Message Authentication Codes Secure against Additively Related-Key Attacks

Keita Xagawa

NTT Secure Platform Laboratories
xagawa.keita@lab.ntt.co.jp

Abstract. Message Authentication Code (MAC) is one of most basic primitives in cryptography. After Biham (EUROCRYPT 1993) proposed related-key attacks (RKAs), RKAs have damaged MAC's security. To relieve MAC of RKA distress, Bellare and Cash proposed pseudo-random functions (PRFs) secure against multiplicative RKAs (EUROCRYPT 2010). They also proposed PRFs secure against additive RKAs, but their reduction requires sub-exponential time. Since PRF directly implies Fixed-Input Length (FIL) MAC, their PRFs result in MACs secure against multiplicative RKAs.
In this paper, we proposed Variable-Input Length (VIL) MAC secure against *additive* RKAs, whose reductions are polynomial time in the security parameter. Our construction stems from MACs from number-theoretic assumptions proposed by Dodis, Kiltz, Pietrzak, Wichs (EUROCRYPT 2012) and public-key encryption schemes secure against additive RKAs proposed by Wee (PKC 2012).

1 Introduction

Message Authentication Code (MAC) is one of most basic primitive in cryptography. It generates a tag, denoted by τ, on a message, denoted by m, of arbitrary length by using a secret key, denoted by κ. A sender and receiver share key κ and verify integrity of the message with τ. It is required that any PPT adversary, who does not know κ, cannot produce a message m and a consistent tag τ.

In the area of secret-key cryptography, we often consider related-key attacks (RKAs). This attack captures correlation between two secret keys κ_1 and κ_2, which may be stemmed from low entropy of PRG, and a part of side-channel attacks and fault-injection attacks. Hence, we have been motivated to construct MACs secure against RKAs from the theoretical and practical views.

Secret-key primitives based on the number-theoretic assumptions: Meanwhile, for provable security, we often construct secret-key primitives based on the number-theoretic assumptions, e.g., Pseudo-Random Functions (PRFs) based on the decisional Diffie-Hellman (DDH) assumption [NR04] and those based on the factoring assumption [NRR02]. Since PRFs directly implies Fixed-Input Length (FIL) MACs, we have secure MACs based on the number-theoretic assumptions, which require costly computations.

When we allow MAC to be *probabilistic*, more efficient constructions are proposed by Dodis, Kiltz, Pietrzak, Wichs [DKPW12]. They proposed MACs from the DDH, gap-CDH, DCR, LWE, and factoring assumptions. Unfortunately, they are vulnerable under RKAs. Let us exemplify a simple RKA against a UF-CMVA secure MAC scheme based on the DDH assumption in Dodis et al. [DKPW12, Section 4.2].

Definition 1.1 (Dodis et al. [DKPW12]). *Let $\mathbb{G}$ be a finite group of prime order q. Let g_1, g_2 be a generator of $\mathbb{G}$. Let $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q$ be a collision-resistant hash function.*

Key Generation: *Choose $\kappa = (x_1, y_1, x_2, y_2) \leftarrow \mathbb{Z}_q^4$ uniformly at random.*
MAC: *Let $m \in \{0,1\}^*$ be a message: sample $r \leftarrow \mathbb{Z}_q$ and compute*

$$\tau = (c_1, c_2, k) = (g_1^r, g_2^r, c_1^{x_1\ell + y_1} \cdot c_2^{x_2\ell + y_2}),$$

where $\ell = \mathsf{H}(c_1, c_2, m)$. [1] *Output tag $\tau = (c_1, c_2, k)$.*

[1] In the 20121029:031553 version of Cryptology ePrint Archive: Report 2012/059, the label is m itself. We can attack this version.

Verification: *On input* m *and* $\tau = (c_1, c_2, k) \in \mathbb{G}^3$*, set* $\ell = \mathsf{H}(c_1, c_2, m)$*, compute* $k' = \Lambda_t^{\ell}(c_1, c_2) = c_1^{x_1\ell+y_1} \cdot c_2^{x_2\ell+y_2}$*, and output* acc *if* $k = k'$*; otherwise, output* rej.

For secret key $t = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$ and difference $\Delta = (\delta_1, \delta_2, \eta_1, \eta_2) \in \mathbb{Z}_q^4$, we define related-key derivation function $\phi_\Delta(t) = (x_1 + \delta_1, x_2 + \delta_2, y_1 + \eta_1, y_2 + \eta_2)$. We can mount a simple related-key attack as follows:

1. Let $m \in \{0, 1\}^*$ be a target message for forgery.
2. Choose non-zero difference $\Delta = (\delta_1, \delta_2, \eta_1, \eta_2) \in \mathbb{Z}_q^4$.
3. Query ϕ_Δ and m to the related-key tag-generation oracle, which makes a tag on m under the key $\phi_\Delta(\kappa)$.
4. Receive $\tau' = (c_1, c_2, k') = (c_1, c_2, c_1^{(x_1+\delta_1)\ell+(y_1+\eta_1)} \cdot c_2^{(x_2+\delta_2)\ell+(y_2+\eta_2)})$, where $c_1 = g_1^r$, $c_2 = g_2^r$, and $\ell = \mathsf{H}(c_1, c_2, m)$.
5. Compute $k = k' \cdot c_1^{-(\delta_1\ell+\eta_1)} \cdot c_2^{-(\delta_2\ell+\eta_2)} = c_1^{x_1\ell+y_1} \cdot c_2^{x_2\ell+y_2}$.
6. Output m and $\tau = (c_1, c_2, k)$.

This RKA exploits the algebraic structure of MAC.

RKA-secure secret-key primitives based on the number-theoretic assumptions: Since Bellare and Cash [BC10]'s breakthrough, several researchers have studied RKA-secure primitives. Bellare and Cash [BC10] proposed PRFs (thus, FIL-MACs) secure against multiplicative RKAs based on the DDH assumption. They also proposed a DDH-based PRF secure against additive RKAs whose reduction requires *exponential* time.

Goyal, O'Neil, and Rao [GOR11] RKA-secure weak PRFs based on q-Diffie-Hellman Inversion assumption (q-DHI assumption, in short).

Under the decisional bilinear Diffie-Hellman (DBDH) assumption, Bellare, Paterson, and Thomson [BPT12] proposed symmetric-key encryption scheme which is CPA and CCA secure against RKAs with respect to $\Phi = \{\phi_{a,b} : x \mapsto ax + b \mid a \in \mathbb{Z}_q^*, b \in \mathbb{Z}_q\}$. Symmetric-key encryption scheme turns into FIL-MAC. Therefore, we already have a Φ^+-RKA-secure FIL-MAC based on the DBDH assumption.

Summarizing the above, we have RKA-secure MAC schemes based on the assumptions related to discrete logarithm.

1.1 Our Contribution

We provide MAC schemes secure against additive RKAs based on the factoring, DDH, and DBDH assumptions.

We take two approaches following Dodis et al. [DKPW12]. Dodis et al. constructed MACs from *labeled* Hash Proof System (HPS) and *labeled* CCA2-secure (public-key or symmetric-key) encryption schemes. (The example appeared in the above is a MAC from a symmetric-key version of the DDH-based labeled HPS.)

The key technique to enhance the security is putting a public key into the labels, where a public key can be considered as a key fingerprint of the secret key. Notice that in the above example, the label ℓ is set as $\mathsf{H}(c_1, c_2, m)$. We replace it with $\mathsf{H}(\mu(t), c_1, c_2, m)$, where $\mu(t)$ is a public key corresponding to t. This simple patch has already appeared in the context of key-substitution attacks and rouge-key attacks [BWM99,MS04], in which an adversary, given a user's verification key vk and signature σ on a message m, and generates its verification key vk' (and signing key if possible) such that (vk', σ, m) passes the verification. (See e.g., [MS04, Section 4].) In addition, this technique is already exploited by Bellare and Cash [BC10] and Wee [Wee12].

Notes: The extended abstract of this paper appeared at SCIS 2013 (January, 2013). In the accepted papers list of FSE 2013, I found a similar paper titled as "Secure Message Authentication against Related-Key Attack" and written by Bhattacharyya and Roy [BR13]. Our paper is an independent result.

2 Definitions

Hereafter, λ denotes the security parameter.

2.1 Message Authentication Codes

A MAC scheme, MAC, consists of four algorithms; the setup algorithm Setup that, on input 1^λ, outputs public parameters π, the key-generation algorithm KG that, on input π, outputs key κ, the tag-generation algorithm TAG that, on input π, κ, and message $m \in \{0,1\}^*$, outputs tag τ, and the verification algorithm VRFY that, on input π, κ, m, and τ, outputs acc or rej.

We say that the MAC scheme is correct if for any π and κ generated by Setup and KG, for any message m, we have

$$\mathsf{VRFY}(\pi, \kappa, m, \mathsf{TAG}(\pi, \kappa, m)) = \mathsf{acc}.$$

Standard security: Let us recall the security notion, unforgeability against chosen-message verification attack, UF-CMVA security, in short. Roughly speaking, we say the scheme is UF-CMVA secure if any PPT adversary cannot forge (m^*, τ^*) even if it is allowed to access to a tag-generation oracle, denoted by Tag. We follow the definition in Bellare, Goldreich, and Mityagin [BGM04].

Definition 2.1 (UF-CMVA security). *Define experiment* $\mathsf{Expt}^{\mathsf{uf\text{-}cmva}}_{\mathsf{MAC},\mathsf{A}}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: *Generate* $\pi \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $\kappa^* \leftarrow \mathsf{KG}(\pi)$*. Initialize* $L \leftarrow \emptyset$*, which will store the queries of* A *and corresponding answers. Run the adversary by feeding* π*.*

Learning: *The adversary could query to the tag-generation and verification oracles defined as follows:*

Tag: *It receives* m*, returns* $\tau \leftarrow \mathsf{TAG}(\pi, \kappa^*, m)$*, and updates* $L \leftarrow L \cup \{(m, \tau)\}$*.*

Vrfy: *It receives* m *and* τ*. It returns* $\mathsf{dec} \leftarrow \mathsf{VRFY}(\pi, \kappa^*, m, \tau)$*.*

Finalization: *The adversary stops with output* (m^*, τ^*)*. Output* 1 *if* $(m^*, *) \notin L$ *and* $\mathsf{VRFY}(\pi, \kappa^*, m^*, \tau^*) = \mathsf{acc}$*. Output* 0 *otherwise.*

We define the advantage of A *as* $\mathsf{Adv}^{\mathsf{uf\text{-}cmva}}_{\mathsf{MAC},\mathsf{A}}(\lambda) = \Pr\left[\mathsf{Expt}^{\mathsf{uf\text{-}cmva}}_{\mathsf{MAC},\mathsf{A}}(\lambda) = 1\right]$*. We say that* MAC *is* UF-CMVA *secure if for any PPT adversary* A*, its advantage* $\mathsf{Adv}^{\mathsf{uf\text{-}cmva}}_{\mathsf{MAC},\mathsf{A}}(\cdot)$ *is negligible.*

We can define strong unforgeability under chosen message and verification attacks (sUF-CMVA security in short) in a similar fashion by relaxing the conditions with $(m^*, \tau^*) \notin L$ and $\mathsf{VRFY}(\pi, \kappa^*, m^*, \tau^*) = \mathsf{acc}$. (The adversary can query m^* to the tag-generation oracle, Tag.)

