-size tags are not prohibitively expensive as the encryption scheme of [26, Section 4] can make do with a constant n (typically, $n = 6$). In the application to fuzzy extractors [45], however, it is desirable to reduce the tag length. In the robustly reusable fuzzy extractor of [45], the core tag component t_c is included in the public helper string P that allows reconstructing a secret key from a noisy biometric reading w. The latter lives in a metric space that should be small enough to fit in the input space $\mathbb{Z}_p^n$ of the underlying LAF family. Even if p is exponentially large in the security parameter λ, a constant n would restrict biometric readings to have linear length in λ. Handling biometric readings of polynomial length thus incurs $n = \omega(1)$, which results in large tags and longer public helper strings. This motivates the design of new LAF candidates with smaller tags.

Our Results. The contribution of this paper is two-fold. We first construct a new LAF with linear-size tags and prove it secure under simple, constant-size assumptions (as opposed to q-type assumptions, which are described using a linear number of elements in the number of adversarial queries) in bilinear groups. The indistinguishability and evasiveness properties of our scheme are implied by the Decision 3-party Diffie-Hellman assumption (more precisely, its natural analogue in asymmetric bilinear maps), which posits the pseudorandomness of tuples $(g, g^a, g^b, g^c, g^{abc})$, for random $a, b, c \in_R \mathbb{Z}_p$. For inputs in $\mathbb{Z}_p^n$, where p is the group order, our core tag components only consist of $O(n)$ group elements.

These shorter tags are obtained without inflating evaluation keys, which remain of length $O(n)$ (as in [26]).

As a second contribution, we provide a second LAF realization with $O(n)$-size tags where the indistinguishability and evasiveness properties are both *almost tightly* related to the underlying hardness assumption. Namely, our security proofs are tight – or almost tight in the terminology of Chen and Wee [16] – in that the gap between the advantages of the adversary and the reduction only depends on the security parameter, and not on the number of adversarial queries. In the LAF suggested by Hofheinz [26], the proof of evasiveness relies on the unforgeability of Waters signatures [43]. As a result, the reduction loses a linear factor in the number of lossy tags obtained by the adversary. In our second construction, we obtain tight reductions by replacing Waters signatures with (a variant of) a message authentication code (MAC) due to Blazy, Kiltz and Pan [7]. As a result, our proof of evasiveness only loses a factor $O(\lambda)$ with respect to the Symmetric eXternal Diffie-Hellman assumption (SXDH). If our scheme is plugged into the robustly reusable fuzzy extractor of Wen and Liu [45], it immediately translates into a tight proof of robustness in the sense of the definition of [45]. While directly using our second LAF in the KDM-CCA-secure scheme of [26] does not seem sufficient to achieve tight key-dependent message security, it may still provide a building block for future constructions of tightly KDM-CCA-secure encryption schemes with short ciphertexts.

Techniques. Like the DLIN-based solution given by Hofheinz [26], our evaluation algorithms proceed by computing a matrix-vector product in the exponent, where the matrix is obtained by pairing group elements taken from the core tag t_{c} with elements of the evaluation key. Here, we reduce the size of t_{c} from $O(n^2)$ to $O(n)$ group elements using a technique suggested by Boyen and Waters [14] in order to compress the evaluation keys of DDH-based lossy trapdoor functions.

In the pairing-based LTF of [14], the evaluation key contains group elements $\{(R_i, S_i) = (g^{r_i}, (h^i \cdot u)^{r_i})\}_{i=1}^n$, $\{(V_j = g^{v_j}, H_j = (h^j \cdot u)^{v_j})\}_{j=1}^n$. Using a symmetric bilinear maps $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, these make it possible to compute the off-diagonal elements of a matrix

$$M_{i,j} = e(g,h)^{r_i \cdot v_j} = \Big(\frac{e(R_i, H_j)}{e(S_i, V_j)}\Big)^{1/(j-i)} \qquad \forall (i,j) \in [n] \times [n] \setminus \{(i,i)\}_{i=1}^n \qquad (1)$$

via a "two equation" technique borrowed from the revocation system of Lewko, Sahai and Waters [34]. By including $\{D_i = e(g,g)^{r_i \cdot v_i} \cdot e(g,g)\}_{i=1}^n$ in the evaluation key, the LTF of [14] allows the evaluator to compute a matrix $(M_{i,j})_{i,j \in [n]}$ such that $M_{i,j} = e(g,g)^{r_i \cdot v_j}$ if $i \neq j$ and $M_{i,i} = e(g,g)^{r_i \cdot v_i} \cdot e(g,g)^{m_i}$ and for which $m_i = 1$ (resp. $m_i = 0$), for all $i \in [n]$, in injective (resp. lossy) functions. The indistinguishability of lossy and injective evaluation keys relies on the fact that (1) is only computable when $i \neq j$, making it infeasible to distinguish $\{D_i = e(g,h)^{r_i \cdot v_i} \cdot e(g,g)\}_{i=1}^n$ from $\{D_i = e(g,h)^{r_i \cdot v_i}\}_{i=1}^n$.

Our first LAF construction relies on the "two equation" technique of [34] in a similar way with the difference that we include $\{(V_j = g^{v_j}, H_j = (h^j \cdot u)^{v_j}\}_{j=1}^n$ in the evaluation key ek, but $\{(R_i, S_i) = (g^{r_i}, (h^i \cdot u)^{r_i})\}_{i=1}^n$ is now part of the core tag components t_{c}. This makes it possible to compute off-diagonal elements of $(M_{i,j})_{i,j\in[n]}$ by pairing elements of ek with those of t_{c}. To enable the computation of diagonal elements $\{M_{i,i}\}_{i=1}^{n}$, we augment core tag components by introducing pairs $(D_i, E_i) \in \mathbb{G}^2$, which play the same role as $\{D_i = e(g,g)^{r_i\cdot v_i} \cdot e(g,g)\}_{i=1}^{n}$ in the LTF of [14]. In lossy tags, $\{(D_i, E_i)\}_{i=1}^{n}$ are of the form

$$(D_i, E_i) = (h^{r_i\cdot v_i} \cdot H_{\mathbb{G}}(\tau)^{\rho_i}, g^{\rho_i}), \tag{2}$$

for a random $\rho_i \in_R \mathbb{Z}_p$, where τ is a chameleon hashing of all tag components. Such pairs $\{(D_i, E_i)\}_{i=1}^{n}$ allow the evaluator to compute

$$M_{i,i} = \frac{e(D_i, g)}{e(H_{\mathbb{G}}(\tau), E_i)} = e(g,h)^{r_i\cdot v_i} \qquad \forall i \in [n],$$

which results in a rank-one matrix $(M_{i,j})_{i,j\in[n]}$, where $M_{i,j} = e(g,h)^{r_i\cdot v_j}$. When computed as per (2), $\{(D_i, E_i)\}_{i=1}^{n}$ can be seen as "blinded" Waters signatures [43]. Namely, $(g, h, V_i = g^{v_i})$ can be seen as a verification key; h^{v_i} is the corresponding secret key; and $r_i \in \mathbb{Z}_p$ serves as a blinding factor that ensures the indistinguishability of (D_i, E_i) from random pairs. Indeed, the Decision 3-party Diffie-Hellman (D3DH) assumption allows proving that $h^{r_i\cdot v_i}$ is computationally indistinguishable from random given (g, h, g^{v_i}, g^{r_i}). In our proof of indistinguishability, however, we need to rely on the proof technique of the Boneh-Boyen IBE [11] in order to apply a hybrid argument that allows gradually replacing pairs $\{(D_i, E_i)\}_{i=1}^{n}$ by random group elements.

In our proof of evasiveness, we rely on the fact that forging a pair of the form $(D_i, E_i) = (h^{r_i\cdot v_i} \cdot H_{\mathbb{G}}(\tau)^{\rho_i}, g^{\rho_i})$ on input of (g, h, g^{v_i}) is as hard as solving the 2-3-Diffie-Hellman problem [33], which consist in finding a non-trivial pair $(g^r, g^{r\cdot ab}) \in \mathbb{G}^* \times \mathbb{G}^*$ on input of (g, g^a, g^b). In turn, this problem is known to be at least as hard as breaking the Decision 3-party Diffie-Hellman assumption.

The above techniques allow us to construct a LAF with $O(n)$-size tags and evaluation keys made of $O(n + \lambda)$ group elements under a standard assumption. Our first LAF is actually described in terms of asymmetric pairings, but it can be instantiated in all types (i.e., symmetric or asymmetric) of bilinear groups. Our second LAF construction requires asymmetric pairing configurations and the Symmetric eXternal Diffie-Hellman (SXDH) assumption. It is very similar to our first construction with the difference that we obtain a tight proof of evasiveness by replacing Waters signatures with a variant of a MAC proposed by Blazy, Kiltz and Pan [7]. In order for the proofs to go through, we need to include n MAC instances (each with its own keys) in lossy tags, which incurs evaluation keys made of $O(n \cdot \lambda)$ group elements. We leave it is an interesting open problem to achieve tight security using shorter evaluation keys.

Related Work. All-but-one lossy trapdoor functions (ABO-LTFs) [40] are similar to LAFs in that they are lossy function families where each function is parametrized by a tag that determines if the function is injective or lossy. They differ from LAFs in two aspects: (i) They should be efficiently invertible using a trapdoor; (ii) For a given evaluation key ek, there exists only one tag for which the function is lossy. The main motivation of ABO-LTFs is the construction of chosen-ciphertext [41] encryption schemes. *All-but-many* lossy trapdoor functions (ABM-LTFs) are an extension of ABO-LTFs introduced by Hofheinz [25]. They are very similar to LAFs in that a trapdoor makes it possible to dynamically create arbitrarily many lossy tags using. In particular, each tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ consists of an auxiliary component t_{a} and a core component t_{c} so that, by computing t_{c} as a suitable function of t_{a}, the pair $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ can be made lossy, but still random-looking. The motivation of ABM-LTFs is the construction chosen-ciphertext-secure public-key encryption schemes in scenarios, such as the selective-opening setting [18,2], which involve multiple challenge ciphertexts [25]. They also found applications in the design of universally composable commitments [20]. Lossy algebraic filters differ from ABM-LTFs in that they may not have a trapdoor enabling efficient inversion but, for any lossy tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, the information revealed by $f_{\mathsf{LAF}}(t, x)$ is always the same (i.e., it is completely determined by x and the evaluation key ek).

LAFs were first introduced by Hofheinz [26] as a building block for KDM-CCA-secure encryption schemes, where they enable some form of "plaintext awareness". In the security proofs of KDM-secure encryption schemes (e.g., [10]), the reduction must be able to simulate encryptions of (functions of) the secret key. When the adversary is equipped with a decryption oracle, the ability to publicly compute encryptions of the decryption key may be a problem as decryption queries could end up revealing that key. LAFs provide a way reconcile the conflicting requirements of KDM and CCA2-security by introducing in each ciphertext a LAF-evaluation of the secret key. By having the simulator encrypt a lossy function of the secret key, one can keep encryption queries from leaking too much secret information. At the same time, adversarially-generated ciphertexts always contain an injective function of the key, which prevents the adversary from learning the secret key by publicly generating encryptions of that key.

Recently, Wen and Liu [45] appealed to LAFs in the design of robustly reusable fuzzy extractors. As defined by Dodis *et al.* [17], fuzzy extractors allow one to generate a random cryptographic key R – together with some public helper string P – out of a noisy biometric reading w. The key R need not be stored as it can be reproduced from the public helper string P and a biometric reading w' which is sufficiently close to w. Reusable fuzzy extractors [13] make it possible to safely generate multiple keys $R_1, \ldots, R_t$ (each with its own public helper string P_i) from correlated readings $w_1, \ldots, w_t$ of the same biometric source. Wen and Liu [45] considered the problem of achieving robustness in reusable fuzzy extractors. In short, robustness prevents adversaries from covertly tampering with the public helper string P_i in order to affect the reproducibility of R_i. The Wen-Liu [45] fuzzy extractor relies on LAFs to simultaneously achieve reusability and robustness assuming a common reference string. Their solution requires the LAF to be homomorphic, meaning that function outputs should live in a group and, for any tag t and inputs x_1, x_2, we have $f_{\mathsf{LAF}}(t, x_1 + x_2) = f_{\mathsf{LAF}}(t, x_1) \cdot f_{\mathsf{LAF}}(t, x_2)$. The candidate proposed by Hofheinz [26] and ours are both usable in robustly reusable fuzzy extractors as they both satisfy this homomorphic property. Our scheme offers the benefit of shorter public helper strings P since these have to contain a LAF tag in the fuzzy extractor of [45].

The tightness of cryptographic security proofs was first considered by Bellare and Rogaway [4] in the random oracle model [3]. In the standard model, it drew a lot of attention in digital signatures and public-key encryption the recent years (see, e.g., [29,16,7,35,36,27,21,28,22]). In the context of all-but-many lossy trapdoor functions, a construction with tight evasiveness was put forth by Hofheinz [25]. A tightly secure lattice-based ABM-LTF was described by Libert *et al.* [37] as a tool enabling tight chosen-ciphertext security from lattice assumptions. To our knowledge, the only other prior work considering tight reductions for lossy trapdoor functions is a recent result of Hofheinz and Nguyen [30]. In particular, tight security has never been obtained in the context of LAFs, nor in fuzzy extractors based on public-key techniques.

2 Background

2.1 Lossy Algebraic Filters

We recall the definition of Lossy Algebraic Filter (LAF) from [26], in which the distribution over the function domain may not be the uniform one.

