Message-Recovery MACs and Verification-Unskippable AE

Shoichi Hirose[1], Yu Sasaki[2], and Kan Yasuda[2]

[1] University of Fukui, Fukui, Japan
hrs_shch@u-fukui.ac.jp
[2] NTT Secure Platform Laboratories, Tokyo, Japan
{sasaki.yu,yasuda.kan}@lab.ntt.co.jp

Abstract. This paper explores a new type of MACs called message-recovery MACs (MRMACs). MRMACs have an additional input R that gets recovered upon verification. Receivers must execute verification in order to recover R, making the verification process unskippable. Such a feature helps avoid mis-implementing verification algorithms. The syntax and security notions of MRMACs are rigorously formulated. In particular, we formalize the notion of unskippability and present a construction of an unskippable MRMAC from a tweakable cipher and a universal hash function. Our construction is provided with formal security proofs. We extend the idea of MRMACs to a new type of authenticated encryption called verification-unskippable AE (VUAE). We propose a generic Enc-then-MRMAC composition which realizes VUAE. The encryption part needs to satisfy a new security notion called one-time undecipherability. We provide three constructions that are one-time undecipherable, and they are proven secure under various security models.
Keywords: message recovery MACs, authenticated encryption, unskippability, one-time undecipherability, CTR mode, Even-Mansour, FX.

1 Introduction

Message authentication is fundamental to secure communication, ensuring that a message that a user received has not been modified from the original content. Authenticity can be achieved both in the public-key (digital signatures) and symmetric-key (message authentication codes, in short MACs) frameworks.

The importance of message authenticity can be emphasized by the case of traffic signal: green, yellow or red must not be modified in any way. This case also tells us that all receivers must perform the verification procedure correctly, or otherwise the reliability of the system would get severely compromised.

Authenticated encryption (AE) provides both authenticity and confidentiality in a single scheme. It is now actively discussed in the CAESAR competition [12]. Many AE schemes take the verify-then-decrypt approach to their decryption process, where the decryption takes place only after the verification succeeds. If decrypted data are released before verification—which is called *decryption-misuse* [1, 2] or *releasing unverified plaintext* (RUP) [4, 5]—security

is no longer guaranteed, often leading to forgery or plaintext recovery. Therefore, the execution of verification is crucial for many AE schemes.

For authenticity, extra bandwidth is inevitable due to the fact that additional information for verification needs to be sent together with the message itself. In the public-key framework, *message-recovery signatures* [23] mitigate the increase in bandwidth. The idea is to construct a signature scheme with which part of the message can be recovered during verification, so that only the remaining part of the message needs to be sent. Namely, it produces a signature Σ taking three inputs: a private key $K_{\mathtt{priv}}$, a recoverable part R and a (non-recoverable) message M. Any user can verify the integrity of (M, Σ) using the corresponding public key $K_{\mathtt{pub}}$, recovering R upon verification.

In general, digital signatures with additional functionalities are realized by exploiting some mathematical properties. It is not clear how to add similar functionalities to MACs that are constructed from only symmetric-key primitives. To the best of our knowledge, the aggregate MAC [17, 21] seems to be the only concrete example having relevant applications.

Our Contributions.

- This paper proposes a new type of MAC with additional features called *message-recovery MAC* (MRMAC), which is a symmetric-key counterpart of the message-recovery signature. It takes three inputs: a pair of keys (K_1, K_2), an r-bit recoverable part R, and arbitrary length message M. Let $\tilde{E}_K : \{0,1\}^\ell \times \{0,1\}^\tau \to \{0,1\}^\tau$ and $H_K : \{0,1\}^* \to \{0,1\}^\ell$ be a tweakable block cipher (TBC) and a universal hash function using K, respectively. The tag generation first concatenates $\tau - r$ bits of zeros to R. Then, a tag T is computed by $\tilde{E}_{K_1}(H_{K_2}(M), R\|0^{\tau-r})$ as illustrated in Fig. 1. Verification first computes $H_{K_2}(M)$ and computes decryption of TBC, $\tilde{E}^{-1}_{K_1}(H_{K_2}(M), T)$, to check if the resultant plaintext P contains $\tau - r$ bits of zeros in the proper position. If verification succeeds, r bits of P is output as R.

  The property that the recoverable part R is recovered during verification makes the verification process unskippable. This prevents mis-implementation of verification algorithms thus improves reliability of the system.

  We formalize the syntax of MRMAC and define security notions. In particular, the property that R cannot be recovered without executing verification is defined as *unskippability*, which is new and particular to MRMAC. We prove security of our scheme with respect to the proposed notions.

- We then propose new type of AE called *verification-unskippable AE* (VUAE) by instantiating MRMAC for the MAC part. We start by showing a generic composition *Enc-then-MRMAC*; combining a particular type of encryption and MRMAC in the Enc-then-MAC manner leads to VUAE. Here, the encryption scheme needs to be *one-time encryption* satisfying the security notion which we call *one-time undecipherability*. One-time encryption uses a secret random string S which essentially acts as a key but is *one-time*, meaning that S is freshly re-sampled upon each execution of encryption. The existence

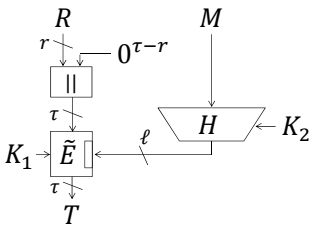

Fig. 1. Message-recovery MAC

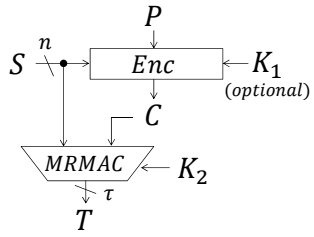

Fig. 2. Enc-then-MRMAC composition

Table 1. Security models of each construction

Construction	Security Models		Efficiency
	unskippability	AE security	(additional requirement)
One-time CTR	standard model	standard model	recomputing key schedule
FX + CTR	ideal cipher	standard model	using K_{enc}
EM + CTR	random permutation	random permutation	-

of S makes the use of a (standard) key K optional. One-time undecipherability is informally described that an adversary who knows K but does not know S cannot distinguish a decryption oracle from a random oracle.

In more details, the sender and receiver first share a key for MAC, K_2, and if necessary a key for encryption, K_1. The sender generates a random one-time string S and computes an encryption scheme $\mathcal{E}$ as $C \leftarrow \mathcal{E}_s(M)$ or $C \leftarrow \mathcal{E}_{s,K_1}(M)$, where subscripts of $\mathcal{E}$ represent keys. It then computes MRMAC by taking S as the recoverable part and C as the message. The composition is illustrated in Fig. 2. The receiver needs S for starting decryption which is recovered during verification. Thus verification unskippability is achieved.

We propose three constructions of encryption schemes satisfying one-time undecipherability; based on CTR mode, based on the FX construction + CTR mode, and based on the Even-Mansour (EM) construction + CTR mode. Security of those are proven in the security model shown in Table 1.

- The discussion so far is about unskippability of standalone MAC and of AE (encryption + MAC). We thus extend it to a layered MAC (MAC + MAC) for the sake of completeness of this topic.

Paper Outline. Section 2 introduces related work. Section 3 formalizes syntax and security notions of MRMAC. Section 4 proposes our construction of MRMAC and proves its security. Section 5 discusses the application of MRMAC to key-wrapping. Section 6 formalizes syntax and security notions of VUAE, and proposes a generic composition, ENC-then-MRMAC. Section 7 proposes three constructions of VUAE and proves their security. Section 8 extends VUAE for handling associated data and applies MRMAC to layered protocols.

2 Related Work

In Section 2.1 we name some previous work related to the idea of message-recovery MACs. In Section 2.2 we name those related to the idea of bypassing verification. We also mention in Section 2.3 some previous work that handles security notions in settings similar to the current work.

2.1 Recovering Information during Verification

The idea of recovering some information during verification has appeared several times in previous work. Some are in the context of MAC, while others in the context of AE. Here we list some of the major ones that we are aware of.

Aggregate MACs. The notion of aggregate MACs was introduced by Katz and Lindell [21]. Its use case—relaying messages and tags in a sensor network—motivates us to study the notion of layered MACs, which we discuss in Section 8.

Eikemeier et al. [17] proposed history-free aggregate MACs by utilizing partially invertible MACs, in which the finalization part (often a block-cipher call) inverts an n-bit tag T. The partial inversion property is used only for the purpose of proving the security of their construction.

Incremental MACs and AE Schemes. Incrementality of a tag generation algorithm $\mathcal{G}(\cdot)$ is a property that once a tag $T = \mathcal{G}(M)$ for input M is stored, the tag $T' = \mathcal{G}(M')$ for a slightly modified input M' can be updated from $\mathcal{G}(M)$ more efficiently than computing $\mathcal{G}(M')$ from scratch. The notion of incrementality was discussed for MACs [6], AE [13] and AEAD [26]. In order to be equipped with incrementality, the schemes need to adopt an invertible finalization so that the state value after processing all the message blocks (typically the XOR of computation results for all blocks) can be recovered from the tag.

Secret Message Numbers (SMN). The CAESAR competition allows to use SMN, which can take a role of a nonce, but can be sent through the communicated channel only after being encrypted. The protected-IV (PIV) construction [27] accommodates SMN; namely, an initialization vector (IV) is encrypted to what they call the reconstruction information (RI), and only the RI (and not IV) is sent. The receiver can recover IV from RI and ciphertext. We note that PIV is a three-pass construction and its verification is done after decryption.

2.2 Protection against Verification Skipping

To the best of our knowledge, there have been no proposed MAC schemes that explicitly claim security against legitimate users skipping verification. For AE, several approaches have been proposed in order to mitigate damages that would be caused by skipping verification.

Robust Authenticated Encryption. One novel approach is *robust authenticated encryption* [20], in which the decrypted "plaintext" would become completely random for any wrong inputs to the decryption algorithm, thus not causing any damage even if verification is done after decryption. Most of the robust AE constructions tend to be inefficient. There have been several pieces of previous work that study the tradeoff between robust-like AE security and efficiency.

Releasing Unverified Plaintext (RUP). The event that decryption results (plaintext candidates) are released before verification is called *releasing unverified plaintext* (RUP). Andreeva et al. [4, 5] formalized security models under the RUP setting. A technique called *nonce decoy* has been proposed, which adds security to nonce-based AE schemes in such a way as the decryption results for wrong (converted) nonce would become random.

Similarly, a dedicated design APE [3] randomizes the decryption results for wrong tag input, and a conversion method tag-feedback [19] randomizes the decryption results for wrong associated data, (converted) nonce, and tag.

A number of approaches have been proposed to provide tradeoff between security against decryption misuse and efficiency. In this paper, we propose a quite different approach that *forces* the execution of verification.

2.3 "Keyless" and "Dual" Security Notions

We shall discuss the unskippability of message-recovery MACs, which is a security notion against legitimate users who know the secret key K. Security notions without a secret key or a randomized element have appeared in different contexts, e.g. message-locked encryption [8, 7]. Also, one of our constructions uses a block cipher keyed in two different ways, acting as a PRF when keyed either way. Such a notion of dual PRFs has been studied by Bellare and Lysyanskaya [9].

3 Message-Recovery MACs

In this section we discuss message-recovery MACs (MRMACs). Here we briefly discuss their similarities and differences as compared to message-recovery signatures and authenticated encryption. We describe the syntax of message-recovery MAC algorithms in Section 3.1 and give security definitions in Section 3.2.

Comparison with Message-Recovery Signatures. Message-recovery MACs are the symmetric-key counterpart of message-recovery signature schemes [23, 24]. However, there is a *significant difference in the security requirement on the recoverable part* of the message.

Recall that a message-recovery signature scheme produces a signature Σ taking three inputs: a private key $K_{\mathtt{priv}}$, a recoverable part R and a (non-recoverable) message M. Anyone can verify the integrity of a message-signature pair (M, Σ) using the corresponding public key $K_{\mathtt{pub}}$ of the signer, recovering

R upon verification. Similarly, a message-recovery MAC produces a tag T taking three inputs: a secret key K, a recoverable part R and a (non-recoverable) message M, but now only those users who know the secret key K can verify the integrity of the message-tag pair (M, T).

Note that while it does not make sense to try to protect the confidentiality of R in the case of signatures (because anyone can recover R using $K_{\mathtt{pub}}$), it *does* make sense in the case of MACs. Indeed, in this work it is crucial that those users who do not know K should not be able to obtain any information about R.

Comparison with (Deterministic) Authenticated Encryption. One may notice that MRMACs are analogous to deterministic authenticated encryption (DAE) [25]. The *difference lies in the length of the recoverable part.*

The analogy between MRMACs and DAE can be drawn by regarding "R for MRMAC" as "plaintext for DAE," "M for MRMAC" as "associated data for DAE" and the MAC tag T as "ciphertext-tag for DAE." However, in the current work, MRMACs and AE play completely different roles, and the analogy ends there. Indeed, we will use an MRMAC to build an AE scheme, where R is used as "a one-time secret string," M as "ciphertext for AE," and T as the tag itself.

For MRMACs, the length of a recoverable part R is limited to a fixed size, which is usually about the size of a security parameter, e.g. 128 bits. This contrasts sharply with (D)AE, where the space of plaintext is usually "huge," so that the (D)AE scheme can accommodate possibly lengthy plaintext. This is exactly where the RUP problem arises [4]. On the other hand, MRMAC is free of such an issue, because we are guaranteed that $|R|$ is fixed to a small value.