RKA Security: We next define UF-CMVA security under related-key attacks and we call it as Unforgeability against related-key and chosen-message verification attack, UF-RK-CMVA security in short. We follow the RKA-security definition for PRFs in Bellare and Khono [BK03] and that for signatures in Bellare, Cash, and Miller [BCM11] rather than Goyal, O'Neill, and Rao [GOR11]. The UF-RK-CMVA security is defined as follows:

Definition 2.2 (UF-RK-CMVA security). *Define experiment* $\mathsf{Expt}^{\mathsf{uf\text{-}rk\text{-}cmva}}_{\mathsf{MAC},\mathsf{A},\Phi}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: *Generate* $\pi \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $\kappa^* \leftarrow \mathsf{KG}(\pi)$*. Initialize* $L \leftarrow \emptyset$*. Run the adversary by feeding* π*.*

Learning: *The adversary could query to the tag-generation and verification oracles defined as follows:*

RK-Tag: *It receives* $\phi \in \Phi$ *and* $m \in \{0,1\}^*$*. It returns* $\tau \leftarrow \mathsf{TAG}(\pi, \phi(\kappa^*), m)$*. If* $\phi(\kappa^*) = \kappa^*$ *then it updates* $L \leftarrow L \cup \{(\phi, m, \tau)\}$*.*

RK-Vrfy: *It receives* $\phi \in \Phi$*,* $m \in \{0,1\}^*$*, and* τ*. It returns* $\mathsf{dec} \leftarrow \mathsf{VRFY}(\pi, \phi(\kappa^*), m, \tau)$*.*

Finalization: *The adversary stops with output* (m^*, τ^*). *Output* 1 *if* $(\phi, m^*, *) \notin L$ *and* $\mathsf{VRFY}(\pi, \kappa, m^*, \tau^*) = \mathsf{acc}$. *Output* 0 *otherwise.*

We define the advantage of A *as* $\mathsf{Adv}^{\text{uf-rk-cmva}}_{\mathsf{MAC},\mathsf{A},\Phi}(\lambda) = \Pr\left[\mathsf{Expt}^{\text{uf-rk-cmva}}_{\mathsf{MAC},\mathsf{A},\Phi}(\lambda) = 1\right]$. *We say that* MAC *is* Φ-UF-RK-CMVA *secure if for any PPT adversary* A, *its advantage* $\mathsf{Adv}^{\text{uf-rk-cmva}}_{\mathsf{MAC},\mathsf{A},\Phi}(\cdot)$ *is negligible.*

As strong UF-CMVA security, we can define strong Φ-sUF-RK-CMVA security; we define it by relaxing the conditions with $(id, m^*, \tau^*) \notin L$ and $\mathsf{VRFY}(\pi, \kappa^*, m^*, \tau^*) = \mathsf{acc}$. (The adversary can query (id, m^*) to the tag-generation oracle, RK-Tag.)

2.2 One-Time Signature

A one-time signature scheme, OTS, consists of three algorithms; the key-generation algorithm $\mathsf{ots.gen}$ that, on input 1^λ, outputs a key pair (ovk, osk), the signing algorithm $\mathsf{ots.sign}$ that, on input osk and message $m \in \{0,1\}^*$, outputs signature σ, and the verification algorithm $\mathsf{ots.vrfy}$ that, on input ovk, m, and σ, outputs acc or rej.

We say that the one-time signature scheme is correct if for any λ and (ovk, osk) generated by $\mathsf{ots.gen}$, for any message m, we have

$$\mathsf{ots.vrfy}(ovk, m, \mathsf{ots.sign}(osk, m)) = \mathsf{acc}.$$

Definition 2.3 (sEUF-OTCMA security). *Define experiment* $\mathsf{Expt}^{\text{seuf-otcma}}_{\mathsf{OTS},\mathsf{A}}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: $(ovk, osk) \leftarrow \mathsf{ots.gen}(1^\lambda)$. *Initialize* $L \leftarrow \emptyset$. *Run the adversary by feeding ovk.*
Learning: *The adversary could query to the signing oracle only at once.*
OT-Sign: *It receives m. Return* $\sigma \leftarrow \mathsf{ots.sign}(osk, m)$ *and update* $L \leftarrow L \cup \{(m, \sigma)\}$.
Finalization: *The adversary stops with output* (m^*, σ^*). *Output* 1 *if* $(m^*, \sigma^*) \notin L$ *and* $\mathsf{ots.vrfy}(ovk, m^*, \sigma^*) = \mathsf{acc}$. *Output* 0 *otherwise.*

We define the advantage of A *as* $\mathsf{Adv}^{\text{seuf-otcma}}_{\mathsf{OTS},\mathsf{A}}(\lambda) = \Pr\left[\mathsf{Expt}^{\text{seuf-otcma}}_{\mathsf{OTS},\mathsf{A}}(\lambda) = 1\right]$. *We say* OTS *is* sEUF-OTCMA *secure if for any PPT adversary* A, *its advantage* $\mathsf{Adv}^{\text{seuf-otcma}}_{\mathsf{OTS},\mathsf{A}}(\cdot)$ *is negligible.*

2.3 Hash Functions

A family of hash functions consists of two algorithms: the setup algorithm Setup takes 1^λ as input and outputs hk; the evaluation algorithm H takes hk and message $m \in \{0,1\}^*$ and outputs digest h.

Definition 2.4 (Collision resistance). *Define experiment* $\mathsf{Expt}^{\text{coll}}_{\mathsf{A},\mathsf{Hash}}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: $hk \leftarrow \mathsf{Setup}(1^\lambda)$, *Run the adversary with hk.*
Finalization: *The adversary stops with output* $m, m' \in \mathcal{M}$. *Output* 1 *if* $m \neq m'$ *and* $\mathsf{H}(hk, m) = \mathsf{H}(hk, m')$; *output* 0 *otherwise.*

We define the advantage of A *as* $\mathsf{Adv}^{\text{coll}}_{\mathsf{Hash},\mathsf{A}}(\lambda). = \Pr\left[\mathsf{Expt}^{\text{coll}}_{\mathsf{Hash},\mathsf{A}}(\lambda) = 1\right]$. *We say* Hash *is collision resistant if for any PPT adversary* A, *its advantage* $\mathsf{Adv}^{\text{coll}}_{\mathsf{Hash},\mathsf{A}}(\cdot)$ *is negligible.*

Remark 2.1. To make presentation simple, we often write "we choose a hash function H" instead of "we choose a hash function $hk \leftarrow \mathsf{Setup}(1^\lambda)$."

2.4 Enhancement of Security

Here, we show by using sEUF-OTCMA-secure signature scheme, we can transform UF-CMVA-secure MAC into sUF-CMVA-secure MAC. This technique was proposed by Huang, Wong, Li, and Zhao [HWLZ08] and related papers [BSW06,TOO08,SPW07] to enhance security of signature. e note that this transformation preserves RKA security.

Definition 2.5 (Transfomation).

$\overline{\mathsf{Setup}}(1^\lambda)$: *output* $\overline{\pi} = \pi \leftarrow \mathsf{Setup}(1^\lambda)$.
$\overline{\mathsf{KG}}(\overline{\pi})$: *output* $\overline{\kappa} = \kappa \leftarrow \mathsf{Setup}(\pi)$.
$\overline{\mathsf{TAG}}(\overline{\pi}, \overline{\kappa}, m)$: *generate* $(ovk, osk) \leftarrow \mathsf{ots.gen}(1^\lambda)$*; generate a tag* $\tau \leftarrow \mathsf{TAG}(\pi, \kappa, ovk)$*; generate a signature* $\sigma \leftarrow \mathsf{ots.sign}(osk, (m, \tau))$*; output* $\overline{\tau} = (ovk, \tau, \sigma)$.
$\overline{\mathsf{VRFY}}(\overline{\pi}, \overline{\kappa}, m, \overline{\tau})$: *parse* $(ovk, \tau, \sigma) \leftarrow \overline{\tau}$*; verify* $\mathsf{ots.vrfy}(ovk, (m, \tau), \sigma) = \mathsf{acc}$ *and* $\mathsf{VRFY}(\pi, \kappa, \tau) = \mathsf{acc}$. *If both are accepted, output* acc. *Otherwise, output* rej.

Lemma 2.1. *Let* MAC *be a* Φ-UF-RK-CMVA*-secure MAC scheme. Let* OTS *be a* sEUF-OTCMA*-secure signature scheme. Then,* $\overline{\mathsf{MAC}}$ *is* Φ-sUF-RK-CMVA *secure.*

3 Construction from Labeled Hash Proof System

In this section, we proposed Φ^+-RKA-secure MAC scheme based on the Labeled Hash Proof Systems (L-HPSs).

3.1 Labeled Hash Proof System

Let us recall L-HPS, which is called extended HPS originally [CS02].

Let $\mathcal{C}$ and $\mathcal{K}$ be finite sets. Consider $\mathcal{V} \subset \mathcal{C}$, which is a valid ciphertext space. The space of labels is denoted by $\mathcal{L}$. The secret- and public-key space is denoted by $\mathcal{T}$ and $\mathcal{F}$, respectively. Let $\Lambda_t : \mathcal{C} \times \mathcal{L} \to \mathcal{K}$ be a hash function indexed by secret $t \in \mathcal{T}$.

We can summarize the properties of labeled hash function as follows:

Projective: We say a labeled hash function $\Lambda_t^\ell(\cdot) = \Lambda_t(\cdot, \ell)$ is *projective* if there exists $\mu : \mathcal{T} \to \mathcal{F}$ which uniquely determines the action of $\Lambda_t^\ell : \mathcal{C} \to \mathcal{K}$ over $\mathcal{V}$.

Universal$_1$: We say that a projective labeled hash function is ϵ_1*-almost universal*$_1$ if for all $c \in \mathcal{C} \setminus \mathcal{V}$ and $\ell \in \mathcal{L}$,

$$\Delta((f, \Lambda_t^\ell(C)), (f, k)) \leq \epsilon_1$$

where $f = \mu(t)$ for $t \leftarrow \mathcal{T}$ and $k \leftarrow \mathcal{K}$. If $\epsilon_1 = 0$, then we often omit "ϵ_1-almost."