Definition 1. *For integers $\ell_{\mathsf{LAF}}(\lambda), n(\lambda) > 0$, an (ℓ_{LAF}, n)-lossy algebraic filter (*LAF*) with security parameter λ consists of the following PPT algorithms:*

Key generation. $\mathsf{LAF.Gen}(1^\lambda)$ *outputs an evaluation key ek and a trapdoor key tk. The evaluation key ek specifies an ℓ_{LAF}-bit prime p as well as the description of a tag space $\mathcal{T} = \mathcal{T}_{\mathsf{c}} \times \mathcal{T}_{\mathsf{a}}$, where $\mathcal{T}_{\mathsf{c}}$ is efficiently samplable. The disjoint sets of injective and non-injective tags are called $\mathcal{T}_{\mathsf{inj}}$ and $\mathcal{T}_{\mathsf{non\text{-}inj}} = \mathcal{T} \backslash \mathcal{T}_{\mathsf{inj}}$, respectively. We also define the subset of lossy tags $\mathcal{T}_{\mathsf{loss}}$ to be a subset of $\mathcal{T}_{\mathsf{non\text{-}inj}}$, which induce very lossy functions. A tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ is described by a core part $t_{\mathsf{c}} \in \mathcal{T}_{\mathsf{c}}$ and an auxiliary part $t_{\mathsf{a}} \in \mathcal{T}_{\mathsf{a}}$. A tag may be injective, or lossy, or neither. The trapdoor tk allows sampling lossy tags.*

Evaluation. $\mathsf{LAF.Eval}(ek, t, X)$ *takes as inputs an evaluation key ek, a tag $t \in \mathcal{T}$ and a function input $X \in \mathbb{Z}_p^n$. It outputs an image $Y = f_{ek,t}(X)$.*

Lossy tag generation. $\mathsf{LAF.LTag}(tk, t_{\mathsf{a}})$ *takes as input the trapdoor key tk, an auxiliary part $t_{\mathsf{a}} \in \mathcal{T}_{\mathsf{a}}$ and outputs a core part t_{c} such that $t = (t_{\mathsf{c}}, t_{\mathsf{a}}) \in \mathcal{T}_{\mathsf{loss}}$ forms a lossy tag.*

In addition, LAF *has to meet the following requirements:*

Lossiness. *For any $(ek, tk) \overset{R}{\leftarrow} \mathsf{LAF.Gen}(1^\lambda)$, the following conditions should be satisfied.*

a. For any $t \in \mathcal{T}_{\mathsf{inj}}$, $f_{ek,t}(.)$ should behave as an injective function (note that $f_{ek,t}^{-1}(.)$ is not required to be efficiently computable given tk).

b. For any auxiliary tag $t_{\mathsf{a}} \in \mathcal{T}_{\mathsf{a}}$ and any $t_{\mathsf{c}} \overset{R}{\leftarrow} \mathsf{LAF.LTag}(tk, t_{\mathsf{a}})$, we have $t = (t_{\mathsf{c}}, t_{\mathsf{a}}) \in \mathcal{T}_{\mathsf{loss}}$, meaning that $f_{ek,t}(.)$ is a lossy function. Moreover, for *any input $X = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ and any $t = (t_\mathsf{c}, t_\mathsf{a}) \in \mathcal{T}_{\mathsf{loss}}$, $f_{ek,t}(X)$ is completely determined by $\sum_{i=1}^{n} v_i \cdot x_i \bmod p$ for coefficients $\{v_i\}_{i=1}^{n}$ that only depend on ek.*

Indistinguishability. *Multiple lossy tags are computationally indistinguishable from random tags, namely:*

$$\mathbf{Adv}_{Q}^{\mathcal{A},\mathsf{ind}}(\lambda) := \left|\Pr[\mathcal{A}(1^\lambda, ek)^{\mathsf{LAF.LTag}(tk,\cdot)} = 1] - \Pr[\mathcal{A}(1^\lambda, ek)^{\mathcal{O}_{\mathcal{T}_c}(\cdot)} = 1]\right|$$

is negligible for any PPT algorithm $\mathcal{A}$, where $(ek, tk) \overset{R}{\leftarrow} \mathsf{LAF.Gen}(1^\lambda)$ and $\mathcal{O}_{\mathcal{T}_c}(\cdot)$ is an oracle that assigns a random core tag $t_\mathsf{c} \overset{R}{\leftarrow} \mathcal{T}_\mathsf{c}$ to each auxiliary tag $t_\mathsf{a} \in \mathcal{T}_\mathsf{a}$ (rather than a core tag that makes $t = (t_\mathsf{c}, t_\mathsf{a})$ lossy). Here Q denotes the number of oracle queries made by $\mathcal{A}$.

Evasiveness. *Non-injective tags are computationally hard to find, even with access to an oracle outputting multiple lossy tags, namely:*

$$\mathbf{Adv}_{Q_1,Q_2}^{\mathcal{A},\mathsf{eva}}(\lambda) := \Pr[\mathcal{A}(1^\lambda, ek)^{\mathsf{LAF.LTag}(tk,\cdot),\ \mathsf{LAF.IsInjective}(tk,\cdot)} \in \mathcal{T}_{\mathsf{non\text{-}inj}}]$$

is negligible for legitimate adversary $\mathcal{A}$, where $(ek, ik, tk) \overset{R}{\leftarrow} \mathsf{LAF.Gen}(1^\lambda)$ and $\mathcal{A}$ is given access to the following oracle:

- $\mathsf{LAF.LTag}(tk,\cdot)$ *which acts exactly as the lossy tag generation algorithm.*
- $\mathsf{LAF.IsInjective}(tk,\cdot)$ *that takes as input a tag $t = (t_\mathsf{c}, t_\mathsf{a})$. It outputs 0 if $t \in \mathcal{T}_{\mathsf{non\text{-}inj}} = \mathcal{T} \backslash \mathcal{T}_{\mathsf{inj}}$ and 1 if $t \in \mathcal{T}_{\mathsf{inj}}$. If $t \notin \mathcal{T}$, the oracle outputs $\perp$.*

We denote by Q_1 and Q_2 the number of queries to $\mathsf{LAF.LTag}(tk,\cdot)$ and $\mathsf{LAF.IsInjective}(tk,\cdot)$, respectively. By "legitimate adversary", we mean that $\mathcal{A}$ is PPT and never outputs a tag $t = (t_\mathsf{c}, t_\mathsf{a})$ such that t_c was obtained by invoking the $\mathsf{LAF.LTag}$ oracle on t_a.

In our construction, the tag space $\mathcal{T}$ will not be dense (i.e., not all elements of the ambient algebraic structure are potential tags). However, elements of the tag space $\mathcal{T}$ will be efficiently recognizable given ek.

We note that the above definition of evasiveness departs from the one used by Hofheinz [26] in that it uses an additional $\mathsf{LAF.IsInjective}(tk,\cdot)$ oracle that uses the trapdoor tk to decide whether a given tag is injective or not. However, this oracle will only be used in our tightly secure LAF (and not in our first construction). Its only purpose is to enable a modular use of our tightly evasive LAF in applications to KDM security [26] or robustly reusable fuzzy extractors [45]. Specifically, by invoking the $\mathsf{LAF.IsInjective}(tk,\cdot)$ oracle, the reduction from the security of a primitive to the underlying LAF's evasiveness does not have to guess which adversarial query involves a non-lossy tag.

2.2 Chameleon Hash Functions

A chameleon hash function [32] is a tuple of algorithms $\mathsf{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$ which contains an algorithm CMKg that, given a security parameter 1^λ, outputs a key pair $(hk, td) \leftarrow \mathcal{G}(1^\lambda)$. The randomized hashing algorithm outputs $y = \mathsf{CMhash}(hk, m, r)$ given the public key hk, a message m and random coins $r \in \mathcal{R}_{hash}$. On input of messages m, m', random coins $r \in \mathcal{R}_{hash}$ and the trapdoor key td, the switching algorithm $r' \leftarrow \mathsf{CMswitch}(td, m, r, m')$ computes $r' \in \mathcal{R}_{hash}$ such that $\mathsf{CMhash}(hk, m, r) = \mathsf{CMhash}(hk, m', r')$. The collision-resistance property mandates that it be infeasible to come up with pairs $(m', r') \neq (m, r)$ such that $\mathsf{CMhash}(hk, m, r) = \mathsf{CMhash}(hk, m', r')$ without knowing the trapdoor key tk. Uniformity guarantees that the distribution of hash values is independent of the message m: in particular, for all hk, and all messages m, m', the distributions $\{r \leftarrow \mathcal{R}_{hash} : \mathsf{CMHash}(hk, m, r)\}$ and $\{r \leftarrow \mathcal{R}_{hash} : \mathsf{CMHash}(hk, m', r)\}$ are identical.

2.3 Hardness Assumptions

Definition 2. *Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be bilinear groups of order p. The* **First Decision 3-Party Diffie-Hellman** *(D3DH1) assumption holds in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ if no PPT distinguisher can distinguish the distribution*

$$D_1 := \{(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, g^{abc}) \mid g \stackrel{R}{\leftarrow} \mathbb{G}, \hat{g} \stackrel{R}{\leftarrow} \hat{\mathbb{G}},\ a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_p\}$$
$$D_0 := \{(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, g^z) \mid g \stackrel{R}{\leftarrow} \mathbb{G}, \hat{g} \stackrel{R}{\leftarrow} \hat{\mathbb{G}},\ a, b, c, z \stackrel{R}{\leftarrow} \mathbb{Z}_p\}.$$

The D3DH1 assumption has a natural analogue where the pseudorandom value lives in $\hat{\mathbb{G}}$ instead of $\mathbb{G}$.

Definition 3. *The* **Second Decision 3-Party Diffie-Hellman** *(D3DH2) assumption holds in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ if no PPT algorithm can distinguish between the distribution*

$$D_1 := \{(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, \hat{g}^{abc}) \mid g \stackrel{R}{\leftarrow} \mathbb{G}, \hat{g} \stackrel{R}{\leftarrow} \hat{\mathbb{G}},\ a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_p\}$$
$$D_0 := \{(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, \hat{g}^z) \mid g \stackrel{R}{\leftarrow} \mathbb{G}, \hat{g} \stackrel{R}{\leftarrow} \hat{\mathbb{G}},\ a, b, c, z \stackrel{R}{\leftarrow} \mathbb{Z}_p\}.$$

We also need a computational assumption which is implied by D3DH2. The 2-3-CDH was initially introduced [33] in ordinary (i.e., non-pairing-friendly) discrete-logarithm hard groups. Here, we extend it to asymmetric bilinear groups.

Definition 4 ([33]). *Let $(\mathbb{G}, \hat{\mathbb{G}})$ be a bilinear groups of order p with generators $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$. The* **2-out-of-3 Computational Diffie-Hellman** *(2-3-CDH) assumption says that, given $(g, g^a, \hat{g}^a, g^b, \hat{g}^b)$ for randomly chosen $a, b \stackrel{R}{\leftarrow} \mathbb{Z}_p$, no PPT algorithm can find a pair $(g^r, g^{r \cdot ab})$ such that $r \neq 0$.*

It is known (see, e.g., [38]) that any algorithm solving the 2-3-CDH problem can be used to break the D3DH2 assumption. On input of $(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, \hat{g}^z)$, where $z = abc$ or $z \in_R \mathbb{Z}_p$, the reduction can simply run a 2-3-CDH solver on input of $(g, g^a, g^b, \hat{g}^a, \hat{g}^b)$. If the solver outputs a non-trivial pair of the form $(R_1, R_2) = (g^r, g^{r \cdot ab})$, the D3DH2 distinguisher decides that $z = abc$ if and only if $e(R_1, \hat{g}^z) = e(R_2, \hat{g}^c)$.

In our constructions, we actually rely on a weaker variant of D3HD1, called wD3HD1, where $\hat{g}^a$ is not given. In our tightly secure construction (which requires asymmetric pairings), we need to rely on the following variant of wD3HD1.

Definition 5. *Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be bilinear groups of order p. The* **Randomized weak Decision 3-Party Diffie-Hellman** *(R-wD3DH1) assumption holds in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ if no PPT distinguisher can distinguish the distribution*

$$
\begin{aligned}
D_1 := \Big\{ \{(g, \hat{g}, g^{a_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, g^{a_i bc})\}_{i=1}^{Q} \mid g \overset{R}{\leftarrow} \mathbb{G}, \hat{g} \overset{R}{\leftarrow} \hat{\mathbb{G}}, \\
a_1, \ldots, a_Q, b, c \overset{R}{\leftarrow} \mathbb{Z}_p \} \Big\} \\
D_0 := \Big\{ \{(g, \hat{g}, g^{a_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, g^{z_i})\}_{i=1}^{Q} \mid g \overset{R}{\leftarrow} \mathbb{G}, \hat{g} \overset{R}{\leftarrow} \hat{\mathbb{G}}, \\
a_1, \ldots, a_Q, z_1, \ldots, z_Q, b, c \overset{R}{\leftarrow} \mathbb{Z}_p \} \Big\}.
\end{aligned}
$$

We do not know if D3DH1 or wD3DH1 can be tightly reduced to R-wD3DH1 (the only reduction we are aware of proceeds via a hybrid argument). In asymmetric pairings, however, we can give a tight reduction between R-wD3DH1 and a combination of wD3DH1 and SXDH.

Lemma 1. *There is a tight reduction from the wD3DH1 assumption and the DDH assumption in $\mathbb{G}$ to the R-wD3DH1 assumption. More precisely, for any R-wD3DH1 adversary $\mathcal{B}$, there exist distinguishers $\mathcal{B}_1$ and $\mathcal{B}_2$ which run in about the same time as $\mathcal{B}$ and such that*

$$
\mathbf{Adv}_{\mathcal{B}}^{\text{R-wD3DH1}}(\lambda) \leq \mathbf{Adv}_{\mathcal{B}_1}^{\text{wD3DH1}}(\lambda) + \mathbf{Adv}_{\mathcal{B}_2}^{\text{DDH}_1}(\lambda),
$$

where the second term denotes $\mathcal{B}_2$'s advantage as a DDH distinguisher in $\mathbb{G}$.

Proof. To prove the result, we consider the following distribution:

$$
\begin{aligned}
D_{int} := \Big\{ \{(g, \hat{g}, g^{a \cdot \alpha_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, g^{z \cdot \alpha_i})\}_{i=1}^{Q} \mid g \overset{R}{\leftarrow} \mathbb{G},\ \hat{g} \overset{R}{\leftarrow} \hat{\mathbb{G}}, \\
\alpha_1, \ldots, \alpha_Q, b, c, z \overset{R}{\leftarrow} \mathbb{Z}_p,\ a \overset{R}{\leftarrow} \mathbb{Z}_p^{\star} \} \Big\}
\end{aligned}
$$

A straightforward reduction shows that, under the wD3DH1 assumption, D_1 is computationally indistinguishable from D_{int}. We show that, under the DDH assumption in $\mathbb{G}$, D_{int} is computationally indistinguishable from D_0. Moreover, the reduction is tight in that the two distinguishers have the same advantage.