3.1 Syntax of Message-Recovery MACs

A message-recovery MAC Θ is a triplet of algorithms $\Theta = (\mathcal{K}, \mathcal{G}, \mathcal{V})$ with a correctness condition that should be satisfied between the algorithms. Each of the three algorithm is defined as follows:

1. **Key setup.** The key setup $\mathcal{K}$ is a randomized algorithm which takes no input and outputs a uniformly random key $K \xleftarrow{\$} \mathbb{K}$. The key space $\mathbb{K}$ may be divided into its "subkey" spaces as $\mathbb{K} = \mathbb{K}_1 \times \mathbb{K}_2 \times \cdots$, in which case we write $K = (K_1, K_2, \ldots)$, depending on "how many" (sub)keys the MRMAC construction requires. We write $K \leftarrow \mathcal{K}(\cdot)$ to denote the invocation of $\mathcal{K}$ and assigning the key value to variable K.
2. **Tag generation.** The generation algorithm $\mathcal{G}$ is a keyed family of deterministic, stateless functions. We have $\mathcal{G}_K : \mathbb{R} \times \mathbb{M} \to \mathbb{T}$ for each $K \in \mathbb{K}$, where we have a recoverable part $R \in \mathbb{R}$, a message $M \in \mathbb{M}$ and a tag $T \in \mathbb{T}$, so that $T \leftarrow \mathcal{G}_K(R, M)$.
3. **Verification with recovery.** The verification algorithm $\mathcal{V}$ is a keyed family of deterministic, stateless functions. It takes the form of $\mathcal{V}_K : \mathbb{M} \times \mathbb{T} \to \mathbb{R} \cup \{\perp\}$. The symbol $\perp$ implies verification failure. We demand that $\perp \notin \mathbb{R}$.

Given $\mathcal{K}$, $\mathcal{G}$ and $\mathcal{V}$ above, we require the following condition between them:

- **Correctness condition.** For any $R \in \mathbb{R}$ and $M \in \mathbb{M}$, we always have $\mathcal{V}_K(M,T) = R$ whenever $T \leftarrow \mathcal{G}_K(R,M)$ and $K \leftarrow \mathcal{K}(\cdot)$,

which simply ensures that we would get the correct value of R after verifying a legitimate pair (M,T) with which the value R was used.

3.2 Security Definitions for Message-Recovery MACs

We provide message-recovery MACs with three different security notions. The first one, *unforgeability*, corresponds to the ordinary MAC security which has been most commonly used. The second one, *indistinguishability*, ensures the confidentiality of the recoverable part. The last one, *unskippability*, is the key notion in the current work. Informally, unskippability says that even a legitimate verifier, who *holds the secret key* K, should not be able to recover R without performing the full verification procedure. It already sounds challenging, and we shall need to make a forcible argument for formalizing this notion.

Security Notations and Conventions. An adversary $\boldsymbol{A}$ is an oracle machine, possibly with a random tape. We write $\boldsymbol{A}^{\mathcal{O}_1(\cdot),\mathcal{O}_2(\cdot),\cdots}$ to mean the value returned by $\boldsymbol{A}$ after interacting with its oracles $\mathcal{O}_1(\cdot), \mathcal{O}_2(\cdot), \ldots$. We assume that $\boldsymbol{A}$ never repeats its queries to its oracle if the oracle is deterministic. We usually upper-bound by q the number of queries that $\boldsymbol{A}$ makes to its oracles. The running time of $\boldsymbol{A}$ includes the time to execute its oracles. A scheme $\Lambda = (\mathcal{S}_1, \mathcal{S}_2, \ldots)$ is a set of algorithms, usually the first one $\mathcal{S}_1$ being a key setup algorithm. An adversary $\boldsymbol{A}$ is given oracle access to some of the algorithms $\mathcal{S}_i$ after a key setup. We write $\mathrm{Adv}^{***}_{\Lambda}(\boldsymbol{A})$ to denote the *advantage* of an adversary $\boldsymbol{A}$ attacking a scheme Λ with respect to a security notion $***$.

Tag Unforgeability. This corresponds to the common security notion for ordinary MACs. We give an adversary $\boldsymbol{A}$ oracle access to the generation $\mathcal{G}_K(\cdot,\cdot)$ and verification $\mathcal{V}_K(\cdot,\cdot)$. We want to make sure that the adversary $\boldsymbol{A}$ cannot *forge*, or come up with a new pair of (M,T) that would make the $\mathcal{V}$-oracle accept the query (M,T) and return some string $R \in \mathbb{R}$ other than the reject symbol $\perp$. We allow multiple trials to the verification query, defining

$$\mathrm{Adv}^{\mathrm{mac}}_{\Theta}(\boldsymbol{A}) := \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\mathcal{V}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\perp(\cdot,\cdot)} = 1\big],$$

where the probabilities are taken over the choice of key $K \leftarrow \mathcal{K}(\cdot)$ and random coins used by the adversary, if any. The symbol $\perp(\cdot,\cdot)$ denotes a "reject oracle;" it accepts a query $(M,T) \in \mathbb{M} \times \mathbb{T}$, having the same interface as $\mathcal{V}_K(\cdot,\cdot)$, and always returns the reject symbol $\perp$. To avoid a trivial win, the adversary $\boldsymbol{A}$ is not allowed to make a query (M,T) to its $\mathcal{V}$-oracle if $\boldsymbol{A}$ has already made a $\mathcal{G}$-query (R,M) with some $R \in \mathbb{R}$ and the $\mathcal{G}$-oracle has returned $T \leftarrow \mathcal{G}_K(R,M)$.

Recoverable-Part Indistinguishability. This notion basically corresponds to protecting the confidentiality of the recoverable part R. We adopt the definition via indistinguishability. Let $\mathcal{G}_K(\$,\cdot)$ denote a "random-R oracle" which, upon receiving a query $(R, M) \in \mathbb{R} \times \mathbb{M}$, picks a random $R' \xleftarrow{\$} \mathbb{R}$ and returns $T' \leftarrow \mathcal{G}_K(R', M)$. An adversary $\boldsymbol{A}$ should not be able to distinguish between the real oracle $\mathcal{G}_K(\cdot,\cdot)$ and the random-R oracle. Succinctly, we define

$$\mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}) := \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\$,\cdot)} = 1\big],$$

where the probabilities are defined over $K \leftarrow \mathcal{K}(\cdot)$, choice of R' by the $\mathcal{G}_K(\$,\cdot)$ oracle, and random coins of the adversary $\boldsymbol{A}$, if any. The following lemma will be useful in Section 6. The lemma simply combines the unforgeability and the indistinguishability notions.

Lemma 1. *Let $\Theta = (\mathcal{K}, \mathcal{G}, \mathcal{V})$ be a message-recovery MAC and $K \leftarrow \mathcal{K}(\cdot)$. For any adversary $\boldsymbol{A}$ having access to $\mathcal{G}$-oracle and $\mathcal{V}$-oracle and making q queries, there exist an adversary $\boldsymbol{B}$ such that*

$$\Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\mathcal{V}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\$,\cdot),\perp(\cdot,\cdot)} = 1\big] \le \mathrm{Adv}^{\mathrm{mac}}_{\Theta}(\boldsymbol{A}) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{B}),$$

where $\boldsymbol{B}$ makes at most q queries and runs in time comparable to that of $\boldsymbol{A}$.

Proof. We have

$$\begin{aligned}
&\Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\mathcal{V}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\$,\cdot),\perp(\cdot,\cdot)} = 1\big] \\
&= \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\mathcal{V}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\perp(\cdot,\cdot)} = 1\big] \\
&\qquad + \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\perp(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\$,\cdot),\perp(\cdot,\cdot)} = 1\big] \\
&\le \mathrm{Adv}^{\mathrm{mac}}_{\Theta}(\boldsymbol{A}) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{B}),
\end{aligned}$$

where $\boldsymbol{B}$ is the ind-adversary naturally derived from $\boldsymbol{A}$. □

Motivating Counterexamples for MRMAC Unskippability. To have a good grasp of unskippability notion, we begin with instructive examples of *skippable* MRMAC constructions. At first glance these constructions appear plausible; indeed, they are unforgeable and indistinguishable. However, they are exactly what we want to *avoid* using, because they are *not* unskippable. We give two examples. The first example is equipped with *hasty* recover-anyway algorithms, which can be incentive for the receiver to skip verification. The second example is not unskippable but comes without hasty recovery, which sounds somewhat contradictory.

1. **Example #1: case with hasty recovery.** This construction comes with hasty recovery, meaning that there are "recover-anyway" algorithms that can be executed faster than the original verification algorithm—exactly the things that we want to avoid. The construction is given in Fig. 3. It can be directly confirmed that this construction Θ is unforgeable and indistinguishable. However, it is easy to recover R without verifying the integrity of the

```
function K(·)
    K1 ←$ K1, K2 ←$ K2, K3 ←$ K3
    return (K1, K2, K3)
end function
function G_{K1,K2,K3}(R, M)
    W ← H_{K2}(M)
    U ← Ẽ_{K1}(W, R)
    U' ← Ẽ'_{K3}(U, R)
    return T ← U‖U'
end function
```

```
function V_{K1,K2,K3}(C, T)
    U‖U' ← T
    W ← H_{K2}(M)
    R ← Ẽ_{K1}(W, U)
    R' ← Ẽ'_{K3}(U, U')
    if R' = R then
        return R
    else
        return ⊥
    end if
end function
```

Fig. 3. Example #1: a *skippable* MRMAC construction $\Theta = (\mathcal{K}, \mathcal{G}, \mathcal{V})$ with hasty recovery, where $\mathcal{G}_{K_1,K_2,K_3} : (R, M) \mapsto T$ for $R \in \mathbb{R} = \{0,1\}^n$, $M \in \mathbb{M}$ and $T \in \mathbb{T} = \{0,1\}^{2n}$. The construction Θ uses a tweakable block cipher $\widetilde{E}_{K_1} : \mathbb{W} \times \{0,1\}^n \to \{0,1\}^n$ with $K_1 \in \mathbb{K}_1$ and a tweak space $\mathbb{W}$, a universal hash function $H_{K_2} : \mathbb{M} \to \mathbb{W}$ with $K_2 \in \mathbb{K}_2$ and another tweakable block cipher $\widetilde{E}'_{K_3} : \mathbb{W}' \times \{0,1\}^n \to \{0,1\}^n$ with $K_3 \in \mathbb{K}_3$ and a tweak space $\mathbb{W}' = \{0,1\}^n$. The construction is illustrated in Fig. 9.

message-tag pair (M, T). There are two ways to do this. The first recover-anyway algorithm simply returns $R' \leftarrow \widetilde{E}'^{-1}_{K_3}(U, U')$, not using the information M at all. The second recover-anyway algorithm returns $R \leftarrow \widetilde{E}^{-1}_{K_1}(W, U)$ after computing $W \leftarrow H_{K_2}(M)$. The second algorithm does not use the information U', which is part of the tag T.

2. **Example 2: case without hasty recovery.** One might expect that non-existence of hasty recovery would suffice for unskippability. The next example demonstrates that such expectation is actually wrong, depending on what we exactly mean by "hasty." So let us re-examine more closely what we mean by "hasty." First, it seems safe to assume that any legitimate verification procedure should end with a code of the form:

```
if (some string) = (some other string) then
    return R
else                                                    (*)
    return ⊥
end if
```

and one could always skip this final "if" part of the verification procedure and come up with a recover-anyway algorithm that would always return R. Whether we should call it hasty or not (probably there is little reason to skip just the final "if," so let us call it not hasty here), this skipping seems inevitable—but we would like to ensure that even such a recover-anyway algorithm would not lead to any significant security loss. Now we show an MRMAC construction that is problematic in Fig. 4; it does not come with hasty recovery, yet if one skips the "if," then one would encounter a severe

```
1: function K(·)
2:     K_1 ←$ 𝕂_1, K_2 ←$ 𝕂_2, K_3 ←$ 𝕂_3
3:     return (K_1, K_2, K_3)
4: end function
```

```
1: function G_{K_1,K_2,K_3}(R, M)
2:     V ← F_{K_3}(R, M)
3:     W ← H_{K_2}(M)
4:     U ← Ẽ_{K_1}(W, R‖V)
5:     return T ← U‖V
6: end function
```

```
1: function V_{K_1,K_2,K_3}(C, T)
2:     U‖V ← T
3:     W ← H_{K_2}(M)
4:     R‖V* ← Ẽ^{-1}_{K_1}(W, U)
5:     if V = V* then
6:         return R
7:     else
8:         return ⊥
9:     end if
10: end function
```

Fig. 4. Example #2: a *skippable* MRMAC construction $\Theta = (\mathcal{K}, \mathcal{G}, \mathcal{V})$ without hasty recovery, where $\mathcal{G}_{K_1,K_2,K_3} : (R, M) \mapsto T$ for $R \in \mathbb{R} = \{0,1\}^{n/2}$, $M \in \mathbb{M}$ and $T \in \mathbb{T} = \{0,1\}^{3n/2}$. The construction Θ uses a tweakable block cipher $\widetilde{E}_{K_1} : \mathbb{W} \times \{0,1\}^n \to \{0,1\}^n$ with $K_1 \in \mathbb{K}_1$ and a tweak space $\mathbb{W}$, a universal hash function $H_{K_2} : \mathbb{M} \to \mathbb{W}$ with $K_2 \in \mathbb{K}_2$ and a pseudo-random function $F_{K_3} : \mathbb{R} \times \mathbb{M} \to \{0,1\}^{n/2}$ with $K_3 \in \mathbb{K}_3$. Note that the algorithm $\mathcal{V}_K$ does not use K_3. The construction is illustrated in Fig. 10.

security problem. It can be directly verified that this construction is unforgeable and indistinguishable. Then let us consider a recover-anyway algorithm that skips the "if" part at line 5 of the $\mathcal{V}$ algorithm in Fig. 4 and always returns R. This is problematic, because it would return the "correct" value of R regardless of the value of V, which is part of the tag T.