Universal$_2$: We say that a projective labeled hash function is ϵ_2*-almost universal*$_2$ if for all $c, c^* \in \mathcal{C} \setminus \mathcal{V}$ and $\ell, \ell^* \in \mathcal{L}$ with $\ell \neq \ell^*$,

$$\Delta((f, \Lambda_t^\ell(c), \Lambda_t^{\ell^*}(c^*)), (f, \Lambda_t^\ell(c), k)) \leq \epsilon_2$$

where $f = \mu(t)$ for $t \leftarrow \mathcal{T}$ and $K \leftarrow \mathcal{K}$. If $\epsilon_2 = 0$, then we often omit "ϵ_2-almost."

Extracting: We say that a projective labeled hash function is ϵ_{ext}*-almost extracting* if for any $c \in \mathcal{C}$ and $\ell \in \mathcal{L}$,

$$\Delta(\Lambda_t^\ell(c), k) \leq \epsilon_{\mathrm{ext}}$$

where $t \leftarrow \mathcal{T}$ and $K \leftarrow \mathcal{K}$.

Syntax: A labeled hash proof system $\mathsf{HPS} = (\mathsf{Setup}_{\mathsf{HPS}}, \mathsf{SampR}, \mathsf{Pub}, \mathsf{Priv})$ consists of four algorithms:

- Setup is a setup algorithm that, on input 1^λ, output public parameters π, which define $\mathcal{C}$, $\mathcal{V}$, $\mathcal{F}$, $\mathcal{T}$, $\{\Lambda_t : \mathcal{C} \times \mathcal{L} \to \mathcal{K} \mid t \in \mathcal{T}, \ell \in \mathcal{L}\}$, and $\mu : \mathcal{T} \to \mathcal{F}$.
- SampR is a sampling algorithm that, on input π and randomness r, outputs $c \in \mathcal{V} \subseteq \mathcal{C}$.
- Pub is a public evaluation algorithm that, on input π, $\mu(t)$, label ℓ, and r used to generate c, and outputs $k = \Lambda_t^\ell(c)$.
- Priv is a private evaluation algorithm that, on input π, t, ℓ, and c, outputs $k = \Lambda_t^\ell(c)$.

Finally, we recall the subset membership problem.

Definition 3.1 (The subset-membershp problem assumption). *Define experiment* $\mathsf{Expt}^{\mathsf{smp}}_{\mathsf{HPS},\mathsf{A}}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: $\pi \leftarrow \mathsf{Setup}(1^\lambda)$. *Choose* $b \leftarrow \{0,1\}$, $c_0 \leftarrow \mathcal{C}$, $c_1 \leftarrow \mathcal{V}$. *Run the adversary by feeding* π *and* c_b.
Finalization: *The adversary stops with output* b'. *Output* 1 *if* $b = b'$. *Output* 0 *otherwise.*

We define the advantage of A *as* $\mathsf{Adv}^{\mathsf{smp}}_{\mathsf{HPS},\mathsf{A}}(\lambda) = \left|\Pr[\mathsf{Expt}^{\mathsf{smp}}_{\mathsf{HPS},\mathsf{A}}(\lambda) = 1] - \frac{1}{2}\right|$. *We say that the subset membership problem is hard if for any PPT adversary* A, *its advantage* $\mathsf{Adv}^{\mathsf{smp}}_{\mathsf{HPS},\mathsf{A}}(\lambda)$ *is negligible in* λ.

As a concrete example, we will take the DDH-based L-HPS. See Section 3.5 for details.

3.2 Our Additional Requirements

In addition, we define the properties of L-HPS as follows:

$\mathcal{K}$'s commutativity: We say that a labeled hash function is $\mathcal{K}$-commutative if $\mathcal{K}$ is a commutative group.
μ's homomorphism: We say that a labeled hash function is μ-homomorphic if its projection function μ is a homomorphism from $\mathcal{T}$ to $\mathcal{F}$. Hereafter, we assume that $\mathcal{T}$ is an additive finite group.
Key-Homomorphism: We say that a labeled hash function is key-homomorphic if Λ_t^ℓ is homomorphic with respect to $t \in \mathcal{T}$. Specifically, for any ℓ, $c \in \mathcal{C}$, $\mu(t)$, and Δ, we can efficiently compute $k' = \Lambda_{t+\Delta}^\ell(c)$ from $k = \Lambda_t^\ell(c)$. For example, $\Lambda_{t+\Delta}^\ell(c) = \Lambda_t^\ell(c) \cdot \Lambda_\Delta^\ell(c)$.
μ's Φ-collision resistance: We say that a labeled hash function is Φ-collision-resistant if the problem on μ's collision with respect to Φ, defined later, is hard.

Definition 3.2 (μ's Φ-collision resistance). *Define experiment* $\mathsf{Expt}^{\mu\text{-coll}}_{\mathsf{HPS},\mathsf{A},\Phi}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: $\pi \leftarrow \mathsf{Setup}(1^\lambda)$. $t \leftarrow \mathcal{T}$. *Run the adversary by feeding* π, $\mu(t)$, *and trapdoor* t.
Finalization: *The adversary stops with output* $\phi \in \Phi$. *Output* 1 *if* $\phi(t) \neq t$ *and* $\mu(\phi(t)) = \mu(t)$. *Output* 0 *otherwise.*

We define the advantage of A *as* $\mathsf{Adv}^{\mu\text{-coll}}_{\mathsf{HPS},\mathsf{A},\Phi}(\lambda) = \Pr[\mathsf{Expt}^{\mu\text{-coll}}_{\mathsf{HPS},\mathsf{A},\Phi}(\lambda) = 1]$. *We say that the subset membership problem is hard if for any PPT adversary* A, *its advantage* $\mathsf{Adv}^{\mu\text{-coll}}_{\mathsf{HPS},\mathsf{A},\Phi}(\lambda)$ *is negligible in* λ.

3.3 Our Construction

Before describing our construction, let us explain the intuition.

We recall the MAC construction from L-HPS by Dodis et al. [DKPW12]. The key generation algorithm outputs $\kappa = t \leftarrow \mathcal{T}$. The tag is $k = \Lambda_t^\ell(c) \in \mathcal{K}$, where $c \leftarrow \mathcal{V}$ and $\ell = \mathsf{H}(c, m)$. [2] The verification is done by checking whether $k = \Lambda_t^\ell(c)$ or not.

[2] In the 20121029:031553 version of Cryptology ePrint Archive: Report 2012/059, the label is m itself. We can attack this version.

We explicitly employ $f = \mu(t)$ as a key-fingerprinting. In our construction, the label is computed as $\ell = \mathsf{H}(f, c, m)$. This slight modification prevents an adversary tamper a secret key. Intuitively speaking, even if the adversary obtains a tag $\tau' = (c, k')$ produced by $\phi(t)$, here ℓ' is $\mathsf{H}(\mu(\phi(t)), c, m)$. Hence, this (c, k') is independent from the tag produced with t and the adversary cannot exploit this tag.

Let $(\mathsf{Setup}_{\mathsf{HPS}}, \mathsf{SampR}, \mathsf{Pub}, \mathsf{Priv})$ be a L-HPS. Our MAC, $\mathsf{MAC}_{\mathsf{HPS}}$, is defined as follows:

$\mathsf{Setup}(1^\lambda)$: $\pi \leftarrow \mathsf{Setup}(1^\lambda)$. Define a hash function $\mathsf{H} : \{0,1\}^* \to \mathcal{K}$. Output public parameters (π, H).
$\mathsf{KG}(\pi, \mathsf{H})$: Choose $t \leftarrow \mathcal{T}$. Output key $\kappa = t$.
$\mathsf{TAG}(\pi, \mathsf{H}, \kappa, m)$: On input key $\kappa = t$ and message $m \in \{0,1\}^*$,

1. compute $f \leftarrow \mu(t)$,
2. choose r uniformly at random,
3. compute $c \leftarrow \mathsf{SampR}(r)$,
4. compute label $\ell \leftarrow \mathsf{H}(f, c, m)$,
5. compute $k \leftarrow \mathsf{Priv}(t, \ell, c)$,
6. set $\tau \leftarrow (c, k)$,
7. and output tag τ.

$\mathsf{VRFY}(\pi, \mathsf{H}, \kappa, m, \tau)$: On input $\kappa = t$, $m \in \{0,1\}^*$, and tag $\tau = (c, k)$,

1. compute $f \leftarrow \mu(t)$,
2. compute $\ell' \leftarrow \mathsf{H}(f, c, m)$,
3. compute $k' \leftarrow \mathsf{Priv}(t, \ell', c)$,
4. and output acc if $k = k'$; otherwise, output rej.

3.4 Security

Theorem 3.1. *Suppose that L-HPS* HPS *is universal*$_2$ *and extracting, and the subset membership problem is hard. Let* $\mathcal{T}$ *be an additive commutative group and define* $\Phi^+ = \{\phi_\Delta : t \mapsto t + \Delta \mid \Delta \in \mathcal{T}\}$. *Moreover, suppose that* HPS *is* $\mathcal{K}$*-commutative,* μ*-homomorphic, key-homomorphic, and* Φ^+*-collision resistant. Then,* MAC *is* Φ^+-UF-RK-CMVA *secure.*

We adopt a game-hopping proof. We mainly follow the sequence of games in [DKPW12], but in some case, there are differences.

Let us show the details.

$\mathsf{Expt}_{\mathrm{real}}$*:* This is the original experiment. Therefore, we have that

$$\mathsf{Adv}^{\mathsf{uf\text{-}rk\text{-}cmva}}_{\mathsf{MAC},\mathsf{A},\Phi^+}(\lambda) = \Pr[\mathsf{Expt}_{\mathrm{real}} = 1].$$

$\mathsf{Expt}'_{\mathrm{real}}$*:* In this game, the challenger outputs $\perp$ if the adversary queries ϕ such that $t + \Delta \neq t$ but $\mu(t + \Delta) = \mu(t)$. We note that there is no non-zero Δ which makes $t + \Delta = t$.

Claim. There exists a PPT adversary B such that

$$|\Pr[\mathsf{Expt}_{\mathrm{real}} = 1] - \Pr[\mathsf{Expt}'_{\mathrm{real}} = 1]| \leq \mathsf{Adv}^{\mu\text{-}\mathsf{coll}}_{\mathsf{HPS},\mathsf{B},\Phi^+}(\lambda).$$

Proof. The two games differ when the adversary queries $\Delta_i \neq 0$ to the oracles which makes $t + \Delta_i \neq t$ and $\mu(t + \Delta_i) = \mu(t)$. It is obvious that this contradicts μ's Φ^+-collision-resistance property. □

$\mathsf{Expt}''_{\mathrm{real}}$*:* Next, the challenger outputs $\perp$ if there exists a collision on computation of ℓ. Note that we now eliminate the collision of μ and ℓ.

