A preliminary version of this paper appears in the proceedings of the 20th Australasian Conference on Information Security and Privacy (ACISP 2015), DOI: 10.1007/978-3-319-19962-7_12. This is the full version.

Linkable Message Tagging: Solving the Key Distribution Problem of Signature Schemes

Felix Günther[1] and Bertram Poettering[2]

[1] Cryptoplexity Group, Technische Universität Darmstadt, Germany
guenther@cs.tu-darmstadt.de
[2] Foundations of Cryptography, Ruhr-Universität Bochum, Germany
www.foc.rub.de

Abstract. Digital signatures are one of the most extensively used cryptographic primitives today. It is well-understood that they guarantee practical security only if the corresponding verification keys are distributed authentically; however, arguably, satisfying solutions for the latter haven't been found yet, or at least aren't in large-scale deployment. This paper introduces a novel approach for cryptographic message authentication where this problem does not arise: A *linkable message tagging* scheme (LMT) identifies pairs of messages and accompanying authentication tags as related if and only if these tags were created using the same secret key. In other words, in contrast to signature schemes, our primitive does not aim at detecting whether individually considered messages originate from an explicitly specified entity, but instead decides whether all messages from a given collection originate from the same (possibly anonymous) source. The appealing consequence is that our primitive fully avoids public keys and hence elegantly sidesteps the key distribution problem of signature schemes.

As an interesting application of LMT we envision an email authentication system with minimal user interaction. Email clients could routinely generate a secret LMT key upon their first invocation, and then equip all outgoing messages with corresponding tags. On the receiver's side, client software could automatically verify for incoming messages whether they indeed originate from the same entity as previously or subsequently received messages with identical sender address. Although this form of authentication does not provide as strong guarantees of message's origin as signature schemes would do, we do believe that trading the apparently discouraging obstacles implied by the authentic distribution of signature verification keys for the assumption that an attacker does not forge *every* message exchanged between parties is quite attractive.

As technical contributions we formalize the notions of LMT and its (more efficient) variant CMT (*classifiable message tagging*), including corresponding notions of unforgeability. For both variants we propose a range of provably secure constructions, basing on different hardness assumptions, with and without requiring random oracles.

Keywords: Message authentication, key distribution problem, message tagging, digital signatures

1 Introduction

Digital signature schemes. Digital signatures are an omnipresent cryptographic primitive in today's security landscape. Most prominently they find application as building blocks in message authentication, entity authentication, and in establishing trust relations in hierarchical public key infrastructures (PKIs). The first formal treatment of the security of signature schemes was given in 1984 when Goldwasser, Micali, and Rivest coined the notion of *existential unforgeability*, requiring that it be infeasible for any adaptive forging adversary to come up with a valid signature on a fresh message, even in the presence of a signing oracle [20, 21]. Despite the apparent strength of this notion, research effort in the past 30 years resulted in a wide range of efficient and provably secure constructions available today [35,9,8,47]. Due to their inalienability, signature schemes are also represented in many international standards, including IEEE P1363, ANSI X9.62, FIPS 186-4, and diverse IETF RFCs.

Public key infrastructures. A remaining challenge for the practical security of signature schemes is the authentic distribution of verification keys: Trustworthy authentication of messages or entities via signatures seems out of reach if assurances on the genuineness of available verification keys cannot be provided. Indeed, if the adversary manages to replace real verification keys by keys of its own, all security is lost.

A variety of proposals to resolve this challenge exist. As a prime example, hierarchical PKIs like those building on the X.509 standard [28] provide certificate-based attestation that a given public key belongs to a certain real-world user. Here, a logical link between a root certification authority (CA) and the user is established, consisting of a chain of (sub-)certificates that are issued by a set of (sub-)CAs; each of the latter vouches only for the identity of the respective subordinate entity.

Social PKIs, like the 'web of trust' underlying OpenPGP [13], constitute an alternative approach towards the authentic distribution of cryptographic keys. Here, putting trust into centralized CAs is not required; instead, users establish trust relationships by deciding on the genuineness of keys with the help and judgment of 'socially close' other users. The result (e.g., 'key is fully trustworthy') is encoded together with the considered key, authenticated with the own signing key, and uploaded to so-called key servers, ready for the retrieval by other users.

Practical obstacles in email authentication. On first sight the two discussed approaches towards the authentic distribution of cryptographic keys appear sound and reliable. Moreover, when considering authentic email communication, software implementations based on the S/MIME (i.e., X.509) and OpenPGP standards are freely available as plugins for all modern mail clients. However, in practice, large-scale deployment of email authentication for individuals, i.e., outside of organizations, fails to appear. Indeed, only a negligible fraction of Internet users secures their email correspondence using cryptography. Partly responsible for this might be the following technical and social obstacles:

- In respect to hierarchical PKIs: a practically unmanageable number of root CAs (current email clients have built-in databases with hundreds of root certificates), unclear trust relations to the CAs (severe security incidents at some CAs have been reported recently, see the compromises of DigiNotar in 2011 [18], TURKTRUST in 2013 [46], and National Informatics Centre of India in 2014 [22]), unclear authentication procedures during certificate requests (some CAs content themselves with requiring that certificate requesters have read-access to emails sent to their alleged email addresses, other CAs have more stringent policies), and unclear revocation procedures.
- In respect to social PKIs: barriers introduced by time-consuming and error-prone authentication procedures (e.g., in 'key-signing parties' where physical identity documents like passports are meant to be mutually verified), complicated user guidance in encryption software [49], and the privacy problem introduced by publicly revealing social relationships on the key servers.

To summarize, all currently established approaches towards strong email authentication come with specific usability barriers that seem insurmountable by the broader public in practice.

A novel approach: history-based message authentication. Assume a setting in which two users, Alice and Bob, routinely communicate via email and at some point have accumulated a communication history of, say, one year. Assume further that an adversary wants to intervene in this communication by injecting her own messages and making Bob accept them as originating from Alice. Such an attack could clearly be recognized and thwarted if Alice would sign all her outgoing messages and Bob would validate all incoming ones using an authentic copy of Alice's public key; however, motivated by the discussion above, in order to keep our considerations consistent with practical user behavior, in the following we abstain from assuming that Alice and Bob actively exchange any verification keys or follow any comparable explicit setup routine.

Our new concept of *history-based message authentication* builds on the idea that the strong authentication of regularly occurring message transmissions can be boot-strapped from a single authentic delivery (even if it is not known which out of many transmissions the authentic one is). These preconditions are met in the context of email communication where, arguably, a large fraction of messages reaches its destination in unmodified form. As we will elaborate, our new paradigm of authentication does not involve any kind of interaction between computer and user (except, possibly, if identified forgeries shall be reported), and in particular does not require an explicit exchange of verification keys. As our method completely side-steps the obstacles of key distribution discussed above, we envision that our new authentication model could contribute to make email authentication practically accessible to the masses.

Theoretical contributions. We explore the concept of history-based message authentication by introducing the notion of *linkable message tagging (LMT)*. Briefly, such schemes allow users holding a *tagging key* to equip given messages with corresponding *tags*; a dedicated *linking predicate* can then be used to decide whether any two message-tag pairs originate from the same (anonymous) sender, i.e., were created using the same key. Importantly, and in contrast to signature schemes, users of LMT schemes do not have explicit identities, e.g., in the form of public keys. However, if required, LMT schemes can be 'upgraded' to the functionality and security properties of signature schemes by authentically exchanging a specific reference token (that takes precisely the role of a verification key). Observe that this can be done at any time, even after first LMT tags have been exchanged, and seems particularly helpful to resolve cases where LMT schemes identify forged messages.

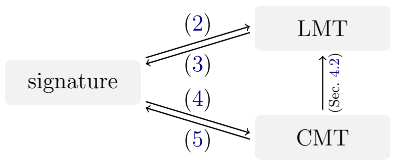
An interesting subclass of LMT is given by *classifiable message tagging (CMT)*. Corresponding schemes are optimized for situations where messages shall be automatically *classified* according to their origin. More concretely, in CMT schemes, a dedicated algorithm computes for messages-tag pairs a corresponding *class identifier*; the LMT linking predicate then reports matching origins if and only if the respective class identifiers match. As we argue below, this extended functionality is quite attractive in applications like email authentication.

To model security of LMT and CMT, we formalize notions of *unforgeability*: Intuitively, schemes are unforgeable if no adversary can create, for any message of its choosing, a tag that is identified by the scheme as originating from the same source as a genuinely crafted message-tag pair.

Practical contributions. On the constructive side we unveil tight connections between the new primitives on the one hand and signature schemes on the other (this might be surprising at first sight, as the new primitives *per se* do not involve public keys). These connections not only allow for generic constructions of LMT and CMT schemes from signature schemes and vice versa, but also explain notable similarities between our constructions and some well-known signature schemes.

We additionally put forward a couple of direct constructions of LMT and CMT schemes, with and without random oracles, based on different hardness assumptions. An overview over our direct constructions and the (less efficient) generic signature-based constructions is given in Figure 1. For details see the corresponding sections of this paper.

We complete our studies of practical LMT and CMT schemes by considering the real-world context of email authentication, in particular the applicability of our concepts to the OpenPGP and S/MIME frameworks. We anticipate here that specific properties of the structure of S/MIME-formatted emails immediately allow CMT functionality. We stress that this holds



Type	Scheme	Setting	Assumption	Model
LMT	BLS-LMT (1)	Pairing	CDH	ROM
CMT	Waters-CMT (6)	Pairing	CDH	CRS
CMT	Schnorr-CMT (7)	Cyclic group	DLP	ROM
LMT	Sig-LMT (2)	as signature	euf-cma	STD
CMT	Sig-CMT (4)	as signature	euf-cma	STD

Fig. 1. The figure on the left shows the conceptual relationship between signature schemes, LMT schemes, and CMT schemes. The table on the right gives an overview over the proposed direct (top) and generic (bottom) LMT and CMT constructions, indicating whether proofs are in the standard (STD), random oracle (ROM), or common reference string (CRS) model. The numbers in parentheses indicate the respective constructions.

without involving code adaption on the sender's side or setup assumptions like the existence of a certification infrastructure.

Envisioned application: automated email authentication. We consider the concept of linkable message tagging valuable for implementing easy-to-use and fully-automated cryptographic authentication of email communication; notably, in contrast to current approaches based on signature schemes and certificates, LMT and CMT do not rely on trusted third parties or on verification keys exchanged a priori. More concretely, we envision that email client software would automatically setup tagging keys upon its first execution, equip all outgoing emails with corresponding LMT tags, and visually group together incoming emails according to their origin (e.g., by color or position) without requiring any user interaction. By consequence, adversarially crafted or manipulated emails would be displayed differently than the authentic ones.

Pitfalls with naïve constructions using signature schemes. We discuss why the following ad-hoc approach towards linkable message tagging is insecure in general: Using a regular signature scheme, users sign all outgoing emails and append the signatures to the messages; after some time, users additionally disclose to each other their respective verification keys. Authenticity of these keys is then verified by checking the validity of all signatures received so far; if all signatures are valid, the corresponding key is considered authentic. What might at first seem to be a sound approach is in fact not: (strongly) unforgeable signature schemes can be vulnerable to so-called *duplicate-signature key selection* (DSKS) attacks in which the adversary aims at finding verification keys that are valid for given message-signature pairs [7, 33, 30]. Clearly, a DSKS-vulnerable signature scheme will not yield a secure LMT scheme when employed in the construction just described. This in particular rules out using standard OpenPGP and S/MIME signatures for this naïve approach, as the latter are based on RSA or DSA which are reported vulnerable in [7].