MRMAC Unskippability Formalization. What was wrong with the previous examples? In these examples, the insecurity originates from the fact that the receiver is able to obtain information about R without verifying the "entire" (M, T). Indeed, $R \leftarrow \widetilde{E}_{K_1}(W, U)$ and $R' \leftarrow \widetilde{E}'_{K_3}(U, U')$ in Fig. 3 do not take a half of the tag U' and the entire M as input, respectively. $\widetilde{E}^{-1}_{K_1}(W, U)$ in Fig. 4 does not take $n/2$ bits of the tag V as input. We would like to demand that the receiver check all bits of the message M and all bits of the tag T and only then become able to compute R and "`some string`" in pseudo-code $(*)$.

How do we impose such a requirement on a legitimate receiver who *knows the secret key* K? Since all of K, M and T are known to the receiver, there is no secret or randomized element left in the game of computing $R \leftarrow \mathcal{V}_K(M, T)$. Under such circumstances, it seems exceedingly difficult to define any security notion in a formal manner.

Fortunately, in our specific case of MRMAC unskippability, we can reasonably convert our requirement on a legitimate receiver $\boldsymbol{U}$ who knows the key K to another requirement on an adversary $\boldsymbol{A}$ who does not know the key K. The conversion is based on a contrapositive reasoning, as follows:

1. Suppose $\boldsymbol{U}$ could recover information about $R \leftarrow \mathcal{V}_K(M, T)$ without checking all bits of M and T.

2. This implies that there is a recover-anyway algorithm $\mathcal{R}_K : \mathbb{M} \times \mathbb{T} \to \mathbb{R}$ such that $R' \leftarrow \mathcal{R}_K(M', T')$, with $(M', T') \neq (M, T)$ and with R' containing some information about the original R.
3. Then, we should be able to construct an adversary $\boldsymbol{A}$ that is given oracle access to $\mathcal{G}_K(\cdot,\cdot)$ and $\mathcal{R}_K(\cdot,\cdot)$ and retrieve information about the "correct" R by making the query (M', T') to its $\mathcal{R}$-oracle.

Therefore, let us define the *unskippability* of an MRMAC scheme $\Theta = (\mathcal{K}, \mathcal{G}, \mathcal{V})$ with respect to a recover-anyway algorithm $\mathcal{R}$, for an adversary $\boldsymbol{A}$ who does not know the key, via the indistinguishability:

$$\mathrm{Adv}^{\mathrm{unskip}}_{\Theta,\mathcal{R}}(\boldsymbol{A}) := \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\mathcal{R}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\$(\cdot,\cdot)} = 1\big],$$

where the probabilities are defined over $K \leftarrow \mathcal{K}(\cdot)$ and random coins used by the oracles and the adversary, if any. The symbol $\$(\cdot,\cdot)$ denotes the random oracle that accepts a query (M, T) and always returns a freshly sampled random string $R^* \xleftarrow{\$} \mathbb{R}$. To avoid a trivial win, we demand that $\boldsymbol{A}$ never make a query (M, T) to its $\mathcal{R}$-oracle such that $\boldsymbol{A}$ has already made a query (R, M) with some $R \in \mathbb{R}$ to its $\mathcal{G}$-oracle and the $\mathcal{G}$-oracle has returned the value T.

The following lemma will become useful in Section 6. The lemma simply combines the two notions of unskippability and indistinguishability.

Lemma 2. *Let $\Theta = (\mathcal{K}, \mathcal{G}, \mathcal{V})$ be a message-recovery MAC and $K \leftarrow \mathcal{K}(\cdot)$. Let $\mathcal{R}$ be a recover-anyway oracle associated with Θ. For any adversary $\boldsymbol{A}$ having access to $\mathcal{G}$-oracle and $\mathcal{R}$-oracle and making q queries, there exist an adversary $\boldsymbol{B}$ such that*

$$\Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\mathcal{R}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\$,\cdot),\$(\cdot,\cdot)} = 1\big] \leq \mathrm{Adv}^{\mathrm{unskip}}_{\Theta,\mathcal{R}}(\boldsymbol{A}) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{B}),$$

where $\boldsymbol{B}$ makes at most q queries and runs in time comparable to that of $\boldsymbol{A}$.

Proof. We have

$$\begin{aligned}
&\Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\mathcal{R}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\$,\cdot),\$(\cdot,\cdot)} = 1\big] \\
&= \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\mathcal{R}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\$(\cdot,\cdot)} = 1\big] \\
&\qquad + \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\cdot,\cdot),\$(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{G}_K(\$,\cdot),\$(\cdot,\cdot)} = 1\big] \\
&\leq \mathrm{Adv}^{\mathrm{unskip}}_{\Theta,\mathcal{R}}(\boldsymbol{A}) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{B}),
\end{aligned}$$

where $\boldsymbol{B}$ is the ind-adversary naturally derived from $\boldsymbol{A}$. □

4 Constructing Unskippable Message-Recovery MACs

We present our construction of unskippable message-recovery MAC. We give security proofs for unforgeability, indistinguishability and unskippability.

4.1 Construction

The proposed construction uses a universal hash function and a tweakable block cipher as its building blocks. It resembles the MAC construction recently proposed by Cogliati, Lee and Seurin at ESC 2017 [15].

Let $\tilde{E}_{K_1} : \{0,1\}^\ell \times \{0,1\}^\tau \to \{0,1\}^\tau$ be a tweakable block cipher, and $H_{K_2} : \mathbb{M} \to \{0,1\}^\ell$ be a universal hash function.

1. **Key setup.** The keys K_1 and K_2 are chosen uniformly at random from $\mathbb{K}_1$ and $\mathbb{K}_2$, respectively.
2. **Tag generation.** For recoverable part $R \in \{0,1\}^r$ and message $M \in \mathbb{M}$, $\mathcal{G}_{K_1,K_2} : \mathbb{R} \times \mathbb{M} \to \{0,1\}^\tau$ is defined by setting the tag value $\mathcal{G}_{K_1,K_2}(R,M)$ to be $\tilde{E}_{K_1}(H_{K_2}(M), R\|0^{\tau-r})$.
3. **Verification with recovery.** For message $M \in \mathbb{M}$ and tag $T \in \{0,1\}^\tau$, $\mathcal{V}_{K_1,K_2}(M,T)$ first computes $P \leftarrow \tilde{E}^{-1}_{K_1}(H_{K_2}(M), T)$. If the least significant $(\tau - r)$ bits of P are zero, then it outputs the most significant r bits of P as a recoverable part. Otherwise, it outputs $\perp$.

The construction is illustrated in Fig. 1. It is obvious that the proposed construction described above satisfies the correctness condition.

4.2 Security Proofs

We will assume that the underlying tweakable block cipher is ideal, that is, it is replaced with a tweakable random permutation $\tilde{\pi}$ chosen uniformly at random. Let us call the proposed construction $\Theta^{\tilde{\pi},H}$.

Theorem 1 (Tag Unforgeability). *For any adversary $\boldsymbol{A}$ making at most q_g queries to the $\mathcal{G}$-oracle and q_v queries to the $\mathcal{V}$-oracle,*

$$\mathrm{Adv}^{\mathrm{mac}}_{\Theta^{\tilde{\pi},H}}(\boldsymbol{A}) \le \frac{q_\mathrm{v}}{2^{\tau-r}} + \frac{(q_\mathrm{g}+q_\mathrm{v})^2}{2^\tau} + \frac{(q_\mathrm{g}+q_\mathrm{v})^2}{2^{\ell+1}}.$$

Proof. First, we replace the ideal tweakable permutation $\tilde{\pi}$ and its inverse with two independent tweakable functions chosen uniformly at random. Let $\hat{\mathcal{G}}_{K_1,K_2}$ and $\hat{\mathcal{V}}_{K_1,K_2}$ be the resultant generation and verification algorithms, respectively. Then, from the PRP/PRF switching lemma [11],

$$\mathrm{Adv}^{\mathrm{mac}}_{\Theta^{\tilde{\pi},H}}(\boldsymbol{A}) \le \Pr\big[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2},\hat{\mathcal{V}}_{K_1,K_2}}\big] - \Pr\big[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2},\perp}\big] + \frac{(q_\mathrm{g}+q_\mathrm{v})^2}{2^\tau}.$$

$(\hat{\mathcal{G}}_{K_1,K_2}, \hat{\mathcal{V}}_{K_1,K_2})$ and $(\hat{\mathcal{G}}_{K_1,K_2}, \perp)$ are identical until $\boldsymbol{A}$ makes a successful verification query which makes $\hat{\mathcal{V}}_{K_1,K_2}$ return a string in $\{0,1\}^r$ as a recoverable part. Let us call the event Suc. Then,

$$\Pr\big[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2},\hat{\mathcal{V}}_{K_1,K_2}}\big] - \Pr\big[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2},\perp}\big] \le \Pr[\mathsf{Suc}] \le \Pr[\mathsf{Col}_H] + \Pr[\mathsf{Suc} \mid \neg\mathsf{Col}_H],$$

where Col_H is the event of collision for H. Since H is a universal hash function, $\Pr[\mathsf{Col}_H] \le (q_\mathrm{g}+q_\mathrm{v})^2/2^{\ell+1}$. $\Pr[\mathsf{Suc} \mid \neg\mathsf{Col}_H] \le q_\mathrm{v}/2^{\tau-r}$. □

Theorem 2 (Recoverable-Part Indistinguishability). *For any adversary* $\boldsymbol{A}$ *making at most* q_{g} *queries to the* $\mathcal{G}$*-oracle,*

$$\mathrm{Adv}^{\mathrm{ind}}_{\Theta^{\tilde{\pi},H}}(\boldsymbol{A}) \leq \left(\frac{1}{2^{\tau}} + \frac{1}{2^{\ell}} + \frac{1}{2^{r+1}}\right) q_{\mathrm{g}}^2.$$

Proof. First, we replace the ideal tweakable permutation $\tilde{\pi}$ with a tweakable function chosen uniformly at random. Let $\hat{\mathcal{G}}_{K_1,K_2}$ be the resultant generation algorithms. Then, from the PRP/PRF switching lemma [11],

$$\mathrm{Adv}^{\mathrm{ind}}_{\Theta^{\tilde{\pi},H}}(\boldsymbol{A}) \leq \Pr\left[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2}(\cdot,\cdot)}\right] - \Pr\left[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2}(\$,\cdot)}\right] + \frac{q_{\mathrm{g}}^2}{2^{\tau}}.$$

Let Col_H be the event of collision for H. Let $\mathsf{Col}_{\$}$ be the event of collision for \$. $\hat{\mathcal{G}}_{K_1,K_2}(\cdot,\cdot)$ returns a tag chosen uniformly at random for each query as long as $\neg\mathsf{Col}_H$. $\hat{\mathcal{G}}_{K_1,K_2}(\$,\cdot)$ returns a tag chosen uniformly at random for each query as long as $\neg\mathsf{Col}_H \wedge \neg\mathsf{Col}_{\$}$. Thus,

$$\Pr\left[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2}(\cdot,\cdot)}\right] - \Pr\left[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2}(\$,\cdot)}\right] \leq 2\Pr[\mathsf{Col}_H] + \Pr[\mathsf{Col}_{\$}] \leq \frac{q_{\mathrm{g}}^2}{2^{\ell}} + \frac{q_{\mathrm{g}}^2}{2^{r+1}}.$$

□

Theorem 3 (Unskippability). *For any adversary* $\boldsymbol{A}$ *making at most* q_{g} *queries to the* $\mathcal{G}$*-oracle and* q_{v} *queries to the* $\mathcal{R}$*-oracle,*

$$\mathrm{Adv}^{\mathrm{unskip}}_{\Theta^{\tilde{\pi},H},\mathcal{R}}(\boldsymbol{A}) \leq \left(\frac{1}{2^{\tau}} + \frac{1}{2^{\ell+1}}\right)(q_{\mathrm{g}} + q_{\mathrm{v}})^2.$$

Proof. First, we replace the ideal tweakable permutation $\tilde{\pi}$ and its inverse with two independent tweakable functions chosen uniformly at random. Let $\hat{\mathcal{G}}_{K_1,K_2}$ and $\hat{\mathcal{R}}_{K_1,K_2}$ be the resultant generation and recover-anyway algorithms, respectively. Then, from the PRP/PRF switching lemma [11],

$$\mathrm{Adv}^{\mathrm{unskip}}_{\Theta^{\tilde{\pi},H},\mathcal{R}}(\boldsymbol{A}) \leq \Pr\left[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2},\hat{\mathcal{R}}_{K_1,K_2}}\right] - \Pr\left[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2},\$}\right] + \frac{(q_{\mathrm{g}} + q_{\mathrm{v}})^2}{2^{\tau}}.$$

$(\hat{\mathcal{G}}_{K_1,K_2}, \hat{\mathcal{R}}_{K_1,K_2})$ and $(\hat{\mathcal{G}}_{K_1,K_2}, \$)$ are identical until the queries made by $\boldsymbol{A}$ cause a collision for H. Since H is a universal hash function,

$$\Pr\left[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2},\hat{\mathcal{R}}_{K_1,K_2}}\right] - \Pr\left[\boldsymbol{A}^{\hat{\mathcal{G}}_{K_1,K_2},\$}\right] \leq (q_{\mathrm{g}} + q_{\mathrm{v}})^2/2^{\ell+1}.$$

□

5 Key-Wrapping with Message-Recovery MACs

In this section we discuss the issue of increasing the length r of the recoverable part, which arises when one wants to wrap key material using a message-recovery MAC. We also discuss an effective means of resolving the issue.

Later in the paper we shall use a message-recovery MAC in order to construct an authenticated encryption scheme. In the construction, the recoverable part is a random, secret string S whose length is equal to the security parameter, e.g. $S \in \{0,1\}^n$ with $n = 128$. Then we may face a problem with the construction described in Section 4, as typically the block length τ of a tweakable cipher is $\tau = n$ bits. Appending $0^{n/2}$ to a recoverable part would result in $r = \tau - n/2 = n/2$. This is too short for key wrapping, being half the size of what we need.