First, we show that, under the wD3DH1 assumption, D_{int} is computationally indistinguishable from D_1.

We can build a wD3DH1 distinguisher $\mathcal{B}_1$ from any distinguisher for D_1 and D_{int}. With $(g, \hat{g}, g^a, g^b, g^c, \hat{g}^b, \hat{g}^c, T)$ as input where $g \overset{R}{\leftarrow} \mathbb{G}$, $\hat{g} \overset{R}{\leftarrow} \hat{\mathbb{G}}$ and $a, b, c \overset{R}{\leftarrow} \mathbb{Z}_p$, $\mathcal{B}_1$ uniformly draws $\alpha_i, \ldots, \alpha_Q \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes

$$
D_{\mathcal{B}_1} := \Big\{ \{(g, \hat{g}, g^{a \cdot \alpha_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, T^{\alpha_i})\}_{i=1}^{Q} \mid \alpha_1, \ldots, \alpha_Q \overset{R}{\leftarrow} \mathbb{Z}_p \Big\}.
$$

It is easy to see that if $T = g^{abc}$, then D_{B_1} is identical to D_1. If $T \in_R \mathbb{G}$, then D_{B_1} is distributed as D_{int}. Hence, any distinguisher between D_1 and D_{int} with $D_{\mathcal{B}_1}$ implies a distinguisher $\mathcal{B}_1$ for the wD3DH1 problem.

Next, we show that, under the DDH assumption in $\mathbb{G}$, D_{int} is computationally indistinguishable from D_0. In order to build a DDH distinguisher $\mathcal{B}_2$ out of a distinguisher between D_{int} and D_0, we use the random self-reducibility of the DDH assumption.

Lemma 2 (Random Self-Reducibility [39]). *Letting $\mathbb{G}$ be a group of prime order p, there exists a PPT algorithm R that takes as input $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, for any $a, b, c \in \mathbb{Z}_p$, and returns a triple $(g^a, g^{b'}, g^{c'}) \in \mathbb{G}^3$ such that:*

- *If $c = ab \mod q$, then b' is uniformly random in $\mathbb{Z}_p$ and $c' = ab'$.*
- *If $c \neq ab \mod q$, then $b', c' \in_R \mathbb{Z}_p$ are independent and uniformly random.*

On input of $(g, g^z, g^\alpha, T) \in \mathbb{G}^4$, where $g \xleftarrow{R} \mathbb{G}$ and $z, \alpha \xleftarrow{R} \mathbb{Z}_p$, $\mathcal{B}_2$ uses algorithm R to generate Q instances $\{(g^z, g^{\alpha_i}, T_i)\}_{i=1}^Q$. Next, $\mathcal{B}_2$ draws $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$, $a, b, c \xleftarrow{R} \mathbb{Z}_p$ and defines the following distribution:

$$D_{\mathcal{B}_2} := \Big\{ \{(g, \hat{g}, (g^{\alpha_i})^a, g^b, g^c, \hat{g}^b, \hat{g}^c, T_i)\}_{i=1}^Q \mid \hat{g} \xleftarrow{R} \hat{\mathbb{G}}, a, b, c \xleftarrow{R} \mathbb{Z}_p \Big\}.$$

We observe that, if $T = g^{z \cdot \alpha}$, we have $T_i = g^{z \cdot \alpha_i}$ for all $i \in [Q]$. In this case, $D_{\mathcal{B}_2}$ is identical to D_{int}. In contrast, if $T \in_R \mathbb{G}$, the random self-reducibility ensures that $T_1, \ldots, T_Q \in_R \mathbb{G}$ are i.i.d, meaning that $D_{\mathcal{B}_2}$ is identical to D_0. Using a distinguisher between D_{int} and D_0 and feeding it with $D_{\mathcal{B}_2}$, we obtain a distinguisher $\mathcal{B}_2$ for the DDH problem in $\mathbb{G}$. □

3 A Lossy Algebraic Filter With Linear-Size Tags

We present a LAF based on DDH-like assumptions with tags of size $O(n)$, where n is the number of input symbols when the input is viewed as a vector over $\mathbb{Z}_p$. Our tags are comprised of $4n$ elements of $\mathbb{G}$, which outperforms the construction of [26] for $n > 4$. In his application to KDM-CCA security [26], Hofheinz uses a LAF scheme with $n = 6$, in which case we decrease the tag size from 43 to 24 group elements[4] and thus shorten ciphertexts by 19 group elements.

The construction is inspired by the lossy TDF of [14] and relies on the revocation technique of Lewko, Sahai and Waters [34] (LSW) in the same way. In asymmetric pairings $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$, the evaluation key contains a set of LSW ciphertexts $\{(\hat{V}_j = \hat{g}^{v_j}, \hat{H}_j = (\hat{h}^j \cdot \hat{u})^{v_j})\}_{j=1}^n$, while each core tag component t_c can be seen as containing a set of LSW secret keys $\{(R_i, S_i) = (g^{r_i}, (h^i \cdot u)^{r_i})\}_{i=1}^n$, allowing the evaluator compute $M_{i,j} = e(g, \hat{h})^{r_i \cdot v_j}$ for any pairwise distinct indices $i \neq j$. In lossy tags $(t_\mathsf{c}, t_\mathsf{a})$, diagonal elements $\{M_{i,i}\}_{i=1}^n$ are handled by having t_c contain Waters signatures $(D_i, E_i) = (h^{r_i \cdot v_i} \cdot H_{\mathbb{G}}(\tau)^{\rho_i}, g^{\rho_i})$, where $\rho_i \in_R \mathbb{Z}_p$ and $H_{\mathbb{G}} : \{0,1\}^L \to \mathbb{G}$ is an algebraic hash function mapping the output τ of a chameleon hash function to the group $\mathbb{G}$. For indistinguishability purposes, pairs

[4] The LAF of [26] was described in terms of symmetric pairings but it extends to asymmetric pairings $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ where tags are comprised of elements in $\mathbb{G}$.

$\{(D_i, E_i)\}_{i=1}^n$ are not immediately recognizable as Waters signatures because the underlying secret key h^{v_i} is blinded by a random exponent $r_i = \log_g(R_i)$. Still, running the verification algorithm of Waters signatures on (D_i, E_i) allows the evaluation algorithm to derive $M_{i,i} = e(g, \hat{h})^{r_i \cdot v_i}$, so that $(M_{i,j})_{i,j\in[n]}$ forms a rank-1 matrix. In injective tags, $\{(D_i, E_i)\}_{i=1}^n$ are uniformly distributed in $\mathbb{G}$, so that $(M_{i,j})_{i,j\in[n]}$ is the sum of a rank-1 matrix and a diagonal matrix.

3.1 Description

Key generation. $\mathsf{LAF.Gen}(1^\lambda)$ conducts the following steps.

1. Choose bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with random generators $g, h, u \xleftarrow{R} \mathbb{G}$ and $\hat{g}, \hat{h}, \hat{u} \xleftarrow{R} \hat{\mathbb{G}}$ subject to the constraints $\log_g(h) = \log_{\hat{g}}(\hat{h})$ and $\log_g(u) = \log_{\hat{g}}(\hat{u})$.
2. Choose a chameleon hash function $\mathsf{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$, where the hashing algorithm $\mathsf{CMhash} : \{0,1\}^* \times \mathcal{R}_{hash} \to \{0,1\}^L$ has output length $L \in \mathsf{poly}(\lambda)$. Generate a pair $(hk_{\mathsf{CMH}}, td_{\mathsf{CMH}}) \leftarrow \mathsf{CMKg}(1^\lambda)$ made of a hashing key hk_{CMH} and a trapdoor td_{CMH}.
3. Choose random exponents $w_0, \ldots, w_L \xleftarrow{R} \mathbb{Z}_p$ and define
$$W_k = g^{w_k}, \qquad \hat{W}_k = \hat{g}^{w_k} \qquad \forall k \in [0, L]$$
that will be used to instantiate two hash functions $H_{\mathbb{G}} : \{0,1\}^L \to \mathbb{G}$, $H_{\hat{\mathbb{G}}} : \{0,1\}^L \to \hat{\mathbb{G}}$ which map any string $\mathsf{m} \in \{0,1\}^L$ to
$$H_{\mathbb{G}}(\mathsf{m}) = W_0 \cdot \prod_{k=1}^{L} W_k^{\mathsf{m}[k]}, \qquad H_{\hat{\mathbb{G}}}(\mathsf{m}) = \hat{W}_0 \cdot \prod_{k=1}^{L} \hat{W}_k^{\mathsf{m}[k]},$$
respectively. Note that $e(g, H_{\hat{\mathbb{G}}}(\mathsf{m})) = e(H_{\mathbb{G}}(\mathsf{m}), \hat{g})$ for any $\mathsf{m} \in \{0,1\}^L$.
4. Let $n \in \mathsf{poly}(n)$ be the desired input length. For each $j \in [n]$, choose $v_j \xleftarrow{R} \mathbb{Z}_p$ and define
$$\hat{V}_j = \hat{g}^{v_j}, \qquad \hat{H}_j = (\hat{h}^j \cdot \hat{u})^{v_j} \qquad \forall j \in [n].$$
5. Output the evaluation key ek and the lossy tag generation key tk, which consist of
$$ek := \Big(hk_{\mathsf{CMH}},\ g,\ h,\ u,\ \hat{g},\ \hat{h},\ \hat{u},\ \{W_k, \hat{W}_k\}_{k=0}^L,\ \ \{\hat{V}_j, \hat{H}_j\}_{j=1}^n\Big),$$
$$tk := \big(td_{\mathsf{CMH}},\ \{v_j\}_{j=1}^n\big).$$

The tag space $\mathcal{T} = \mathcal{T}_{\mathsf{c}} \times \mathcal{T}_{aux}$ is defined as a product of $\mathcal{T}_{\mathsf{a}} = \{0,1\}^*$ and
$$\mathcal{T}_{\mathsf{c}} := \{\big(\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash}\big) \mid r_{hash} \in \mathcal{R}_{\mathsf{CMH}} \ \wedge \\ \forall i \in [n] : (R_i, S_i, D_i, E_i) \in \mathbb{G}^{*4} \ \wedge \ e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})\},$$
where $\mathbb{G}^* := \mathbb{G} \setminus \{1_{\mathbb{G}}\}$. The range of the function family is $\mathsf{Rng}_\lambda = \mathbb{G}_T^{n+1}$ and its domain is $\mathbb{Z}_p^n$.

Lossy tag generation. $\mathsf{LAF.LTag}(tk, t_{\mathsf{a}})$ takes in an auxiliary tag component $t_{\mathsf{a}} \in \{0,1\}^*$ and uses $tk = \big(td_{\mathsf{CMH}}, \{v_j\}_{j=1}^n, \{w_k\}_{k=0}^L\big)$ to generate a lossy tag as follows.

1. For each $i \in [n]$, choose $r_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and compute
$$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n]. \tag{3}$$
2. For each $i \in [n]$, choose $\rho_i \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute
$$D_i = h^{r_i \cdot v_i} \cdot H_{\mathbb{G}}(\tau)^{\rho_i}, \qquad E_i = g^{\rho_i} \qquad \forall i \in [n],$$
where $\tau \in \{0,1\}^L$ is chosen uniformly in the range of CMhash.
3. Use the trapdoor td_{CMH} to find $r_{hash} \in \mathcal{R}_{hash}$ such that
$$\tau = \mathsf{CMhash}\big(hk_{hash}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i\}_{i=1}^n), r_{hash}\big) \in \{0,1\}^L$$
and output the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, where $t_{\mathsf{c}} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$.

Each lossy tag is associated with a matrix $\big(M_{i,j}\big)_{i,j \in [n]} = \big(e(g, \hat{h})^{r_i \cdot v_j}\big)_{i,j}$, which is a rank-1 matrix in the exponent. Its diagonal entries consist of
$$M_{i,i} = \frac{e(D_i, \hat{g})}{e(E_i, H_{\hat{\mathbb{G}}}(\tau))} = e(g, \hat{h})^{r_i \cdot v_i} \qquad \forall i \in [n], \tag{4}$$
while its non-diagonal entries
$$M_{i,j} = \Big(\frac{e(R_i, \hat{H}_j)}{e(S_i, \hat{V}_j)}\Big)^{1/(j-i)} = e(g, \hat{h})^{r_i \cdot v_j} \qquad \forall (i,j) \in [n] \times [n] \setminus \{(i,i)\}_{i=1}^n, \tag{5}$$
are obtained by pairing tag component (R_i, S_i) with evaluation key components $(\hat{V}_j, \hat{H}_j)$.

Random Tags. A random tag can be publicly sampled as follows.

1. For each $i \in [n]$, choose $r_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and compute $\{R_i, S_i\}_{i=1}^n$ as in (3).
2. For each $i \in [n]$, choose $(D_i, E_i) \overset{R}{\leftarrow} \mathbb{G}^* \times \mathbb{G}^*$ uniformly at random.
3. Choose $r_{hash} \overset{R}{\leftarrow} \mathcal{R}_{hash}$.

Finally, output the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, where $t_{\mathsf{c}} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$.

We note that, in both random and lossy tags, we have $e(R_i, \hat{u}^i \cdot \hat{h}) = e(S_i, \hat{g})$ for all $i \in [n]$, so that elements of $\mathcal{T}$ are publicly recognizable.

Evaluation. $\mathsf{LAF.Eval}(ek, t, \mathbf{x})$ takes in the function input $\mathbf{x} \in \mathbb{Z}_p^n$ as well as the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$. It parses t_{c} as $(\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ and proceeds as follows.