Claim. There exists a PPT adversary B such that

$$|\Pr[\mathsf{Expt}'_{\mathrm{real}} = 1] - \Pr[\mathsf{Expt}''_{\mathrm{real}} = 1]| \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash},\mathsf{B}}(\lambda).$$

Proof. It is obvious that this contradicts the collision-resistance property of Hash. □

$\mathsf{Expt}_{i,j}$: We next change the two oracles repeatedly. For $i \in [Q_T]$ and $j \in \{0,1,2,3,4\}$, we define games $\mathsf{Expt}_{i,j}$ as follows:

$\mathsf{Expt}_{i,0}$: In this game, RK-Tag is defined as follows: On the first $i-1$ queries, answer with random $(c,k) \leftarrow C \times \mathcal{K}$. On the rest queries, it answers the tag as the original.

$\mathsf{Expt}_{i,1}$: In this game, RK-Tag is defined as follows: On the first $i-1$ queries, answer with random $(c,k) \leftarrow C \times \mathcal{K}$ as in $\mathsf{Expt}_{i,0}$. On i-th query, the oracle chooses $c \leftarrow C$, computes $\ell \leftarrow H(\mu(t+\varDelta), c, m)$, computes $k \leftarrow \mathsf{Priv}(t+\varDelta, \ell, c)$, and answers (c,k). On the rest queries, it answers the tag as the original.

$\mathsf{Expt}_{i,2}$: We next change the oracle RK-Vrfy. It rejects if $c \in C \setminus \mathcal{V}$.

$\mathsf{Expt}_{i,3}$: We again change behaviour of RK-Tag. On i-th query, the oracle chooses $c \leftarrow C$, computes $\ell \leftarrow H(\mu(t+\varDelta), c, m)$, computes $k \leftarrow \mathcal{K}$,

$\mathsf{Expt}_{i,4}$: We reset the verification oracle. Oracle RK-Vrfy answers as in the original.

The following table summarizes the difference on the i-th answer from RK-Tag and the behavior of RK-Vrfy.

	The i-th answer from RK-Tag	RK-Vrfy
$\mathsf{Expt}_{i,0}$	$c \leftarrow \mathcal{V}$, $\ell \leftarrow \mathsf{H}(\mu(t+\varDelta), c, m)$, $k \leftarrow \mathsf{Priv}(t+\varDelta, \ell, c)$.	Real
$\mathsf{Expt}_{i,1}$	$c \leftarrow C$, $\ell \leftarrow \mathsf{H}(\mu(t+\varDelta), c, m)$, $k \leftarrow \mathsf{Priv}(t+\varDelta, \ell, c)$.	Real
$\mathsf{Expt}_{i,2}$	$c \leftarrow C$, $\ell \leftarrow \mathsf{H}(\mu(t+\varDelta), c, m)$, $k \leftarrow \mathsf{Priv}(t+\varDelta, \ell, c)$.	Reject if $c \in C \setminus \mathcal{V}$
$\mathsf{Expt}_{i,3}$	$c \leftarrow C$, $\ell \leftarrow \mathsf{H}(\mu(t+\varDelta), c, m)$, $k \leftarrow \mathcal{K}$.	Reject if $c \in C \setminus \mathcal{V}$
$\mathsf{Expt}_{i,4}$	$c \leftarrow C$, $\ell \leftarrow \mathsf{H}(\mu(t+\varDelta), c, m)$, $k \leftarrow \mathcal{K}$.	Real

Claim. For all $i \in [Q_T]$, there exists a PPT adversary B such that

$$|\Pr[\mathsf{Expt}_{i,0} = 1] - \Pr[\mathsf{Expt}_{i,1} = 1]| \leq \mathsf{Adv}^{\mathsf{smp}}_{\mathsf{HPS},\mathsf{B}}(\lambda).$$

Proof. The difference between $\mathsf{Expt}_{i,0}$ and $\mathsf{Expt}_{i,1}$ is the answer for i-th query to RK-Tag. In $\mathsf{Expt}_{i,0}$, the oracle computes a tag

$$c \leftarrow \mathcal{V}, \ell \leftarrow \mathsf{H}(\mu(t+\varDelta), c, m), k \leftarrow \mathsf{Priv}(t+\varDelta, \ell, c), \text{ and } \tau \leftarrow (c,k).$$

In $\mathsf{Expt}_{i,1}$, the oracle computes a tag

$$c \leftarrow C, \ell \leftarrow \mathsf{H}(\mu(t+\varDelta), c, m), k \leftarrow \mathsf{Priv}(t+\varDelta, \ell, c), \text{ and } \tau \leftarrow (c,k).$$

Therefore, it is easy to show that this contradicts the hardness of the subset membership problem. □

Claim. For all $i \in [Q_T]$, we have that $|\Pr[\mathsf{Expt}_{i,1} = 1] - \Pr[\mathsf{Expt}_{i,2} = 1]| \leq Q_V/\#\mathcal{K}$.

Proof. The two games differ if the adversary queries $\varDelta_j$, m_j, and $\tau_j = (c_j, k_j)$ to the related-key verification oracle such that $c_j \in C \setminus \mathcal{V}$ and $k_j = \Lambda^{\ell_j}_{t+\varDelta_j}(c_j)$ with $\ell_j = H(\mu(t+\varDelta_j), c_j, m_j)$. Otherwise, the two games are equivalent.

We first note that the adversary can learn such an inconsistent tag only from i-th tag-generation query, c^* and k^* with label $\ell^* = H(\mu(sk+\varDelta^*), c^*, m^*)$, which is valid in $\mathsf{Expt}_{i,1}$ but invalid in $\mathsf{Expt}_{i,2}$. We next note that, in order to run $\mathsf{Expt}_{i,2}$, one should have trapdoor information on C denoted by w.

From the hypothesis on the adversary's verification queries, we have that $(c^*, k^*, m^*, \varDelta^*) \neq (c_j, k_j, m_j, \varDelta_j)$ for any $j \in [Q_V]$. We can classify the j-th query into two cases:

- $(c^*, m^*, \varDelta^*) = (c_j, m_j, \varDelta_j)$: In this case, $k^* \neq k_j$ holds. Hence, the j-th verification query is rejected in $\mathsf{Expt}_{i,1}$. Since this query is also rejected in $\mathsf{Expt}_{i,2}$, the adversary learns nothing.

– $(c^*, m^*, \varDelta^*) \neq (c_j, m_j, \varDelta_j)$: Recall that $\ell^* \neq \ell_j$, since we have eliminated the label reuse at $\mathsf{Expt}''_{\mathrm{real}}$. Here, the adversary knows the tag k^* with $(c^*, k^*, m^*, \varDelta^*)$, where $k^* = \varLambda^{\ell^*}_{t+\varDelta^*}(c^*)$. By the universal$_2$ property, a bad $k_j = \varLambda^{\ell_j}_{t+\varDelta_j}(c_j) = \varLambda^{\ell_j}_{t}(c_j) \cdot \varLambda^{\ell_j}_{\varDelta_j}(c_j)$ is uniform at random even after seeing $f = \mu(t)$ and $\varLambda^{\ell^*}_{t}(c^*)$. Hence, we upper-bound the probability that such an event occurs as we want.

This completes the proof. □

Claim. For all $i \in [Q_T]$, we have that $|\Pr[\mathsf{Expt}_{i,2} = 1] - \Pr[\mathsf{Expt}_{i,3} = 1]| \leq 1/\#\mathcal{K}$.

Proof. By the similar argument to the proof of the previous claim, the bound follows from universal$_2$ property. □

Claim. For all $i \in [Q_T]$, we have that $|\Pr[\mathsf{Expt}_{i,3} = 1] - \Pr[\mathsf{Expt}_{i,4} = 1]| \leq Q_V/\#\mathcal{K}$.

Proof. By the same argument to that in order to bound $\mathsf{Expt}_{i,1}$ and $\mathsf{Expt}_{i,2}$, the bound follows from universal$_2$ property.

Claim. For all $i \in [Q_T]$, we have that $\Pr[\mathsf{Expt}_{i,4} = 1] = \Pr[\mathsf{Expt}_{i+1,0} = 1]$.

Proof. From the definitions of games, the statement follows. □

$\mathsf{Expt}_{\mathrm{final}}$*:* In the final game, RK-Vrfy rejects all queries.

Claim. We have that $|\Pr[\mathsf{Expt}_{Q_T,4} = 1] - \Pr[\mathsf{Expt}_{\mathrm{final}} = 1]| \leq Q_V/\#\mathcal{K}$.

Proof. We note that the adversary obtains no information about sk until it makes the first verification query. Therefore, from the universal$_1$ property of HPS, a query $(m_1, c_1, k_1, \varDelta_1)$ hits the right $k_1 = \varLambda^{\ell_1}_{t+\varDelta_1}(c_1) = \varLambda^{\ell_1}_{t}(c_1) \cdot \varLambda^{\ell_1}_{\varDelta_1}(c_1)$ is at most $1/\#\mathcal{K}$. By the hybrid argument, the distance is upper-bounded by $Q_V/\#\mathcal{K}$. □

Claim. We have that $|\Pr[\mathsf{Expt}_{\mathrm{final}} = 1]| \leq 1/\#\mathcal{K}$.

Proof. Since, in the game, the tag-generation oracle returns random elements and the verification oracle rejects any attempts, now, the challenger need not to know t and $f = \mu(t)$. Hence, the adversary cannot learn even the projective key f. From the universal$_1$ property of HPS, the forge (m^*, c^*, k^*) produced by the adversary is valid at most probability $1/\#\mathcal{K}$. □

3.5 Instantiation from DDH

We review the DDH assumption and a labeled HPS in Cramer and Shoup [CS02].

The DDH assumptions: $\mathsf{GroupG_{DDH}}$ outputs $(\mathbb{G}, q, g)$, where $\mathbb{G}$ be a cyclic group of prime order q and g is a generator of $\mathbb{G}$.