The basic idea to resolve the issue is to use a tweakable cipher whose block length is equal to $\tau = 3n/2$ bits, so that we have $r = \tau - n/2 = n$. Fortunately there exist such tweakable strong pseudo-random permutations (SPRP) constructed from oridnary n-bit primitives. We refer to the HCTR enciphering scheme [28]. HCTR gives a variable-block-length, fixed-tweak-size SPRP using an ordinary n-bit block cipher and a polynomial hash function. In Alg. 1 we describe the HCTR scheme with the block length being fixed to $\tau = 3n/2$ bits and the tweak size fixed to $\ell = n$ bits.

Chakraborty and Nandi [14] prove the security of HCTR up to the birthday bound $O(2^{n/2})$. They prove that HCTR is indistinguishable from an ideal tweakable SPRP. To define an ideal tweakable SPRP for the case $\tau = 3n/2$, let $\mathrm{Perm}(3n/2)$ denote the set of permutations on $\{0,1\}^{3n/2}$ and $\mathrm{Perm}(3n/2)^{\mathbb{W}}$ the set of functions mapping a tweak space $\mathbb{W}$ to $\mathrm{Perm}(3n/2)$. An ideal tweakable SPRP is a collection, index by $\mathbb{W}$, of random permutations $\boldsymbol{\pi}$ and their inverses $\boldsymbol{\pi}^{-1}$, i.e. an element uniformly drawn from the space $\mathrm{Perm}(3n/2)^{\mathbb{W}}$. We define

$$\mathrm{Adv}^{\mathrm{tsprp}}_{\mathrm{HCTR}}(\boldsymbol{A}) := \Pr\left[\boldsymbol{A}^{\widetilde{E}_{K_1,K_2}(\cdot,\cdot),\widetilde{E}^{-1}_{K_1,K_2}(\cdot,\cdot)} = 1\right] - \Pr\left[\boldsymbol{A}^{\boldsymbol{\pi}(\cdot,\cdot),\boldsymbol{\pi}^{-1}(\cdot,\cdot)} = 1\right],$$

where the first probability is defined over the choice of $(K_1, K_2) \leftarrow \mathcal{K}(\cdot)$ and the second probability over the choice of $\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}(3n/2)^{\mathbb{W}}$. Here, we cite their result when the block length is $3n/2$ bits and the tweak size is n bits:

Proposition 1 ([14]). *For any adversary $\boldsymbol{A}$ making at most q queries,*

$$\mathrm{Adv}^{\mathrm{tsprp}}_{\widehat{\mathrm{HCTR}}}(\boldsymbol{A}) \leq 18q^2/2^n,$$

where $\widehat{\mathrm{HCTR}}$ is Alg. 1 with E_{K_1} replaced with an ideal permutation.

6 Verification-Unskippable AE

Now we discuss our main application of message-recovery MACs, namely, verification-unskippable AE schemes. The motivation behind this notion is that, in authenticated encryption, we would like to avoid recovering the plaintext P without verifying the integrity of ciphertext-tag pair (C, T), intentionally or unintentionally. Informally, we say that an AE scheme is *unskippable* if it is equipped with such a mechanism of disabling decryption prior to verification.

Section 6 is mostly devoted to exploring the possibility and general methodology of realizing unskippable AE schemes. We begin with giving the syntax of

Algorithm 1 HCTR. E_{K_1} is an SPRP over $\{0,1\}^n$ and G_{K_2} is a polynomial hash function from $(\{0,1\}^n)^*$ to $\{0,1\}^n$.

function $\mathcal{K}(\cdot)$
  $K_1 \xleftarrow{\$} \mathbb{K}_1$, $K_2 \xleftarrow{\$} \mathbb{K}_2$
  return (K_1, K_2)
end function

function $\widetilde{E}_{K_1,K_2}(W, X)$ ▷ $|X|=3n/2$
  $X_1 \| X_2 \leftarrow X$ ▷ $|X_2|=n/2$
  $U \leftarrow X_1 \oplus G_{K_2}(X_2 \| 0^{n/2} \| W)$
  $V \leftarrow E_{K_1}(U)$
  $Y_2 \leftarrow X_2 \oplus \mathtt{msb}_{n/2}(E_{K_1}(U \oplus V))$
  $Y_1 \leftarrow V \oplus G_{K_2}(Y_2 \| 0^{n/2} \| W)$
  return $Y_1 \| Y_2$
end function

function $\widetilde{E}^{-1}_{K_1,K_2}(W, Y)$ ▷ $|Y|=3n/2$
  $Y_1 \| Y_2 \leftarrow Y$ ▷ $|Y_2|=n/2$
  $Y_2 \leftarrow [Y]_{n/2}$
  $V \leftarrow Y_1 \oplus G_{K_2}(Y_2 \| 0^{n/2} \| W)$
  $U \leftarrow E^{-1}_{K_1}(V)$
  $X_2 \leftarrow Y_2 \oplus [E_{K_1}(U \oplus V)]_{n/2}$
  $X_1 \leftarrow U \oplus G_{K_2}(X_2 \| 0^{n/2} \| W)$
  return $X_1 \| X_2$
end function

randomized AE schemes in Section 6.1. For the purpose of simpler presentation we only consider the case without associated data. It is not difficult to include associated data into our AE constructions, as we shall discuss in Section 8.

6.1 Syntax of Randomized AE Schemes

We consider *randomized* AE schemes, where the encryption-generation algorithm is randomized (the verification-decryption algorithm remains deterministic). A randomized AE scheme $\Pi\Theta$ is a triplet of algorithms $\Pi\Theta = (\mathcal{K}, \mathcal{EG}, \mathcal{DV})$ with a correctness condition that should be satisfied between the algorithms. The three algorithms have the following syntax:

1. **Key setup.** The key setup $\mathcal{K}$ is a randomized algorithm that takes no input and returns a uniformly sampled key K from the key space $\mathbb{K}$. We write $K \leftarrow \mathcal{K}(\cdot)$, simply meaning $K \xleftarrow{\$} \mathbb{K}$. Typically we have $\mathbb{K} = \mathbb{K}_1 \times \mathbb{K}_2 \times \cdots$ with "subkey" spaces $\mathbb{K}_1, \mathbb{K}_2, \ldots$ and $K = (K_1, K_2, \ldots)$ with subkeys $K_1, K_2, \ldots$, depending on "how many" keys the AE construction requires.
2. **Encryption-generation.** The encryption-generation $\mathcal{EG}$ is a *randomized*, stateless algorithm keyed by $K \in \mathbb{K}$. We have $\mathcal{EG}_K : \mathbb{P} \to \mathbb{C} \times \mathbb{T}$, where $\mathbb{P}$, $\mathbb{C}$ and $\mathbb{T}$ denote the plaintext, ciphertext and tag spaces, respectively. Note that randomization is implicit; we actually have some random element $S \xleftarrow{\$} \mathbb{S}$ sampled and used within the algorithm $\mathcal{EG}$, so that at each invocation we may get different results for $\mathcal{EG}_K(P)$ even with the same K and the same P.
3. **Verification-decryption.** The verification-decryption $\mathcal{DV}$ is a deterministic, stateless algorithm keyed by $K \in \mathbb{K}$. We have $\mathcal{DV}_K : \mathbb{C} \times \mathbb{T} \to \mathbb{P} \cup \{\perp\}$, where the symbol $\perp$ represents verification failure. We demand $\perp \notin \mathbb{P}$.

As usual we demand the following:

- **Correctness condition.** We always have $P = \mathcal{DV}_K(C, T)$ if $K \leftarrow \mathcal{K}$ and $(C, T) \leftarrow \mathcal{EG}_K(P)$ for any $P \in \mathbb{P}$.

The condition says that one should always get the correct plaintext P by the decryption-verification algorithm $\mathcal{DV}$ if the correct K, C and T are used.

AE Indistinguishability and Unforgeability. We define the AE security of a randomized scheme $\Pi\Theta = (\mathcal{K}, \mathcal{EG}, \mathcal{DV})$ via the indistinguishability:

$$\mathrm{Adv}^{\mathrm{ae}}_{\Pi\Theta}(\boldsymbol{A}) := \Pr\big[\boldsymbol{A}^{\mathcal{EG}_K(\cdot),\mathcal{DV}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{EG}_K(\$),\perp(\cdot,\cdot)} = 1\big], \qquad (1)$$

where the probabilities are taken over the choice of key $K \leftarrow \mathcal{K}(\cdot)$ and random coins used by the oracles and the adversary $\boldsymbol{A}$, if any. Here the oracle $\mathcal{EG}_K(\$)$ accepts a query $P \in \mathbb{P} \subset \{0,1\}^*$, picks a random $P^* \xleftarrow{\$} \mathbb{P}$ such that $|P^*| = |P|$ and returns $C^* \leftarrow \mathcal{EG}_K(P^*)$. The oracle $\perp(\cdot,\cdot)$ accepts a query (C, T) and always returns the reject symbol $\perp$. To avoid a trivial win, adversary $\boldsymbol{A}$ is not allowed to make a query (C, T) to its $\mathcal{DV}$-oracle if (C, T) has been returned by the $\mathcal{EG}$-oracle. This definition corresponds to nothing but the conventional notion of security that is most commonly used for AE schemes.

6.2 Discussion on AE Unskippability

Now given the syntax and the conventional security definition, let us come back to unskippable AE schemes. To capture the notion of AE unskippability, first we try to give a formal definition.

Trying Extending MRMAC Unskippability Definition to AE. A naive approach to formalizing AE unskippability is to try extending our definition of MRMAC unskippability. As a result of such an approach, we obtain nothing but the notion of *RUP security* [4]: for "decrypt-anyway" oracle $\mathcal{DA}$, we put

$$\mathrm{Adv}^{\mathrm{rup}}_{\Pi\Theta,\mathcal{DA}}(\boldsymbol{A}) := \Pr\big[\boldsymbol{A}^{\mathcal{EG}_K(\cdot),\mathcal{DA}_K(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{EG}_K(\cdot),\$(\cdot,\cdot)} = 1\big], \qquad (2)$$

where the probabilities are defined over the choice of key $K \leftarrow \mathcal{K}(\cdot)$ and random coins used by the oracles and the adversary. The oracle $\$(\cdot,\cdot)$ accepts a query (C, T), computes $P \leftarrow \mathcal{DA}_K(C, T)$ and returns a random string $P^* \xleftarrow{\$} \mathbb{P}$ such that $|P^*| = |P|$. To avoid a trivial win, adversary $\boldsymbol{A}$ is not allowed to make the same query to its $\mathcal{DA}$-oracle as returned by the $\mathcal{EG}$-oracle.

Unfortunately, it turns out that the above notion, which is the strongest in RUP security, is not sufficient for ensuring unskippability. This can be easily seen by a typical example of the encode-then-encipher scheme: let $\boldsymbol{E}_K : \{0,1\}^* \to \{0,1\}^*$ be a variable-length SPRP. The encryption-generation $(C, T) \leftarrow \mathcal{EG}_K(P)$ is defined as $C\|T \leftarrow \boldsymbol{E}_K(S\|0^{n/2}\|P)$ with a randomizing string $S \xleftarrow{\$} \{0,1\}^n$. The decryption-generation $\mathcal{DA}_K(C, T)$ simply computes $(S', B, P') \leftarrow \boldsymbol{E}_K^{-1}(C\|T)$, and returns P' if $B = 0^{n/2}$ or $\perp$ otherwise. One can decrypt-anyway by always

returning P' regardless of value B. This encode-then-encipher scheme is RUP secure. As one can see, verification is done at the "if $B = 0^{n/2}$" part, which occurs at the end of $\mathcal{DV}$ procedure. By that time, the decryption process (i.e. computing the plaintext candidate P') has already completed, meaning that the encode-then-encipher scheme is not unskippable.

The above observation is compatible with the word "RUP," as it stands for releasing *unverified* plaintext. In AE schemes that aim for RUP security, it might be natural to perform decryption and release the plaintext candidate without verification. Clearly this is *not* what we are aiming for in this paper.

Re-evaluating the Enc-then-MAC Composition. We have seen that naively extending the MRMAC unskippability would not work. So we change our direction. Recall that the basic motivation behind AE unskippability was simple: we would like to perform "Verify-then-Dec," rather than "Dec-then-Verify" or "Dec-and-Verify." How do we achieve this? We can find our answer in the Enc-then-MAC composition [10], which has been known for its better composition security as compared to Enc-and-MAC or MAC-then-Enc. We re-evaluate the Enc-then-MAC composition by focusing on the *order* of execution itself.

Naturally, the $\mathcal{DV}$-procedure of an Enc-then-MAC composition $\Pi\Theta$ *should* take the style of Verify-then-Dec, just inverting the $\mathcal{EG}$-procedure (i.e., Enc-then-MAC). We would like to change this "should" to a "must." That is, we would like to *force* the receiver to perform verification first, and then decryption. We would like to exclude any possibility that the receiver might change the order of execution. By closely examining the Enc-then-MAC composition, we reach a conclusion of using a message-recovery MAC and ensuring the unskippability of $\Pi\Theta$ by fulfilling both of the following conditions:

1. **The MRMAC part is unskippable.** The receiver, who knows the secret key, cannot recover any information about a string $S \in \mathbb{S} = \mathbb{R}$ unless he performs the full verification of (C, T). The string S is the recoverable part.
2. **The encryption part is "undecipherable."** The receiver, who knows the secret key, cannot recover any information about plaintext P without knowing any information about the string $S \in \mathbb{S} = \mathbb{R}$ (the recoverable part) that was used to produce the corresponding ciphertext C.