Related work. Cryptographic solutions for message authentication in settings without pre-shared keys are generally based on signature schemes [16], most often in conjunction with hierarchical or non-hierarchical PKIs (like in X.509 [28] or OpenPGP [13]). As argued above, such approaches suffer from a variety of usability problems, particularly when used outside of organizations [49].

An alternative to public-key-based message authentication is given by *identity-based signatures* [45] that side-step the key distribution problem by replacing participants' verification keys by just their identities. Problematic here, however, is the required trustworthiness of the central authority (CA) that issues the private keys: the *key escrow problem*, namely that the CA can compute secret keys for any identity of its choosing and thus impersonate it, is inherent to id-based cryptography.

Certificateless signatures [1] can be seen as hybrid between PKI-based and identity-based signatures, avoiding both the use of certificates and the key escrow problem. Here, the CA issues only partial private keys; these need to be complemented by the corresponding user to obtain the actual private/public key pair. The public key can then be distributed non-authentically.

We note that the concept of *message recognition* [48, 32] partially aims in the direction of linkable message tagging. It was coined for settings where devices with low computational power and communication bandwidth should, after exchanging a small amount of authentic data, be able to afterwards recognize each other's messages. Comparable goals were also approached in ad-hoc networks based on location-limited side channels [4]. Note that both these notions require an a priori or side-channel-based exchange of authenticated data. With other words, they do not achieve the functionality of LMT schemes.

2 Preliminaries

Notation. Let A, B be sets. For $Q \subseteq A \times B$ and $a \in A$ we write $(a, \cdot) \in Q$ if $\exists b \in B : (a, b) \in Q$; we write $(a, \cdot) \notin Q$ if $\forall b \in B : (a, b) \notin Q$.

Fact 1 (Equivalence kernel). *Let A, B be sets and let $f \colon A \to B$ be a function. The equivalence relation $\sim_f$ on A defined by $a_1 \sim_f a_2 \Leftrightarrow f(a_1) = f(a_2)$ is said to be the* equivalence kernel *of f. For all equivalence relations $\sim$ on A we observe that $\sim$ is the equivalence kernel of projection $A \to A/_\sim$; $a \mapsto [a]$, where $[a]$ denotes the equivalence class of a.*

Definition 1 (Reflexive closure). *Let A be a set and let $R \subseteq A \times A$ be a relation on A. We define $\mathrm{Fix}(A) = \{(a, a) : a \in A\}$ and say that $R \cup \mathrm{Fix}(A)$ is the* reflexive closure *of R.*

2.1 Digital signature schemes

Signature schemes belong to the most important primitives in cryptography [29]. We recap their functionality, correctness, and security properties, introducing explicit notation for referring to the message space, the signature space, and the verification key space.

Definition 2 (Signature scheme). *A* signature scheme *$\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ consists of a message space $\mathcal{M} = \{0,1\}^*$, a signature space $\mathcal{S}ig$, a verification key space $\mathcal{VK}$, and the following efficient algorithms:*

$\mathsf{KGen}(1^\lambda)$: *On input the security parameter, this probabilistic algorithm outputs a signing key sk and a verification key $vk \in \mathcal{VK}$.*

$\mathsf{Sign}(sk, m)$: *On input a signing key and a message $m \in \mathcal{M}$, this algorithm outputs a signature $\sigma \in \mathcal{S}ig$.*

$\mathsf{Ver}(vk, m, \sigma)$: *On input a verification key, a message, and a candidate signature, this deterministic algorithm outputs either 0 or 1.*

The scheme is correct *if, for all $\lambda \in \mathbb{N}$, all $(sk, vk) \leftarrow_R \mathsf{KGen}(1^\lambda)$, all $m \in \{0,1\}^*$, and all $\sigma \leftarrow_R \mathsf{Sign}(sk, m)$, we have $\mathsf{Ver}(vk, m, \sigma) = 1$.*

As security requirements for signature schemes we define unforgeability ($\mathsf{euf\text{-}cma}$) and strong unforgeability ($\mathsf{suf\text{-}cma}$). It is readily seen that the latter notion is (strictly) stronger than the former.

Definition 3 (Unforgeability of signature schemes). *A signature scheme Σ is* (strongly) existentially unforgeable under adaptive chosen-message attacks ($\mathsf{euf\text{-}cma}$, respectively $\mathsf{suf\text{-}cma}$) *if for all efficient adversaries $\mathcal{A}$ the success probability $\mathrm{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}$ (resp., $\mathrm{Succ}^{\mathsf{suf\text{-}cma}}_{\Sigma,\mathcal{A}}$) is a negligible function where*

$$\mathrm{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(\lambda) = \Pr\left[\mathrm{Expt}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(1^\lambda) = 1\right] \quad \textit{and} \quad \mathrm{Succ}^{\mathsf{suf\text{-}cma}}_{\Sigma,\mathcal{A}}(\lambda) = \Pr\left[\mathrm{Expt}^{\mathsf{suf\text{-}cma}}_{\Sigma,\mathcal{A}}(1^\lambda) = 1\right]$$

```
Expt$^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(1^\lambda)$:
    (a) $(sk, vk) \leftarrow_R \mathsf{KGen}(1^\lambda)$
    (b) $(m^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(vk)$
    (c) Return 1 iff all hold:
            – $(m^*, \cdot) \notin Q$
            – $\mathsf{Ver}(vk, m^*, \sigma^*) = 1$
```

```
Expt$^{\mathsf{suf\text{-}cma}}_{\Sigma,\mathcal{A}}(1^\lambda)$:
    (a) $(sk, vk) \leftarrow_R \mathsf{KGen}(1^\lambda)$
    (b) $(m^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(vk)$
    (c) Return 1 iff all hold:
            – $(m^*, \sigma^*) \notin Q$
            – $\mathsf{Ver}(vk, m^*, \sigma^*) = 1$
```

```
Processing of $\mathcal{O}_{\mathsf{Sign}}(m)$:
    (a) $\sigma \leftarrow_R \mathsf{Sign}(sk, m)$
    (b) $Q \leftarrow Q \cup \{(m, \sigma)\}$
    (c) Return $\sigma$
```

Fig. 2. Experiments for (strong) unforgeability of signature schemes. We assume that queries to $\mathcal{O}_{\mathsf{Sign}}$ oracle are answered as specified on the right, and that set Q is initialized as $Q \leftarrow \emptyset$.

are defined in respect to the experiments from Figure 2, and the probabilities are taken over the random coins of the respective experiment and the adversary.

3 Linkable message tagging schemes

Our central contribution is the introduction of linkable message tagging (LMT) schemes. This novel cryptographic primitive allows users, after having generated a (secret) tagging key, to take arbitrary messages and compute corresponding LMT tags; other users can test for any two such message-tag pairs whether they have the same origin, i.e., were generated using the same key. Notably, this verification is done without any further (public or secret) inputs. Briefly speaking, security requires that tags be unforgeable, i.e., no adversary can find a message-tag pair that appears to have the same origin as a genuine one.

3.1 Syntax and security

We formalize LMT schemes by specifying their syntax, correctness, and security properties.

Definition 4 (LMT scheme). *A* linkable message tagging (LMT) scheme $\mathsf{L} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Link})$ *consists of a message space $\mathcal{M} = \{0,1\}^*$, a tag space $\mathcal{T}$, and the following efficient algorithms:*

$\mathsf{KGen}(1^\lambda)$**:** *On input the security parameter, this probabilistic algorithm outputs a tagging key tk.*
$\mathsf{Tag}(tk, m)$**:** *On input a tagging key tk and message $m \in \mathcal{M}$, this algorithm outputs a tag $\tau \in \mathcal{T}$.*
$\mathsf{Link}(m_1, \tau_1, m_2, \tau_2)$**:** *On input message-tag pairs $(m_1, \tau_1), (m_2, \tau_2) \in \mathcal{M} \times \mathcal{T}$, this deterministic algorithm outputs either 0 or 1. As a shortcut, we write $(m_1, \tau_1) \sim (m_2, \tau_2)$ if $\mathsf{Link}(m_1, \tau_1, m_2, \tau_2) = 1$. Otherwise, we write $(m_1, \tau_1) \not\sim (m_2, \tau_2)$.*

An LMT scheme shall offer a meaningful way to partition messages-tag pairs according to their origin. We hence require that the Link algorithm defines an equivalence relation on $\mathcal{M} \times \mathcal{T}$, and that all message-tag pairs created using a fixed tagging key belong to the same equivalence class.

Definition 5 (Correctness of LMT schemes). *An LMT scheme* L *is* correct *if (a) for all $\lambda \in \mathbb{N}$, all $tk \leftarrow_R \mathsf{KGen}(1^\lambda)$, all $m_1, m_2 \in \mathcal{M}$, and all $\tau_1 \leftarrow_R \mathsf{Tag}(tk, m_1)$ and $\tau_2 \leftarrow_R \mathsf{Tag}(tk, m_2)$, we have $(m_1, \tau_1) \sim (m_2, \tau_2)$, i.e., $\mathsf{Link}(m_1, \tau_1, m_2, \tau_2) = 1$, and (b) the* Link *algorithm defines an equivalence relation on $\mathcal{M} \times \mathcal{T}$, i.e., if for all $(m_1, \tau_1), (m_2, \tau_2), (m_3, \tau_3) \in \mathcal{M} \times \mathcal{T}$ we have*

– $(m_1, \tau_1) \sim (m_1, \tau_1)$ *(Reflexivity)*
– $(m_1, \tau_1) \sim (m_2, \tau_2) \Leftrightarrow (m_2, \tau_2) \sim (m_1, \tau_1)$ *(Symmetry)*
– $(m_1, \tau_1) \sim (m_2, \tau_2) \wedge (m_2, \tau_2) \sim (m_3, \tau_3) \Rightarrow (m_1, \tau_1) \sim (m_3, \tau_3)$ *(Transitivity)*

```
$\mathrm{Expt}^{\mathsf{lmt\text{-}euf}}_{\mathsf{L},\mathcal{A}}(1^\lambda)$:
  (a) $tk \leftarrow_R \mathsf{KGen}(1^\lambda)$
  (b) $(m^*, \tau^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Tag}}}(1^\lambda)$
  (c) Return 1 iff all hold:
    – $(m^*, \cdot) \notin Q$
    – $\exists (m, \tau) \in Q : (m^*, \tau^*) \sim (m, \tau)$
```

```
$\mathrm{Expt}^{\mathsf{lmt\text{-}suf}}_{\mathsf{L},\mathcal{A}}(1^\lambda)$:
  (a) $tk \leftarrow_R \mathsf{KGen}(1^\lambda)$
  (b) $(m^*, \tau^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Tag}}}(1^\lambda)$
  (c) Return 1 iff all hold:
    – $(m^*, \tau^*) \notin Q$
    – $\exists (m, \tau) \in Q : (m^*, \tau^*) \sim (m, \tau)$
```

```
Processing of $\mathcal{O}_{\mathsf{Tag}}(m)$:
  (a) $\tau \leftarrow_R \mathsf{Tag}(tk, m)$
  (b) $Q \leftarrow Q \cup \{(m, \tau)\}$
  (c) Return $\tau$
```

Fig. 3. Experiments for (strong) unforgeability of LMT schemes. We assume that queries to $\mathcal{O}_{\mathsf{Tag}}$ oracle are answered as specified on the right, and that set Q is initialized as $Q \leftarrow \emptyset$.

```
$\mathsf{KGen}(1^\lambda)$:
  $x \leftarrow_R \mathbb{Z}_q$
  $X \leftarrow g^x$
  Output $(sk, vk) = (x, X)$

$\mathsf{Sign}(sk, m)$:
  Output $\sigma = H(m)^x$

$\mathsf{Ver}(vk, m, \sigma)$:
  If $e(\sigma, g) = e(H(m), X)$
    output 1,
    else output 0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
$\mathsf{KGen}(1^\lambda)$:
  $x \leftarrow_R \mathbb{Z}_q$
  Output $tk = x$

$\mathsf{Tag}(tk, m)$:
  Output $\tau = H(m)^x$

$\mathsf{Link}(m_1, \tau_1, m_2, \tau_2)$:
  If $e(H(m_1), \tau_2) = e(H(m_2), \tau_1)$
    output 1,
    else output 0
```

Fig. 4. Signature scheme BLS (top) and LMT scheme $\mathsf{BLS\text{-}LMT}$ (bottom). For further details see Construction 1.