Definition 3.3 (DDH assumption). *For an adversary,* A*, we define its advantage as*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ddh}}_{\mathsf{GroupG_{DDH}},\mathsf{A}}(\lambda) = \Pr[&(\mathbb{G}, q, g) \leftarrow \mathsf{GroupG_{DDH}}(1^\lambda), a, b \leftarrow \mathbb{Z}_q : \mathsf{A}(\mathbb{G}, q, g, g^a, g^b, g^{ab}) = 1] \\ &- \Pr[(\mathbb{G}, q, g) \leftarrow \mathsf{GroupG_{DDH}}(1^\lambda), a, b, c \leftarrow \mathbb{Z}_q : \mathsf{A}(\mathbb{G}, q, g, g^a, g^b, g^c) = 1].\end{aligned}$$

We say that A *(t, ϵ)-solves the DDH problem if* A *runs in time t and its advantage is larger than ϵ. We say that the DDH assumption (w.r.t.* $\mathsf{GroupG_{DDH}}$*) holds if for any PPT adversary* A*, its advantage is negligible in λ.*

Labeled hash functions: Cramer and Shoup proposed the DDH-based labeled hash functions defined as follows: Let g_1 and g_2 be generators of $\mathbb{G}$. Let $\mathcal{C} = \mathbb{G}^2$, $\mathcal{K} = \mathbb{G}$, and $\mathcal{V} = \{(g_1^r, g_2^r) : r \in \mathbb{Z}_q\}$. Let $\mathcal{L} = \mathbb{Z}_q^*$, $\mathcal{T} = \mathbb{Z}_q^4$, and $\mathcal{F} = \mathbb{G}^2$. For $t = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$, we define a projection function as

$$\mu : \mathcal{T} \to \mathcal{F} : (x_1, x_2, y_1, y_2) \mapsto (X, Y) = (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}).$$

For $t = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$, $c = (c_1, c_2) \in \mathbb{G}^2$, $\ell \in \mathbb{Z}_q$, we define a labeled hash function as

$$\Lambda_t^{\ell}(c_1, c_2) = c_1^{x_1\ell + y_1} \cdot c_2^{x_2\ell + y_2}.$$

L-HPS: Let us review the labeled hash proof system in [CS02].

$\mathsf{Setup}_{\mathsf{HPS}}$: $(\mathbb{G}, q, g) \leftarrow \mathsf{GroupG}_{\mathsf{DDH}}(1^{\lambda})$, $w \leftarrow \mathbb{Z}_q$, $g_1 \leftarrow g$, $g_2 \leftarrow g^w$. Choose a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q^*$. Output $\pi = (\mathbb{G}, q, g_1, g_2, \mathsf{H})$. (We implicitly set w as the language trapdoor for $\mathcal{V} = \{(g_1^r, g_2^r)\}$.)

Pub: On input $f = (X, Y) \in \mathbb{G}^2$, $\ell \in \mathbb{Z}_q$, $r \in \mathbb{Z}_q$, which defines $(c_1, c_2) = (g_1^r, g_2^r)$, the public evaluate algorithm computes $\mathsf{Pub}(f, \ell, c, r) = (X^{\ell} Y)^r$.

Priv: The private evaluate algorithm, on input $(c_1, c_2) \in \mathbb{G}^2$ and $\ell \in \mathbb{Z}_q$, computes $\mathsf{Priv}(sk, \ell, c) = c_1^{x_1\ell + y_1} c_2^{x_2\ell + y_2}$.

We verify that the above L-HPS satisfies our requirements.

μ's homomorphism: We have that, for any $t = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$ and $\Delta = (\delta_1, \delta_2, \eta_1, \eta_2) \in \mathbb{Z}_q^4$,

$$\mu(t + \Delta) = (g_1^{x_1+\delta_1} g_2^{x_2+\delta_2}, g_1^{y_1+\eta_1} g_2^{y_2+\eta_2}) = (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}) \cdot (g_1^{\delta_1} g_2^{\delta_2}, g_1^{\eta_1} g_2^{\eta_2}) = \mu(t) \cdot \mu(\Delta)$$

as we want.

Key homomorphism of Λ: We have that for any $t = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$, $\Delta = (\delta_1, \delta_2, \eta_1, \eta_2) \in \mathbb{Z}_q^4$, $\ell \in \mathbb{Z}_q^*$, $c = (c_1, c_2) \in \mathbb{G}^2$,

$$\Lambda_{t+\Delta}^{\ell}(c_1, c_2) = c_1^{(x_1+\delta_1)\ell + y_1 + \eta_1} \cdot c_2^{(x_2+\delta_2)\ell + y_2 + \eta_2} = c_1^{x_1\ell + y_1} c_2^{x_2\ell + y_2} \cdot c_1^{\delta_1\ell + \eta_1} c_2^{\delta_2\ell + \eta_2} = \Lambda_t^{\ell}(c_1, c_2) \cdot \Lambda_{\Delta}^{\ell}(c_1, c_2)$$

as we want.

μ's collision resistance: It is easy to show it from the discrete logarithm assumption on (g_1, g_2).

4 Construction from Tag-based Adaptive Trapdoor Relations

Kiltz, Mohassel, and O'Neill proposed a new notion for constructing public-key encryption, *(tag-based) adaptive trapdoor functions (T-ATDF)* [KMO10]. Roughly speaking, the tag-based trapdoor functions are adaptive if the T-ATDFs remain one-way $y = f_{\text{TAG}^*}(r)$ even if the adversary is allowed to access to an inversion oracle $f^{-1}_{\text{TAG} \neq \text{TAG}^*}(\cdot)$ on distinct tags.

Wee weakened this notion into *(tag-based) adaptive trapdoor relations (T-ATDR)* [Wee10]. In the TDFs, the trapdoor should invert the original. But, in ATDR definition, the sender and receiver shares the intermediate value s rather than the original randomness r.

4.1 (Tag-based) Adaptive Trapdoor Relations

Let us recall T-ATDR [Wee10]. Let $\mathcal{Y}$, $\mathcal{R}$, and $\mathcal{S}$ be finite sets. The space of tags is denoted by $\mathcal{TS}$. The secret- and public-key space is denoted by $\mathcal{T}$ and $\mathcal{F}$, respectively. The key space is denoted by $\mathcal{K}$. Let $F_f(\text{TAG}, \cdot) : \mathcal{S} \to \mathcal{Y}$ be a (tagged) injective function indexed by public information $f \in \mathcal{F}$ with tag $\text{TAG} \in \mathcal{TS}$.

Syntax. A tag-based adaptive trapdoor relation system $\mathsf{ATDR} = (\mathsf{Setup}, \mathsf{TrapGen}, \mathsf{Samp}, \mathsf{Inv}, \mathsf{G})$ consists of five algorithms:

- Setup is a setup algorithm that, on input 1^λ, outputs public parameters π, which define $\mathcal{Y}$, $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, $\mathcal{F}$, $\mathcal{TS}$, $\mathcal{K}$, and $\{F_f : \mathcal{S} \times \mathcal{TS} \to \mathcal{Y} \mid f \in \mathcal{F}\}$.
- $\mathsf{TrapGen}$ is a key-generation algorithm that, on input π, outputs a pair of keys $(f, t) \in \mathcal{F} \times \mathcal{T}$.
- Samp is a public sampling algorithm that, on input π, f, tag $\textsc{tag}$, and randomness $r \in \mathcal{R}$, outputs session randomness s and $y = F_f(\textsc{tag}, s)$.
- Inv is an inversion algorithm that, on input π, t, $\textsc{tag}$, and y, outputs $s = F_f^{-1}(\textsc{tag}, y)$.
- G is an extracting algorithm that, on input s, outputs $k \in \mathcal{K}$.

Security. Intuitively speaking, we say that tag-based ATDR ATDR is adaptively pseudorandom if any PPT adversary cannot distinguish $(y, \mathsf{G}(s))$ and (y, k), where $y = F_f^{\textsc{tag}^*}(s)$ and $k \leftarrow \mathcal{K}$ on the tag $\textsc{tag}^*$ chosen by the adversary at the beginning of the game, even if it is allowed to access the inversion oracle for any $\textsc{tag} \neq \textsc{tag}^*$.

Definition 4.1 (Adaptive Pseudorandomness). *Define experiment* $\mathsf{Expt}^{\mathsf{adapt\text{-}pr}}_{\mathsf{ATDR},\mathsf{A}}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: *Run the adversary with* 1^λ *and obtain* $\textsc{tag}^*$. *Generate* $\pi \leftarrow \mathsf{Setup}(1^\lambda)$, $(f^*, t^*) \leftarrow \mathsf{TrapGen}(\pi)$, *and* $(s^*, y^*) \leftarrow \mathsf{Samp}(\pi, f^*, \textsc{tag}^*)$. *Flip a coin* $b^* \leftarrow \{0, 1\}$. *Extract* $k_0 \leftarrow \mathsf{G}(s^*)$ *and generate* $k_1 \leftarrow \mathcal{K}$. *Run the adversary by feeding* (π, f^*, y^*, k_{b^*}).

Learning: *The adversary could query to the inversion oracle defined as follows:*

Inv: *It receives* $\textsc{tag}$ *and* y. *If* $\textsc{tag} = \textsc{tag}^*$ *then it returns* $\perp$. *Otherwise, it returns* $s \leftarrow \mathsf{Inv}(\pi, t^*, \textsc{tag}, y)$.

Finalization: *The adversary stops with output* b. *Output* 1 *if* $b = b^*$. *Output* 0 *otherwise.*

We define the advantage of A *as*

$$\mathsf{Adv}^{\mathsf{adapt\text{-}pr}}_{\mathsf{ATDR},\mathsf{A}}(\lambda) = \left|\Pr\left[\mathsf{Expt}^{\mathsf{adapt\text{-}pr}}_{\mathsf{ATDR},\mathsf{A}}(\lambda) = 1\right] - \frac{1}{2}\right|.$$

We say that ATDR *is adaptive pseudorandom if for any PPT adversary* A, *its advantage* $\mathsf{Adv}^{\mathsf{adapt\text{-}pr}}_{\mathsf{ATDR},\mathsf{A}}(\cdot)$ *is negligible.*

4.2 Wee's Additional Requirements

Wee [Wee12] defined two properties on ATDR.

One is Φ-key homomorphism. Intuitively speaking, if one can convert y under the public key $\mu(t)$ and tag $\textsc{tag}$ into y' under the derived public key $\mu(\phi(t))$ by ϕ and the same tag $\textsc{tag}$ keeping the seed s unchanged.

Definition 4.2 (Φ-key homomorphism [Wee12]). *We say that* ATDR *is* Φ*-key homomorphic if there exists a PPT algorithm* T *such that for all* $\phi \in \Phi$, *for any* π, t, $\textsc{tag}$, y, $\mathsf{Inv}(\pi, \phi(t), \textsc{tag}, y) = \mathsf{Inv}(\pi, t, \textsc{tag}, T(\pi, \phi, \textsc{tag}, y))$ *holds.*

This property is a weak variant of key malleability in [BC10] and key homomorphism in [AHI11].