The above two conditions define the unskippability of an AE scheme that takes the style of Enc-then-MRMAC. The former condition can be met by simply using an unskippable MRMAC Θ, but it remains unclear what types of encryption schemes to be combined with and what we formally mean by *undecipherability*. We look into these matters using the rest of Section 6.

6.3 One-Time Encryption Schemes

The type of encryption schemes that is suitable for Enc-then-MRMAC turns out to be the one that uses the string S as a *one-time* key material. Formally, a

one-time encryption scheme $\Pi = (\mathcal{S}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is a set of four algorithms with a correctness condition, as defined in the following:

1. **Key-material generation.** Our encryption scheme uses a secret random string S which is uniformly drawn from the space $\mathbb{S}$. We write $S \leftarrow \mathcal{S}(\cdot)$ to mean $S \xleftarrow{\$} \mathbb{S}$. The string S essentially acts as a key but is *one-time*, meaning that S is freshly re-sampled as $S \leftarrow \mathcal{S}(\cdot)$ upon each execution of the encryption algorithm $\mathcal{E}$.
2. **Key setup (optional).** The key setup $\mathcal{K}$ simply defines a secret (long-term) key K. We write $K \leftarrow \mathcal{K}(\cdot)$ to mean $K \xleftarrow{\$} \mathbb{K}$. This is optional; it may be the case that there is no (long-term) key K. In such a case we set $\mathbb{K} := \{\varnothing\}$.
3. **Encryption.** The encryption algorithm $\mathcal{E}$ has the syntax $\mathcal{E}_{S,K} : \mathbb{P} \to \mathbb{C}$, where $\mathbb{P}$ and $\mathbb{C}$ represent the plaintext and ciphertext spaces, respectively.
4. **Decryption.** The decryption algorithm $\mathcal{D}$ has the syntax $\mathcal{D}_{S,K} : \mathbb{C} \to \mathbb{P}$.

As usual we require a correctness condition:

- **Correctness condition.** We always have $\mathcal{D}_{S,K}(C) = P$ whenever $S \leftarrow \mathcal{S}(\cdot)$, $K \leftarrow \mathcal{K}(\cdot)$ and $C \leftarrow \mathcal{E}_{S,K}(P)$ for any $P \in \mathbb{P}$,

which simply guarantees that one would get the correct plaintext P when decrypting a ciphertext C using the same (S, K) as was used when encrypting.

Indistinguishability for One-Time Encryption Schemes. This notion basically requires that the encryption algorithm $\mathcal{E}$ should behave like a random oracle, if $S \xleftarrow{\$} \mathbb{S}$ is freshly sampled each time the algorithm is invoked. We define

$$\mathrm{Adv}^{\text{ind-cpa}}_{\Pi}(\boldsymbol{A}) := \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\mathcal{S}(\cdot),K}(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\mathcal{S}(\cdot),K}(\$)} = 1\big], \tag{3}$$

where the probabilities are defined over the choice of key $K \leftarrow \mathcal{K}(\cdot)$, if any, and random coins used by the oracles and the adversary. The encryption oracle $\mathcal{E}_{\mathcal{S}(\cdot),K}(\cdot)$ accepts a query $P \in \mathbb{P}$ and returns $C \leftarrow \mathcal{E}_{S,K}(P)$ where $S \leftarrow \mathcal{S}(\cdot)$ is freshly sampled each time the query is made. The oracle $\mathcal{E}_{\mathcal{S}(\cdot),K}(\$)$ accepts a query $P \in \mathbb{P}$ and picks a random $P' \xleftarrow{\$} \mathbb{P}$ such that $|P'| = |P|$ and returns $C \leftarrow \mathcal{E}_{S,K}(P')$, where $S \leftarrow \mathcal{S}(\cdot)$ is freshly sampled each time the query is made.

One-Time Undecipherability Formalization. Recall that by undecipherability we wanted to ensure that the receiver, who knows the (optional) secret key K, cannot recover any information about plaintext P without knowing any information about the string $S \in \mathbb{S}$ that was used to produce the corresponding ciphertext C. Formalizing such a notion is less challenging than the MRMAC unskippability as we saw in Section 3, because now we have a secret information $S \xleftarrow{\$} \mathbb{S}$ which is not known to the receiver. Still, it turns out that undecipherability is a relatively strong requirement on a one-time encryption scheme Π.

The notion of one-time undecipherability amounts to a type of indistinguishability between the decryption oracle and a random oracle, ensuring that the adversary (receiver) is not obtaining any information about P. We define

$$\mathrm{Adv}^{\mathrm{undec}}_{\Pi}(\boldsymbol{A}) := \\ \Pr\big[\boldsymbol{A}(K)^{\mathcal{E}_{\mathcal{S}(\cdot),K}(\cdot),\,\mathcal{D}_{\$,K}(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}(K)^{\mathcal{E}_{\mathcal{S}(\cdot),K}(\$),\,\$(\cdot)} = 1\big], \qquad (4)$$

where the decryption oracle $\mathcal{D}_{\$,K}(\cdot)$ accepts a query $C \in \mathbb{C}$, picks a random $S^* \xleftarrow{\$} \mathbb{S}$ and returns $P \leftarrow \mathcal{D}_{S^*,K}(C)$. The random oracle $\$(\cdot)$ accepts a query $C \in \mathbb{C}$, picks a random $S^* \xleftarrow{\$} \mathbb{S}$, computes $P \leftarrow \mathcal{D}_{S^*,K}(C)$ and returns a random plaintext $P^* \xleftarrow{\$} \mathbb{P}$ such that $|P^*| = |P|$. The probabilities are defined over the choice of key $K \leftarrow \mathcal{K}(\cdot)$ and random coins used by the oracles and the adversary. Note that the key value K is input to adversary $\boldsymbol{A}$.

One-Time Undecipherability Implies Decryption Indistinguishability. We briefly point out that one-time undecipherability implies indistinguishability of the decryption algorithm in the secret-key setting. This result will be used later when we discuss the security of Enc-then-MRMAC composition. Let us define another indistinguishability notion for Π by

$$\mathrm{Adv}^{\text{ind-cca}}_{\Pi}(\boldsymbol{A}) := \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\mathcal{S}(\cdot),K}(\cdot),\,\mathcal{D}_{\$,K}(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\mathcal{S}(\cdot),K}(\cdot),\,\$(\cdot)} = 1\big], \qquad (5)$$

where the oracles are as defined before, and the probabilities are defined over the choice of key K and random coins used by the oracles and the adversary. Note that here the secret key K is not given to adversary $\boldsymbol{A}$. This notion says that the decryption algorithm should behave like a random oracle to those who do not know the secret key K.

Lemma 3. *If a one-time encryption scheme Π is undecipherable, then it is decryption-indistinguishable. That is, for any ind-cca-adversary $\boldsymbol{A}$ making q queries, there exists an undec-adversary $\boldsymbol{B}$ such that*

$$\mathrm{Adv}^{\text{ind-cca}}_{\Pi}(\boldsymbol{A}) \le \mathrm{Adv}^{\mathrm{undec}}_{\Pi}(\boldsymbol{B}), \qquad (6)$$

where $\boldsymbol{B}$ makes at most q queries to the decryption oracle, and the running time of $\boldsymbol{B}$ is the sum of the running time of $\boldsymbol{A}$ plus the time to execute the encryption algorithm required to answer the queries made by $\boldsymbol{A}$.

Proof. The adversary $\boldsymbol{B}$ takes as input a key K and has oracle access to either $\mathcal{D}_{\$,K}(\cdot)$ or $\$(\cdot)$. We simply let $\boldsymbol{B}$ run $\boldsymbol{A}$ and simulate $\mathcal{E}_{\mathcal{S}(\cdot),K}(\cdot)$ to answer the queries made by $\boldsymbol{A}$. □

6.4 Formulating Generic Enc-then-MRMAC Composition

Now we are ready to formally state the generic Enc-then-MRMAC composition for constructing an unskippable AE scheme, binding a one-time, undecipherable encryption scheme Π with an unskippable message-recovery MAC Θ. As

Algorithm 2 Generic Enc-then-MRMAC composition $\Pi\Theta = (\mathcal{K}, \mathcal{EG}, \mathcal{DV})$

function $\mathcal{K}(\cdot)$
  $K_1 \leftarrow \mathcal{K}_1(\cdot)$, $K_2 \leftarrow \mathcal{K}_2(\cdot)$
  return (K_1, K_2)
end function

function $\mathcal{EG}_{K_1,K_2}(P)$
  $S \leftarrow \mathcal{S}(\cdot)$
  $C \leftarrow \mathcal{E}_{S,K_1}(P)$, $T \leftarrow \mathcal{G}_{K_2}(S, C)$
  return (C, T)
end function

function $\mathcal{DV}_{K_1,K_2}(C, T)$
  $S \leftarrow \mathcal{V}_{K_2}(C, T)$
  if $S = \perp$ **then**
    return $\perp$
  else
    $P \leftarrow \mathcal{D}_{S,K_1}(C)$
    return P
  end if
end function

expected, we demonstrate that AE unskippability is a stronger notion than the RUP security. Our composition is generic enough to cover three different constructions which will be described in Section 7.

The Enc-then-MRMAC composition uses a one-time encryption scheme Π and a message-recovery MAC Θ. More specifically, these are:

1. **A one-time encryption scheme $\boldsymbol{\Pi}$.** We have $\Pi = (\mathcal{S}, \mathcal{K}_1, \mathcal{E}, \mathcal{D})$, with $S \leftarrow \mathcal{S}(\cdot)$, $S \in \mathbb{S}$, $K_1 \leftarrow \mathcal{K}_1(\cdot)$, $K_1 \in \mathbb{K}_1$, $\mathcal{E}_{S,K} : \mathbb{P} \to \mathbb{C}$ and $\mathcal{D}_{S,K} : \mathbb{C} \to \mathbb{P}$ as described in Section 6.3. This is used to encrypt plaintext P to ciphertext C using a one-time string S and possibly with a key K.
2. **A message-recovery MAC $\boldsymbol{\Theta}$.** We have $\Theta = (\mathcal{K}_2, \mathcal{G}, \mathcal{V})$, with $K_2 \leftarrow \mathcal{K}_2(\cdot)$, $K_2 \in \mathbb{K}_2$, $\mathcal{G}_{K_2} : \mathbb{R} \times \mathbb{M} \to \mathbb{T}$ and $\mathcal{V}_{K_2} : \mathbb{M} \times \mathbb{T} \to \mathbb{R} \cup \{\perp\}$ as described in Section 3.1. This is used to produce a tag T from the ciphertext C (as the non-recoverable message) and the one-time string S (as the recoverable part).

To combine these two schemes, we need the following condition:

- **Compatibility requirement.** We need to have $\mathbb{S} = \mathbb{R}$ and $\mathbb{C} = \mathbb{M}$.

Now given a one-time encryption scheme Π and a message-recovery MAC Θ satisfying the compatibility requirement above, we can construct an AE scheme $\Pi\Theta$, which is defined in Alg. 2.

Indistinguishability and Unforgeability for Enc-then-MRMAC. We show that the Enc-then-MRMAC $\Pi\Theta$ is AE-secure (indistinguishable and unforgeable) if the underlying components Π and Θ are both secure. This corresponds to the well-known Enc-then-MAC composition result [10]. Specifically, we prove the following composition theorem:

Theorem 4. *Let $\Pi\Theta$ be the Enc-then-MRMAC scheme composed of a one-time encryption scheme Π and a message-recovery MAC Θ, as described in Alg. 2. If both Π and Θ are secure, then so is $\Pi\Theta$. Specifically, for any AE-adversary $\boldsymbol{A}$ attacking $\Pi\Theta$ and making at most q_{e} queries to the $\mathcal{EG}$-oracle and q_{v} queries to the $\mathcal{DV}$-oracle, there exist adversaries $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$ such that*

$$\mathrm{Adv}^{\mathrm{ae}}_{\Pi\Theta}(\boldsymbol{A}) \leq \mathrm{Adv}^{\text{ind-cpa}}_{\Pi}(\boldsymbol{A}_1) + 2\,\mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A_2}) + \mathrm{Adv}^{\mathrm{mac}}_{\Theta}(\boldsymbol{A}_3), \tag{7}$$

Proof.

function $\mathcal{EG}^{\$}_{K_1,K_2}(P)$
  $S \leftarrow \mathcal{S}(\cdot)$, $C \leftarrow \mathcal{E}_{S,K_1}(P)$
  $S^* \xleftarrow{\$} \mathbb{S}$, $T \leftarrow \mathcal{G}_{K_2}(S^*, C)$
  return (C, T)
end function

function $\mathcal{E}^{\$}\mathcal{G}^{\$}_{K_1,K_2}(P)$
  $P^* \xleftarrow{\$} \mathbb{P}$ with $|P^*| = |P|$
  $S \leftarrow \mathcal{S}(\cdot)$, $C \leftarrow \mathcal{E}_{S,K_1}(P^*)$
  $S^* \xleftarrow{\$} \mathbb{S}$, $T \leftarrow \mathcal{G}_{K_2}(S^*, C)$
  return (C, T)
end function

Fig. 5. Intermediate oracles $\mathcal{EG}^{\$}_{K_1,K_2}(\cdot)$ and $\mathcal{E}^{\$}\mathcal{G}^{\$}_{K_1,K_2}(\cdot)$

where

- *The adversary $\boldsymbol{A}_1$ makes at most q_{e} queries to the encryption oracle. The running time of $\boldsymbol{A}_1$ is the sum of the running time of $\boldsymbol{A}$ plus the time to execute $\mathcal{K}_2$, $\mathcal{G}$ and $\perp$ required to answer the queries made by $\boldsymbol{A}$.*
- *The adversary $\boldsymbol{A}_2$ makes at most q_{e} queries to the tag-generation oracle. The running time of $\boldsymbol{A}_2$ is the sum of the running time of $\boldsymbol{A}$ plus the time to execute $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\perp$ required to answer the queries made by $\boldsymbol{A}$.*
- *The adversary $\boldsymbol{A}_3$ makes at most q_{e} queries to the tag-generation oracle and q_{v} queries to the verification-with-recovery oracle. The running time of $\boldsymbol{A}_3$ is the sum of that of $\boldsymbol{A}$ plus the time to execute $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{D}$ required to answer the queries made by $\boldsymbol{A}$.*