1. Return $\perp$ if there exists $i \in [n]$ such that $e(R_i, \hat{h}^i \cdot \hat{u}) \neq e(S_i, \hat{g})$.

2. Compute the matrix $\big(M_{i,j}\big)_{i,j\in[n]} \in \mathbb{G}_T^{n\times n}$ as

$$M_{i,i} = \frac{e(D_i, \hat{g})}{e(E_i, H_{\hat{\mathbb{G}}}(\tau))} \qquad\qquad \forall i \in [n] \qquad , \tag{6}$$

where $\tau = \mathsf{CMhash}\big(hk_{hash}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i\}_{i=1}^{n}), r_{hash}\big)$, and

$$M_{i,j} = \Big(\frac{e(R_i, \hat{H}_j)}{e(S_i, \hat{V}_j)}\Big)^{1/(j-i)} \qquad \forall (i,j) \in [n]\times[n] \setminus \{(i,i)\}_{i=1}^{n}, \tag{7}$$

Note that, since $R_i = g^{r_i}$ and $S_i = (h^i \cdot u)^{r_i}$ for some $r_i \in \mathbb{Z}_q$, we have

$$\begin{aligned} M_{i,i} &= e(g,\hat{h})^{r_i\cdot v_i + \omega_i}, &\qquad& \forall i \in [n] \\ M_{i,j} &= e(g,\hat{h})^{r_i\cdot v_j}, && \forall i \neq j, \end{aligned} \tag{8}$$

for some vector $(\omega_1, \ldots, \omega_n)^\top \in \mathbb{Z}_p^n$ that only contains non-zero entries if $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ is injective.

3. Compute the vector $\big(V_{T,j}\big)_{j\in[n]}$ as $V_{T,j} = e(h, \hat{V}_j) = e(g,\hat{h})^{v_j}$ for each $j \in [n]$.
4. Use the input $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{Z}_p^n$ to compute

$$\begin{aligned} Y_0 &= \prod_{j=1}^{n} V_{T,j}^{x_j} \\ Y_i &= \prod_{j=1}^{n} M_{i,j}^{x_j} &\qquad& \forall i \in [n] \end{aligned} \tag{9}$$

and output $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_n)^\top \in \mathbb{G}_T^{n+1}$.

While the above construction inherits the $\Theta(\lambda)$-size public keys of Waters signatures [43], we believe that it can be adapted to other signature schemes in the standard model (e.g., [8,31]) so as to obtain shorter evaluation keys.

Injectivity and lossiness. For any injective tag, all entries of the vector $(\omega_1, \ldots, \omega_n)^\top$ are non-zero in (8). We can use Y_0 to ensure that the function is injective. As long as $\omega_i \neq 0$ for all $i \in [n]$, the evaluation algorithm (9) yields a vector $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_n) \in \mathbb{G}_T^{n+1}$ of the form

$$\begin{aligned} Y_0 &= e(g,\hat{h})^{\sum_{j=1}^{n} v_j\cdot x_j} \\ Y_i &= e(g,\hat{h})^{\omega_i\cdot x_i + r_i\cdot\sum_{j=1}^{n} v_j\cdot x_j} &\qquad& \forall i \in [n], \end{aligned}$$

meaning that $x_i \in \mathbb{Z}_p$ is uniquely determined by (Y_0, Y_i) and (R_i, D_i, E_i) (note that the triple (R_i, D_i, E_i) uniquely defines ω_i).

For any lossy tag, the evaluation outputs $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_n) \in \mathbb{G}_T^{n+1}$ such that

$$\begin{aligned} Y_0 &= e(g, \hat{h})^{\sum_{j=1}^{n} v_j \cdot x_j} \\ Y_i &= e(g, \hat{h})^{r_i \cdot \sum_{j=1}^{n} v_j \cdot x_j} \qquad\qquad \forall i \in [n], \end{aligned}$$

which always reveals the same information $\sum_{j=1}^{n} v_j \cdot x_j \bmod p$ about the input vector $\mathbf{x} = (x_1, \ldots, x_n)^\top$, no matter which tag is used.

3.2 Security

The proof of indistinguishability relies on the wD3DH1 assumption via a hybrid argument over the queries to the $\mathsf{LAF.LTag}(tk, \cdot)$ oracle and over the pairs $\{(D_i, E_i)\}_{i=1}^{n}$ produced by $\mathsf{LAF.LTag}(tk, \cdot)$ at each query. Using the R-wD3DH1 assumption, it is possible to modify the proof so as to use a hybrid argument over the pairs $\{(D_i, E_i)\}_{i=1}^{n}$ only (meaning that all queries to $\mathsf{LAF.LTag}(tk, \cdot)$ are processed in parallel at each game transition). However, this proof would require the SXDH assumption – which only holds in asymmetric pairings – to apply the result of Lemma 1. In contrast, the proof of Theorem 1 allows instantiations in all bilinear group configurations, even in symmetric pairings.

The proof of Theorem 1 uses a hybrid argument to gradually replace pairs $\{(D_i, E_i)\}_{i=1}^{n}$ by truly random group elements in outputs of the lossy tag generation oracle. To this end, it relies on the proof technique of the Boneh-Boyen IBE [11] in the proof of Lemma 3. Namely, in order to embed a D3DH1 instance $(g, h, g^{v_k}, g^{r_k}, T \overset{?}{=} h^{r_k \cdot v_k})$ in the k-th pair (D_k, E_k), for indexes $i > k$, the reduction has to simulate $h^{r_i \cdot v_k}$ for a known $r_i \in \mathbb{Z}_p$ and an unknown h^{v_k}.

Theorem 1. *The above LAF provides indistinguishability under the wD3DH1 assumption in* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$.

Proof. We first recall that, for any injective or non-injective tag $t = (t_\mathsf{c}, t_\mathsf{a})$, the core component $t_\mathsf{c} = (\{R_i, S_i, D_i, E_i\}_{i=1}^{n}, r_{hash})$ imply a matrix $\big(M_{i,j}\big)_{i,j \in [n]}$ where the off-diagonal entries are $M_{i,j} = e(g, \hat{h})^{r_i \cdot v_j}$ and the diagonal entries are of the form (8). In injective tags, the vector $(\omega_1, \ldots, \omega_n)^\top \in \mathbb{Z}_p^n$ only contains non-zero entries. In lossy tags, we have $(\omega_1, \ldots, \omega_n)^\top = \mathbf{0}^n$. We define a sequence of hybrid games. In $\mathsf{Game}_{(0,0)}$, the adversary has access to the real oracle $\mathsf{LAF.LTag}(tk, .)$ oracle that always outputs lossy tags. In $\mathsf{Game}_{(Q,n)}$, the adversary is given access to an oracle $\mathcal{O}_{\mathcal{T}_\mathsf{c}}(.)$ that always outputs random tags.

$\mathsf{Game}_{(\ell,k)}$ $(1 \leq \ell \leq Q, 1 \leq k \leq n)$**:** In this game, the adversary interacts with a hybrid oracle $\mathsf{LAF.LTag}^{(\ell,k)}(tk, .)$. At the μ-th query, this oracle outputs tags $t^{(\mu)} = (t_\mathsf{c}^{(\mu)}, t_\mathsf{a}^{(\mu)})$ such that

- If $\mu < \ell$, the tag $t_\mathsf{c}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^{n}, r_{hash})$ implies a matrix $\big(M_{i,j}^{(\mu)}\big)_{i,j \in [n]}$ of the form (8) where $(\omega_1^{(\mu)}, \ldots, \omega_n^{(\mu)})^\top$ is uniform over $\mathbb{Z}_p^n$

- If $\mu = \ell$, $t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ implies a matrix $\big(M_{i,j}^{(\mu)}\big)_{i,j \in [n]}$ of the form (8) where the first k entries of $(\omega_1^{(\mu)}, \ldots, \omega_n^{(\mu)})^\top$ are uniform over $\mathbb{Z}_p$ and its last $n-k$ entries are zeroes.
- If $\mu > \ell$, the matrix $\big(M_{i,j}^{(\mu)}\big)_{i,j \in [n]}$ implied by the core tag component $t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ is a rank-1 matrix in the exponent since $(\omega_1^{(\mu)}, \ldots, \omega_n^{(\mu)})^\top = \mathbf{0}^n$.

Lemma 3 shows that, for all pairs $(\ell, k) \in [Q] \times [n]$, these games are computationally indistinguishable from one another, which yields the stated result. □

Lemma 3. *For each $k \in [n]$ and $\ell \in [Q]$, $\mathsf{Game}_{(\ell,k)}$ is computationally indistinguishable from $\mathsf{Game}_{(\ell,k-1)}$ if the wD3DH1 assumption holds. Under the same assumption, $\mathsf{Game}_{(\ell,1)}$ is computationally indistinguishable from $\mathsf{Game}_{(\ell-1,n)}$.*

Proof. For the sake of contradiction, assume that there exists $\ell \in [Q]$, $k \in [n]$ such that the adversary $\mathcal{A}$ can distinguish $\mathsf{Game}_{(\ell,k)}$ from $\mathsf{Game}_{(\ell,k-1)}$ with noticeable advantage (the indistinguishability of $\mathsf{Game}_{(\ell-1,n)}$ and $\mathsf{Game}_{(\ell,1)}$ can be proved in a completely similar way). We build a wD3DH1 distinguisher $\mathcal{B}$ that inputs $(g, \hat{g}, g^a, g^b, g^c, \hat{g}^b, \hat{g}^c, T)$ with the goal of deciding if $T = g^{abc}$ or $T \in_R \mathbb{G}$.

To this end, $\mathcal{B}$ defines $h = g^b$, $\hat{h} = \hat{g}^b$ and $\hat{V}_k = \hat{g}^c$. It picks $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and defines $\hat{u} = \hat{h}^{-k} \cdot \hat{g}^\alpha$ as well as $u = h^{-k} \cdot g^\alpha$, which implicitly sets $v_k = c$. This allows defining

$$\hat{H}_k = (\hat{h}^k \cdot \hat{u})^c = (\hat{g}^c)^\alpha,$$

In addition, $\mathcal{B}$ defines $(W_0, W_1, \ldots, W_L) \in \mathbb{G}^{L+1}$ and $(\hat{W}_0, \hat{W}_1, \ldots, \hat{W}_L) \in \hat{\mathbb{G}}^{L+1}$ by setting

$$W_i = (g^b)^{\alpha_i} \cdot g^{\beta_i}, \qquad \hat{W}_i = (\hat{g}^b)^{\alpha_i} \cdot \hat{g}^{\beta_i} \quad \forall i \in \{0, \ldots, L\}$$

for randomly chosen $\alpha_0, \ldots, \alpha_L \stackrel{R}{\leftarrow} \mathbb{Z}_p$, $\beta_0, \ldots, \beta_L \stackrel{R}{\leftarrow} \mathbb{Z}_p$. Then, $\mathcal{B}$ chooses $v_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ for each $i \in [n] \setminus \{k\}$ and defines the rest of the evaluation key ek by setting

$$\hat{V}_i = \hat{g}^{v_i}, \qquad \hat{H}_i = (\hat{h}^i \cdot \hat{u})^{v_i}, \qquad \forall i \in [n] \setminus \{k\}$$

Then, at each invocation of the $\mathsf{LAF.LTag}(tk, .)$ oracle, $\mathcal{B}$ responds as follows. At the μ-th query $t_{\mathsf{a}}^{(\mu)}$, it generates a core tag $t_{\mathsf{c}}^{(\mu)}$ such that

- If $\mu < \ell$, $t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ contains $\{\hat{D}_i, \hat{E}_i\}_{i=1}^n$ uniformly random pairs whereas $\{R_i, \hat{S}_i\}_{i=1}^n$ are chosen as in the real algorithm sampling random tags.
- If $\mu = \ell$, $t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ is generated as follows. It sets

$$R_k = g^a, \qquad S_k = (g^a)^\alpha.$$

As for indexes $i \neq k$, it chooses $r_1, \ldots, r_{k-1}, r_{k+1}, \ldots, r_n \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sets

$$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n] \setminus \{k\}.$$

It generates the pairs $\{D_i, E_i\}_{i=1}^{n}$ by choosing $(D_i, E_i) \overset{R}{\leftarrow} \mathbb{G}^2$ at random for each $i \in [k-1]$. The k-th pair (D_k, E_k) is defined as

$$D_k = T \cdot H_{\mathbb{G}}(\tau)^{\rho_k}, \qquad\qquad E_k = g^{\rho_k}. \tag{10}$$

for a randomly chosen $\rho_k \overset{R}{\leftarrow} \mathbb{Z}_p$. As for $\{D_i, E_i\}_{i=k+1}^{n}$, they are obtained by choosing a random $\tau = \tau[1] \dots \tau[L] \in \{0,1\}^L$ in the range of CMhash and choosing $\rho_i \overset{R}{\leftarrow} \mathbb{Z}_p$ before setting

$$\begin{aligned} D_i &= H_{\mathbb{G}}(\tau)^{\rho_i} \cdot (g^c)^{-r_i \cdot \frac{\beta_0 + \sum_{i=1}^{L} \beta_i \cdot \tau[i]}{\alpha_0 + \sum_{i=1}^{L} \alpha_i \cdot \tau[i]}} \\ E_i &= g^{\rho_i} \cdot (g^c)^{-\frac{r_i}{\alpha_0 + \sum_{i=1}^{L} \alpha_i \cdot \tau[i]}} \end{aligned} \tag{11}$$

which can be written

$$\begin{aligned} D_i &= g^{bc \cdot r_i} \cdot H_{\mathbb{G}}(\tau)^{\tilde{\rho}_i} = h^{v_k \cdot r_i} \cdot H_{\mathbb{G}}(\tau)^{\tilde{\rho}_i} \\ E_i &= g^{\tilde{\rho}_i} \end{aligned}$$

if we define $\tilde{\rho}_i = \rho_i - \frac{c \cdot r_i}{\alpha_0 + \sum_{i=1}^{L} \alpha_i \cdot \tau[i]}$. Note that the reduction $\mathcal{B}$ fails if $\alpha_0 + \sum_{i=1}^{L} \alpha_i \cdot \tau[i] = 0$ but this only occurs with negligible chance since the coordinates $(\alpha_0, \dots, \alpha_L) \in \mathbb{Z}_p^L$ are independent of $\mathcal{A}$'s view. Finally, $\mathcal{B}$ uses the trapdoor td_{CMH} of the chameleon hash function to find coins $r_{hash} \in \mathcal{R}_{\mathsf{CMH}}$ such that $\tau = \mathsf{CMhash}\big(hk_{hash}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i\}_{i=1}^{n}), r_{hash}\big)$.