The other is Φ-fingerprinting property defined as follows:

Definition 4.3 (Φ-fingerprinting [Wee12]). *Define experiment* $\mathsf{Expt}^{\mathsf{fp}}_{\mathsf{ATDR},\mathsf{A},\Phi}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: *Generate* $\pi \leftarrow \mathsf{Setup}(1^\lambda)$. *Run the adversary with* π *and receive* $\textsc{tag}^*$. *Generate* $(f^*, t^*) \leftarrow \mathsf{TrapGen}(\pi)$. *Generate* $(s, y) \leftarrow \mathsf{Samp}(\pi, f^*, \textsc{tag}^*)$. *Run the adversary by feeding* (π, f^*, t^*, y).

Finalization: *The adversary stops with output $\phi \in \Phi$. Output* 1 *if* $\mathsf{Inv}(\pi, \phi(t^*), \text{TAG}^*, y) \neq \perp$, $\phi(t^*) \neq t^*$, *and* $\mu(\phi(t^*)) = \mu(t^*)$. *Output* 0 *otherwise.*

We define the advantage of A *as*

$$\mathsf{Adv}^{\mathsf{fp}}_{\mathsf{ATDR},\mathsf{A},\Phi}(\lambda) = \Pr\left[\mathsf{Expt}^{\mathsf{fp}}_{\mathsf{ATDR},\mathsf{A},\Phi}(\lambda) = 1\right].$$

We say that ATDR *admits Φ-fingerprinting if for any PPT adversary* A*, its advantage* $\mathsf{Adv}^{\mathsf{fp}}_{\mathsf{ATDR},\mathsf{A},\Phi}(\cdot)$ *is negligible.*

4.3 Our Additional Requirements

$\mathcal{K}$'s commutativity: We say that ATDR is $\mathcal{K}$-commutative if $\mathcal{K}$ is a commutative group.

μ's homomorphism: We say that ATDR is μ-homomorphic if there exists homomorphism $\mu : \mathcal{T} \to \mathcal{F}$ such that TrapGen can be written in the form as follows: $t \leftarrow \mathcal{T}$, $f \leftarrow \mu(t)$. Key-generation function μ is a homomorphism from $\mathcal{T}$ to $\mathcal{F}$. Hereafter, we assume that $\mathcal{T}$ is an additive finite group.

μ's Φ-collision resistance: As in the previous definition on labeled hash functions, we require that μ is collision resistant with respect to Φ even if we know a trapdoor. See the following definition.

Definition 4.4 (μ's Φ-collision resistance). *Define experiment* $\mathsf{Expt}^{\mu\text{-coll}}_{\mathsf{ATDR},\mathsf{A},\Phi}(\lambda)$ *between adversary* A *and the challenger as follows:*

Initialization: *Generate* $\pi \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(f^*, t^*) \leftarrow \mathsf{TrapGen}(\pi)$. *Run the adversary by feeding* (π, f^*, t^*).

Finalization: *The adversary stops with output $\phi \in \Phi$. Output* 1 *if* $\phi(t^*) \neq t^*$ *and* $\mu(\phi(t^*)) = \mu(t^*)$. *Output* 0 *otherwise.*

We define the advantage of A *as*

$$\mathsf{Adv}^{\mu\text{-coll}}_{\mathsf{ATDR},\mathsf{A},\Phi}(\lambda) = \Pr\left[\mathsf{Expt}^{\mu\text{-coll}}_{\mathsf{ATDR},\mathsf{A},\Phi}(\lambda) = 1\right].$$

We say that ATDR *is μ is Φ-collision resistant if for any PPT adversary* A*, its advantage* $\mathsf{Adv}^{\mu\text{-coll}}_{\mathsf{ATDR},\mathsf{A},\Phi}(\cdot)$ *is negligible.*

We note that μ's Φ-CR property is stronger than Φ-finger printing.

4.4 Our Construction

Let ATDR be a tag-based ATDR system associative with $\mathcal{Y}, \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{F}, \mathcal{TS}, \mathcal{K}$, and $\{F_f : \mathcal{S} \times \mathcal{TS} \to \mathcal{Y} \mid f \in \mathcal{F}\}$. Let OTS = (ots.gen, ots.sign, ots.vrfy) be a one-time signature scheme.

We define our MAC(ATDR, OTS) as follows:

$\mathsf{Setup}(1^\lambda)$: $\pi \leftarrow \mathsf{Setup}_{\mathsf{ATDR}}(1^\lambda)$. Output public parameters π.

$\mathsf{KG}(\pi, \mathsf{H})$: Generate $(f, t) \leftarrow \mathsf{TrapGen}(\pi)$. (Here, we note that $f = \mu(t)$.) Choose $p \leftarrow \mathcal{K}$. Output $\kappa = (t, p)$.

$\mathsf{TAG}(\pi, \kappa, m)$: On input key $\kappa = (t, p)$ and message $m \in \{0, 1\}^*$,

1. compute $f \leftarrow \mu(t)$,
2. generate $(ovk, osk) \leftarrow \mathsf{ots.gen}(1^\lambda)$,
3. compute $(s, y) \leftarrow \mathsf{Samp}(\pi, f, ovk)$,
4. compute $c \leftarrow \mathsf{G}(s) + p$,
5. generate $\sigma \leftarrow \mathsf{ots.sign}(osk, (f, y, c, m))$,
6. set $\tau \leftarrow (ovk, y, c, \sigma)$, and
7. output tag τ.

$\mathsf{VRFY}(\pi, \kappa, m, \tau)$: On input key $\kappa = (t, p)$, message $m \in \{0, 1\}^*$, and tag $\tau = (ovk, y, c, \sigma)$,

1. compute $f \leftarrow \mu(t)$,
2. If $\mathsf{ots.vrfy}(ovk, (f, y, c, m)) = \mathsf{acc}$ then go next, otherwise, stop with outputting rej.
3. Compute $s \leftarrow \mathsf{Inv}(\pi, t, ovk, y)$.
4. If $p = c - \mathsf{G}(s)$ then output acc; otherwise, output rej.

4.5 Security

Theorem 4.1. *Let* ATDR *be a $\mathcal{K}$-commutative tag-based ATDR system. Suppose $\mathcal{T} \times \mathcal{K}$ be a commutative group and let $\Phi^+ = \{\phi_{\delta,\eta} : (t, p) \mapsto (t + \delta, p + \eta) \mid (\delta, \eta) \in \mathcal{T} \times \mathcal{K}\}$. Suppose that* ATDR *is adaptively pseudorandom and key homomorphic. Moreover, suppose that* ATDR *is μ-homomorphic and μ-collision resistant with respect to Φ^+. Then,* MAC(ATDR, OTS) *is Φ^+*-UF-RK-CMVA *secure.*

The proof is obtained by combining those of Dodis et al. and Wee. We employ game-based proof. Hereafter, without loss of generality, we suppose that the adversary never query a verification query (ϕ, m, τ) if τ is generated by RK-Tag on query (ϕ, m). The adversary specifies ϕ by $(\delta, \eta) \in \mathcal{T} \times \mathcal{K}$.

$\mathsf{Expt}_{\mathrm{real}}$. This is the original experiment. Hence, we have that

$$\mathsf{Adv}^{\mathsf{uf\text{-}rk\text{-}cmva}}_{\mathsf{MAC,A,}\Phi}(\lambda) = \Pr[\mathsf{Expt}_{\mathrm{real}} = 1].$$

$\mathsf{Expt}'_{\mathrm{real}}$. We next eliminate unexpected $\phi_i = (\delta_i, \eta_i)$ which makes a bad collision. For simplicity, we let $(\delta_0, \eta_0) = (0, 0)$. The oracles reject if $t^* + \delta_i \neq t^* + \delta_j$ and $\mu(t^* + \delta_i) = \mu(t^* + \delta_j)$ for $\delta_i \neq \delta_j$.

Claim. There exists a PPT algorithm B that $\left|\Pr[\mathsf{Expt}_{\mathrm{real}} = 1] - \Pr[\mathsf{Expt}'_{\mathrm{real}} = 1]\right| \leq \mathsf{Adv}^{\mu\text{-}\mathsf{coll}}_{\mathsf{ATDR,B,}\Phi^+}(\lambda)$.

Proof. The two games differ when the adversary queries (δ_i, η_i) and (δ_j, η_j) to the oracles satisfying $\delta_i \neq \delta_j$, $t^* + \delta_i \neq t^* + \delta_j$, and $\mu(t^* + \delta_i) = \mu(t^* + \delta_j)$. But, it is obvious that this contradicts μ's collision-resistance property. □

$\mathsf{Expt}''_{\mathrm{real}}$. We next eliminate reuse of *ovk*. At the initialization phase, the challenger picks up $(ovk_i, osk_i) \leftarrow \mathsf{KG}(1^\lambda)$ for $i \in [Q_T]$ to use them in the oracle RK-Tag. If the oracle RK-Vrfy receives a query including $ovk = ovk_i$ then the oracle rejects it anyway.

Claim. There exists a PPT algorithm B that $\left|\Pr[\mathsf{Expt}'_{\mathrm{real}} = 1] - \Pr[\mathsf{Expt}''_{\mathrm{real}} = 1]\right| \leq Q_T \cdot \mathsf{Adv}^{\mathsf{seuf\text{-}otcma}}_{\mathsf{OTS,B}}(\lambda)$.

Proof. Let $\phi_i = (\delta_i, \eta_i)$ and m_i be the i-the RK-tagging query and let $\tau_i = (ovk_i, y_i, c_i, \sigma_i)$ be the answer to the query. The difference occurs when, the adversary queries to the RK-verification oracle $\phi = (\delta, \eta)$, m, and $\tau = (ovk, y, c, \sigma)$ such that $ovk = ovk_i$ for some $i \in [Q_T]$ which accepts the RK-verification oracle. Let us check the difference:

- $(\delta, \eta) = (0, 0)$: In this case, we have $m \neq m_i$ or $(y, c, \sigma) \neq (y_i, c_i, \sigma_i)$; otherwise, it makes no differences. In the both cases, we obtain a forgery.
- $\delta \neq \delta_i$: In this case, we have that $\mu(t + \delta_i) = f_i \neq f' = \mu(t + \delta)$, since we already cut this event. Therefore, σ is a forgery of a new message (f', y, c, m) and this contradicts sEUF-OTCMA-security of OTS.
- $\delta = 0$ but $\eta \neq 0$: In this case, we have the sub-cases as follows:
  - $(m, y, c, \sigma) = (m_i, y_i, c_i, \sigma_i)$: Notice that in this case, the RK-verification query is rejected. This is because $p + \eta$, which is correct, never equals to $p + \eta$, which cannot pass 4-th check.
  - Otherwise, we have the one of m, y, c, or σ differs from the one in the i-th query. therefore, we get a forgery.