Let us define "intermediate" oracles $\mathcal{EG}^{\$}_{K_1,K_2}(\cdot)$ and $\mathcal{E}^{\$}\mathcal{G}^{\$}_{K_1,K_2}(\cdot)$ both of which accept a query $P \in \mathbb{P}$ and return $(C, T) \in \mathbb{C} \times \mathbb{T}$, as described in Fig. 5. Then we have

$$
\begin{aligned}
&\mathrm{Adv}^{\mathrm{ae}}_{\Pi\Theta}(\boldsymbol{A}) \\
&= \Pr\left[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\cdot),\mathcal{DV}_{K_1,K_2}(\cdot,\cdot)} = 1\right] - \Pr\left[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\$),\perp(\cdot,\cdot)} = 1\right] && (8)\\
&= \Pr\left[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\cdot),\mathcal{DV}_{K_1,K_2}(\cdot,\cdot)} = 1\right] - \Pr\left[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\perp(\cdot,\cdot)} = 1\right] \\
&\qquad + \Pr\left[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\perp(\cdot,\cdot)} = 1\right] - \Pr\left[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\$),\perp(\cdot,\cdot)} = 1\right] && (9)\\
&\le \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,1}) + \mathrm{Adv}^{\mathrm{mac}}_{\Theta}(\boldsymbol{A}_3) \\
&\qquad + \Pr\left[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\perp(\cdot,\cdot)} = 1\right] - \Pr\left[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\$),\perp(\cdot,\cdot)} = 1\right] && (10)\\
&\le \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,1}) + \mathrm{Adv}^{\mathrm{mac}}_{\Theta}(\boldsymbol{A}_3) \\
&\qquad + \Pr\left[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\perp(\cdot,\cdot)} = 1\right] - \Pr\left[\boldsymbol{A}^{\mathcal{E}^{\$}\mathcal{G}^{\$}_{K_1,K_2}(\cdot),\perp(\cdot,\cdot)} = 1\right] \\
&\qquad + \Pr\left[\boldsymbol{A}^{\mathcal{E}^{\$}\mathcal{G}^{\$}_{K_1,K_2}(\cdot),\perp(\cdot,\cdot)} = 1\right] - \Pr\left[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\$),\perp(\cdot,\cdot)} = 1\right] && (11)\\
&\le \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,1}) + \mathrm{Adv}^{\mathrm{mac}}_{\Theta}(\boldsymbol{A}_3) + \mathrm{Adv}^{\text{ind-cpa}}_{\Pi}(\boldsymbol{A}_1) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,2}), && (12)
\end{aligned}
$$

where we can use Lemma 1 for the conversion from Eq. (9) to Eq. (10). The adversary $\boldsymbol{A}_3$ makes at most q_{e} queries to the tag-generation oracle and q_{v} queries to the verification-with-recovery oracle. The running time of $\boldsymbol{A}_3$ is the

sum of the running time of $\boldsymbol{A}$ plus the time to execute $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{D}$ required to answer the queries made by $\boldsymbol{A}$.

The adversary $\boldsymbol{A}_{2,1}$ makes at most q_{e} queries to the tag-generation oracle. The running time of $\boldsymbol{A}_{2,1}$ is the sum of the running time of $\boldsymbol{A}$ and the time to execute $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\perp$ required to answer the queries made by $\boldsymbol{A}$.

The adversary $\boldsymbol{A}_1$ makes at most q_{e} queries to the encryption oracle. The running time of $\boldsymbol{A}_1$ is the sum of that of $\boldsymbol{A}$ plus the time to execute $\mathcal{K}_2$, $\mathcal{G}$ and $\perp$ required to answer the queries made by $\boldsymbol{A}$. The adversary $\boldsymbol{A}_{2,2}$ is similar to $\boldsymbol{A}_{2,1}$.

Now observe that there exists some adversary $\boldsymbol{A}_2$ such that $\mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,1}) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,2}) \le 2\,\mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_2)$, where $\boldsymbol{A}_2$ makes at most q_{e} queries to the tag-generation oracle, and the running time of $\boldsymbol{A}_2$ is the sum of the running time of $\boldsymbol{A}$ plus the time to execute $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\perp$ required to answer the queries made by $\boldsymbol{A}$. This completes the proof. □

AE Unskippability Nullifies RUP. By now it is almost clear that AE unskippability implies the RUP security; AE unskippability guarantees that there exists no such thing as "unverified plaintext," so there is nothing to release. We formulate this implication in the following theorem; recall that decryption indistinguishability is implied by decipherability by Lemma 3.

Theorem 5. *Let $\Pi\Theta$ be the Enc-then-MRMAC composition binding a one-time encryption scheme Π with a message-recovery MAC Θ, as described in Alg. 2. If Π is decryption-indistinguishable and Θ is both indistinguishable and unskippable, then the composed AE scheme $\Pi\Theta$ is RUP-secure. Specifically, for any RUP-adversary $\boldsymbol{A}$ attacking $\Pi\Theta$ with a decrypt-anyway oracle $\mathcal{DA}$ by making at most q_{e} encryption queries and q_{d} decrypt-anyway queries, there exist adversaries $\boldsymbol{A}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{A}_3$ such that*

$$\mathrm{Adv}^{\mathrm{rup}}_{\Pi\Theta,\mathcal{DA}}(\boldsymbol{A}) \le \mathrm{Adv}^{\text{ind-cca}}_{\Pi}(\boldsymbol{A}_1) + 2\,\mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A_2}) + \mathrm{Adv}^{\mathrm{unskip}}_{\Theta}(\boldsymbol{A}_3), \qquad (13)$$

where

- *The adversary $\boldsymbol{A}_1$ makes at most q_{d} decryption queries. The running time of $\boldsymbol{A}_1$ is the sum of the running time of $\boldsymbol{A}$ plus the time to execute $\mathcal{E}$, $\mathcal{K}_2$ and $\mathcal{G}$ required to answer the queries made by $\boldsymbol{A}$.*
- *The adversary $\boldsymbol{A}_2$ makes at most q_{e} tag-generation queries. The running time of $\boldsymbol{A}_2$ is the sum of the running time of $\boldsymbol{A}$ plus the time to execute $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{D}$ required to answer the queries made by $\boldsymbol{A}$.*
- *The adversary $\boldsymbol{A}_3$ makes at most q_{e} tag-generation queries and q_{d} recover-anyway queries. The running time of $\boldsymbol{A}_3$ is the sum of that of $\boldsymbol{A}$ plus the time to execute $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{D}$ required to answer the queries made by $\boldsymbol{A}$.*

Proof. Let us define an intermediate oracle $\mathcal{DR}^{\$}_{K_1,K_2}(C,T)$ that accepts a query (C,T) and returns P as $S^* \xleftarrow{\$} \mathbb{S}$, $P \leftarrow \mathcal{D}_{S^*,K_1}(C)$. Other than the fact that

$\mathcal{DR}^{\$}_{K_1,K_2}(C,T)$ accepts an extra key K_2 and an extra argument T, it is identical to the $\mathcal{D}_{\$,K_1}(C)$ oracle. Now we have

$$
\begin{aligned}
&\mathrm{Adv}^{\mathrm{rup}}_{\Pi\Theta,\mathcal{DA}}(\boldsymbol{A}) \\
&= \Pr\big[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\cdot),\mathcal{DA}_{K_1,K_2}(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\cdot),\$(\cdot,\cdot)} = 1\big] && (14)\\
&= \Pr\big[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\cdot),\mathcal{DA}_{K_1,K_2}(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\mathcal{DR}^{\$}_{K_1,K_2}(\cdot,\cdot)} = 1\big] \\
&\qquad + \Pr\big[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\mathcal{DR}^{\$}_{K_1,K_2}(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\cdot),\$(\cdot,\cdot)} = 1\big] && (15)\\
&\le \mathrm{Adv}^{\mathrm{unskip}}_{\Theta,\mathcal{R}}(\boldsymbol{A}_3) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,1}) \\
&\qquad + \Pr\big[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\mathcal{DR}^{\$}_{K_1,K_2}(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\cdot),\$(\cdot,\cdot)} = 1\big] && (16)\\
&\le \mathrm{Adv}^{\mathrm{unskip}}_{\Theta,\mathcal{R}}(\boldsymbol{A}_3) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,1}) \\
&\qquad + \Pr\big[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\mathcal{DR}^{\$}_{K_1,K_2}(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\$(\cdot,\cdot)} = 1\big] \\
&\qquad + \Pr\big[\boldsymbol{A}^{\mathcal{EG}^{\$}_{K_1,K_2}(\cdot),\$(\cdot,\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{EG}_{K_1,K_2}(\cdot),\$(\cdot,\cdot)} = 1\big] && (17)\\
&\le \mathrm{Adv}^{\mathrm{unskip}}_{\Theta,\mathcal{R}}(\boldsymbol{A}_3) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,1}) + \mathrm{Adv}^{\text{ind-cca}}_{\Pi}(\boldsymbol{A}_1) + \mathrm{Adv}^{\mathrm{ind}}_{\Theta}(\boldsymbol{A}_{2,2}), && (18)
\end{aligned}
$$

where from Eq. (15) to Eq. (16) we have used Lemma 2. The adversary $\boldsymbol{A}_3$ runs $\boldsymbol{A}$ and executes $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{D}$ to answer the queries made by $\boldsymbol{A}$. The adversary $\boldsymbol{A}_3$ makes at most q_{e} tag-generation queries and q_{d} recover-anyway queries.

The adversary $\boldsymbol{A}_{2,1}$ runs $\boldsymbol{A}$ and executes $\mathcal{K}_1$, $\mathcal{E}$, $\mathcal{S}$ and $\mathcal{D}$ to answer the queries made by $\boldsymbol{A}$. The adversary $\boldsymbol{A}_{2,1}$ makes at most q_{e} tag-generation queries. The adversary $\boldsymbol{A}_{2,2}$ is similar to $\boldsymbol{A}_{2,1}$.

The adversary $\boldsymbol{A}_1$ runs $\boldsymbol{A}$ and executes $\mathcal{K}_2$ and $\mathcal{G}$ to simulate oracles of $\boldsymbol{A}$. The adversary $\boldsymbol{A}_1$ makes at most q_{e} encryption and q_{d} decryption queries. □

7 Constructing One-Time Undecipherable Schemes

It remains to come up with actual constructions for one-time undecipherable encryption schemes. The notion of one-time undecipherability is stronger than that of the ordinary indistinguishability, and in fact it turns out that many naive constructions of one-time encryption schemes fail to achieve this requirement.

We start with a counterexample; we show that a naive CTR mode of operation with a random (and secret) IV does not satisfy one-time undecipherability. Then we present three different constructions of one-time undecipherable encryption schemes. The first and the third constructions use a block cipher. The second one is based on Even-Mansour construction [18] and hence uses a public permutation. Each has advantages and disadvantages, giving us a tradeoff between security models in which the construction can be proven and efficiency.

7.1 A Motivating Counterexample of One-Time Undecipherability

We start with a counter (CTR) mode with a random initialization vector (IV) and show that such a construction is *not* one-time undecipherable. Throughout

1: **function** $\mathtt{Ctr}^{IV}[E_K](P)$
2: $(P_1, P_2, \ldots, P_m) \leftarrow P$
3: **for** $i = 1$ to m **do**
4: $j \leftarrow \mathtt{int}(IV) + i \bmod 2^n$
5: $J \leftarrow \langle j \rangle_n$
6: $Z_i \leftarrow E_K(J)$
7: $C_i \leftarrow P_i \oplus \mathtt{msb}_{|P_i|}(Z_i)$
8: **end for**
9: **return** $C \leftarrow C_1 \| C_2 \| \cdots \| C_m$
10: **end function**

1: **function** $\mathtt{EM}_L[\pi](X)$
2: $Y \leftarrow L \oplus \pi(L \oplus X)$
3: **return** Y
4: **end function**

1: **function** $\mathtt{FX}_L[E_K](X)$
2: $Y \leftarrow L \oplus E_K(L \oplus X)$
3: **return** Y
4: **end function**

Fig. 6. Left: the CTR mode $\mathtt{Ctr}^{IV}[E_K] : P \mapsto C$ using a block cipher $E_K : \{0,1\}^n \to \{0,1\}^n$ with $K \in \mathbb{K}$ and $IV \in \{0,1\}^n$, **upper right:** Even-Mansour construction $\mathtt{EM}_L[\pi] : X \mapsto Y$ using a public permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ with $L \in \{0,1\}^n$, and **lower right:** the FX construction $\mathtt{FX}_L[E_K] : X \mapsto Y$. In the CTR mode, plaintext P is divided into n-bit blocks with the last block length $|P_m| \leq n$. The symbol $\mathtt{int}(IV)$ represents the integer converted from the bit string IV with the left most bit being most significant. The symbol $\langle j \rangle_n$ is the n-bit string converted from integer $j < 2^n$ with the most significant bit being the left most. The symbol $\mathtt{msb}_{|P_i|}(Z_i)$ denotes the left most $|P_i|$ bits of the string Z_i.