- If $\mu > \ell$, the tags are generated as lossy tags. To this end, $\mathcal{B}$ proceeds as in the previous case, except that all elements $\{D_i, E_i\}_{i=1}^{n}$ (and not only the last $n-k$ ones) are generated as per (11).

It is easy to see that, if $T = g^{abc}$, the pair (D_k, E_k) of (10) can be written

$$D_k = h^{v_k \cdot r_k} \cdot H_{\mathbb{G}}(\tau)^{\rho_k}, \qquad\qquad E_k = g^{\rho_k},$$

meaning that $\mathcal{A}$'s view is the same as in $\mathsf{Game}_{(\ell,k-1)}$. In contrast, if $T \in_R \mathbb{G}$, then (D_k, E_k) can be written

$$D_k = h^{\omega_k + v_k \cdot r_k} \cdot H_{\mathbb{G}}(\tau)^{\rho_k}, \qquad\qquad E_k = g^{\rho_k},$$

for some uniformly random $\omega_k \in_R \mathbb{Z}_p$. In this case, $\mathcal{A}$'s view corresponds to $\mathsf{Game}_{(\ell,k)}$. □

The evasiveness property is established by Theorem 2 for which a proof is given in the full version of the paper.

Theorem 2. *The above LAF provides evasiveness assuming that: (i)* CMH *is a collision-resistant chameleon hash function; (ii) The wD3DH1 and* 2-3-*CDH assumptions both hold in* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$.

Recall that the wD3DH1 and 2-3-CDH assumptions are implied by the D3DH1 and D3DH2 assumptions, respectively. Theorem 1 and Theorem 2 thus guarantee the D3DH1 and D3DH2 assumptions suffice to ensure the indistinguishability and evasiveness properties of our LAF construction (indeed, chameleon hash functions also exist under these assumptions).

3.3 Towards All-But-Many Lossy Trapdoor Functions

Our LAF construction can be modified to construct an all-but-many lossy trapdoor function [25]. Recall that ABM-LTFs do not require evaluations on lossy tags to always output the same information about the input: on any lossy tag, the image size is only required to be much smaller. On the other hand, ABM-LTFs require that, for any injective tag, the function be efficiently invertible using a trapdoor.

Our construction can be turned into an ABM-LTF in the following way. In the evaluation algorithm, a binary input vector $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \{0,1\}^n$ is mapped to the output $(Y_0, \ldots, Y_n) \in \mathbb{G}_T^{n+1}$, where $Y_0 = \prod_{i=1}^n e(R_i, \hat{h})^{x_i}$ and

$$Y_j = \prod_{i=1}^n M_{i,j}^{x_i} \qquad \forall j \in [n],$$

which can be written

$$\begin{aligned} Y_0 &= e(g, \hat{h})^{\sum_{i=1}^n r_i \cdot x_i} \\ Y_j &= e(g, \hat{h})^{\omega_j \cdot x_j + v_j \cdot \sum_{i=1}^n r_i \cdot x_i} \qquad \forall j \in [n]. \end{aligned}$$

Using $ik = (v_1, \ldots, v_n) \in \mathbb{Z}_p^n$ as an inversion key, one can decode the j-th input bit as $x_j = 0$ (resp. $x_j = 1$) if $Y_j / Y_0^{v_j} = 1_{\mathbb{G}_T}$ (resp. $Y_j / Y_0^{v_j} \neq 1_{\mathbb{G}_T}$).

Unfortunately, the above ABM-LTF does not seem immediately usable in the application to selective-opening chosen-ciphertext security, which was suggested in [25]. The reason is that our tags have a special and publicly recognizable structure, where (R_i, S_i) both depend on the same exponent $r_i \in \mathbb{Z}_p$. In the selective-opening setting, the problem arises when the adversary chooses to corrupt some senders, at which point the reduction should reveal the random coins used to create lossy/injective tags. In our construction, this would entail to reveal $r_i \in \mathbb{Z}_p$, which is incompatible with our proofs of indistinguishability and evasiveness. In the ABM-LTF constructions of [25,37], lossy tags are explainable because they are pseudorandom, which allows the reduction to pretend that they have been randomly sampled in their ambient space. Here, the special structure of lossy/injective tags prevents us from explaining the generation of lossy tags in the same way for corrupted senders. The only apparent way to sample a pair (R_i, S_i) satisfying $e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})$ is to choose $r_i \in \mathbb{Z}_p$ and compute $(R_i, S_i) = (g^{r_i}, (h^i \cdot u)^{r_i})$.

We thus leave it as an open problem to build an ABM-LTF with explainable linear-size tags under DDH-like assumptions.

4 A Lossy Algebraic Filter With Tight Security

In this section, we modify our first LAF construction in such a way that we can prove it tightly secure under constant-size assumptions.[5] To this end, we replace

[5] While the assumption of Definition 5 is described using $O(Q)$ group elements, it tightly reduces to wD3DH1 and DDH which both take a constant number of group elements to describe.

Waters signatures by a variant of the MAC described by Blazy, Kiltz and Pan [7], which is itself inspired by the Naor-Reingold PRF [39].

4.1 A Variant of the BKP MAC

The MAC construction below is identical to the signature scheme implied by the IBE scheme of [7, Appendix D] with two differences which prevent public verification in order to obtain a pseudo-random MAC instead of a digital signature. The signature scheme of [7] was actually designed by transposing a pseudo-random MAC from standard DDH-hard groups to bilinear groups in order to enable public verification. Here, we cannot immediately use the MAC of [7] because we need bilinear maps in the evaluation algorithm of our LAF.

In order to obtain a pseudo-random MAC, we thus modify the signature scheme of [7] by introducing an additional randomizer $r \in \mathbb{Z}_p$ and an extra group element h, of which the discrete logarithm $\log_g(h)$ serves as a private verification key.

Keygen$(1^\lambda, 1^L)$**:** Given a security parameter λ and a message length $L \in \mathsf{poly}(\lambda)$, choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with generators $g, h \stackrel{R}{\leftarrow} \mathbb{G}$, $\hat{g} \stackrel{R}{\leftarrow} \hat{\mathbb{G}}$.

1. Choose $\theta, \alpha, \beta \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and compute $\hat{g}^\theta \in \hat{\mathbb{G}}$. For each $\mu \in \{0,1\}$, choose vectors $\boldsymbol{x}_\mu = (x_{1,\mu}, \ldots, x_{L,\mu}) \stackrel{R}{\leftarrow} \mathbb{Z}_p^L$, $\boldsymbol{y}_\mu = (y_{1,\mu}, \ldots, y_{L,\mu}) \stackrel{R}{\leftarrow} \mathbb{Z}_p^L$.
2. Set $v = \alpha + \theta \cdot \beta$ and $\boldsymbol{z}_\mu = \boldsymbol{x}_\mu + \theta \cdot \boldsymbol{y}_\mu \in \mathbb{Z}_p^L$. Compute $\hat{V} = \hat{g}^v$ and, for each $\mu \in \{0,1\}$, define $\hat{\boldsymbol{Z}}_\mu = (\hat{Z}_{1,\mu}, \ldots, \hat{Z}_{L,\mu}) = \hat{g}^{\boldsymbol{z}_\mu}$.

Output a secret key $\mathsf{sk}_{mac} = (\alpha, \beta, \boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{y}_0, \boldsymbol{y}_1, \eta)$, where $\eta = \log_g(h)$, and public parameters consisting of $\mathsf{pp} = \big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, h, \hat{g}^\theta, (\hat{V}, \hat{\boldsymbol{Z}}_0, \hat{\boldsymbol{Z}}_1)\big)$.

Mac.Sig$(\mathsf{pp}, \mathsf{sk}_{mac}, M)$**:** To generate a MAC for $M = m[1] \ldots m[L] \in \{0,1\}^L$ using $\mathsf{sk}_{mac} = (x, y, \boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{y}_0, \boldsymbol{y}_1, \eta)$, choose $r, \rho \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and compute

$$\begin{aligned}\sigma_1 &= h^{\alpha \cdot r} \cdot g^{\rho \cdot (\sum_{k=1}^L x_{k,m[k]})} \\ \sigma_2 &= h^{\beta \cdot r} \cdot g^{\rho \cdot (\sum_{k=1}^L y_{k,m[k]})} \\ \sigma_3 &= g^\rho \\ \sigma_4 &= g^r\end{aligned}$$

Mac.Ver$(\mathsf{pp}, \mathsf{sk}_{mac}, M, \sigma)$**:** Given $\mathsf{sk}_{mac} = (\alpha, \beta, \boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{y}_0, \boldsymbol{y}_1, \eta)$ and an L-bit message $M = m[1] \ldots m[L]$, a purported MAC $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ is accepted if and only if

$$e(\sigma_1, \hat{g}) \cdot e(\sigma_2, \hat{g}^\theta) = e(\sigma_4, \hat{V})^\eta \cdot e(\sigma_3, \prod_{k=1}^L \hat{Z}_{k,m[k]}). \qquad (12)$$

We note that the verification algorithm can be modified in such a way that it does not require any pairing evaluation. The above description is just meant to simplify the presentation of the security proof of our LAF construction.

The proof is essentially identical to that of [7] but we give it for completeness. We note that, in the security definitions of MACs, the adversary is generally allowed to make verification queries. Here, for simplicity, we prove unforgeability in a game where the adversary knows $\eta = \log_g(h)$, which allows it to run the verification oracle itself. This dispenses us with the need for a verification oracle.

Lemma 4. *The above construction is an unforgeable MAC assuming that the SXDH assumption holds in* $(\mathbb{G}, \hat{\mathbb{G}})$*. Namely, any forger* $\mathcal{A}$ *making* Q *MAC queries and* Q_V *verification queries within running time* $t_\mathcal{A}$ *has advantage at most*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{uf-mac}}(\lambda) \leq \mathbf{Adv}_{\mathcal{B}_1}^{\text{DDH}_2}(\lambda) + 2L \cdot \mathbf{Adv}_{\mathcal{B}_2}^{\text{DDH}_1}(\lambda),$$

where $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *are PPT distinguishers against the DDH assumption in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively, which run in time* $t_\mathcal{A} + (Q + Q_V) \cdot \mathsf{poly}(\lambda)$.

Proof. To prove the result, we consider a sequence of games. For each index i, we call W_i the event that the challenger outputs 1 in Game_i.

Game_0: This is the real game MAC security game, where the adversary $\mathcal{A}$ is additionally given $\eta = \log_g(h)$ in such a way that it can run the verification algorithm (and test whether equation (12) holds) by itself. The challenger outputs 1 if and only if $\mathcal{A}$ eventually outputs a pair $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ satisfying

$$e(\sigma_1^\star, \hat{g}) \cdot e(\sigma_2^\star, \hat{g}^\theta) = e(\sigma_4^\star, \hat{V})^\eta \cdot e(\sigma_3^\star, \prod_{k=1}^{L} \hat{Z}_{k,m^\star[k]}), \tag{13}$$

where $M^\star = m^\star[1] \dots m^\star[L] \in \{0,1\}^L$, although $M^\star$ was not previously queried to the MAC oracle. By definition, $\Pr[W_0] = \mathbf{Adv}_{\mathcal{A}}^{\text{uf-mac}}(\lambda)$.

Game_1: In this game, we modify again the verification oracle as follows. When $\mathcal{A}$ outputs a pair $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ such that $M^\star$ was not queried to the MAC oracle but $(M^\star, \sigma^\star)$ still satisfies (13), the challenger checks if

$$\begin{aligned} \sigma_1^\star &= {\sigma_4^\star}^{\eta \cdot \alpha} \cdot {\sigma_3^\star}^{\sum_{k=1}^{L} x_{k,m^\star[k]}} \\ \sigma_2^\star &= {\sigma_4^\star}^{\eta \cdot \beta} \cdot {\sigma_3^\star}^{\sum_{k=1}^{L} y_{k,m^\star[k]}}. \end{aligned} \tag{14}$$

We call E_1 the event that equalities (14) are satisfied. If they are not satisfied, the challenger outputs 0. Otherwise, it outputs 1 as it did in Game_0. If we denote by E_0 the analogue of event E_1 in Game_0, we have

$$\begin{aligned} \Pr[W_0] &= \Pr[W_0 \wedge E_0] + \Pr[W_0 \wedge \neg E_0] \\ &= \Pr[W_1 \wedge E_1] + \Pr[W_0 \wedge \neg E_0] = \Pr[W_1] + \Pr[W_0 \wedge \neg E_0] \end{aligned}$$

since $\Pr[W_1 \wedge \neg E_1] = 0$. Lemma 5 shows that event $W_0 \wedge \neg E_0$ would contradict the DDH assumption in $\hat{\mathbb{G}}$: namely, we have $\Pr[W_0 \wedge \neg E_0] \leq \mathbf{Adv}^{\text{DDH}_2}(\lambda)$, which implies $|\Pr[W_1] - \Pr[W_0]| \leq \mathbf{Adv}^{\text{DDH}_2}(\lambda)$.

We now use a sub-sequence of L hybrid games over the input bits of queried messages. For convenience, we define $\mathsf{Game}_{2.0}$ to be identical to Game_1.

$\mathsf{Game}_{2.i}$ $(1 \leq i \leq L)$**:** In this sub-sequence of games, we modify the key generation phase and the MAC oracle in the following way.