The essential security property of LMT schemes is unforgeability: Given a collection Q of genuine message-tag pairs, it shall be impossible to come up with a tag τ^* on a fresh message m^* such that $(m^*, \tau^*) \sim (m, \tau)$ for any pair $(m, \tau) \in Q$. Akin to the security notions for signature schemes, we distinguish between two variants of unforgeability: $\mathsf{lmt\text{-}euf}$, and the (strictly) stronger $\mathsf{lmt\text{-}suf}$.

Definition 6 (Unforgeability of LMT schemes). *An LMT scheme* L *is* (strongly) unforgeable *(*$\mathsf{lmt\text{-}euf}$*, respectively* $\mathsf{lmt\text{-}suf}$*) if for all efficient adversaries* $\mathcal{A}$ *the success probability* $\mathrm{Succ}^{\mathsf{lmt\text{-}euf}}_{\mathsf{L},\mathcal{A}}$ *(resp.,* $\mathrm{Succ}^{\mathsf{lmt\text{-}suf}}_{\mathsf{L},\mathcal{A}}$*) is a negligible function where*

$$\mathrm{Succ}^{\mathsf{lmt\text{-}euf}}_{\mathsf{L},\mathcal{A}}(\lambda) = \Pr\left[\mathrm{Expt}^{\mathsf{lmt\text{-}euf}}_{\mathsf{L},\mathcal{A}}(1^\lambda) = 1\right] \quad \textit{and} \quad \mathrm{Succ}^{\mathsf{lmt\text{-}suf}}_{\mathsf{L},\mathcal{A}}(\lambda) = \Pr\left[\mathrm{Expt}^{\mathsf{lmt\text{-}suf}}_{\mathsf{L},\mathcal{A}}(1^\lambda) = 1\right]$$

are defined in respect to the experiments from Figure 3, and the probabilities are taken over the random coins of the respective experiment and the adversary.

3.2 A direct LMT construction

We show the existence of LMT schemes by giving a simple yet efficient example. Our construction is loosely related to the BLS signature scheme by Boneh, Lynn, and Shacham [9, 10] (cf. Figure 4) which enjoys strong unforgeability under the CDH assumption in a pairing-friendly group, in the random oracle model.

Construction 1 (LMT scheme $\mathsf{BLS\text{-}LMT}$). *The LMT scheme* $\mathsf{BLS\text{-}LMT}$ *is defined in respect to a (symmetric) bilinear group* $\mathbb{G}$ *and a hash function* $H \colon \mathcal{M} \to \mathbb{G} \setminus \{1\}$*. More precisely, for a set* $(\mathbb{G}, \mathbb{G}_T, q, g, e)$ *it is assumed that* $\mathbb{G} = \langle g \rangle$ *is a cyclic group of prime order* q *and that* $e \colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *is a bilinear map. The tag space of* $\mathsf{BLS\text{-}LMT}$ *is* $\mathcal{T} = \mathbb{G}$*, and its algorithms are specified in Figure 4.*

Proof of Correctness. Consider function $f\colon \mathcal{M}\times\mathcal{T}\to\mathbb{Z}_q;\ (m,\tau)\mapsto \log_{H(m)}(\tau)$. For arbitrary $(m_1,\tau_1),(m_2,\tau_2)\in\mathcal{M}\times\mathcal{T}$ let $\alpha = f(m_1,\tau_1)$ and $\beta = f(m_2,\tau_2)$, i.e., $\tau_1 = H(m_1)^\alpha$ and $\tau_2 = H(m_2)^\beta$. By bilinearity and symmetry of e we obtain

$$(m_1,\tau_1)\sim(m_2,\tau_2)\ \Leftrightarrow\ e(H(m_1),H(m_2))^\beta = e(H(m_2),H(m_1))^\alpha\ \Leftrightarrow\ \alpha=\beta. \tag{1}$$

Together with Fact 1 this characterization of the Link algorithm allows an immediate verification of the requirements from Definition 5. □

We next analyze the security properties of BLS-LMT.

Theorem 1 (Security of BLS-LMT). *The LMT scheme* BLS-LMT *is strongly unforgeable (*lmt-suf*) if the CDH assumption holds in $\mathbb{G}$ and H is modeled as a random oracle. By consequence, the scheme is also unforgeable (*lmt-euf*).*

Proof. We reduce the strong unforgeability of the BLS-LMT scheme to the strong unforgeability of the BLS signature scheme (defined over the same bilinear group). Together with Fact 2 below, this establishes the statement.

From an efficient adversary $\mathcal{A}$ against lmt-suf of BLS-LMT we construct an efficient adversary $\mathcal{B}$ against suf-cma of BLS. Concretely, $\mathcal{B}(vk)$ runs $\mathcal{A}(1^\lambda)$ as a subroutine. Each oracle query $\mathcal{O}_{\mathsf{Tag}}(m)$ posed by $\mathcal{A}$ is forwarded by $\mathcal{B}$ as $\mathcal{O}_{\mathsf{Sign}}(m)$ to its own oracle, and the result is relayed to $\mathcal{A}$ (observe that this is a perfect simulation). Finally, when $\mathcal{A}$ outputs a candidate tag forgery (m^*,τ^*), $\mathcal{B}$ outputs the same pair and stops.

Assume $\mathcal{A}$ is successful, i.e., (m^*,τ^*) is a valid tag forgery. By definition we then have $e(H(m^*),H(m)^x) = e(H(m),\tau^*)$ for $x=\log_g vk$ and some m. This equation can only hold if $\tau^* = H(m^*)^x$, i.e., τ^* is a valid BLS signature on m^*. Further on, as τ^* is not the result of an $\mathcal{O}_{\mathsf{Tag}}(m^*)$ query, this signature was also not output on input m^* by $\mathcal{B}$'s $\mathcal{O}_{\mathsf{Sign}}$ oracle, i.e., adversary $\mathcal{B}$ is successful in forging a signature. We hence have $\mathrm{Succ}^{\mathsf{suf\text{-}cma}}_{\mathsf{BLS},\mathcal{B}}(\lambda) = \mathrm{Succ}^{\mathsf{lmt\text{-}suf}}_{\mathsf{BLS\text{-}LMT},\mathcal{A}}(\lambda)$. As the left-hand side is negligible by assumption, this completes the proof. □

Fact 2 (Unforgeability of BLS signature scheme [9]). *The* BLS *signature scheme is existentially unforgeable (*euf-cma*) if the CDH assumption holds in $\mathbb{G}$ and H is modeled as a random oracle. Moreover, the scheme has unique signatures and is hence strongly unforgeable (*suf-cma*).*

3.3 On the generic relation between signature and LMT schemes

We now explore the relationship between signature schemes and LMT schemes. Perhaps surprisingly, from an engineer's point of view, the notions are quite close: LMT schemes and signature schemes can be constructed from each other, and the corresponding transformations are natural and efficient. On the other side of the coin this tight connection implies that there is little hope to obtain practical LMT constructions from just symmetric primitives like blockciphers, hash functions, or PRFs, although LMT schemes *per se* do not require public keys.

We begin with transforming signature schemes into LMT schemes. The idea behind our construction is to use signing keys as tagging keys and signatures with attached verification keys as tags, i.e., $\mathcal{T} = \mathcal{S}ig\times\mathcal{VK}$; precisely, a tag for a message m is a pair $\tau = (\sigma, vk)$ such that σ is a signature on m in respect to verification key vk. Intuitively, in order to test whether two message-tag pairs are in LMT relation, one checks that the verification keys match and both signatures are valid. This minimum requirement on the Link algorithm is formally expressed by binary relation R_M, defined on $\mathcal{M}\times\mathcal{T} = \mathcal{M}\times\mathcal{S}ig\times\mathcal{VK}$ as follows:

$$(m_1,\sigma_1,vk_1)\ R_M\ (m_2,\sigma_2,vk_2)\quad\Leftrightarrow\quad vk_1 = vk_2\ \wedge\ \mathsf{Ver}(vk_1,m_1,\sigma_1) = 1 = \mathsf{Ver}(vk_2,m_2,\sigma_2).$$

Observe that this relation is symmetric and transitive. However, it is not reflexive, and hence does not induce a correct LMT scheme, as message-tag pairs with invalid signatures are not in relation to themselves. Fixing this problem by adding further elements to R_M requires special care: intuitively, LMT security is diluted by putting valid and invalid tags into relation. Generic candidate elements that can be safely added to R_M seem to be those from binary relation R_0, defined on $\mathcal{M} \times \mathcal{S}ig \times \mathcal{VK}$ such that

$$(m_1, \sigma_1, vk_1)\ R_0\ (m_2, \sigma_2, vk_2) \quad \Leftrightarrow \quad \mathsf{Ver}(vk_1, m_1, \sigma_1) = 0 = \mathsf{Ver}(vk_2, m_2, \sigma_2).$$

Concluding, we say that an equivalence relation R on $\mathcal{M} \times \mathcal{S}ig \times \mathcal{VK}$ is *admissible* if $R_M \subseteq R \subseteq R_M \cup R_0$. Many such relations exist in general, but two natural choices for R are

– $R_M^{1:1} = R_M \cup \mathrm{Fix}(\mathcal{M} \times \mathcal{S}ig \times \mathcal{VK})$, i.e., the reflexive closure of R_M (see Definition 1). This is the finest possible admissible relation, where no invalid message-tag pair is related to any other (valid or invalid) message-tag pair.
– $R_M^{*:1} = R_M \cup R_0$, i.e., the coarsest possible admissible relation. Here, all invalid message-tag pairs are in relation to each other, i.e., belong to the same equivalence class.

Lemma 1. *$R_M^{1:1}$ and $R_M^{*:1}$ are admissible equivalence relations.*

Proof. Let $\mathcal{X} = \mathcal{M} \times \mathcal{S}ig \times \mathcal{VK}$. Observe that for each $a = (m, \sigma, vk) \in \mathcal{X}$ we have either $\mathsf{Ver}(vk, m, \sigma) = 1$ and $(a, a) \in R_M$, or $\mathsf{Ver}(vk, m, \sigma) = 0$ and $(a, a) \in R_0$. In other words, $\mathrm{Fix}(\mathcal{X}) \subseteq R_M \cup R_0$.

We first consider $R_M^{1:1}$. Reflexivity and symmetry are clear. To show transitivity, let $a, b, c \in \mathcal{X}$ with $(a, b), (b, c) \in R_M^{1:1}$. Nothing is to show if $(a, b) \in \mathrm{Fix}(\mathcal{X})$ or $(b, c) \in \mathrm{Fix}(\mathcal{X})$. Thus assume $(a, b), (b, c) \in R_M$. Transitivity of R_M implies $(a, c) \in R_M \subseteq R_M^{1:1}$. Admissibility follows from $\mathrm{Fix}(\mathcal{X}) \subseteq R_M \cup R_0$.