Summarizing the above, this contradicts to the sEUF-OTCMA security of the one-time signature scheme OTS. □

Expt_i *and* Expt'_i *for* $i \in [0, Q_T]$. We next change the RK-tagging oracle as follows:

Expt_i: On the first i queries, RK-TAG replaces p with 0. The rest $Q_T - i$ queries, it does the original.

Expt_i': On the first $i-1$ queries, RK-TAG replaces p with 0. On the i-th query, it used c chosen uniformly at random. The rest $Q_T - i$ queries, it does the original.

For the summary, see the table.

	The answer of RK-TAG on the i-th query
Expt_i	$c \leftarrow \mathsf{G}(s) + p + \eta$
Expt'_i	$c \leftarrow \mathcal{K}$
Expt_{i+1}	$c \leftarrow \mathsf{G}(s) + \eta$

We note that $\mathsf{Expt}_0 = \mathsf{Expt}'_0 = \mathsf{Expt}'_{\mathrm{real}}$.

In the following, we claim that Expt_i and Expt'_i are computationally indistinguishable and Expt'_i and Expt_{i+1} are also.

Claim. For all $i \in [Q_T]$, there exists a PPT algorithm B that $\left|\Pr[\mathsf{Expt}_i = 1] - \Pr[\mathsf{Expt}'_i = 1]\right| \leq \mathsf{Adv}^{\mathsf{adapt\text{-}pr}}_{\mathsf{ATDR},\mathsf{B}}(\lambda)$.

Proof. We construct an adversary B for the $\mathsf{adapt\text{-}pr}$ game from the adversary A which distinguishes Expt_{i-1} and Expt_i as follows:

INITIALIZATION: On input 1^λ, B generates $(ovk^*, osk^*) \leftarrow \mathsf{Gen}(1^\lambda)$ and declares ovk^* as a target tag. It then receives (π, f^*, y^*, k_{b^*}) from its challenger, where $\pi \leftarrow \mathsf{Setup}(1^\lambda)$, $(f^*, t^*) \leftarrow \mathsf{TrapGen}(\pi)$, $(s^*, y^*) \leftarrow \mathsf{Samp}(\pi, f^*, ovk^*)$, $b^* \leftarrow \{0,1\}$, $k_0 \leftarrow \mathsf{G}(s^*)$, and $k_1 \leftarrow \mathcal{K}$. It chooses $p \leftarrow \mathcal{K}$ uniformly at random. Run the adversary A with π.

LEARNING PHASE: B simulates the oracles as follows:

- RK-TAG: It receives $\phi = (\delta, \eta)$ and m.
  - (On the first $i-1$ queries:) compute $f' = f^* \cdot \mu(\delta) = \mu(t^* + \delta)$ from μ's homomorphism, generate $(ovk, osk) \leftarrow \mathsf{ots.gen}(1^\lambda)$, compute $(s, y) \leftarrow \mathsf{Samp}(\pi, f', ovk)$, compute $c \leftarrow \mathsf{G}(s) + \eta$, generate $\sigma \leftarrow \mathsf{ots.sign}(osk, (f', y, c, m))$, set $\tau \leftarrow (ovk, y, c, \sigma)$. and return τ to A.
  - compute $f' = f^* \cdot \mu(\delta) = \mu(t^* + \delta)$ from μ's homomorphism, compute $y' \leftarrow T(\pi, \phi(t^*), ovk^*, y^*)$, compute $c \leftarrow k_{b^*} + \eta$, generate $\sigma \leftarrow \mathsf{ots.sign}(osk^*, (f', y, c, m))$, set $\tau \leftarrow (ovk^*, y, c, \sigma)$. and return τ to A.
  - (On the last $Q_T - i$ queries:) compute $f' = f^* \cdot \mu(\delta) = \mu(t^* + \delta)$ from μ's homomorphism, generate $(ovk, osk) \leftarrow \mathsf{ots.gen}(1^\lambda)$, compute $(s, y) \leftarrow \mathsf{Samp}(\pi, f', ovk)$, compute $c \leftarrow \mathsf{G}(s) + p + \eta$, generate $\sigma \leftarrow \mathsf{ots.sign}(osk, (f', y, c, m))$, set $\tau \leftarrow (ovk, y, c, \sigma)$. and return τ to A.
- RK-VRFY: B receives ϕ, m, and $\tau = (ovk, y, c, \sigma)$. If $ovk = ovk^*$, then it returns $\perp$. Else it can use its inversion oracle since $ovk \neq ovk^*$: B computes $f' = f \cdot \mu(\delta)$ and verifies $\mathsf{ots.vrfy}(ovk, (f', y, c, m))$. If it passes the verification, B computes $y' = T(\pi, \eta, ovk, y)$ and query y' to its decryption oracle with tag ovk. Then, B receives $s' = \mathsf{Inv}(\pi, t+\delta, ovk, y)$ Finally, B checks $c = \mathsf{G}(s') + p + \eta$ or not.

FINALIZATION: Finally, A outputs m and $\tau = (ovk, y, c, \sigma)$. Since $ovk \neq ovk^*$ again, B can check the validity of m and τ as in the simulation of RK-VRFY. If A wins, B outputs 1. Otherwise, B outputs 0.

By the definition of B, B perfectly simulates Expt_i if $b^* = 0$ and Expt'_i if $b^* = 1$, since c^* in τ_i is uniformly at random. □

Claim. For all $i \in [0, Q_T - 1]$, there exists a PPT algorithm B that $\left|\Pr[\mathsf{Expt}'_i = 1] - \Pr[\mathsf{Expt}_{i+1} = 1]\right| \leq \mathsf{Adv}^{\mathsf{adapt\text{-}pr}}_{\mathsf{ATDR},\mathsf{B}}(\lambda)$.

Since the proof is the same as the previous proof, we omit it.

$\mathsf{Expt}_{\mathrm{final}}$: In this final game, the oracle RK-VRFY rejects all queries. Hence, the adversary has no chance to obtain any information on k in the learning phase by RK-tagging oracle. Therefore, it gains information only from RK-VRFY.

We show the following two claims.

Claim. We have that $\left|\Pr[\mathsf{Expt}_{Q_T} = 1] - \Pr[\mathsf{Expt}_{\mathrm{final}} = 1]\right| \leq Q_V/\#\mathcal{K}$.

Proof. The games differ if the adversary queries ϕ_j, m_j, and $\tau_j = (ovk_j, y_j, c_j, \sigma_j)$ to the RK-verification oracle, which is correct in Expt_{Q_T}. Notice that even in Expt_{Q_T}, the adversary cannot obtain information of p from the RK-tagging oracle. Therefore, the first RK-verification query is correct with probability at most $1/\#\mathcal{K}$. The above upper bound follows from the hybrid argument. □

Claim. We have that $\Pr[\mathsf{Expt}_{\mathrm{final}} = 1] \leq 1/\#\mathcal{K}$.

Proof. In this game, the challenger does not give the adversary *any information about p*. Therefore, the advantage is at most $1/\#\mathcal{K}$.

4.6 Instantiation from Factoring

Let us briefly recall the properties of the signed quadratic residues [HK09a,HK09b]. Fix a Blum integer $N = PQ$ for safe primes $P = 2p + 1$ and $Q = 2q + 1$ such that $P, Q \equiv 3 \pmod 4$. Let $\mathbb{J}_N$ be a set of elements whose Jacobi symbol is 1. For $x \in \mathbb{Z}_N$, let $|x| \in \mathbb{Z}_N$ be the absolute value of x, where x is in $\{-(N-1)/2, \dots, (N-1)/2\}$. Let $\mathbb{QR}_N$ be the quadratic residue group. Notice that $-1 \notin \mathbb{QR}_N$. Finally, we define

$$\mathbb{QR}_N^+ = \{|x| \mid x \in \mathbb{QR}_N\}.$$

This is the signed quotient group $\mathbb{QR}_N^+ = \mathbb{J}_N/(\pm 1)$, a cyclic group of order $(p-1)(q-1)/4$, and efficiently recognizable by computing Jacobi symbol, since $\mathbb{QR}_N^+ = \mathbb{J}_N^+ := \{|x| \mid x \in \mathbb{J}_N\}$. Let g be a random generator of $\mathbb{QR}_N^+$.

Definition 4.5 (Factoring assumption). *For an adversary,* A*, we define its advantage as* $\mathsf{Adv}^{\mathsf{fact}}_{\mathsf{InstG},\mathsf{A}}(\lambda) = \Pr[(N, p, q) \leftarrow \mathsf{InstG}(1^\lambda) : \mathsf{A}(N) \in \{p, q\}]$. *We say that* A (t, ϵ)*-factors composite integers if* A *runs in time t and its advantage is larger than ϵ. We say that the factoring assumption (w.r.t.* InstG*) holds if for any PPT adversary* A*, its advantage is negligible in λ.*

We suppose that InstG always output a Blum integer N, that is, $P, Q \equiv 3 \pmod 4$ and they are safe primes.

Tag-based ATDR. For easiness of notation, we let $\Omega = 2^\omega$ and $\Lambda = 2^\lambda$. The space of labels is $[0, \Omega - 1]$ and the space of extracted key is $[0, \Lambda - 1]$.

$\mathsf{Setup}(1^\lambda)$: Generate two strong primes P, Q whose bit lengths are λ. Compute $N = PQ$ and generate a random generator g of $\mathbb{QR}_N^+$. Output $\pi = (N, g)$.

$\mathsf{TrapGen}(\pi)$: On input (N, g), choose $t \leftarrow [(N-1)/4]$. Compute $f \leftarrow g^{\Lambda\Omega t}$ and output (f, t).

$\mathsf{Samp}(f, \textsc{tag}; r)$: On randomness $r \leftarrow ?$, compute $(s, u) \leftarrow (g^{\Omega r}, g^{\Lambda\Omega r})$. Compute $w = (f \cdot g^{\textsc{tag}})^r$. Output s and $y = (u, w)$.

$\mathsf{Inv}(t, \textsc{tag}, y)$: On $y = (u, w)$,

1. verify $u, w \in \mathbb{QR}_N^+$; otherwise, output $\perp$ and stop;
2. verify $w^{\Lambda\Omega} = u^{\textsc{tag}+\Lambda\Omega t}$; otherwise, output $\perp$ and stop;
3. compute $a, b, c \in \mathbb{Z}$ such that $2^c = a \cdot \textsc{tag} + b \cdot \Lambda\Omega$ in $\mathbb{Z}$,
4. compute $s' \leftarrow (w^a \cdot u^{b-a\cdot t})^{2^{\textsc{tag}-c}}$,
5. and output s'.