- At the beginning of the game, the challenge defines $\hat{V} = \hat{g}^v$ for a random $v \stackrel{R}{\leftarrow} \mathbb{Z}_p$.
- MAC queries are handled as follows. Let $R : \{0,1\}^i \rightarrow \mathbb{Z}_p$ be a truly random function mapping i-bit input to $\mathbb{Z}_p$. At each message M queried by $\mathcal{A}$, the challenger computes $(\sigma_3, \sigma_4) = (g^\rho, g^r)$ for random $\rho, r \stackrel{R}{\leftarrow} \mathbb{Z}_p$. Then, it outputs $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$, where

$$\sigma_1 = h^{(v-\theta\cdot R(m[1]\dots m[i]))\cdot r} \cdot g^{\rho\cdot(\sum_{k=1}^{L} x_{k,m[k]})}$$
$$\sigma_2 = h^{R(m[1]\dots m[i])\cdot r} \cdot g^{\rho\cdot(\sum_{k=1}^{L} y_{k,m[k]})}$$

When the adversary outputs $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ satisfying (13) for a new message $M^\star$, the challenger checks if the following equalities are satisfied:

$$\begin{aligned} \sigma_1^\star &= \sigma_4^{\star\,\eta\cdot(v-\theta\cdot R(m^\star[1]\dots m^\star[i]))} \cdot \sigma_3^{\star\,\sum_{k=1}^{L} x_{k,m^\star[k]}} \\ \sigma_2^\star &= \sigma_4^{\star\,\eta\cdot R(m^\star[1]\dots m^\star[i])} \cdot \sigma_3^{\star\,\sum_{k=1}^{L} y_{k,m^\star[k]}}. \end{aligned} \tag{15}$$

If so, the challenger outputs 1. Otherwise, it outputs 0. Lemma 6 shows that $\mathsf{Game}_{2.i}$ is indistinguishable from $\mathsf{Game}_{2.(i-1)}$ under the DDH assumption in $\mathbb{G}$. Namely, $|\Pr[W_{2.i}] - \Pr[W_{2.(i-1)}]| \leq \mathbf{Adv}^{\mathrm{DDH}_1}(\lambda)$.

In $\mathsf{Game}_{2.L}$, we claim that $\Pr[W_{2.L}] = 1/p$. Indeed, the equalities (15) can only hold by pure chance when $i = L$ because $m^\star[1]\dots m^\star[L]$ was never involved in an output of the MAC oracle. Hence, the random function output $R(m^\star[1]\dots m^\star[L])$ is perfectly independent of $\mathcal{A}$'s view. Since $\Pr[W_{2.0}] = \Pr[W_1]$, we obtain the claimed upper bound for $\Pr[W_0]$. □

Lemma 5. *In* Game_0*, we have* $\Pr[W_0 \wedge \neg E_0] \leq \mathbf{Adv}^{\mathrm{DDH}_2}(\lambda)$.

Proof. Towards a contradiction, let us assume that, in Game_1, the adversary $\mathcal{A}$ can output a pair $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ satisfying (13) but not (14). We construct a distinguisher $\mathcal{B}$ for the DDH assumption in $\hat{\mathbb{G}}$. Our distinguisher $\mathcal{B}$ takes as input $(\hat{g}, \hat{g}^\theta, \hat{g}^\omega, \hat{T}) \in \hat{\mathbb{G}}^4$ and decides if $\hat{T} = \hat{g}^{\alpha\cdot\omega}$ or $\hat{T} \in_R \hat{\mathbb{G}}$. To this end, $\mathcal{B}$ will compute a pair of the form $(w, w^\theta) \in \mathbb{G}^2$ with $w \neq 1_{\mathbb{G}}$, which allows solving the given DDH instance in $\hat{\mathbb{G}}$ by testing if $e(w, \hat{T}) = e(w^\theta, \hat{g}^\omega)$. Indeed, the latter equality holds if and only if $\hat{T} = \hat{g}^{\alpha\cdot\omega}$.

The reduction $\mathcal{B}$ runs the real key generation algorithm and answers all MAC and verification queries exactly as in Game_1. By hypothesis, $\mathcal{B}$ has non-negligible probability of outputting a pair $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ satisfying (13) although

$$\sigma_1^\star \neq \sigma_4^{\star\,\eta\cdot\alpha} \cdot \sigma_3^{\star\,\sum_{k=1}^{L} x_{k,m^\star[k]}}, \qquad \sigma_2^\star \neq \sigma_4^{\star\,\eta\cdot\beta} \cdot \sigma_3^{\star\,\sum_{k=1}^{L} y_{k,m^\star[k]}}.$$

At this point, $\mathcal{B}$ uses sk_{mac} to construct a different valid MAC $(\sigma_1', \sigma_2', \sigma_3^\star, \sigma_4^\star)$ satisfying (13) and such that $(\sigma_1', \sigma_2') \neq (\sigma_1^\star, \sigma_2^\star)$. Namely, $\mathcal{B}$ computes

$$\sigma_1' = {\sigma_4^\star}^{\eta \cdot \alpha} \cdot {\sigma_3^\star}^{\sum_{k=1}^{L} x_{k,m^\star[k]}}, \qquad \sigma_2' = {\sigma_4^\star}^{\eta \cdot \beta} \cdot {\sigma_3^\star}^{\sum_{k=1}^{L} y_{k,m^\star[k]}}.$$

By dividing the two verification equations for $(\sigma_1', \sigma_2', \sigma_3^\star, \sigma_4^\star)$ and $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star)$, we get

$$e(\sigma_1^\star/\sigma_1', \hat{g}) \cdot e(\sigma_2^\star/\sigma_2', \hat{g}^\theta) = 1_{\mathbb{G}_T}$$

meaning that $\sigma_1^\star/\sigma_1' = (\sigma_2'/\sigma_2^\star)^\theta$. Since $\sigma_1^\star \neq \sigma_1'$, this provides $\mathcal{B}$ with a non-trivial pair $(w, w^\theta) = \big(\sigma_2'/\sigma_2^\star, \sigma_1^\star/\sigma_1'\big)$, which is sufficient to solve DDH in $\hat{\mathbb{G}}$. □

Lemma 6. *Under the DDH assumption in* $\mathbb{G}$, *the challenger outputs* 1 *with about the same probabilities in* $\mathsf{Game}_{3.(i-1)}$ *and* $\mathsf{Game}_{3.i}$. *We have*

$$|\Pr[W_{2.i}] - \Pr[W_{2.(i-1)}]| \leq 2 \cdot \mathbf{Adv}^{\mathrm{DDH}_1}(\lambda).$$

(The proof is given in the full version of the paper.)

4.2 The LAF Construction

In order to apply a hybrid argument in our proof of indistinguishability, we need to use n instances of the MAC of Section 4.1, each of which has its own secret key $\mathsf{sk}_{mac,j}$ and its own set of public parameters $\mathsf{pp}_j = \big(g, \hat{g}, h, \hat{g}^{\theta_j}, (\hat{V}_j, \hat{\boldsymbol{Z}}_{j,0}, \hat{\boldsymbol{Z}}_{j,1})\big)$. As a result, we need an evaluation key containing $\Theta(n \cdot L)$ group elements. We leave it as an open problem to shorter the evaluation while retaining tight security and short tags.

Key generation. $\mathsf{LAF.Gen}(1^\lambda)$ conducts the following steps.

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with generators $g, h \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$ and let $\eta = \log_g(h)$.
2. Choose a chameleon hash function $\mathsf{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$, where the hashing algorithm $\mathsf{CMhash} : \{0,1\}^* \times \mathcal{R}_{hash} \to \{0,1\}^L$ has output length $L \in \mathsf{poly}(\lambda)$. Generate a pair $(hk_{\mathsf{CMH}}, td_{\mathsf{CMH}}) \leftarrow \mathsf{CMKg}(1^\lambda)$ made of a hashing key hk_{CMH} and a trapdoor td_{CMH}.
3. Generate n keys for the MAC of Section 4.1 which all share the same parameters $g, h \in \mathbb{G}$, $\hat{g} \in \hat{\mathbb{G}}$. Namely, for each $j \in [n]$, conduct the following steps.
    a. Choose $\theta_j \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{g}^{\theta_j} \in \hat{\mathbb{G}}$.
    b. For each $\mu \in \{0,1\}$, choose vectors $\boldsymbol{x}_{j,\mu} = (x_{j,1,\mu}, \ldots, x_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$ and $\boldsymbol{y}_{j,\mu} = (y_{j,1,\mu}, \ldots, y_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$.
    c. Compute $\boldsymbol{z}_{j,\mu} = \boldsymbol{x}_{j,\mu} + \theta_j \cdot \boldsymbol{y}_{j,\mu}$ and $\hat{\boldsymbol{Z}}_{j,\mu} = \hat{g}^{\boldsymbol{z}_{j,\mu}} = (\hat{g}^{z_{j,1,\mu}}, \ldots, g^{z_{j,L,\mu}})$ for each $\mu \in \{0,1\}$.
    d. Choose $\alpha_j, \beta_j \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{V}_j = \hat{g}^{\alpha_j + \theta_j \cdot \beta_j}$.

   e. Define $\mathsf{sk}_{mac,j} = (\alpha_j, \beta_j, \boldsymbol{x}_{j,0}, \boldsymbol{x}_{j,1}, \boldsymbol{y}_{j,0}, \boldsymbol{y}_{j,1})$.

4. Choose $u \xleftarrow{R} \mathbb{G}$ and $\hat{h}, \hat{u} \xleftarrow{R} \hat{\mathbb{G}}$ subject to the constraints $\log_g(h) = \log_{\hat{g}}(\hat{h})$ and $\log_g(u) = \log_{\hat{g}}(\hat{u})$.
5. Define

$$\hat{H}_j = (\hat{h}^j \cdot \hat{u})^{\alpha_j + \theta_j \cdot \beta_j} \qquad \forall j \in [n].$$

6. Output the evaluation key ek and the lossy tag generation key tk, which consist of

$$ek := \Big(g,\ h,\ u,\ \hat{g},\ \hat{h},\ \hat{u},\ \{\hat{g}^{\theta_j}\}_{j=1}^n, \{\hat{\boldsymbol{Z}}_{j,\mu}\}_{j\in[n],\mu\in\{0,1\}},\ \{\hat{V}_j, \hat{H}_j\}_{j=1}^n, hk_{\mathsf{CMH}}\Big),$$
$$tk := (\{\mathsf{sk}_{mac,j}\}_{j=1}^n, \eta, td_{\mathsf{CMH}}).$$

The tag space $\mathcal{T} = \mathcal{T}_{\mathsf{c}} \times \mathcal{T}_{aux}$ is defined as a product of $\mathcal{T}_{\mathsf{a}} = \{0,1\}^*$ and

$$\mathcal{T}_{\mathsf{c}} := \{(\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash}) \mid r_{hash} \in \mathcal{R}_{hash} \ \wedge \\ \forall i \in [n] : (R_i, S_i, D_i, E_i, F_i) \in \mathbb{G}^5 \ \wedge\ e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})\}.$$

The range of the function family is $\mathsf{Rng}_\lambda = \mathbb{G}_T^{n+1}$ and its domain is $\mathbb{Z}_p^n$.

Lossy tag generation. $\mathsf{LAF.LTag}(tk, t_{\mathsf{a}})$ takes in an auxiliary tag component $t_{\mathsf{a}} \in \{0,1\}^*$ and uses $tk = (\{\mathsf{sk}_{mac,j}\}_{j=1}^n, \eta)$ to generate a lossy tag as follows.

1. For each $i \in [n]$, choose $r_i \xleftarrow{R} \mathbb{Z}_p$ and compute

$$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n]. \tag{16}$$

2. Choose a random string $\tau \in \{0,1\}^L$ in the range of CMhash. Then, for each $i \in [n]$, choose $\rho_i \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\begin{aligned} D_i &= h^{\alpha_i \cdot r_i} \cdot g^{\rho_i \cdot (\sum_{k=1}^L x_{i,k,\tau[k]})}, \\ E_i &= h^{\beta_i \cdot r_i} \cdot g^{\rho_i \cdot (\sum_{k=1}^L y_{i,k,\tau[k]})}, \qquad \forall i \in [n] \\ F_i &= g^{\rho_i}. \end{aligned} \tag{17}$$

3. Use the trapdoor td_{CMH} of the chameleon hash function to find random coins $r_{hash} \in \mathcal{R}_{hash}$ such that

$$\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash}) \in \{0,1\}^L.$$

4. Output the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, where $t_{\mathsf{c}} = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash})$.

Each lossy tag corresponds to a matrix $\big(M_{i,j}\big)_{i,j\in[n]} = \big(e(g,\hat{h})^{r_i \cdot (\alpha_j + \theta_j \cdot \beta_j)}\big)_{i,j}$, which forms a rank-1 matrix in the exponent. Its diagonal entries consist of

$$M_{i,i} = \frac{e(D_i, \hat{g}) \cdot e(E_i, \hat{g}^{\theta_i})}{e(F_i, \prod_{k=1}^L \hat{Z}_{i,k,\tau[k]})} = e(g, \hat{h})^{r_i \cdot (\alpha_i + \theta_i \cdot \beta_i)} \qquad \forall i \in [n], \tag{18}$$

while its non-diagonal entries

$$M_{i,j} = \Big(\frac{e(R_i, \hat{H}_j)}{e(S_i, \hat{V}_j)}\Big)^{1/(j-i)} \tag{19}$$
$$= e(g, \hat{h})^{r_i \cdot (\alpha_j + \theta_j \cdot \beta_j)} \qquad \forall (i,j) \in [n] \times [n] \setminus \{(i,i)\}_{i=1}^{n},$$

are obtained by pairing tag component (R_i, S_i) with evaluation key components $(\hat{V}_j, \hat{H}_j)$.

Random Tags. A random tag can be publicly sampled as follows.

1. For each $i \in [n]$, choose $r_i \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute $\{R_i, S_i\}_{i=1}^{n}$ as in (16).
2. For each $i \in [n]$, choose $(D_i, E_i, F_i) \overset{R}{\leftarrow} \mathbb{G}^3$ uniformly at random.
3. Choose $r_{hash} \overset{R}{\leftarrow} \mathcal{R}_{hash}$.

Output the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, where $t_{\mathsf{c}} = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^{n}, r_{hash})$.

We note that, in both random and lossy tags, we have $e(R_i, \hat{u}^i \cdot \hat{h}) = e(S_i, \hat{g})$ for all $i \in [n]$, so that elements of $\mathcal{T}$ are publicly recognizable.