We next consider $R_M^{*:1}$. Reflexivity follows from $\mathrm{Fix}(\mathcal{X}) \subseteq R_M \cup R_0 = R_M^{*:1}$. Symmetry is clear. To show transitivity, let $a, b, c \in \mathcal{X}$ with $(a, b), (b, c) \in R_M^{*:1}$. Then either $(a, b), (b, c) \in R_M$ or $(a, b), (b, c) \in R_0$. In both cases $(a, c) \in R_M^{*:1}$ follows. Admissibility is clear. □

We are now ready to specify our signature-based LMT scheme. More precisely, we define a family of schemes, parameterized by an admissible equivalence relation R. We stress that the security achieved by our LMT construction is independent of the specific relation in use (as long as it is admissible) as the latter influences only how *invalid* message-tag pairs are grouped together. Defining our construction in respect to arbitrary admissible relations leaves flexibility to the user of the scheme—it is the application that determines whether $R_M^{1:1}$, $R_M^{*:1}$, or any other admissible relation is the favorable one.

Construction 2 (LMT scheme $\mathsf{Sig\text{-}LMT}$ from a generic signature scheme). *Let Σ be a signature scheme with message space $\mathcal{M} = \{0,1\}^*$, signature space $\mathcal{S}ig$, and verification key space $\mathcal{VK}$. Let R be an admissible equivalence relation on $\mathcal{M} \times \mathcal{S}ig \times \mathcal{VK}$. Define the LMT scheme $\mathsf{Sig\text{-}LMT}$ in respect to R with tag space $\mathcal{T} = \mathcal{S}ig \times \mathcal{VK}$ as follows:*

$\mathsf{KGen}(1^\lambda)$**:** *Set $(sk, vk) \leftarrow_R \Sigma.\mathsf{KGen}(1^\lambda)$ and output tagging key $tk = (sk, vk)$.*
$\mathsf{Tag}(tk, m)$**:** *Compute $\sigma \leftarrow_R \mathsf{Sign}(sk, m)$ and output tag $\tau = (\sigma, vk)$.*
$\mathsf{Link}(m_1, \tau_1, m_2, \tau_2)$**:** *Parse τ_1 as (σ_1, vk_1) and τ_2 as (σ_2, vk_2). Output 1 if $(m_1, \sigma_1, vk_1)\ R\ (m_2, \sigma_2, vk_2)$. Otherwise, output 0.*

Correctness of $\mathsf{Sig\text{-}LMT}$ follows directly from the definition of admissibility. The scheme's unforgeability properties are analyzed in Theorem 2; as mentioned above, they are independent of the particular choice of relation R.

Theorem 2 (Security of $\mathsf{Sig\text{-}LMT}$). *For any admissible equivalence relation R, the LMT scheme $\mathsf{Sig\text{-}LMT}$ defined in respect to R is (strongly) unforgeable if the underlying signature scheme Σ is (strongly) unforgeable.*

Proof. We reduce the unforgeability of the $\mathsf{Sig\text{-}LMT}$ scheme to the unforgeability of the Σ signature scheme. The proof can easily be adapted to cover the strong variants of the corresponding security notions.

From an efficient adversary $\mathcal{A}$ against $\mathsf{lmt\text{-}euf}$ of $\mathsf{Sig\text{-}LMT}$ we construct an efficient adversary $\mathcal{B}$ against $\mathsf{euf\text{-}cma}$ of Σ. Concretely, $\mathcal{B}(vk)$ runs $\mathcal{A}(1^\lambda)$ as a subroutine. Each oracle query $\mathcal{O}_{\mathsf{Tag}}(m)$ posed by $\mathcal{A}$ is forwarded by $\mathcal{B}$ as $\sigma \leftarrow \mathcal{O}_{\mathsf{Sign}}(m)$ to its own oracle, and $\tau = (\sigma, vk)$ is answered to $\mathcal{A}$ (observe that this is a perfect simulation). Let Q denote the collection of all occurring such pairs (m, σ). Finally, when $\mathcal{A}$ outputs a candidate tag forgery (m^*, τ^*), $\mathcal{B}$ parses τ^* as (σ^*, vk^*), outputs σ^*, and stops.

Assume $\mathcal{A}$ is successful, i.e., (m^*, τ^*) is a valid tag forgery. By definition we then have $(m^*, \sigma^*, vk^*)\ R\ (m, \sigma, vk)$ for some $(m, \sigma) \in Q$. As $\mathsf{Ver}(vk, m, \sigma) = 1$ by construction, we particularly have $(m^*, \sigma^*, vk^*)\ R_M\ (m, \sigma, vk)$, i.e., $\mathsf{Ver}(vk, m^*, \sigma^*) = 1$. Further on, as m^* was never queried to the $\mathcal{O}_{\mathsf{Tag}}$ oracle, it was also not queried to $\mathcal{B}$'s $\mathcal{O}_{\mathsf{Sign}}$ oracle, i.e., adversary $\mathcal{B}$ is successful in forging a signature. We hence have $\mathrm{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{B}}(\lambda) = \mathrm{Succ}^{\mathsf{lmt\text{-}euf}}_{\mathsf{Sig\text{-}LMT},\mathcal{A}}(\lambda)$. As the left-hand side is negligible by assumption, this completes the proof. □

We next transform LMT schemes into signature schemes. The intuition behind our construction is to use tags as signatures and to include a reference message-tag pair in the verification key against which candidate signatures (i.e., tags) can be aligned.[3] However, doing so naïvely would not result in an unforgeable signature scheme: an adversary could just output the reference pair as a forgery. We deploy a technical solution to this problem in Construction 3: the fixed prefix `"1"` is prepended to all signed messages, preventing collision with the `"0"` message used for the reference tag.

Construction 3 (Signature scheme $\mathsf{LMT\text{-}Sig}$ from a generic LMT scheme). *Let L be an LMT scheme with tag space $\mathcal{T}$. Define the signature scheme $\mathsf{LMT\text{-}Sig}$ with $\mathcal{S}ig = \mathcal{VK} = \mathcal{T}$ as follows:*

$\mathsf{KGen}(1^\lambda)$: *Set $tk \leftarrow_R \mathsf{L.KGen}(1^\lambda)$, compute $\tau_0 \leftarrow \mathsf{Tag}(tk, \texttt{"0"})$, and output key pair $(sk, vk) = (tk, \tau_0)$.*
$\mathsf{Sign}(sk, m)$: *Compute $\tau \leftarrow \mathsf{Tag}(tk, \texttt{"1"} \| m)$ and output signature $\sigma = \tau$.*
$\mathsf{Ver}(vk, m, \sigma)$: *Output $\mathsf{Link}(\texttt{"1"} \| m, \sigma, \texttt{"0"}, \tau_0)$.*

Correctness of $\mathsf{LMT\text{-}Sig}$ follows from correctness of L. The unforgeability of Construction 3 is assessed as follows.

Theorem 3 (Security of $\mathsf{LMT\text{-}Sig}$). *The signature scheme $\mathsf{LMT\text{-}Sig}$ is (strongly) unforgeable if the underlying LMT scheme L is (strongly) unforgeable.*

Proof. We reduce the unforgeability of the $\mathsf{LMT\text{-}Sig}$ signature scheme to the unforgeability of LMT scheme L. The proof can easily be adapted to cover the strong variants of the corresponding security notions.

From an efficient adversary $\mathcal{A}$ against $\mathsf{euf\text{-}cma}$ of $\mathsf{LMT\text{-}Sig}$ we construct an efficient adversary $\mathcal{B}$ against $\mathsf{lmt\text{-}euf}$ of L. Concretely, $\mathcal{B}(1^\lambda)$ starts by posing a $\tau_0 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\texttt{"0"})$ query to its tagging oracle. Then $\mathcal{B}$ runs $\mathcal{A}$, on input $vk = \tau_0$, as a subroutine. Each oracle query $\mathcal{O}_{\mathsf{Sign}}(m)$

[3] Note that the BLS signature scheme can be seen as the result of applying this technique to the LMT scheme $\mathsf{BLS\text{-}LMT}$. The reference tag is $X = g^x$ and (virtually) corresponds to a message in $H^{-1}(g)$.

posed by $\mathcal{A}$ is forwarded by $\mathcal{B}$ as $\mathcal{O}_{\mathsf{Tag}}(\texttt{"1"} \parallel m)$ to its own oracle, and the result is relayed to $\mathcal{A}$ (observe that this is a perfect simulation). Finally, when $\mathcal{A}$ outputs a candidate signature forgery (m^*, σ^*), $\mathcal{B}$ outputs $(\texttt{"1"} \parallel m^*, \sigma^*)$ and stops.

Assume $\mathcal{A}$ is successful, i.e., (m^*, σ^*) is a valid signature forgery. By definition we then have $\mathsf{Link}(\texttt{"1"} \parallel m^*, \sigma^*, \texttt{"0"}, \tau_0) = 1$. Further on, as m^* was never queried to the $\mathcal{O}_{\mathsf{Sign}}$ oracle, $\texttt{"1"} \parallel m^*$ was also not queried to the $\mathcal{O}_{\mathsf{Tag}}$ oracle, i.e., adversary $\mathcal{B}$ is successful in forging a tag. We hence have $\mathrm{Succ}^{\mathsf{lmt\text{-}euf}}_{\mathsf{L},\mathcal{B}}(\lambda) = \mathrm{Succ}^{\mathsf{euf\text{-}cma}}_{\mathsf{LMT\text{-}Sig},\mathcal{A}}(\lambda)$. As the left-hand side is negligible by assumption, this completes the proof. □

4 Classifiable message tagging schemes

The main functionality of LMT schemes is the partitioning of message-tag pairs into sets according to their origin. Correspondingly, Definition 5 requires the Link algorithm to induce an equivalence relation $\sim$ on the message-tag space $\mathcal{M} \times \mathcal{T}$ such that, essentially, each equivalence class corresponds to one origin. Note that, so far, the used notion of origin is only virtual: it does not explicitly appear in syntax, correctness, or security definition of LMT schemes. In other words, although the Link algorithm always implies existence of projection

$$\pi\colon \mathcal{M} \times \mathcal{T} \to (\mathcal{M} \times \mathcal{T})/_{\sim};\ (m, \tau) \mapsto [m, \tau]$$

that maps message-tag pairs to their 'origin' (cf. Fact 1), there is no formal requirement in Section 3 that demands that this mapping be effectively computable.

We argue that for many natural applications of the LMT primitive the direct computability of π would be quite desirable. Assume, for instance, a mail client that is instructed to automatically classify received emails according to their origin. In such a setting, for each incoming email m with corresponding tag τ, the Link algorithm of a plain LMT scheme would have to be invoked several times, namely at least once for each priorly identified class of origin. If, however, we assume existence of an efficient projection π as defined above, the mail client could sort any email directly, i.e., without any further pair-wise computation, into a folder of name $\rho \circ \pi(m, \tau)$, where injective function $\rho\colon (\mathcal{M} \times \mathcal{T})/_{\sim} \to \{0,1\}^*$ assigns a (unique) label to each origin.

In this section we introduce an LMT variant called classifiable message tagging (CMT); such schemes are equipped with a dedicated algorithm for the efficient computation of $\rho \circ \pi$, for an adequate labeling scheme ρ. We will define the adapted syntax and security notions, study formal relations between the CMT and LMT primitives, and finally identify generic CMT constructions. We leave the direct construction of (more efficient) instances to Section 5.

4.1 Syntax and security

The syntactical difference between LMT and CMT schemes is that the Link algorithm is replaced by the new $\mathsf{Classify}$ algorithm that shall compute $\rho \circ \pi$. In contrast to the introduction given above, we assume that the labels output by $\mathsf{Classify}$ are elements of an abstract set $\mathcal{C}$ of *class identifiers*; the only restriction on $\mathcal{C}$ is that its elements can be tested for equality.