Section 7, we use subroutines defined in Fig. 6. Here let us define an encryption scheme $\Pi = (\mathcal{S}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ that is not one-time undecipherable, as follows. Let $S \leftarrow \mathcal{S}(\cdot)$ be defined as $S \xleftarrow{\$} \mathbb{S} = \{0,1\}^n$ and $K \leftarrow \mathcal{K}(\cdot)$ as $K \xleftarrow{\$} \mathbb{K}$. The encryption algorithm $C \leftarrow \mathcal{E}_{S,K}(P)$ is defined as $C \leftarrow \mathtt{Ctr}^S[E_K](P)$ and the decryption algorithm $P \leftarrow \mathcal{D}_{S,K}(C)$ as $P \leftarrow \mathtt{Ctr}^S[E_K](C)$.

Let $\boldsymbol{U}$ be a legitimate receiver who knows K. Then the receiver $\boldsymbol{U}$ can easily break this scheme Π in the sense of one-time undecipherability, by just making one query to its decryption oracle. The receiver $\boldsymbol{U}$ simply chooses arbitrary two-block ciphertext $C = C_1 \| C_2$ and make a query C to its decryption oracle. Let $P = P_1 \| P_2$ be the plaintext returned by the oracle. Then the receiver $\boldsymbol{U}$ computes $s_1 \leftarrow \mathtt{int}\big(E_K^{-1}(C_1 \oplus P_1)\big) - 1 \bmod 2^n$ and $s_2 \leftarrow \mathtt{int}\big(E_K^{-1}(C_2 \oplus P_2)\big) - 2 \bmod 2^n$. Here remember that $\boldsymbol{U}$ knows K. If the decryption oracle is real, i.e. $\mathcal{D}_{\$,K}(\cdot)$, then we must have $s_1 = s_2$. On the other hand, if the decryption oracle is random, most likely we get $s_1 \neq s_2$, thus distinguishing the two oracles.

7.2 Construction #1: One-Time CTR Scheme

What was wrong with the CTR mode with a random IV above? It did not work because S, which is the *only* secret information to a legitimate user $\boldsymbol{U}$ who knows K, was actually working not as a "key" but just as a random IV. The next scheme $\Pi_{\mathtt{OT}} = (\mathcal{S}, \mathcal{K}, \mathcal{E}, \mathcal{G})$ is inspired by this counterexample. The new scheme $\Pi_{\mathtt{OT}}$, which we call the *one-time CTR scheme*, uses S as a key of the underlying block cipher E rather than as an IV. The one-time CTR scheme $\Pi_{\mathtt{OT}}$ is defined in Alg. 3 and illustrated in Fig. 11.

PRP Assumption about Block Cipher. We shall prove that the one-time CTR scheme $\Pi_{0\mathtt{T}}$ is IND-CPA and undecipherable under the assumption that the underlying block cipher is secure. So first we need to define our assumption about block ciphers. The PRP advantages of a block cipher E is defined as

$$\mathrm{Adv}^{\mathrm{prp}}_{E}(\boldsymbol{A}) := \Pr\big[\boldsymbol{A}^{E_K(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\pi(\cdot)} = 1\big], \tag{19}$$

where the probabilities are defined over the choice of key $K \xleftarrow{\$} \mathbb{K}$, the choice of permutation $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ and random coins used by the adversary $\boldsymbol{A}$, if any.

Indistinguishability of One-Time CTR Scheme. We start with the conventional security notion of IND-CPA. We prove it based on the assumption that the underlying block cipher is a secure PRP.

Theorem 6. *The one-time CTR mode $\Pi_{0\mathtt{T}}$ is IND-CPA if the underlying block cipher E is a secure PRP. Specifically, for any adversary $\boldsymbol{A}$ making q queries with each query being at most μ blocks, there is an adversary $\boldsymbol{B}$ such that*

$$\mathrm{Adv}^{\text{ind-cpa}}_{\Pi_{0\mathtt{T}}}(\boldsymbol{A}) \le q \cdot \mathrm{Adv}^{\mathrm{prp}}_{E}(\boldsymbol{B}) + \frac{q\mu^2}{2^{n+1}},$$

where $\boldsymbol{B}$ makes at most μ queries to its block cipher oracle and has running time equal to the sum of the running time of $\boldsymbol{A}$ plus the time to execute $\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot)$ to reply to the queries made by $\boldsymbol{A}$.

Proof. We define an intermediate oracle $\mathcal{E}_{\pi_1\cdots\pi_q,\cdot}(\cdot)$ as follows. Let $S_1, S_2, \ldots, S_q \in \mathbb{S}$ be the values of S induced by the queries made by $\boldsymbol{A}$, i.e. $S_i \leftarrow \mathcal{S}(\cdot)$. Let $\pi_1, \pi_2, \ldots, \pi_q \xleftarrow{\$} \mathrm{Perm}(n)$ be independent random permutations on n-bit strings. Then the oracle $\mathcal{E}_{\pi_1\cdots\pi_q,\cdot}(\cdot)$ behaves exactly like the real oracle $\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot)$ except that now the block ciphers $E_{S_1}(\cdot), E_{S_2}(\cdot), \ldots, E_{S_q}(\cdot)$ are replaced with random permutations $\pi_1(\cdot), \pi_2(\cdot), \ldots, \pi_q(\cdot)$. Now we have

$$\begin{aligned}\mathrm{Adv}^{\text{ind-cpa}}_{\Pi_{0\mathtt{T}}}(\boldsymbol{A}) = {} & \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\pi_1\cdots\pi_q,\cdot}(\cdot)} = 1\big] && (20)\\ & + \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\pi_1\cdots\pi_q,\cdot}(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\$)} = 1\big], && (21)\end{aligned}$$

and we can bound Eq. (20) $\le q \cdot \mathrm{Adv}^{\mathrm{prp}}_{E}(\boldsymbol{B})$ with some adversary $\boldsymbol{B}$ making at most μ queries. To bound Eq. (21), we let $\varphi_1, \varphi_2, \ldots, \varphi_q \xleftarrow{\$} \mathrm{Func}(n)$ be independent random functions on n-bit strings and define an oracle $\mathcal{E}_{\varphi_1\cdots\varphi_q,\cdot}(\cdot)$ accordingly. Then we claim that this oracle $\mathcal{E}_{\varphi_1\cdots\varphi_q,\cdot}(\cdot)$ and the random oracle $\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\$)$ result in the exactly same distribution. To see this, we observe that in the latter oracle, the distribution of "random plaintext" $P^*_i \xleftarrow{\$} \mathbb{P}$ $(i = 1, 2, \ldots, q)$ carries over to the oracle output, because it is just xored with the mask $E_{S_i}\big(\langle 1\rangle_n\big) \| E_{S_i}\big(\langle 2\rangle_n\big) \| \cdots$ which is independent of P^*_i. Also in the former oracle, the output is the plaintext P_i xored with the mask $\varphi_i\big(\langle 1\rangle_n\big) \| \varphi_i\big(\langle 2\rangle_n\big) \| \cdots$ which is uniformly random and independent from P_i. So we get Eq. (21) $= \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\pi_1\cdots\pi_q,\cdot}(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}^{\mathcal{E}_{\varphi_1\cdots\varphi_q,\cdot}(\cdot)} = 1\big] \le q \cdot \binom{\mu}{2}/2^n \le q\mu^2/2^{n+1}$, using the PRP/PRF switching lemma. This proves the theorem. □

Algorithm 3 One-time CTR scheme $\Pi_{\mathtt{0T}} = (\mathcal{S}, \mathcal{K}, \mathcal{E}, \mathcal{D})$

function $\mathcal{S}(\cdot)$
  $S \xleftarrow{\$} \{0,1\}^n$
  return S
end function

function $\mathcal{K}(\cdot)$
  return $\varnothing$
end function

function $\mathcal{E}_{S,\cdot}(P)$
  return $C \leftarrow \mathtt{Ctr}^0[E_S](P)$
end function

function $\mathcal{D}_{S,\cdot}(C)$
  return $P \leftarrow \mathtt{Ctr}^0[E_S](C)$
end function

Undecipherability of One-Time CTR Scheme. Next we prove that the one-time CTR scheme $\Pi_{\mathtt{0T}}$ is undecipherable. Owing to the lack of long-term key K in $\Pi_{\mathtt{0T}}$ and the duality between the encryption algorithm $\mathcal{E}_{S,\cdot}(\cdot)$ and the decryption algorithm $\mathcal{D}_{S,\cdot}(\cdot)$ (i.e. they are identical), we can deduce the undecipherability of $\Pi_{\mathtt{0T}}$ almost directly from Theorem 6.

Theorem 7. *The one-time CTR scheme $\Pi_{\mathtt{0T}}$ is undecipherable. Specifically, for any adversary $\boldsymbol{A}$ making q queries with each query being at most μ blocks, there is an adversary $\boldsymbol{B}$ such that*

$$\mathrm{Adv}^{\mathrm{undec}}_{\Pi_{\mathtt{0T}}}(\boldsymbol{A}) \le 2q \cdot \mathrm{Adv}^{\mathrm{prp}}_{E}(\boldsymbol{B}) + \frac{q\mu^2}{2^n},$$

where $\boldsymbol{B}$ makes at most μ queries to its block cipher oracle and has running time equal to the sum of the running time of $\boldsymbol{A}$ plus the time to execute $\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot)$ and $\mathcal{D}_{\$,\cdot}(\cdot)$ to reply to the queries made by $\boldsymbol{A}$.

Proof. We have

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{undec}}_{\Pi_{\mathtt{0T}}}(\boldsymbol{A}) &= \Pr\big[\boldsymbol{A}(\varnothing)^{\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot),\,\mathcal{D}_{\$,\cdot}(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}(\varnothing)^{\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\$),\,\$(\cdot)} = 1\big] \\
&= \Pr\big[\boldsymbol{A}(\varnothing)^{\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot),\,\mathcal{D}_{\$,\cdot}(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}(\varnothing)^{\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot),\,\$(\cdot)} = 1\big] \quad (22) \\
&\quad + \Pr\big[\boldsymbol{A}(\varnothing)^{\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot),\,\$(\cdot)} = 1\big] - \Pr\big[\boldsymbol{A}(\varnothing)^{\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\$),\,\$(\cdot)} = 1\big], \quad (23)
\end{aligned}$$

and Eq. (23) $\le \mathrm{Adv}^{\text{ind-cpa}}_{\Pi_{\mathtt{0T}}}(\boldsymbol{A})$. To bound Eq. (22), we note that it amounts to the indistinguishability between the oracles $\mathcal{D}_{\$,\cdot}(\cdot)$ and $\$(\cdot)$. Now since $\mathcal{D}_{\$,\cdot}(\cdot)$ is identical to $\mathcal{E}_{\mathcal{S}(\cdot),\cdot}(\cdot)$, we get Eq. (22) $\le \mathrm{Adv}^{\text{ind-cpa}}_{\Pi_{\mathtt{0T}}}(\boldsymbol{A})$. Hence $\mathrm{Adv}^{\mathrm{undec}}_{\Pi_{\mathtt{0T}}}(\boldsymbol{A}) \le 2\,\mathrm{Adv}^{\text{ind-cpa}}_{\Pi_{\mathtt{0T}}}(\boldsymbol{A}) \le 2q\,\mathrm{Adv}^{\mathrm{prp}}_{E}(\boldsymbol{B}) + q\mu^2/2^n$ from Theorem 6. □

7.3 Construction #2: Even-Mansour CTR Scheme

We have seen that the one-time CTR scheme $\Pi_{\mathtt{0T}}$ is IND-CPA and undecipherable. Unfortunately, it has an efficiency problem—every time the encryption or decryption algorithm is invoked, a new value of $S \in \mathbb{S}$ is used, which is input to

the underlying block cipher E as a key. This means that the key schedule of E needs to be computed upon every encryption or decryption.

This motivates us to consider using the Even-Mansour construction $\mathtt{EM}_S[\pi]$, as defined in Fig. 6, in place of the block cipher E_S in $\Pi_{\mathtt{OT}}$. We call the resulting scheme *Even-Mansour CTR scheme* $\Pi_{\mathtt{EM}}$, which is illustrated in Fig. 12. The scheme $\Pi_{\mathtt{EM}}$ is free of the key schedule issue. In principle $\Pi_{\mathtt{EM}}$ can be proven IND-CPA and undecipherable, but it comes with a tradeoff; its security can be proven only in the random permutation model, regarding the underlying public permutation π as a random permutation $\pi \xleftarrow{\$} \mathrm{Perm}(n)$.

The PRP security of $\mathtt{EM}_S[\pi]$ follows from the analysis by Dunkelman et al. [16]. Basically, $\mathtt{EM}_S[\pi]$ is a secure PRP in the random permutation model. However, we cannot simply combine their result with those in Section 7.2, because our results are carried out in the standard model. Therefore, strictly speaking, we shall need to directly prove the IND-CPA and undecipherability of $\Pi_{\mathtt{EM}}$ in the random permutation model. Fortunately, Mouha and Luykx [**?**] have made analysis of $\mathtt{EM}_S[\pi]$ in the multi-user setting, and their results [**?**, Th.1] almost directly imply the IND-CPA and undecipherability of $\Pi_{\mathtt{EM}}$, with multi-user keys corresponding to the multiple keys $S_1, S_2, \ldots, S_q$ in our setting.

7.4 Construction #3: FX CTR Scheme

The third construction, unlike the previous two, utilizes a long-term key $K \in \mathbb{K}$. The scheme is based on the one-time CTR $\Pi_{\mathtt{OT}}$ with the underlying block cipher being the FX construction [22]. The FX construction $\mathtt{FX}_S[E_K]$ is defined in Fig. 6. We call the resulting scheme *FX CTR scheme* $\Pi_{\mathtt{FX}}$, which is illustrated in Fig. 13. The FX CTR scheme lies between the one-time CTR scheme $\Pi_{\mathtt{OT}}$ and the Even-Mansour CTR scheme $\Pi_{\mathtt{EM}}$. Namely, the IND-CPA security of the FX CTR scheme $\Pi_{\mathtt{FX}}$ can be proven in the standard model, even though the one-time undecipherability of $\Pi_{\mathtt{FX}}$ is proven in the ideal cipher model. The FX CTR scheme uses a block cipher and hence needs to compute the key schedule, but it only needs to do it once, because the key K is long-term.