Evaluation. $\mathsf{LAF.Eval}(ek, t, \mathbf{x})$ takes in the input $\mathbf{x} \in \mathbb{Z}_p^n$ and the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$. It parses t_{c} as $(\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^{n}, r_{hash})$ and does the following.

1. Return $\perp$ if there exists $i \in [n]$ such that $e(R_i, \hat{h}^i \cdot \hat{u}) \neq e(S_i, \hat{g})$.
2. Compute the matrix $\big(M_{i,j}\big)_{i,j \in [n]} \in \mathbb{G}_T^{n \times n}$ as

$$M_{i,i} = \frac{e(D_i, \hat{g}) \cdot e(E_i, \hat{g}^{\theta_i})}{e(F_i, \prod_{k=1}^{L} \hat{Z}_{i,k,\tau[k]})} \qquad \forall i \in [n] \quad , \tag{20}$$

where $\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^{n}), r_{hash}) \in \{0,1\}^L$, and

$$M_{i,j} = \Big(\frac{e(R_i, \hat{H}_j)}{e(S_i, \hat{V}_j)}\Big)^{1/(j-i)} \qquad \forall (i,j) \in [n] \times [n] \setminus \{(i,i)\}_{i=1}^{n}, \tag{21}$$

Since $R_i = g^{r_i}$ and $S_i = (h^i \cdot u)^{r_i}$ for some $r_i \in \mathbb{Z}_q$, we have

$$M_{i,i} = e(g, \hat{h})^{r_i \cdot (\alpha_i + \theta_i \cdot \beta_i) + \omega_i}, \qquad \forall i \in [n] \tag{22}$$
$$M_{i,j} = e(g, \hat{h})^{r_i \cdot (\alpha_j + \theta_j \cdot \beta_j)}, \qquad \forall i \neq j,$$

for some vector $(\omega_1, \ldots, \omega_n)^\top \in \mathbb{Z}_p^n$ that only contains non-zero entries if $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ is injective.

3. Compute the vector $\big(V_{T,j}\big)_{j \in [n]}$ as $V_{T,j} = e(h, \hat{V}_j) = e(g, \hat{h})^{\alpha_j + \theta_j \cdot \beta_j}$ for each $j \in [n]$.

4. Use the input $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{Z}_p^n$ to compute

$$Y_0 = \prod_{j=1}^{n} V_{T,j}^{x_j} \tag{23}$$
$$Y_i = \prod_{j=1}^{n} M_{i,j}^{x_j} \qquad \forall i \in [n]$$

and output $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_n)^\top \in \mathbb{G}_T^{n+1}$.

The lossiness/injectivity properties can be analyzed exactly in the same way as in the construction of Section 3. Indeed, by defining $v_j = \alpha_j + \theta_j \cdot \beta_j$ for each $j \in [n]$, we find that $\{\hat{V}\}_{j=1}^n$ and $(M_{ij})_{i,j\in[n]}$ are distributed as in Section 3.

4.3 Security

Theorem 3. *The above LAF provides indistinguishability assuming that the wD3DH1 assumption holds in* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ *and that the DDH assumptions holds in* $\mathbb{G}$. *The advantage of any PPT distinguisher* $\mathcal{A}$ *making* Q *queries within time* $t_{\mathcal{A}}$ *is bounded by*

$$\mathbf{Adv}^{\mathrm{indist}}(\lambda) \leq n \cdot (\mathbf{Adv}_{\mathcal{B}_1}^{\mathrm{wD3DH1}}(\lambda) + \mathbf{Adv}_{\mathcal{B}_2}^{\mathrm{DDH}_1}(\lambda))$$

for PPT algorithm $\mathcal{B}_1$, $\mathcal{B}_2$ *running in time* $t_{\mathcal{A}} + Q \cdot \mathsf{poly}(\lambda)$.

Proof. We define a sequence of hybrid games. In Game_0, the adversary has access to the real oracle $\mathsf{LAF.LTag}(tk, .)$ oracle that always outputs lossy tags. In Game_n, the adversary is given access to an oracle $\mathcal{O}_{\mathcal{T}_\mathsf{c}}(.)$ that always outputs random tags in the tag space $\mathcal{T}$.

Game_ξ' $(1 \leq \xi \leq n)$: The adversary interacts with an oracle $\mathsf{LAF.LTag}^{(\ell,k)}(tk, .)$ that outputs tags $t = (t_\mathsf{c}, t_\mathsf{a})$ with the following hybrid distribution. In the core component $t_\mathsf{c} = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash})$, the first $\xi - 1$ tuples $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=1}^{\xi}$ of t_c are random group elements satisfying the equality $e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})$. The last $n - \xi$ tuples $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=\xi+1}^n$ are generated exactly as in lossy tags. The ξ-th tuple $(R_\xi, S_\xi, D_\xi, E_\xi, F_\xi)$ has a special distribution where $e(R_\xi, \hat{h}^\xi \cdot \hat{u}) = e(S_\xi, \hat{g})$, D_ξ is completely random in $\mathbb{G}$ and

$$E_\xi = h^{\beta_\xi \cdot \log_g(R_\xi)} \cdot g^{\rho_\xi \cdot \sum_{k=1}^{L} y_{\xi,k,\tau[k]}},$$
$$F_\xi = g^{\rho_\xi}$$

Game_ξ $(1 \leq \xi \leq n)$: The adversary interacts with an oracle $\mathsf{LAF.LTag}^{(\ell,k)}(tk, .)$ that outputs $t = (t_\mathsf{c}, t_\mathsf{a})$ such that the first ξ tuples $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=1}^{\xi}$ of t_c are random subject to the constraint $e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})$ while $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=\xi+1}^n$ are generated as in lossy tags.

For each index $\xi \in [n]$, Lemma 7 shows that Game'_ξ is computationally indistinguishable from $\mathsf{Game}_{\xi-1}$ if the R-wD3DH1 assumption holds. In a second step, Lemma 8 shows that Game'_ξ is indistinguishable from Game_ξ under the DDH assumption in $\mathbb{G}$. By applying Lemma 1, we obtain that the scheme provides indistinguishability under tight reductions from the hardness of wD3DH1 and that of the DDH problem in $\mathbb{G}$. □

Lemma 7. *Game'_ξ is computationally indistinguishable from $\mathsf{Game}_{\xi-1}$ under the R-wD3DH1 assumption. The advantage of any PPT distinguisher between the two games can be bounded by $\mathbf{Adv}^{\xi'\text{-}(\xi-1)}(\lambda) \leq \mathbf{Adv}^{\text{R-wD3DH1}}(\lambda)$.*

Proof. Let us assume that there exists $\xi \in [n]$ such that the adversary $\mathcal{A}$ can distinguish Game'_ξ from $\mathsf{Game}_{\xi-1}$ with non-negligible advantage. We build a R-wD3DH1 distinguisher $\mathcal{B}$ that takes as input $\{(g, \hat{g}, g^{a_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, T_i)\}_{i=1}^Q$ with the goal of deciding if $T_i = g^{a_i bc}$ for each $i \in [Q]$ or if $\{T_i\}_{i=1}^Q$ are all independent and uniformly distributed over $\mathbb{G}$.

To this end, $\mathcal{B}$ defines $h = g^b$, $\hat{h} = \hat{g}^b$. It also picks $\theta'_\xi, \beta'_\xi \xleftarrow{R} \mathbb{Z}_p$ uniformly and sets

$$\hat{g}^{\theta_\xi} = (\hat{g}^b)^{\theta'_\xi}, \qquad \hat{V}_\xi = (\hat{g})^c \cdot \hat{g}^{\theta'_\xi \cdot \beta'_\xi},$$

which implicitly defines

$$\alpha_\xi = c, \qquad \beta_\xi = \beta'_\xi / b, \qquad \theta_\xi = b \cdot \theta'_\xi.$$

It chooses $\nu \xleftarrow{R} \mathbb{Z}_p$ and defines $\hat{u} = \hat{h}^{-\xi} \cdot \hat{g}^\nu$ as well as $u = h^{-\xi} \cdot g^\nu$. This allows defining

$$\hat{H}_\xi = (\hat{h}^\xi \cdot \hat{u})^{c + \theta'_\xi \cdot \beta'_\xi} = (\hat{V}_\xi)^\nu,$$

For all indexes $j \in [n] \setminus \{\xi\}$, it chooses $\alpha_j, \beta_j, \theta_j \xleftarrow{R} \mathbb{Z}_p$ and faithfully computes $\hat{V}_j = \hat{g}^{\alpha_j + \theta_j \cdot \beta_j}$ and

$$\hat{H}_j = (\hat{h}^j \cdot \hat{u})^{\alpha_j + \theta_j \cdot \beta_j}.$$

Then, it constructs the MAC secret keys $\{\boldsymbol{x}_{j,\mu}, \boldsymbol{y}_{j,\mu}\}_{j=1}^n$ for randomly chosen vectors $\boldsymbol{x}_{j,\mu} = (x_{j,1,\mu}, \ldots, x_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$, $\boldsymbol{y}_{j,\mu} = (y_{j,1,\mu}, \ldots, y_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$. For each $j \in [n]$, it defines

$$\begin{aligned} \hat{\boldsymbol{Y}}_{j,\mu} &= (\hat{Y}_{j,1,\mu}, \ldots, \hat{Y}_{j,L,\mu}) = \hat{g}^{\boldsymbol{y}_{j,\mu}}, & \boldsymbol{Y}_{j,\mu} &= (Y_{j,1,\mu}, \ldots, Y_{j,L,\mu}) = g^{\boldsymbol{y}_{j,\mu}} \\ \hat{\boldsymbol{X}}_{j,\mu} &= (\hat{X}_{j,1,\mu}, \ldots, \hat{X}_{j,L,\mu}) = \hat{g}^{\boldsymbol{x}_{j,\mu}}, & \boldsymbol{X}_{j,\mu} &= (X_{j,1,\mu}, \ldots, X_{j,L,\mu}) = g^{\boldsymbol{x}_{j,\mu}}. \end{aligned}$$

Then, it computes

$$\begin{aligned} \hat{\boldsymbol{Z}}_{j,\mu} &= \hat{\boldsymbol{X}}_{j,\mu} \cdot \hat{\boldsymbol{Y}}_{j,\mu}^{\theta_j} & \forall j \in [n] \setminus \{\xi\} \\ \hat{\boldsymbol{Z}}_{\xi,\mu} &= \hat{\boldsymbol{X}}_{\xi,\mu} \cdot (\hat{g}^b)^{\boldsymbol{y}_{\xi,\mu} \cdot \theta'_\xi} & \end{aligned}$$

At the t-th invocation of the $\mathsf{LAF.LTag}(tk,.)$ oracle, $\mathcal{B}$ sets

$$R_\xi = g^{a_t}, \qquad\qquad S_\xi = (g^{a_t})^{\nu} = (h^{\xi} \cdot u)^{a_t},$$

where g^{a_t} is fetched from the t-th input tuple $(g, \hat{g}, g^{a_t}, g^b, g^c, \hat{g}^b, \hat{g}^c, T_t)$. For all indexes $i \neq \xi$, it chooses $r_1, \ldots, r_{\xi-1}, r_{\xi+1}, \ldots, r_n \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sets

$$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n] \setminus \{\xi\}.$$

It generates the triples $\{D_i, E_i, F_i\}_{i=1}^n$ by choosing $(D_i, E_i, F_i) \stackrel{R}{\leftarrow} \mathbb{G}^3$ at random for each $i \in [\xi - 1]$. The ξ-th triple (D_k, E_k, F_k) is defined as

$$\begin{aligned} D_\xi &= T_t \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]} \big)^{\rho_\xi}, \\ E_\xi &= (g^{a_t})^{\beta'_\xi} \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]} \big)^{\rho_\xi}, \\ F_\xi &= g^{\rho_\xi}. \end{aligned}$$

for a randomly chosen $\rho_\xi \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $\tau \stackrel{R}{\leftarrow} \{0,1\}^L$. As for $\{D_i, E_i, F_i\}_{i=\xi+1}^n$, they are obtained by choosing choosing $\rho_i, r_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ before setting

$$D_i = (g^b)^{\alpha_i \cdot r_i} \big(\prod_{k=1}^{L} X_{\xi,k,\tau[k]} \big)^{\rho_i}, \qquad E_i = (g^b)^{\beta_i \cdot r_i} \big(\prod_{k=1}^{L} Y_{\xi,k,\tau[k]} \big)^{\rho_i}, \qquad F_i = g^{\rho_i}.$$

Then, it uses the trapdoor td_{CMH} of the chameleon hash function to find coins $r_{hash} \in \mathcal{R}_{hash}$ such that $\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash})$.

It is easy to see that, if $T_t = g^{a_t bc}$, the triple (D_ξ, E_ξ, F_ξ) can be written

$$\begin{aligned} D_\xi &= h^{\alpha_\xi \cdot r_\xi} \cdot \big(\prod_{k=1}^{L} \hat{X}_{\xi,k,\tau[k]} \big)^{\rho_\xi}, \\ E_\xi &= h^{\beta_\xi \cdot r_\xi} \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]} \big)^{\rho_\xi} \\ F_\xi &= g^{\rho_\xi}, \end{aligned}$$

meaning that $\mathcal{A}$'s view is the same as in $\mathsf{Game}_{\xi-1}$. In contrast, if $T_t \in_R \mathbb{G}$, it can be written $T_t = g^{a_t bc + z_t}$ for some uniformly random $z_t \in_R \mathbb{Z}_p$. In this case, (D_ξ, E_ξ, F_ξ) can be written

$$\begin{aligned} D_\xi &= h^{z_t + \alpha_\xi \cdot r_\xi} \cdot \big(\prod_{k=1}^{L} \hat{X}_{\xi,k,\tau[k]} \big)^{\rho_\xi}, \\ E_\xi &= h^{\beta_\xi \cdot r_\xi} \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]} \big)^{\rho_\xi}, \\ F_\xi &= g^{\rho_\xi}, \end{aligned}$$

for some random $z_t \in_R \mathbb{Z}_p$ that does not appear anywhere else. In this case, $\mathcal{A}$'s view corresponds to Game'_ξ. □

Lemma 8. Game_ξ *is computationally indistinguishable from* Game'_ξ *under the DDH assumption in* $\mathbb{G}$*. The advantage of any PPT distinguisher between the two games can be bounded by* $\mathbf{Adv}^{\xi\text{-}\xi'}(\lambda) \leq \mathbf{Adv}^{\mathrm{DDH}_1}(\lambda)$.