Definition 7 (CMT scheme). *A* classifiable message tagging (CMT) scheme $\mathsf{C} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Classify})$ *consists of a message space $\mathcal{M} = \{0,1\}^*$, a tag space $\mathcal{T}$, a class identifier space $\mathcal{C}$, and the following efficient algorithms:*

$\mathsf{KGen}(1^\lambda)$**:** *On input the security parameter, this probabilistic algorithm outputs a tagging key tk.*
$\mathsf{Tag}(tk, m)$**:** *On input a tagging key tk and message $m \in \mathcal{M}$, this algorithm outputs a tag $\tau \in \mathcal{T}$.*
$\mathsf{Classify}(m, \tau)$**:** *On input a message-tag pair $(m, \tau) \in \mathcal{M} \times \mathcal{T}$, this deterministic algorithm outputs a class identifier $cid \in \mathcal{C}$.*

```
$\mathrm{Expt}^{\mathsf{cmt\text{-}euf}}_{\mathsf{C},\mathcal{A}}(1^\lambda)$:
    (a) $tk \leftarrow_R \mathsf{KGen}(1^\lambda)$
    (b) $(m^*, \tau^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Tag}}}(1^\lambda)$
    (c) Return 1 iff all hold:
        – $(m^*, \cdot) \notin Q$
        – $\mathsf{Classify}(m^*, \tau^*) = cid_{tk}$

$\mathrm{Expt}^{\mathsf{cmt\text{-}suf}}_{\mathsf{C},\mathcal{A}}(1^\lambda)$:
    (a) $tk \leftarrow_R \mathsf{KGen}(1^\lambda)$
    (b) $(m^*, \tau^*) \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathsf{Tag}}}(1^\lambda)$
    (c) Return 1 iff all hold:
        – $(m^*, \tau^*) \notin Q$
        – $\mathsf{Classify}(m^*, \tau^*) = cid_{tk}$

Processing of $\mathcal{O}_{\mathsf{Tag}}(m)$:
    (a) $\tau \leftarrow_R \mathsf{Tag}(tk, m)$
    (b) $Q \leftarrow Q \cup \{(m, \tau)\}$
    (c) Return $\tau$
```

Fig. 5. Experiments for (strong) unforgeability of CMT schemes. We assume that queries to $\mathcal{O}_{\mathsf{Tag}}$ oracle are answered as specified on the right, and that set Q is initialized as $Q \leftarrow \emptyset$.

Correctness of CMT schemes requires that for any two pairs of message and corresponding tag (created with the same tagging key) the $\mathsf{Classify}$ algorithm outputs the same class identifier:

Definition 8 (Correctness of CMT schemes). *A CMT scheme* C *is* correct *if for all* $\lambda \in \mathbb{N}$, *all* $tk \leftarrow_R \mathsf{KGen}(1^\lambda)$, *all* $m_1, m_2 \in \mathcal{M}$, *and all* $\tau_1 \leftarrow_R \mathsf{Tag}(tk, m_1)$ *and* $\tau_2 \leftarrow_R \mathsf{Tag}(tk, m_2)$, *we have* $\mathsf{Classify}(m_1, \tau_1) = \mathsf{Classify}(m_2, \tau_2)$.

Definition 9 (Key-specific class identifier cid_{tk}). *Consider a (correct) CMT scheme* C *and a fixed key* $tk \leftarrow_R \mathsf{KGen}(1^\lambda)$. *For any message* $m \in \mathcal{M}$, *compute* $\tau \leftarrow_R \mathsf{Tag}(tk, m)$ *and* $cid \leftarrow \mathsf{Classify}(m, \tau)$. *Then the correctness of* C *ensures that cid is independent of both m and the randomness used in the* Tag *algorithm. In other words, for each tagging key tk there exists a (uniquely) corresponding* key-specific class identifier cid_{tk} *defined in this way.*

Similarly to LMT schemes, we define unforgeability as the essential security property of CMT schemes: Given a collection of genuine message-tag pairs for a fixed class identifier cid_{tk}, it shall be impossible to come up with a tag τ^* on a fresh message m^* such that $\mathsf{Classify}(m^*, \tau^*) = cid_{tk}$. We again distinguish two variants of unforgeability, namely $\mathsf{cmt\text{-}euf}$, and the (strictly) stronger $\mathsf{cmt\text{-}suf}$.

Definition 10 (Unforgeability of CMT schemes). *A CMT scheme* C *is* (strongly) unforgeable *(*$\mathsf{cmt\text{-}euf}$*, respectively* $\mathsf{cmt\text{-}suf}$*) if for all efficient adversaries* $\mathcal{A}$ *the success probability* $\mathrm{Succ}^{\mathsf{cmt\text{-}euf}}_{\mathsf{C},\mathcal{A}}$ *(resp.,* $\mathrm{Succ}^{\mathsf{cmt\text{-}suf}}_{\mathsf{C},\mathcal{A}}$*) is a negligible function where*

$$\mathrm{Succ}^{\mathsf{cmt\text{-}euf}}_{\mathsf{C},\mathcal{A}}(\lambda) = \Pr\left[\mathrm{Expt}^{\mathsf{cmt\text{-}euf}}_{\mathsf{C},\mathcal{A}}(1^\lambda) = 1\right] \quad \textit{and} \quad \mathrm{Succ}^{\mathsf{cmt\text{-}suf}}_{\mathsf{C},\mathcal{A}}(\lambda) = \Pr\left[\mathrm{Expt}^{\mathsf{cmt\text{-}suf}}_{\mathsf{C},\mathcal{A}}(1^\lambda) = 1\right]$$

are defined in respect to the experiments from Figure 5, and the probabilities are taken over the random coins of the respective experiment and the adversary.

Remark 1 (Convenient class identifier spaces). Let $\mathsf{C} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Classify})$ be a CMT scheme with class identifier space $\mathcal{C}$. Let further $H : \mathcal{C} \to \mathcal{C}'$ be a collision-resistant hash function, where $\mathcal{C}'$ is an arbitrary set. It is easy to see that if C is (strongly) unforgeable then so is $\mathsf{C}' = (\mathsf{KGen}, \mathsf{Tag}, H \circ \mathsf{Classify})$. In practice this leads to the conclusion that convenient class identifier spaces like $\mathcal{C}' = \{0,1\}^{160}$ can be assumed without loss of (much) generality.

4.2 CMT schemes are special LMT schemes

We motivated the introduction of CMT schemes with the interest in a specially optimized LMT variant. However, it is not immediately apparent that schemes defined in accordance with Definitions 7–10 are indeed of the LMT type. We next give formal evidence for this relation, both syntax-wise and security-wise. Interestingly, as we will see, the reverse relation does not hold in general, i.e., there exist LMT schemes that do not have a CMT analogue.

Let $\mathsf{C} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Classify})$ be an arbitrary CMT scheme with message space $\mathcal{M}$ and tag space $\mathcal{T}$. Define the following auxiliary algorithm:

$\mathsf{Link}(m_1, \tau_1, m_2, \tau_2)$**:** On input message-tag pairs $(m_1, \tau_1), (m_2, \tau_2) \in \mathcal{M} \times \mathcal{T}$ output 0 or 1 such that the following condition is met:

$$\mathsf{Link}(m_1, \tau_1, m_2, \tau_2) = 1 \quad \Leftrightarrow \quad \mathsf{Classify}(m_1, \tau_1) = \mathsf{Classify}(m_2, \tau_2). \tag{2}$$

Given this, define the LMT scheme corresponding to C as $\mathsf{L} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Link})$. Regarding its correctness, note that condition (a) of Definition 5 is fulfilled due to the correctness of C, and that condition (b) follows from (2) in conjunction with Fact 1. In addition, a comparison of Figures 3 and 5 readily establishes the following relation between security notions.

Lemma 2 (Unforgeability of C is equivalent to unforgeability of L). *A CMT scheme is* cmt-euf *(resp.* cmt-suf*) if and only if the corresponding LMT scheme is* lmt-euf *(resp.* lmt-suf*).* □

Remark 2 (Not all LMT schemes have CMT analogues). It is interesting to note that not all LMT schemes have a corresponding CMT scheme. Consider, for instance, the BLS-LMT scheme from Construction 1. A comparison of equations (1) and (2) suggests that the role of the key-specific class identifier in a potential CMT analogue of BLS-LMT would be taken by $cid_{tk} = tk$, i.e., the tagging key itself. However, this would imply inefficiency of the Classify algorithm as it would have to solve DLP instances (from $H(m)$ and $H(m)^{tk}$ compute tk). This problem wouldn't be mitigated by instead setting $cid_{tk} = g^{tk}$, as Classify now would have to solve the CDH problem (from $H(m)$ and $H(m)^{tk}$ and g compute g^{tk}). Observe also that switching to a group $\mathbb{G}$ in which DLP or CDH are easy is not an option as the unforgeability of BLS-LMT relies on their hardness; indeed, efficient DLP/CDH-solving Classify algorithms as discussed above would lead immediately to efficient adversaries against unforgeability.

4.3 On the generic relation between signature and CMT schemes

In Section 3.3 we showed how to construct LMT schemes from signature schemes and vice versa. We analyze next if similar results also hold in the CMT setting.

We will first focus on how to generically obtain CMT schemes from signature schemes. Recall that we based our corresponding transformation in the LMT setting on an arbitrary admissible equivalence relation R that determines how invalid tags are grouped together (cf. Construction 2). This offers a degree of freedom that we implicitly exploit when constructing our signature-based CMT scheme Sig-CMT presented below: the key observation is that Construction 2 with $R = R_M^{*;1}$ yields exactly the same LMT scheme as is obtained by understanding Sig-CMT as an LMT scheme in the terms of Section 4.2. Indeed, in both cases message-tag pairs are in relation exactly when either both tags are valid and the verification keys match (the latter serve as key-specific class identifiers in Sig-CMT), or when the tags are both invalid.

Construction 4 (CMT scheme Sig-CMT from a generic signature scheme). *Let Σ be a signature scheme with signature space $\mathcal{S}ig$ and verification key space $\mathcal{VK}$. Define the CMT scheme* Sig-CMT *with tag space $\mathcal{T} = \mathcal{S}ig \times \mathcal{VK}$ and class identifier space $\mathcal{C} = \mathcal{VK} \,\dot\cup\, \{\bot\}$ (where $\dot\cup$ denotes the disjoint union) as follows:*

$\mathsf{KGen}(1^\lambda)$**:** *Set $(sk, vk) \leftarrow_R \Sigma.\mathsf{KGen}(1^\lambda)$ and output tagging key $tk = (sk, vk)$.*
$\mathsf{Tag}(tk, m)$**:** *Compute $\sigma \leftarrow_R \mathsf{Sign}(sk, m)$ and output tag $\tau = (\sigma, vk)$.*
$\mathsf{Classify}(m, \tau)$**:** *Parse τ as (σ, vk). Output $cid = vk$ if $\mathsf{Ver}(vk, m, \sigma) = 1$. Otherwise, output $cid = \bot$.*

Correctness of the scheme is obvious. Concerning its security claim, by Lemma 2 it suffices to prove the (strong) unforgeability of the LMT scheme corresponding to Sig-CMT. Following the observations above, as the latter scheme results from Construction 2 with $R = R_M^{*;1}$, Theorem 2 and Lemma 1 establish the following security result.

Theorem 4 (Security of Sig-CMT). *The CMT scheme* Sig-CMT *is (strongly) unforgeable if the underlying signature scheme Σ is (strongly) unforgeable.* $\square$

The next step is to generically construct a signature scheme from a CMT scheme. That this works in principle is not surprising by our earlier results: CMT schemes are just special LMT schemes, and the latter already imply signature schemes. We observe, however, that the following generic construction is slightly more efficient.