The IND-CPA security of $\Pi_{\mathtt{FX}}$ is simply implied by the indistinguishability of the CTR mode with a random IV. Here the block cipher key K is kept secret from the adversary, and S acts as a random IV.

The one-time undecipherability of $\Pi_{\mathtt{FX}}$ is implied by that of $\Pi_{\mathtt{EM}}$; for each key $K \in \mathbb{K}$, which is known to the adversary, we can regard E_K as a public random permutation if we regard E as an ideal cipher. Then $\mathtt{FX}_S[E_K]$ becomes nothing but the Even-Mansour construction if the adversary knows K but not S. Hence the undecipherability immediately follows.

8 Further Extension and Application

8.1 Incorporating Associated Data into Enc-then-MRMAC

It is easy to incorporate associated data A into the generic Enc-then-MRMAC composition. To do this, let $\mathbb{A}$ denote the space of associated data. The compat-

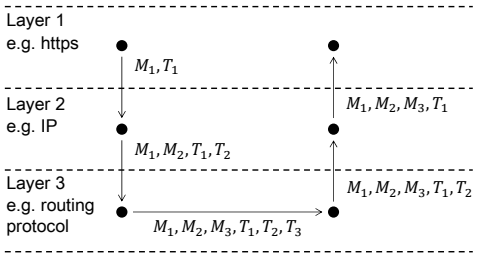

Fig. 7. Layered protocol

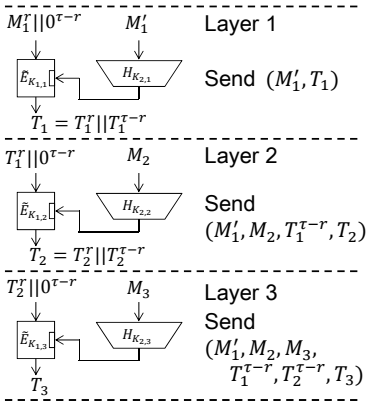

Fig. 8. Tag generation of layered MAC

ibility requirement $\mathbb{C} = \mathbb{M}$ now changes to the existence of an encoding function $\varepsilon : \mathbb{A} \times \mathbb{C} \to \mathbb{M}$ with an efficient decoding function $\varepsilon^{-1} : \mathbb{M} \to \mathbb{A} \times \mathbb{C}$. We can then construct $\mathcal{EG}_{K_1,K_2} : \mathbb{A} \times \mathbb{P} \to \mathbb{C} \times \mathbb{T}$ via $S \leftarrow \mathcal{S}(\cdot)$, $C \leftarrow \mathcal{E}_{S,K_1}(P)$, $T \leftarrow \mathcal{G}_{K_2}\big(S, \varepsilon(A, C)\big)$. An example of ε is $\varepsilon(A, C) := \langle |A| \rangle_n \| A \| C$, where $\langle x \rangle_n$ is an n-bit representation of integer x, assuming $|A| < 2^n$.

8.2 Application to Layered Protocols

Section 4 proposed MRMAC (a stand alone MAC) and Section 7 proposed its application to AE. Here we discuss the usage of MRMAC in layered protocols such as in the OSI reference model.

Setting. There are N_L layers from layer 1 (highest) to layer N_L (lowest). A sender in layer 1 wants to send (M_1, T_1) to a receiver in layer 1, where they share a secret key. These two players never communicate directly, instead, (M_1, T_1) is sent to layer 2. Each middle layer adds a new message M_i to the received one, and tag T_i is computed possibly by taking $(M_1, \cdots, M_i, T_1, \cdots, T_{i-1})$ as input. If the current layer is not the lowest, $(M_1, \cdots, M_i, T_1, \cdots, T_i)$ is sent to layer $i + 1$. Otherwise, it is sent to the receiver side in the lowest layer. The receiver side checks integrity from the lower to higher layers. Only if verification succeeds, the data is sent from layer i to $i - 1$.

Construction. The proposed construction uses the MRMAC proposed in Section 4. To recall the construction, given a recoverable part R and a message M, MRMAC $\mathcal{G}_{K_1,K_2}(R, M)$ produces $\tilde{E}_{K_1}(H_{K_2}(M), R\|0^{\tau-r})$ as a tag.

Key setup. The keys for the i-th layer, $K_{1,i}$ and $K_{2,i}$, $1 \le i \le N_L$, are chosen uniformly at random from $\mathbb{K}_1$ and $\mathbb{K}_2$, respectively.

Tag generation (illustrated in Fig. 8). For layer 1, a message $M_1 \in \mathbb{M}$ is divided into the r bits of recoverable part $M_1^r \in \{0,1\}^r$ and the remaining part M_1'. Namely, $M_1 = M_1^r \| M_1'$. A tag T_1 is computed by $\mathcal{G}_{K_{1,1},K_{2,1}}(M_1^r \| 0^{\tau-r}, M_1')$. Then (M_1', T_1) is sent to layer 2.

For layer $i(= 2, 3, \cdots, N_L)$, the tag from the previous layer, T_{i-1}, is divided into r bits T_{i-1}^r and $\tau - r$ bits $T_{i-1}^{\tau-r}$. The tag T_i is computed by $\mathcal{G}_{K_{1,i},K_{2,i}}(T_{i-1}^r \| 0^{\tau-r}, M_i)$ and (M_i, T_i) is added to and T_{i-1}^r is removed from the data to send. Namely, $(M_1', M_2, \ldots, M_i, T_1^{\tau-r}, \cdots, T_{i-1}^{\tau-r}, T_i)$ is sent to layer $i+1$ when $i < N_L$ and sent to the verifier when $i = N_L$.

Verification with recovery. For layer $i(= N_L, N_L - 1, \cdots, 2)$, it first computes $P \leftarrow \tilde{E}_{K_{1,i}}^{-1}(H_{K_{2,i}}(M_i), T_i)$. If the least significant $\tau - r$ bits of P are zero, it outputs the most significant r bits of P as T_{i-1}^r and sends it layer $i-1$. Otherwise, it outputs $\perp$.

For layer 1, it first computes $P \leftarrow \tilde{E}_{K_{1,1}}^{-1}(H_{K_{2,1}}(M_1'), T_1)$. If the least significant $\tau - r$ bits of P are zero, it outputs the most significant r bits of P as M_1^r. Otherwise, it outputs $\perp$.

By hiding $T_1^r, \cdots, T_{N_L-1}^r$, the construction forces the verification order from layer N_L to layer 1 and the execution of verification for all the layers. Encoding of $\tau - r$ bits of zeros in each layer takes a role of intermediate tags. The protocol can detect in which layer verification failed. In addition, the protocol can stop immediately when verification fails in some layer.

References

1. Abed, F., Fluhrer, S., Foley, J., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: Pipelineable on-line encryption. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, Springer (2014)
2. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: RIV for robust authenticated encryption. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 23–42. Springer (2016)
3. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated permutation-based encryption for lightweight cryptography. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, Springer (2014)
4. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to randomize forged plaintext. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, Springer (2014)
5. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. IACR Cryptology ePrint Archive 144 (2014)
6. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography and application to virus protection. In: Leighton, F.T., Borodin, A. (eds.) STOC 1995. pp. 45–56. ACM (1995)
7. Bellare, M., Keelveedhi, S.: Interactive message-locked encryption and secure deduplication. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 516–538. Springer (2015)

8. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 296–312. Springer (2013)
9. Bellare, M., Lysyanskaya, A.: Symmetric and dual prfs from standard assumptions: A generic validation of an HMAC assumption. IACR Cryptology ePrint Archive 2015, 1198 (2015)
10. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer (2000)
11. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331 (2004), `http://eprint.iacr.org/2004/331`
12. Bernstein, D.: CAESAR Competition. `http://competitions.cr.yp.to/caesar.html` (2013)
13. Buonanno, E., Katz, J., Yung, M.: Incremental unforgeable encryption. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 109–124. Springer (2001)
14. Chakraborty, D., Nandi, M.: An improved security bound for HCTR. In: FSE 2008. pp. 289–302 (2008)
15. Cogliati, B., Lee, J., Seurin, Y.: New MAC constructions from (tweakable) block ciphers. Presented at Early Symmetric Crypto (ESC) 2017 (2017)
16. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: The Even-Mansour scheme revisited. In: EUROCRYPT 2012. pp. 336–354 (2012)
17. Eikemeier, O., Fischlin, M., Götzmann, J., Lehmann, A., Schröder, D., Schröder, P., Wagner, D.: History-free aggregate message authentication codes. In: Garay, J.A., Prisco, R.D. (eds.) SCN 2010. LNCS, vol. 6280, pp. 309–328. Springer (2010)
18. Even, S., Mansour, Y.: A construction of a cioher from a single pseudorandom permutation. In: ASIACRYPT '91. pp. 210–224 (1991)
19. Hirose, S., Sasaki, Y., Yasuda, K.: IV-FV authenticated encryption and triplet-robust decryption. Presented at Early Symmetric Crypto (ESC) 2015 (2015)
20. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 15–44. Springer (2015)
21. Katz, J., Lindell, A.Y.: Aggregate message authentication codes. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 155–169. Springer (2008)
22. Kilian, J., Rogaway, P.: How to protect DES against exhaustive key search. In: CRYPTO '96. pp. 252–267 (1996)
23. Nyberg, K., Rueppel, R.A.: A new signature scheme based on the DSA giving message recovery. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) CCS '93. pp. 58–61. ACM (1993)
24. Pintsov, L.A., Vanstone, S.A.: Postal revenue collection in the digital age. In: FC 2000. pp. 105–120 (2000)
25. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer (2006)
26. Sasaki, Y., Yasuda, K.: A new mode of operation for incremental authenticated encryption with associated data. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 397–416. Springer (2015)
27. Shrimpton, T., Terashima, R.S.: A modular framework for building variable-input-length tweakable ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 405–423. Springer (2013)

28. Wang, P., Feng, D., Wu, W.: HCTR: A variable-input-length enciphering mode. In: Feng, D., Lin, D., Yung, M. (eds.) CISC 2005. LNCS, vol. 3822, pp. 175–188. Springer (2005)

A Supplementary Material: Illustration of Examples of Skippable MRMAC

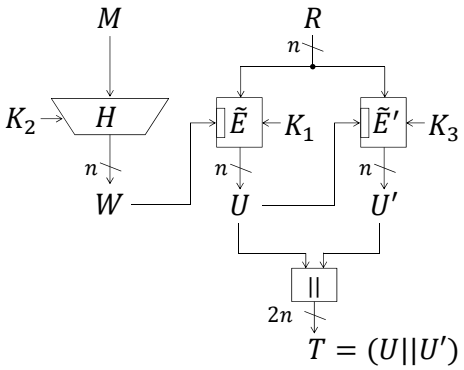

Fig. 9. Example #1: a *skippable* MRMAC construction with hasty recovery

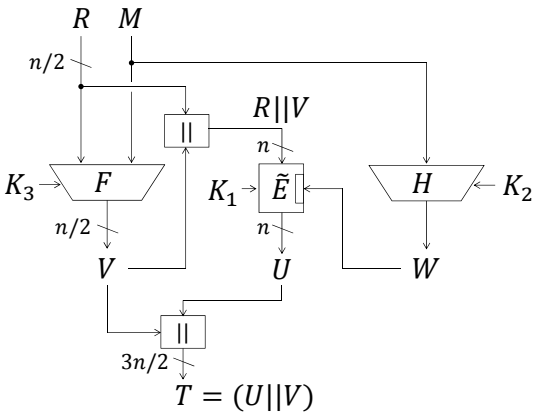

Fig. 10. Example 2: a *skippable* MRMAC construction without hasty recovery

B Supplementary Material: Illustration of One-Time Undecipherable Encryption Schemes

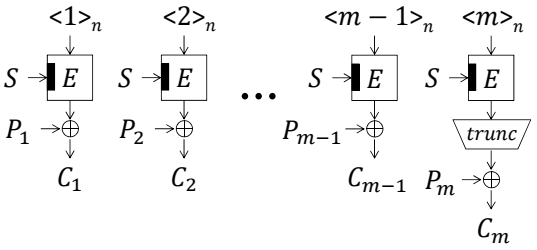

Fig. 11. One-time CTR scheme

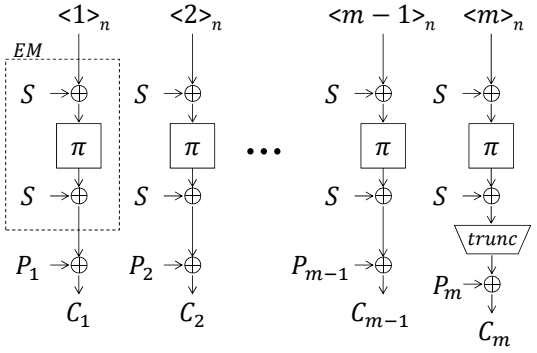

Fig. 12. Even-Mansour CTR scheme

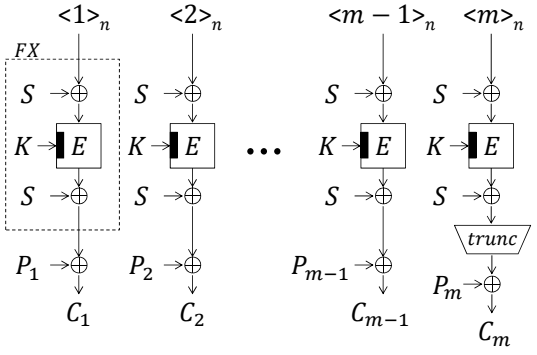

Fig. 13. FX CTR scheme