Proof. We assume that there exists $\xi \in [n]$ such that $\mathcal{A}$ can tell apart Game'_ξ from Game_ξ with noticeable advantage. We build a distinguisher $\mathcal{B}$ that takes as input Q tuples $\{(g, g^{a_i}, g^{a_i \cdot b}, g^b, T_i)\}_{i=1}^Q$ in $\mathbb{G}^5$ with the goal of deciding if $T_i = g^{a_i b}$ for each $i \in [Q]$ or if $\{T_i\}_{i=1}^Q$ are independent and uniformly distributed over $\mathbb{G}$. This assumption is known (see, e.g., [39, Lemma 4.4]) to have a tight reduction from the DDH assumption.

To this end, $\mathcal{B}$ defines $h = g^\eta$, $\hat{h} = \hat{g}^\eta$ for a random $\eta \stackrel{R}{\leftarrow} \mathbb{Z}_p$. It also computes $\hat{g}^{\theta_\xi}$ for a randomly chosen $\theta_\xi \stackrel{R}{\leftarrow} \mathbb{Z}_p$. Then, it picks $v_\xi \stackrel{R}{\leftarrow} \mathbb{Z}_p$ uniformly and sets

$$\hat{V}_\xi = \hat{g}^{v_\xi}.$$

Implicitly, $\mathcal{B}$ will define

$$\beta_\xi = b, \qquad \alpha_\xi = v_\xi - b \cdot \theta_\xi$$

although it does not know (α_ξ, β_ξ). It chooses $\hat{u} \in \hat{\mathbb{G}}$ and $u \in \mathbb{G}$ by setting $u = g^\nu$ and $\hat{u} = \hat{g}^\nu$ for a random $\nu \stackrel{R}{\leftarrow} \mathbb{Z}_p$. Then, $\mathcal{B}$ defines

$$\hat{H}_\xi = (\hat{h}^\xi \cdot \hat{u})^{v_\xi}.$$

For all indexes $j \in [n] \setminus \{\xi\}$, it chooses $\alpha_j, \beta_j, \theta_j \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and faithfully computes $\hat{V}_j = \hat{g}^{\alpha_j + \theta_j \cdot \beta_j}$ and

$$\hat{H}_j = (\hat{h}^j \cdot \hat{u})^{\alpha_j + \theta_j \cdot \beta_j}.$$

Then, it constructs the MAC secret keys $\{\boldsymbol{x}_{j,\mu}, \boldsymbol{y}_{j,\mu}\}_{j=1}^n$ by for randomly chosen vectors $\boldsymbol{x}_{j,\mu} = (x_{j,1,\mu}, \ldots, x_{j,L,\mu}) \stackrel{R}{\leftarrow} \mathbb{Z}_p^L$, $\boldsymbol{y}_{j,\mu} = (y_{j,1,\mu}, \ldots, y_{j,L,\mu}) \stackrel{R}{\leftarrow} \mathbb{Z}_p^L$. For each $j \in [n]$, it defines

$$\begin{aligned} \hat{\boldsymbol{Y}}_{j,\mu} &= (\hat{Y}_{j,1,\mu}, \ldots, \hat{Y}_{j,L,\mu}) = \hat{g}^{\boldsymbol{y}_{j,\mu}}, & \boldsymbol{Y}_{j,\mu} &= (Y_{j,1,\mu}, \ldots, Y_{j,L,\mu}) = g^{\boldsymbol{y}_{j,\mu}} \\ \hat{\boldsymbol{X}}_{j,\mu} &= (\hat{X}_{j,1,\mu}, \ldots, \hat{X}_{j,L,\mu}) = \hat{g}^{\boldsymbol{x}_{j,\mu}}, & \boldsymbol{X}_{j,\mu} &= (X_{j,1,\mu}, \ldots, X_{j,L,\mu}) = g^{\boldsymbol{x}_{j,\mu}}. \end{aligned}$$

Then, it computes

$$\hat{\boldsymbol{Z}}_{j,\mu} = \hat{\boldsymbol{X}}_{j,\mu} \cdot \hat{\boldsymbol{Y}}_{j,\mu}^{\theta_j} \qquad \forall j \in [n].$$

For each $t \in [Q]$, the t-th invocation of the $\mathsf{LAF.LTag}(tk, .)$ oracle is handled by setting

$$R_\xi = g^{a_t}, \qquad\qquad S_\xi = (g^{a_t})^{\eta \cdot \xi + \nu} = (h^\xi \cdot u)^{a_t},$$

where g^{a_t} is fetched from the t-th input tuple $(g, g^{a_t}, g^{a_t \cdot b}, g^b, T_t)$. For all indexes $i \neq \xi$, it chooses $r_1, \ldots, r_{\xi-1}, r_{\xi+1}, \ldots, r_n \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sets

$$R_i = g^{r_i}, \qquad\qquad S_i = (h^i \cdot u)^{r_i} \qquad\qquad \forall i \in [n] \setminus \{\xi\}.$$

It generates the triples $\{D_i, E_i, F_i\}_{i=1}^n$ by choosing $(D_i, E_i, F_i) \stackrel{R}{\leftarrow} \mathbb{G}^3$ at random for each $i \in [\xi - 1]$. The ξ-th triple (D_k, E_k, F_k) is defined by sampling $D_\xi \stackrel{R}{\leftarrow} \mathbb{G}$ uniformly and setting

$$\begin{aligned} E_\xi &= T_t^\eta \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]} \big)^{\rho_\xi}, \\ F_\xi &= g^{\rho_\xi}. \end{aligned}$$

for randomly chosen $\rho_\xi \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $\tau \stackrel{R}{\leftarrow} \{0,1\}^L$. As for $\{D_i, E_i, F_i\}_{i=\xi+1}^n$, they are obtained by choosing choosing $\rho_i, r_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ before setting

$$D_i = h^{\alpha_i \cdot r_i} \big(\prod_{k=1}^{L} X_{\xi,k,\tau[k]} \big)^{\rho_i}, \qquad E_i = h^{\beta_i \cdot r_i} \big(\prod_{k=1}^{L} Y_{\xi,k,\tau[k]} \big)^{\rho_i}, \qquad F_i = g^{\rho_i}.$$

Then, it uses the trapdoor td_{CMH} of the chameleon hash function to obtain coins $r_{hash} \in \mathcal{R}_{hash}$ such that $\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash})$.

We observe that, if $T_t = g^{a_t \cdot b}$ for each $t \in [Q]$, the triples (D_ξ, E_ξ, F_ξ) are distributed as $D_\xi \in_R \mathbb{G}$ and

$$\begin{aligned} E_\xi &= h^{\beta_\xi \cdot \log_g(R_\xi)} \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]} \big)^{\rho_\xi} \\ F_\xi &= g^{\rho_\xi}, \end{aligned}$$

so that $\mathcal{A}$'s view is the same as in Game'_ξ. In contrast, if $T_t \in_R \mathbb{G}$, it can be written $T_t = g^{a_t b + z_t}$ for some uniformly random $z_t \in_R \mathbb{Z}_p$ that does not appear anywhere else. In this case, (D_ξ, E_ξ, F_ξ) is just a triple of uniformly random group elements, meaning that $\mathcal{A}$'s view is the same as in Game_ξ. □

Theorem 4. *The above LAF provides evasiveness under the SXDH and wD3DH1 assumptions, assuming that* CMH *is a collision-resistant chameleon hash function. Namely, for any PPT evasiveness adversary, there exist efficient algorithms* $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ *with comparable running time and such that*

$$\begin{aligned} \mathbf{Adv}_Q^{\mathcal{A},\mathsf{eva}} \leq\ & \mathbf{Adv}_{\mathcal{B}_0}^{\mathsf{CMH\text{-}CR}}(\lambda) + n \cdot \mathbf{Adv}_{\mathcal{B}_1}^{\mathrm{wD3DH1}}(\lambda) \\ & + n \cdot \mathbf{Adv}_{\mathcal{B}_2}^{\mathrm{DDH}_2}(\lambda) + 2n \cdot (1 + L) \cdot \mathbf{Adv}_{\mathcal{B}_3}^{\mathrm{DDH}_1}(\lambda), \end{aligned}$$

(The proof is given in the full version of the paper.)

Acknowledgements

We thank the reviewers for their careful reading. This work was funded in part by the French ANR ALAMBIC project (ANR-16-CE39-0006).

References

1. M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *Asiacrypt*, 2009. 1
2. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *Eurocrypt*, 2009. 1, 1
3. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM-CCS*, 1993. 1
4. M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *Eurocrypt*, 1996. 1
5. M. Bellare and S. Yilek. Encryption schemes secure under selective opening attack. Cryptology ePrint Archive: Report 2009/101, 2009. 1
6. J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *SAC*, 2002. 1
7. O. Blazy, E. Kiltz, and J. Pan. (Hierarchical) Identity-Based Encryption from Affine Message Authentication. In *Crypto*, 2014. 1, 1, 4, 4.1, 4.1
8. F. Böhl, D. Hofheinz, T. Jager, J. Koch, J.-H. Seo, C. Striecks. Practical Signatures from Standard Assumptions. In *Eurocrypt*, 2013. 3.1
9. S. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Crypto*, 2008. 1
10. D. Boneh, S. Halevi, M. Hamburg, R. Ostrovsky. Circular-Secure Encryption from Decision Diffie-Hellman. In *Crypto*, 2008. 1
11. D. Boneh, X. Boyen. Efficient Selective Identity-Based Encryption Without Random Oracles. In *Eurocrypt*, 2004. 1, 3.2
12. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *Crypto*, 2004. 1
13. X. Boyen. Reusable cryptographic fuzzy extractors. In *ACM-CCS*, 2004. 1
14. X. Boyen and B. Waters. Shrinking the keys of discrete-log-type lossy trapdoor functions. In *ACNS*, 2010. 1, 1, 3
15. Z. Brakerski and G. Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In *Crypto*, 2011. 1
16. J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In *Crypto*, 2013. 1, 1
17. Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Eurocrypt*, 2004. 1
18. C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer. Magic functions. *J. of the ACM*, 50(6), 2003. 1
19. D. Freeman, O. Goldreich, E. Kiltz, and G. Rosen, A.and Segev. More constructions of lossy and correlation-secure trapdoor functions. *J. of Cryptology*, 26(1), 2013. 1
20. E. Fujisaki. All-but-many encryption - a new framework for fully-equipped UC commitments. In *Asiacrypt*, 2014. 1
21. R. Gay, D. Hofheinz, E. Kiltz, and H. Wee. Tightly CCA-secure encryption without pairings. In *Eurocrypt*, 2016. 1
22. R. Gay, D. Hofheinz, L. Kohl. Kurosawa-Desmedt Meets Tight Security. In *Crypto*, 2017. 1
23. B. Hemenway and R. Ostrovsky. Extended-DDH and lossy trapdoor functions. In *PKC*, 2012. 1

24. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In *Crypto*, 2008.
25. D. Hofheinz. All-but-many lossy trapdoor functions. In *Eurocrypt*, 2012. 1, 1, 3.3
26. D. Hofheinz. Circular chosen-ciphertext security with compact ciphertexts. In *Eurocrypt*, 2013. Cryptology ePrint Archive: Report 2012/150. 1, 1, 2.1, 2.1, 3, 4
27. D. Hofheinz. Algebraic partitioning: Fully compact and (almost) tightly secure cryptography. In *TCC-A*, 2016. 1
28. D. Hofheinz. Adaptive partitioning. In *Eurocrypt*, 2017. 1
29. D. Hofheinz and T. Jager. Tightly secure signatures and public-key encryption. In *Crypto*, 2012. 1
30. D. Hofheinz and N.-K. Nguyen. On Tightly Secure Primitives in the Multi-Instance Setting. Cryptology ePrint Archive: Report 2018/958. 1
31. C. Jutla, A. Roy. Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces. In *Asiacrypt*, 2013. 3.1
32. H. Krawczyk and T. Rabin. Chameleon Signatures. In *NDSS*, 2000. 2.2
33. S. Kunz-Jacques, D. Pointcheval. About the Security of MTI/C0 and MQV. In *SCN*, 2006. 1, 2.3, 4
34. A. Lewko, A. Sahai, B. Waters. Revocation Systems with Very Small Private Keys. *IEEE Symposium on Security and Privacy*, 2010. 1, 3
35. B. Libert, M. Joye, T. Peters, and M. Yung. Concise Multi-Challenge CCA-secure Encryption and Signatures with Almost Tight Security. In *Asiacrypt*, 2014. 1
36. B. Libert, T. Peters, M. Joye, and M. Yung. Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In *Asiacrypt*, 2015. 1
37. B. Libert, A. Sakzad, D. Stehlé, and R. Steinfeld. All-But-Many Lossy Trapdoor Functions and Selective-Opening Chosen-Ciphertext Security. In *Crypto*, 2017. 1, 3.3
38. B. Libert, D. Vergnaud. Multi-Use Unidirectional Proxy Re-Signatures. In *ACM-CCS*, 2008. 2.3
39. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, 1997. 2, 4, 4.3
40. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, 2008. 1, 1
41. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Crypto*, 1991. 1
42. A. Raghunathan, G. Segev, and S. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In *Eurocrypt'13*, 2013. 1
43. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt*, 2005. 1, 1, 3.1
44. H. Wee. Dual projective hashing and its applications - lossy trapdoor functions and more. In *Eurocrypt*, 2012. 1
45. Y. Wen, S. Liu. Robustly Reusable Fuzzy Extractor from Standard Assumptions. In *Asiacrypt*, 2018. 1, 1, 2.1
46. M. Zhandry. The magic of ELFs. In *Crypto*, 2016. 1