In CMT-Sig we use tagging keys tk as signing keys sk, and key-specific class identifiers cid_{tk} as verification keys vk. By Remark 1 the verification key space $\mathcal{VK}$ of CMT-Sig can hence be chosen to be of a form convenient in practice[4], e.g., $\mathcal{VK} = \{0,1\}^{160}$. Observe that here we do not require to prepend fixed strings to messages in order to split the message space into two disjoint subdomains, as it was necessary in the LMT case (cf. Construction 3).

Construction 5 (Signature scheme CMT-Sig from a generic CMT scheme). *Let* C *be a CMT scheme with tag space $\mathcal{T}$ and class identifier space $\mathcal{C}$. Define the signature scheme* CMT-Sig *with signature space $\mathcal{S}ig = \mathcal{T}$ and verification key space $\mathcal{VK} = \mathcal{C}$ as follows:*

$\mathsf{KGen}(1^\lambda)$**:** *Set $tk \leftarrow_R$ $\mathsf{C.KGen}(1^\lambda)$ and compute corresponding key-specific class identifier cid_{tk}. Output key pair $(sk, vk) = (tk, cid_{tk})$.*

$\mathsf{Sign}(sk, m)$**:** *Compute $\tau \leftarrow_R \mathsf{Tag}(tk, m)$ and output signature $\sigma = \tau$.*

$\mathsf{Ver}(vk, m, \sigma)$**:** *Output 1 if $\mathsf{Classify}(m, \tau) = cid_{tk}$. Otherwise, output 0.*

Correctness of CMT-Sig follows from the correctness of C. Observe that, regarding its security statement, we had to strengthen the precondition in comparison to Theorem 3 for a technical reason.

Theorem 5 (Security of CMT-Sig). *If $\mathcal{M}$ has super-polynomial cardinality, the signature scheme* CMT-Sig *is (strongly) unforgeable if the underlying CMT scheme* C *is (strongly) unforgeable.*

Proof. We reduce the unforgeability of the CMT-Sig signature scheme to the unforgeability of CMT scheme C. The proof can easily be adapted to cover the strong variants of the corresponding security notions.

From an efficient adversary $\mathcal{A}$ against euf-cma of CMT-Sig we construct an efficient adversary $\mathcal{B}$ against cmt-euf of C. Concretely, after fixing a finite subset $\mathcal{M}_0 \subseteq \mathcal{M}$ of super-polynomial size, $\mathcal{B}(1^\lambda)$ starts by picking a random message $m_0 \leftarrow_R \mathcal{M}_0$, posing a $\tau_0 \leftarrow \mathcal{O}_{\mathsf{Tag}}(m_0)$ query to its tagging oracle, and computing key-specific class identifier $cid \leftarrow \mathsf{Classify}(m_0, \tau_0)$. Then $\mathcal{B}$ runs $\mathcal{A}$, on input $vk = cid$, as a subroutine. Each oracle query $\mathcal{O}_{\mathsf{Sign}}(m)$ posed by $\mathcal{A}$ is forwarded by $\mathcal{B}$ as $\mathcal{O}_{\mathsf{Tag}}(m)$ to its own oracle, and the result is relayed to $\mathcal{A}$ (observe that this is a perfect simulation). Finally, when $\mathcal{A}$ outputs a candidate signature forgery (m^*, σ^*), $\mathcal{B}$ outputs the same pair and stops.

Assume $\mathcal{A}$ is successful, i.e., (m^*, σ^*) is a valid signature forgery. By definition we then have $\mathsf{Classify}(m^*, \sigma^*) = cid$. Further on, as m^* was never queried to the $\mathcal{O}_{\mathsf{Sign}}$ oracle, either we have $m^* = m_0$ (with negligible probability $\epsilon \leq 1/|\mathcal{M}_0|$), or m^* was not queried to $\mathcal{B}$'s $\mathcal{O}_{\mathsf{Tag}}$ oracle, i.e., adversary $\mathcal{B}$ is successful in forging a tag. We hence have $\mathrm{Succ}^{\mathsf{cmt\text{-}euf}}_{\mathsf{C},\mathcal{B}}(\lambda) = \mathrm{Succ}^{\mathsf{euf\text{-}cma}}_{\mathsf{CMT\text{-}Sig},\mathcal{A}}(\lambda) - \epsilon$. As both the left-hand side and ϵ are negligible by assumption, this completes the proof. $\square$

[4] Note that, in a different context, a similar approach is discussed by Katz [29, Section 2.4.2]. However, our construction is more efficient than his as our signatures do not need to explicitly carry the verification key.

5 Practical classifiable message tagging constructions

We focus next on practical, efficiency-optimized CMT constructions. Our motivation comes from the observation that the generic methods from Construction 4 demand a signature scheme's verification key to be included in every tag; as we reveal, this is not required in direct, i.e., non-generic constructions. Indeed, we propose two CMT schemes with more compact tags: one defined over a prime-order cyclic group with security under the DLP assumption in the random oracle model, and one based on pairings that does not require random oracles. Interestingly, both constructions are implicitly based on signature schemes where the verification key can be *uniquely reconstructed* from any (valid) signature.

We additionally note that we did not succeed in constructing efficient CMT (or LMT) schemes with security based on the hardness of factoring integers: in all potential schemes we identified, the corresponding (RSA) modulus had to be included in the tags. As such moduli are typically quite large (1024–4096 bits), the efficiency benefits obtainable by a direct construction in comparison to our generic methods are rather small. Indeed, all considered schemes based on RSA or factoring were easily outperformed by our non-factoring-based proposals. See Section 5.4 for more details about our approach to find factoring-based CMT schemes.

We finally point out that in Section 6 we consider standardized email authentication schemes (OpenPGP and S/MIME) and analyze whether they can be reinterpreted as CMT (or LMT) schemes.

5.1 A direct CMT construction without random oracles

Recall that the BLS-LMT scheme from Construction 1 is related to the (pairing-based) BLS signature scheme; as noted in Remark 2 it does not seem to be convertible to a CMT scheme. However, the also pairing-based Waters signature scheme (cf. Waters [47] and Figure 6), which is provably secure in the standard model under the CDH assumption, gives rise to the following CMT scheme.

Construction 6 (CMT scheme Waters-CMT). *The CMT scheme* Waters-CMT *is defined in respect to a (symmetric) bilinear group $\mathbb{G}$ and a hash function $H\colon \mathcal{M} \to \{0,1\}^k$, for an integer $k \in \mathbb{N}$. More precisely, for a set $(\mathbb{G}, \mathbb{G}_T, q, g, e)$ it is assumed that $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order q and that $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map. For fixed $u', u_1, \ldots, u_k \in \mathbb{G}$ define functions $W\colon \{0,1\}^k \to \mathbb{G}$; $(m_1, \ldots, m_k) \mapsto u' \prod_{i=1}^{k} (u_i)^{m_i}$ and $H' = W \circ H$. The tag space of* Waters-CMT *is $\mathcal{T} = \mathbb{G} \times \mathbb{G}$, the class identifier space is $\mathcal{C} = \mathbb{G}_T$, and its algorithms are specified in Figure 6.*

Correctness of Waters-CMT is due to the fact that for all $m_1, m_2 \in \mathcal{M}$ and $r_1, r_2 \in \mathbb{Z}_q$ we have

$$e(g^x H'(m_1)^{r_1}, g)/e(H'(m_1), g^{r_1}) = e(g,g)^x = e(g^x H'(m_2)^{r_2}, g)/e(H'(m_2), g^{r_2}).$$

In other words, for tagging key tk we have the key-specific class identifier $cid_{tk} = e(g,g)^{tk}$ (which corresponds exactly with the verification key in the Waters scheme). We next analyze the security properties of Waters-CMT.

Theorem 6 (Security of Waters-CMT). *The CMT scheme* Waters-CMT *is unforgeable (*cmt-euf*) if the CDH assumption holds in $\mathbb{G}$ and H is collision-resistant, in the common reference string (CRS) model.*

Proof. We reduce the unforgeability of Waters-CMT to the existential unforgeability of the Waters signature scheme (defined over the same bilinear group). Intuitively, together with Fact 3

```
KGen$(1^\lambda)$:
  $x \leftarrow_R \mathbb{Z}_q$
  $X \leftarrow e(g,g)^x$
  Output $(sk, vk) = (x, X)$

Sign$(sk, m)$:
  $r \leftarrow_R \mathbb{Z}_q$
  $\sigma_1 \leftarrow g^x H'(m)^r$
  $\sigma_2 \leftarrow g^r$
  Output $\sigma = (\sigma_1, \sigma_2)$

Ver$(vk, m, \sigma)$:
  Parse $\sigma$ as $(\sigma_1, \sigma_2)$
  If $e(\sigma_1, g)/e(H'(m), \sigma_2) = X$
    output 1,
    else output 0
- - - - - - - - - - - - - - - - - - - - - - - -
KGen$(1^\lambda)$:
  $x \leftarrow_R \mathbb{Z}_q$
  Output $tk = x$

Tag$(tk, m)$:
  $r \leftarrow_R \mathbb{Z}_q$
  $\tau_1 \leftarrow g^x H'(m)^r$
  $\tau_2 \leftarrow g^r$
  Output $\tau = (\tau_1, \tau_2)$

Classify$(m, \tau)$:
  Parse $\tau$ as $(\tau_1, \tau_2)$
  $cid \leftarrow e(\tau_1, g)/e(H'(m), \tau_2)$
  Output $cid$
```

Fig. 6. Signature scheme Waters (top) and CMT scheme Waters-CMT (bottom). For further details see Construction 6.

below, this establishes the statement. However, due to the fact that we let all entities share a common set of elements $u', u_1, \ldots, u_k \in \mathbb{G}$ for function W, the security reduction technically requires these elements to be specified in a CRS.

From an efficient adversary $\mathcal{A}$ against cmt-euf of Waters-CMT we construct an efficient adversary $\mathcal{B}$ against euf-cma of Waters. Concretely, $\mathcal{B}(vk)$ runs $\mathcal{A}(1^\lambda)$ as a subroutine. Each oracle query $\mathcal{O}_{\mathsf{Tag}}(m)$ posed by $\mathcal{A}$ is forwarded by $\mathcal{B}$ as $\mathcal{O}_{\mathsf{Sign}}(m)$ to its own oracle, and the result is relayed to $\mathcal{A}$ (observe that this is a perfect simulation). Finally, when $\mathcal{A}$ outputs a candidate tag forgery (m^*, τ^*), $\mathcal{B}$ outputs the same pair and stops.