$\mathsf{G}(s)$: This is the Blum-Blum-Shub PRG. On input $s \in \mathbb{QR}_N^+$, output

$$\left(\mathsf{lsb}_N(s), \mathsf{lsb}_N(s^2), \mathsf{lsb}_N(s^4), \ldots, \mathsf{lsb}_N(s^{2^{\lambda-1}})\right) \in \{0,1\}^\lambda,$$

where $\mathsf{lsb}_N(u)$ is the least significant bit of $u \in [-(N-1)/2, (N-1)/2]$.

Let us verify that the above tag-based ATDR system satisfies Wee's and our requirements.

Φ^+-key homomorphism: Wee observed this property [Wee12]. For completeness, we prove it again. Let us fix $\delta \in \mathbb{Z}$, $t \in [(N-1)/4]$, and $\textsc{tag} \in [0, \Lambda - 1]$. We have that $F_{\mu(t)}(\ell, s) = (u, w) = (g^{\Lambda\Omega r}, g^{\Lambda\Omega t r} \cdot g^{\textsc{tag} r})$, where $s = g^{\Omega r}$. We have that

$$F_{\mu(t+\delta)}(\ell, s) = (g^{\Lambda\Omega r}, (g^{\Lambda\Omega(t+\delta)} \cdot g^{\textsc{tag}})^r) = (g^{\Lambda\Omega r}, (g^{\Lambda\Omega t} \cdot g^{\textsc{tag}})^r \cdot g^{\Lambda\Omega r\delta}) = (u, w \cdot u^\delta)$$

as we want. Hence, T is defined as follows: On input $\pi = (N, g)$, $\delta \in \mathbb{Z}$, $f \in \mathbb{QR}_N^+$, $\textsc{tag} \in [0, \Lambda - 1]$, and $y = (u, w) \in \mathbb{QR}_N^+$, T outputs $(u, w \cdot u^\delta)$.

μ's Φ^+-collision resistance: Suppose that there exists non-trivial function $\phi : [(N-1)/4] \to [-N, N]$ which yields $f = g^{\Lambda\Omega t} = g^{\Lambda\Omega\cdot\phi(t)} \bmod N$. Since the order of g is $\phi(N)/4 = pq$ and $\Lambda\Omega = 2^{\lambda+\ell}$ is coprime with pq, this implies $t \equiv \phi(t) \pmod{pq}$. Thus, $\phi(t) - t$ reveals non-trivial divisor of pq and we can factor N. This contradicts the factoring assumption (on the Blum integers).

$\mathcal{K}$'s commutativity: If we treat $\{0,1\}^\lambda$ as $\mathrm{GF}(2)^\lambda$, it is a commutative group.

μ's homomorphism: For any $\Delta \in \mathbb{Z}$ and $t \in [(N-1)/4]$, we have that

$$\mu(t + \Delta) = g^{\Lambda\Omega(t+\Delta)} = g^{\Lambda\Omega t} \cdot g^{\Lambda\Omega\Delta} = \mu(t) \cdot \mu(\Delta).$$

4.7 Instantiation from DBDH

Let $\mathsf{GroupG}_{\mathsf{DBDH}}$ be a PPT algorithm that on input a security parameter 1^λ outputs $(\mathbb{G}, \mathbb{G}_T, e, q, g)$ such that; $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of prime order q, g is a generator of $\mathbb{G}$, and a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies

- (Bilinear:) for any $g, h \in \mathbb{G}$ and any $a, b \in \mathbb{Z}_q$, $e(g^a, h^b) = e(g, h)^{ab}$,
- (Non-degenerate:) $e(g, g)$ has order q in $\mathbb{G}_T$, and
- (Efficiently computable:) e is efficiently computable.

Definition 4.6 (DBDH assumption). *For an adversary,* A*, we define its advantage as*

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{dbdh}}_{\mathsf{GroupG}_{\mathsf{DBDH}},\mathsf{A}}(\lambda) \\
&\quad = \Pr[(\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathsf{GroupG}(1^\lambda), a, b, c \leftarrow \mathbb{Z}_q : \mathsf{A}(\mathbb{G}, \mathbb{G}_T, q, g, e, g^a, g^b, g^c, e(g,g)^{abc}) = 1] \\
&\quad\quad - \Pr[(\mathbb{G}, \mathbb{G}_T, q, g, e) \leftarrow \mathsf{GroupG}(1^\lambda), a, b, c, d \leftarrow \mathbb{Z}_q : \mathsf{A}(\mathbb{G}, \mathbb{G}_T, q, g, e, g^a, g^b, g^c, e(g,g)^{d}) = 1]
\end{aligned}$$

We say that A *(t, ϵ)-solves the DBDH problem if* A *runs in time t and its advantage is larger than ϵ. We say that the DBDH assumption (w.r.t.* GroupG*) holds if for any PPT adversary* A*, its advantage is negligible in λ.*

Tag-based ATDR. The space of labels is $\mathbb{Z}_q^*$. $\mathbb{G}_T$ is a key space.

$\mathsf{Setup}(1^\lambda)$: Generate bilinear pairing group $(\mathbb{G}, \mathbb{G}_T, q, e, g)$. Choose $v, h \leftarrow \mathbb{G}$. Output public parameters $\pi = (\mathbb{G}, \mathbb{G}_T, q, e, g, v, h)$.

$\mathsf{TrapGen}(\pi)$: On input $\pi = (\mathbb{G}, \mathbb{G}_T, q, e, g, v, h)$, choose $t \leftarrow \mathbb{Z}_q$ and compute $f \leftarrow g^t$. Output (f, t).

$\mathsf{Samp}(f, \textsc{tag}; r)$: On randomness $r \in \mathbb{Z}_q$, compute $(s, u) \leftarrow (v^r, g^r)$ and $w = (f \cdot v^{\textsc{tag}})^r$. Output s and $y = (u, w)$.

$\mathsf{Inv}(t, \textsc{tag}, y)$: On $y = (u, w)$,
1. verify $u, w \in \mathbb{G}$; if not, output $\perp$ and stop;
2. compute $s' \leftarrow (w \cdot u^{-t})^{\textsc{tag}^{-1}}$,
3. verify $e(g, s') = e(v, u)$; if not, output $\perp$ and stop;
4. output s'.

$\mathsf{G}(s)$: On input $s \in \mathbb{G}$, output $e(s, h) \in \mathbb{G}_T$.

Let us verify that the above tag-based ATDR system satisfies Wee's and our requirements.

Φ^+-key homomorphism: Wee [Wee12] showed Φ^+-key homomorphism. For completeness, we prove it again. For any $\Delta \in \mathbb{Z}_q$, $t \in \mathbb{Z}_q$, and $\textsc{tag} \in \mathbb{Z}_q$, we have $F_{\mu(t)}(\ell, s) = (u, w) = (g^r, (f \cdot v^{\textsc{tag}})^r) = (g^r, (g^t v^{\textsc{tag}})^r)$, where we set $s = v^r$. We have that

$$F_{\mu(t+\Delta)}(\ell, s) = (g^r, (g^{t+\Delta} \cdot v)^r) = (g^r, (g^t v)^r \cdot g^{\Delta r}) = (u, w \cdot u^{\Delta})$$

as we want.

μ's Φ^+-collision resistance: Suppose that there exists non-trivial function $\phi : \mathbb{Z}_q \to \mathbb{Z}_q$ which yields $f = g^t = g^{\phi(t)}$. This implies $t = \phi(t)$.

$\mathcal{K}$'s commutativity: $\mathbb{G}_T$ is a multiplicative commutative group.

μ's homomorphism: For any $\Delta \in \mathbb{Z}_q$ and $t \in \mathbb{Z}_q$, we have that

$$\mu(t + \Delta) = g^{t+\Delta} = g^t \cdot g^{\Delta} = \mu(t) \cdot \mu(\Delta).$$

References

AHI11. Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In Bernard Chazelle, editor, *ICS 2011*, pages 45–60. Tsinghua University Press, 2011. The full version is available at `http://eprint.iacr.org/2010/544`.

BC10. Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Heidelberg, 2010.

BCM11. Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 486–503. Springer, Heidelberg, 2011. The full version is available at `http://eprint.iacr.org/2011/252`.

BGM04. Mihir Bellare, Oded Goldreich, and Anton Mityagin. The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309, 2004. Available at `http://eprint.iacr.org/2004/319`.

BK03. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, 2003.

BPT12. Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. RKA security beyond the linear barrier: IBE, encryption and signatures. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 331–348. Springer, Heidelberg, 2012. The full version is available at `http://eprint.iacr.org/2012/514`.

BR13. Rishiraj Bhattacharyya and Arnab Roy. Secure message authentication against related-key attack, 2013. To appear in FSE 2013.

BWM99. Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In Hideki Imai and Yuliang Zheng, editors, *PKC '99*, volume 1560 of *LNCS*, pages 154–170. Springer, Heidelberg, 1999.

BSW06. Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 229–240. Springer, Heidelberg, 2006.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, 2002.

DKPW12. Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 355–374. Springer, Heidelberg, 2012.

GOR11. Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In Yuval Isahi, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 182–200. Springer, Heidelberg, 2011. The full version is available at `http://eprint.iacr.org/2011/233`.

HK09a. Dennis Hofheinz and Eike Kiltz. The group of signed quadratic residues and applications. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 637–653. Springer, Heidelberg, 2009.

HK09b. Dennis Hofheinz and Eike Kiltz. Practical chosen ciphertext secure encryption from factoring. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 313–332. Springer, Heidelberg, 2009.

HWLZ08. Qiong Huang, Duncan S. Wong, Jin Li, and Yi-Ming Zhao. Generic transformation from weakly to strongly unforgeable signatures. *Journal of Computer Science and Technology*, 23(2):240–252, March 2008. A preliminary version appeared in *ACNS 2007*, 2007. See also `http://eprint.iacr.org/2006/346`.

KMO10. Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 673–692. Springer, Heidelberg, 2010.

MS04. Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *Designs, Codes and Cryptography*, 33(3):261–274, November 2004.

NR04. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, 2004. A preliminary version appeared in *FOCS '97*, 1997.

NRR02. Moni Naor, Omer Reingold, and Alon Rosen. Pseudorandom functions and factoring. *SIAM Journal on Computing*, 31(5):1383–1404, 2002. A preliminary version appeared in *STOC 2000*, 2000.

SPW07. Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In Masayuki Abe, editor, *CT-RSA 2007*, volume 4377 of *LNCS*, pages 357–371. Springer, Heidelberg, 2007.

TOO08. Isamu Teranishi, Takuro Oyama, and Wakaha Ogata. General conversion for obtaining strongly existentially unforgeable signatures. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 91-A(1):94–106, 2008. A preliminary version appeared in *INDOCRYPT 2006*, 2006.

Wee10. Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 314–332. Springer, Heidelberg, 2010.

Wee12. Hoeteck Wee. Public key encryption against related key attacks. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 262–279. Springer, Heidelberg, 2012.