Assume $\mathcal{A}$ is successful, i.e., (m^*, τ^*) with $\tau^* = (\tau_1^*, \tau_2^*)$ is a valid tag forgery. By definition we then have $e(\tau_1^*, g)/e(H'(m^*), \tau_2^*) = cid_{tk} = e(g,g)^{tk}$, hence in particular τ^* is a valid Waters signature on m^*. Further on, as m^* was never queried to the $\mathcal{O}_{\mathsf{Tag}}$ oracle, it was also not queried to $\mathcal{B}$'s $\mathcal{O}_{\mathsf{Sign}}$ oracle, i.e., adversary $\mathcal{B}$ is successful in forging a signature. We hence have $\mathrm{Succ}^{\mathsf{euf\text{-}cma}}_{\mathsf{Waters},\mathcal{B}}(\lambda) = \mathrm{Succ}^{\mathsf{cmt\text{-}euf}}_{\mathsf{Waters\text{-}CMT},\mathcal{A}}(\lambda)$. As the left-hand side is negligible by assumption, this completes the proof. □

Fact 3 (Unforgeability of Waters signature scheme [47]). *The Waters signature scheme is existentially unforgeable (euf-cma) if the CDH assumption holds in $\mathbb{G}$ and H is collision-resistant. The scheme is rerandomizable and hence not strongly unforgeable.*

5.2 A highly practical CMT scheme based on the DLP

Our second CMT scheme is even more efficient than Waters-CMT, as it requires no pairing operation but is instead defined over a regular prime-order group. More precisely, it is based on the Schnorr signature scheme (cf. Schnorr [43,44] and Figure 7) and exploits the fact that in the latter the verification key can be reconstructed from any valid signature, requiring only one (multi-)exponentiation. Interestingly this does not hold for the related signature schemes DSA and ECDSA, as we explore in Section 5.3. We specify the new scheme as follows:

Construction 7 (CMT scheme Schnorr-CMT). *The CMT scheme Schnorr-CMT is defined in respect to a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q and a hash function $H\colon \mathcal{M} \times \mathbb{G} \to \mathbb{Z}_q$. The tag space of Schnorr-CMT is $\mathcal{T} = \mathbb{G} \times \mathbb{Z}_q$, the class identifier space is $\mathcal{C} = \mathbb{G}$, and its algorithms are specified in Figure 7.*

For the correctness of Schnorr-CMT observe that for all $m_1, m_2 \in \mathcal{M}$ and $r_1, r_2 \in \mathbb{Z}_q$ we have

$$(g^{s_1}/R_1)^{1/H(m_1,R_1)} = (g^{k_1+xc_1}/g^{k_1})^{1/c_1} = g^x = (g^{k_2+xc_2}/g^{k_2})^{1/c_2} = (g^{s_2}/R_2)^{1/H(m_2,R_2)}.$$

```
KGen(1^λ):
    $x \leftarrow_R \mathbb{Z}_q$
    $X \leftarrow g^x$
    Output $(sk, vk) = (x, X)$

Sign($sk, m$):
    $k \leftarrow_R \mathbb{Z}_q$
    $R \leftarrow g^k$
    $c \leftarrow H(m, R)$
    $s \leftarrow k + xc \bmod q$
    Output $\sigma = (R, s)$

Ver($vk, m, \sigma$):
    Parse $\sigma$ as $(R, s)$
    If $g^s = RX^{H(m,R)}$
        output 1,
        else output 0
```

```
KGen(1^λ):
    $x \leftarrow_R \mathbb{Z}_q$
    Output $tk = x$

Tag($tk, m$):
    $k \leftarrow_R \mathbb{Z}_q$
    $R \leftarrow g^k$
    $c \leftarrow H(m, R)$
    $s \leftarrow k + xc \bmod q$
    Output $\tau = (R, s)$

Classify($m, \tau$):
    Parse $\tau$ as $(R, s)$
    $cid \leftarrow (g^s/R)^{1/H(m,R)}$
    Output $cid$
```

Fig. 7. Signature scheme Schnorr (top) and CMT scheme Schnorr-CMT (bottom). For further details see Construction 7.

In other words, for tagging key tk we have the key-specific class identifier $cid_{tk} = g^{tk}$ (which corresponds exactly with the verification key in the Schnorr scheme). We establish the security of Schnorr-CMT as follows.

Theorem 7 (Security of Schnorr-CMT). *The CMT scheme Schnorr-CMT is strongly unforgeable (cmt-suf) if the DLP is hard in $\mathbb{G}$ and H is modeled as a random oracle. By consequence, the scheme is also unforgeable (cmt-euf).*

Proof. We reduce the strong unforgeability of Schnorr-CMT to the strong existential unforgeability of the Schnorr signature scheme. Together with Fact 4 below, this establishes the statement.

From an efficient adversary $\mathcal{A}$ against cmt-suf of Schnorr-CMT we construct an efficient adversary $\mathcal{B}$ against suf-cma of Schnorr. Concretely, $\mathcal{B}(vk)$ runs $\mathcal{A}(1^\lambda)$ as a subroutine. Each oracle query $\mathcal{O}_{\mathsf{Tag}}(m)$ posed by $\mathcal{A}$ is forwarded by $\mathcal{B}$ as $\mathcal{O}_{\mathsf{Sign}}(m)$ to its own oracle, and the result is relayed to $\mathcal{A}$ (observe that this is a perfect simulation). Finally, when $\mathcal{A}$ outputs a candidate tag forgery (m^*, τ^*), $\mathcal{B}$ outputs the same pair and stops.

Assume $\mathcal{A}$ is successful, i.e., (m^*, τ^*) with $\tau^* = (R^*, s^*)$ is a valid tag forgery. By definition we then have $(g^{s^*}/R^*)^{1/H(m^*,R^*)} = cid_{tk} = g^{tk}$, i.e., $g^{s^*} = R^*(g^{tk})^{H(m^*,R^*)}$, hence in particular τ^* is a valid Schnorr signature on m^*. Further on, as m^* was never queried to the $\mathcal{O}_{\mathsf{Tag}}$ oracle with answer τ^*, it was also not queried to $\mathcal{B}$'s $\mathcal{O}_{\mathsf{Sign}}$ oracle with that answer, i.e., adversary $\mathcal{B}$ is successful in forging a signature. We hence have $\mathrm{Succ}^{\mathsf{suf\text{-}cma}}_{\mathsf{Schnorr},\mathcal{B}}(\lambda) = \mathrm{Succ}^{\mathsf{cmt\text{-}suf}}_{\mathsf{Schnorr\text{-}CMT},\mathcal{A}}(\lambda)$. As the left-hand side is negligible by assumption, this completes the proof. □

Fact 4 (Unforgeability of Schnorr signature scheme [38, 39]). *The Schnorr signature scheme is strongly existentially unforgeable (suf-cma) if DLP is hard in $\mathbb{G}$ and H is modeled as a random oracle.*[5]

Remark 3 (Efficiency of Schnorr-CMT). The practical efficiency of Schnorr-CMT primarily depends on the choice of the underlying cyclic group. One of the fastest available implementations of the Schnorr signature scheme is the elliptic-curve-based Ed25519 by Bernstein *et al.* [6], with a reported 110000 and 70000 signing and verification operations per second, respectively, on a current high-end CPU. After studying the corresponding source code[6] we can confirm that

[5] Although the statement proven by Pointcheval and Stern [39] considers only euf-cma security, their results can readily be extended to also cover the suf-cma notion.

[6] Available under a free license at http://bench.cr.yp.to/supercop.html.

```
KGen($1^\lambda$):
    $x \leftarrow_R \mathbb{Z}_q$
    $X \leftarrow g^x$
    Output $(sk, vk) = (x, X)$

Sign($sk, m$):
    $k \leftarrow_R \mathbb{Z}_q$
    $r \leftarrow \varphi(g^k)$
    $s \leftarrow k^{-1}(H(m) + rx) \bmod q$
    Output $\sigma = (r, s)$

Ver($vk, m, \sigma$):
    Parse $\sigma$ as $(r, s)$
    If $\varphi\big((g^{H(m)} X^r)^{1/s}\big) = r$
        output 1,
        else output 0
```

Fig. 8. Signature schemes DSA and ECDSA. See Section 5.3 for further details. For clarity of exposition we omit tests demanded by the standards [35, 2] that ensure $r \neq 0 \neq s$.

Ed25519 signatures indeed follow the pattern $\sigma = (R, s)$ from Figure 7, i.e., they can be understood as CMT tags according to our construction (with a minimal modification, though: in Ed25519 the verification key vk is an auxiliary input to hash function H; removing it will turn the signature scheme into a CMT scheme without affecting its existential unforgeability). While, due to the added inversion in $\mathbb{Z}_q$, the Classify timings will be slightly less impressive than those of the original Ver routine, it seems conservative to expect the feasibility of 50000 CMT classifications per second.

5.3 On DSA- and ECDSA-based CMT schemes

Two variants of the Schnorr signature scheme are given by DSA [35] and ECDSA [2]. We point out that, so far, all formal results in support of the schemes' unforgeability are in the generic group model [12], i.e., more direct reductions to, say, the DLP are not known. Nevertheless, due to the great popularity and progress in standardization of DSA and ECDSA, we discuss in the following the feasibility of understanding corresponding signatures as CMT tags, in a way similar to that of Construction 7. Note that here we intentionally do not consider schemes that rely on modified versions of DSA or ECDSA, as corresponding changes would typically imply code adaptions on the tag-creation side, and in settings where this is acceptable our Schnorr-CMT from Section 5.2 seems to be the favorable option anyway (ultimately for the reason that its security is supported by a reduction to a standard hardness assumption).

Both DSA and ECDSA are defined in respect to a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q and a hash function $H\colon \mathcal{M} \to \mathbb{Z}_q$. Specifically, for DSA we have $\mathbb{G} \subseteq \mathbb{Z}_p^*$ for a suitable prime p, and for ECDSA we have $\mathbb{G} \subseteq E(K)$ for an elliptic curve defined over a finite field K. Both schemes additionally require a dedicated hash function $\varphi\colon \mathbb{G} \to \mathbb{Z}_q$; for DSA, assuming the canonical representation of elements $h \in \mathbb{Z}_p^*$ such that $h \in \{1, \ldots, p-1\}$, we have $\varphi\colon h \mapsto (h \bmod q)$, and for ECDSA we have $\varphi\colon P \mapsto e(P.x)$, i.e., each curve point is mapped to an encoding of its x-value. As DSA and ECDSA coincide in all remaining aspects, their specific algorithms are jointly reproduced in Figure 8.

Let us try to understand DSA and ECDSA as CMT schemes in a way similar to that of Construction 7, i.e., we aim at recovering verification keys from signatures. Inspection of the verification equation in Figure 8 shows that for the verification key X we have

$$X \in \Big((\varphi^{-1}(r))^s / g^{H(m)} \Big)^{1/r} .$$

That is, we might have a chance to recover X from $\sigma = (r, s)$ if preimages of φ can efficiently be computed. In the DSA case there seems to be no hope: for every $r \in \mathbb{Z}_q$ there exist $(p-1)/q$-many preimages $\varphi^{-1}(r)$ in $\mathbb{Z}_p^*$, and it seems impossible to efficiently identify from those the 'right one', i.e., the (hopefully unique) preimage that is in $\mathbb{G}$.[7]

[7] Observe that inverting φ would be possible if the index of $\mathbb{G}$ in $\mathbb{Z}_p^*$ was small (e.g., in the 'safe prime' setting where $p = 2q+1$); however, the DSA standard requires $|\mathbb{Z}_p^*|/|\mathbb{G}|$ to be large (e.g., $|\mathbb{Z}_p^*| \approx 2^{1024}$ and $|\mathbb{G}| \approx 2^{160}$).

At first sight, the ECDSA case looks more promising as we have $|\varphi^{-1}(r)| \leq 2$ for all $r \in \mathbb{Z}_q$. Generally, for each occurring value r two different preimages $\varphi^{-1}(r)$ can be efficiently determined through solving a simple quadratic equation in a finite field [11, Section 2.3.4]. However, it remains unclear how to identify among the two solutions the one that leads to the recovery of the right verification key X. Clearly, if we modified the Sign algorithm such that it either would include in the signatures a small piece of information that would allow identifying the right preimage of the particular value r [8], or such that it would iteratively generate signatures until one is obtained where a specific realization of φ^{-1} reconstructs the right preimage, then this would immediately give rise to an ECDSA-based CMT scheme. However, given that an efficiency decrease in the tag generation algorithm seems unavoidable, we don't see a convincing reason that would make the deployment of such a CMT scheme favorable over our Schnorr-CMT from Section 5.2, in particular as the latter can be instantiated over the same elliptic curve as ECDSA.

5.4 On CMT constructions from Fiat-Shamir-transformed signature schemes

As can be noticed from Figure 7, the Schnorr signature scheme and our Schnorr-CMT scheme have quite similar structure. The fact that the former is conceptually rooted in the Fiat-Shamir paradigm (that allows to generically transform certain identification schemes into signature schemes [17]) encourages to analyze whether also other Fiat-Shamir-based signature schemes yield efficient CMT schemes analogously.

We evaluated, besides Schnorr signatures, a total of four standard Fiat-Shamir-based signature schemes in respect to their amenability towards construction of CMT. More precisely, we considered the schemes by Goldwasser, Micali, and Rackoff [19], by Fiat and Shamir [17], by Guillou and Quisquater [23], and by Micali, Ong, and Schnorr [34, 36] (for details on the schemes see Katz [29]). However, we did not succeed in finding attractive CMT solutions basing on these schemes, for several reasons. First of all, the four schemes rely on the hardness of factoring integers, and it seems that the corresponding moduli $N = pq$ have to be included in every tag of a derived CMT scheme (see the discussion in the introduction of this section). Additionally, considering the scheme by Fiat and Shamir, its long verification key (consisting of λ elements in $\mathbb{Z}_N$, where λ is the security parameter) prevents an efficient CMT construction. Finally, for recovering the verification key from signatures (as we did in Sections 5.1 and 5.2), in the schemes by Guillou and Quisquater, and by Micali, Ong, and Schnorr, inversions modulo $\varphi(N)$ have to be computed without knowing factors p and q, which is impossible according to the schemes' underlying hardness assumptions.

Therefore, the Schnorr-CMT scheme remains our only direct and efficient construction derived from a Fiat-Shamir-transformed signature scheme.

6 Linkable message tagging from email authentication schemes

In the course of this paper we introduced and analyzed several LMT and CMT schemes. One of the envisioned applications is to ease email authentication by enabling corresponding client software to automatically decide whether any two incoming messages originate from the same sender—without demanding any prior exchange of (public) key material. On first sight it seems that using such schemes would require extending email clients by additional software components. However, in Sections 3.3 and 4.3 we learned that LMT and CMT schemes are closely related to signature schemes. As most modern email clients have built-in support for the latter (e.g., following the OpenPGP or S/MIME standards), it is an interesting question whether signatures created by standardized email authentication schemes can be (mis-)used to realize

[8] A similar technique is proposed in [11, Section 4.1.6].

LMT/CMT functionality. If this is the case, while on the recipient's side specialized software would still be required, on the side of the sender a standard email client could suffice.

6.1 OpenPGP

A well-known software product that makes available strong encryption and authentication to the masses is marketed since 1991 under the name of *Pretty Good Privacy* (PGP). The tool combines symmetric and asymmetric cryptographic primitives to implement reasonably practical encryption and signing functionality on basis of a decentralized public key infrastructure, the *web of trust*. In reaction to the steady increase of its worldwide popularity, standardization efforts at IETF started in 1996 and resulted in the specification of cryptographical algorithms and message formats in a series of RFCs [3, 14, 13], all subsumed under the name of OpenPGP.

Relevant for this paper are the algorithms that OpenPGP employs for digital signing: RSA and DSA. Precisely, RFC 4880 in Sections 5.2.2 and 5.2.3 mandates signatures to be created either with RSA and PKCS #1 v1.5 padding [27, Section 9.2], or with DSA. While in Section 5.3 we already came to the conclusion that DSA signatures do not yield an LMT scheme, it is also unclear how to use PKCS #1 v1.5 signatures in the LMT sense, leave alone CMT, without knowing the public modulus N (which is not transmitted with every message). With other words, it seems that signatures crafted in compliance with the OpenPGP framework cannot be used in the way envisioned in this paper.

6.2 S/MIME

The second important standard for email encryption and authentication is S/MIME [40, 41]. It builds on the *Cryptographic Message Syntax* [24–26] from PKCS #7 and presumes authenticated distribution of public keys through a X.509 hierarchical PKI. Supported signature algorithms include RSA (PKCS #1 v1.5 [5] and RSA-PSS [42]), DSA and ECDSA (based on SHA-1 or SHA-2, see [5, 15]), and a form of the GOST signature scheme [31].

In contrast to the OpenPGP case, a characteristic feature of S/MIME allows us to reinterpret its signatures in the CMT sense. More specifically, signatures in S/MIME are usually transmitted together with a certificate (issued either by a superordinate CA or by the signing key itself) which includes, among others, the verification key of the signer. This makes Construction 4 readily applicable to obtain CMT functionality. Aiming at a simple proof-of-concept, we used the `openssl` toolset [37] to generate a self-signed S/MIME signature key. We imported it into the Thunderbird email client and used it for authenticating outgoing emails. As expected, on the receiver side, the signer's verification key could be extracted from the PKCS #7-formatted signatures, again with the `openssl` tools.

7 Conclusion and open problems

This paper approaches the key distribution problem of classical signature schemes from an entirely new direction: Our notions of linkable and classifiable message tagging (LMT and CMT) allow to unambiguously decide whether pairs of messages and authentication tags originate from the same source (i.e., tagging key). This is achieved without requiring a special setup (e.g., a pre-shared authentic verification key or a PKI). We construct secure instantiations of the new primitives, both with and without random oracles, based on different hardness assumptions. Our most efficient scheme is ready for practical deployment, allowing the generation and classification of expected 110000 and 50000 tags per second, respectively, on a recent CPU.

The LMT concept illustrates the elusive boundary between symmetric and asymmetric cryptography: On the one hand corresponding schemes do not involve public keys, indicating that the concept is mainly symmetric. On the other hand we reveal tight connections between LMT/CMT schemes and signature schemes, documenting that there is little hope to build practical LMT schemes from just symmetric primitives like blockciphers or hash functions—indeed, any LMT or CMT construction based on such primitives would immediately yield a signature scheme of similar efficiency. This connection is further substantiated by our direct constructions that also exhibit certain similarities with standard signature schemes.

We leave for future research the exact characterization of those signature schemes that have natural LMT or CMT analogues. For instance, for the signature schemes by Boneh and Boyen [8] we were not able to identify corresponding message tagging schemes.

Acknowledgments

The authors thank all anonymous reviewers for their valuable comments. Both authors were supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE, and B. Poettering additionally by EPSRC Leadership Fellowship EP/H005455/1 and a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation. This work has been co-funded by the German Research Foundation (DFG) as part of project S4 within the CRC 1119 CROSSING.

References

1. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In C.-S. Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 452–473. Springer, Nov. / Dec. 2003.
2. American National Standard for Financial Services. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA) (ANS X9.62-2005), Nov. 2005.
3. D. Atkins, W. Stallings, and P. Zimmermann. PGP Message Exchange Formats. RFC 1991 (Informational), Aug. 1996. Obsoleted by RFC 4880.
4. D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS 2002*. The Internet Society, Feb. 2002.
5. L. Bassham, W. Polk, and R. Housley. Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3279 (Proposed Standard), Apr. 2002. Updated by RFCs 4055, 4491, 5480, 5758.
6. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. In B. Preneel and T. Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 124–142. Springer, Sept. / Oct. 2011.
7. S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In H. Imai and Y. Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 154–170. Springer, Mar. 1999.
8. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004.
9. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Dec. 2001.
10. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, Sept. 2004.
11. D. Brown. Certicom Research, Standards for Efficient Cryptography Group (SECG) — SEC 1: Elliptic Curve Cryptography. `http://www.secg.org/download/aid-780/sec1-v2.pdf`, May 21, 2009. Version 2.0.
12. D. R. L. Brown. Generic groups, collision resistance, and ECDSA. *Designs, Codes and Cryptography*, 35(1):119–152, Apr. 2005.
13. J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP Message Format. RFC 4880 (Proposed Standard), Nov. 2007. Updated by RFC 5581.
14. J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP Message Format. RFC 2440 (Proposed Standard), Nov. 1998. Obsoleted by RFC 4880.
15. Q. Dang, S. Santesson, K. Moriarty, D. Brown, and T. Polk. Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA. RFC 5758 (Proposed Standard), Jan. 2010.
16. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

17. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Aug. 1987.
18. Fox-IT. Black Tulip — Report of the investigation into the DigiNotar Certificate Authority breach. http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update/black-tulip-update.pdf, Aug. 2012.
19. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
20. S. Goldwasser, S. Micali, and R. L. Rivest. A "paradoxical" solution to the signature problem (abstract) (impromptu talk). In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, page 467. Springer, Aug. 1985.
21. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.
22. Google Online Security Blog. Maintaining digital certificate security. http://googleonlinesecurity.blogspot.de/2014/07/maintaining-digital-certificate-security.html, July 2014.
23. L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 216–231. Springer, Aug. 1990.
24. R. Housley. Cryptographic Message Syntax (CMS). RFC 3369 (Proposed Standard), Aug. 2002. Obsoleted by RFC 3852.
25. R. Housley. Cryptographic Message Syntax (CMS). RFC 3852 (Proposed Standard), July 2004. Obsoleted by RFC 5652, updated by RFCs 4853, 5083.
26. R. Housley. Cryptographic Message Syntax (CMS). RFC 5652 (INTERNET STANDARD), Sept. 2009.
27. J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), Feb. 2003.
28. B. Kaliski. PKCS #7: Cryptographic Message Syntax Version 1.5. RFC 2315 (Informational), Mar. 1998.
29. J. Katz. *Digital Signatures*. Springer, 2010. ISBN 978-0387277110.
30. N. Koblitz and A. Menezes. Another look at security definitions. *Advances in Mathematics of Communications*, 7(1):1–38, Feb. 2013.
31. S. Leontiev and D. Shefanovski. Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 4491 (Proposed Standard), May 2006.
32. A. Mashatan and S. Vaudenay. A message recognition protocol based on standard assumptions. In J. Zhou and M. Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 384–401. Springer, June 2010.
33. A. Menezes and N. P. Smart. Security of signature schemes in a multi-user setting. *Designs, Codes and Cryptography*, 33(3):261–274, Nov. 2004.
34. S. Micali. A secure and efficient digital signature algorithm. Technical Memo MIT/LCS/TM-501b, Massachusetts Institute of Technology, Laboratory for Computer Science, Apr. 1994.
35. National Institute of Standards and Technology. Digital Signature Standard (DSS) (FIPS PUB 186-4), July 2013.
36. H. Ong and C.-P. Schnorr. Fast signature generation with a Fiat-Shamir-like scheme. In I. Damgård, editor, *EUROCRYPT'90*, volume 473 of *LNCS*, pages 432–440. Springer, May 1990.
37. OpenSSL Project. Open Source Secure Sockets Layer and Transport Layer Security implementation. http://www.openssl.org.
38. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, May 1996.
39. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
40. B. Ramsdell. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. RFC 3851 (Proposed Standard), July 2004. Obsoleted by RFC 5751.
41. B. Ramsdell and S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751 (Proposed Standard), Jan. 2010.
42. J. Schaad, B. Kaliski, and R. Housley. Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 4055 (Proposed Standard), June 2005. Updated by RFC 5756.
43. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Aug. 1990.
44. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
45. A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Aug. 1985.
46. TURKTRUST Information Security Services Inc. Public Announcements. http://turktrust.com.tr/en/kamuoyu-aciklamasi-en.html, Jan. 2013.

47. B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005.
48. A. Weimerskirch and D. Westhoff. Zero common-knowledge authentication for pervasive networks. In M. Matsui and R. J. Zuccherato, editors, *SAC 2003*, volume 3006 of *LNCS*, pages 73–87. Springer, Aug. 2004.
49. A. Whitten and J. D. Tygar. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8*, SSYM'99, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.