

Security of Symmetric Primitives under Incorrect Usage of Keys

Pooya Farshim, Claudio Orlandi, Răzvan Roşie

▶ **To cite this version:**

Pooya Farshim, Claudio Orlandi, Răzvan Roşie. Security of Symmetric Primitives under Incorrect Usage of Keys. IACR Transactions on Symmetric Cryptology, 2017, 10.13154/tosc.v2017.i1.449-473 . hal-01470885

HAL Id: hal-01470885
https://ens.hal.science/hal-01470885

Submitted on 17 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Security of Symmetric Primitives under Incorrect Usage of Keys

Pooya Farshim[1], Claudio Orlandi[2] and Răzvan Roşie[1]

[1] ENS, CNRS, INRIA & PSL Research University, Paris, France
[2] Aarhus Univeristy, Aarhus, Denmark

pooya.farshim@gmail.com orlandi@cs.au.dk rosie@di.ens.fr

Abstract. We study the security of symmetric primitives under the *incorrect* usage of keys. Roughly speaking, a *key-robust* scheme does not output ciphertexts/tags that are valid with respect to distinct keys. Key-robustness is a notion that is often tacitly expected/assumed in protocol design — as is the case with anonymous auction, oblivious transfer, or public-key encryption. We formalize simple, yet strong definitions of key robustness for authenticated-encryption, message-authentication codes and PRFs. We show standard notions (such as AE or PRF security) guarantee a basic level of key-robustness under honestly generated keys, but fail to imply key-robustness under adversarially generated (or known) keys. We show robust encryption and MACs compose well through generic composition, and identify robust PRFs as the main primitive used in building robust schemes. Standard hash functions are expected to satisfy key-robustness and PRF security, and hence suffice for practical instantiations. We however provide further theoretical justifications (in the standard model) by constructing robust PRFs from (left-and-right) collision-resistant PRGs.

Keywords: incorrect key usage · key-robustness · authenticated encryption · MAC · generic composition · collision-resistant PRF · collision-resistant PRG

1 Introduction

Cryptography is complex and hard to understand. While the wide and diverse landscape of cryptographic notions of security is a useful resource for the academic community (as it allows to describe exactly what kind of security a certain cryptographic scheme guarantees – and implicitly which one it does not), this complexity often hinders the ability of practitioners and users of cryptography to implement truly secure cryptographic systems. In the eyes of the users, cryptography is often seen as an *all-or-nothing* process: once cryptography is "turned on," data gets encrypted and therefore the system is secure, hopefully with as little fine print as possible. The shortcomings of this all-or-nothing property has been shown by a long series of attacks on real-world cryptographic protocols.

The academic community is reacting to this real-world need with simpler and more comprehensive notions of security. The most clear example of this is the introduction of the notion of *Authenticated Encryption* (AE) [Rog02, RS06]. While early cryptography considered confidentiality to be the only goal of encryption, over the years it has become apparent that virtually every application requiring confidentiality would also benefit from some form of authenticity guarantees. Therefore, instead of letting the users pick an encryption and a MAC scheme (and combine them in appropriate ways), cryptographers are currently designing schemes that guarantee all properties at once (cf. the CAESER competition). Other examples in this direction are the study of *misuse-resistant AE schemes* [RS06], which guarantee best possible security even in the presence of repeating

Licensed under Creative Commons License CC-BY 4.0.

nonces, security under related-key attacks (RKAs) [Bih94, BK03], and security in the presence of key-dependent messages [BRS02].

In this quest towards coming up with encryption schemes that are as *ideally secure* as possible, we introduce the notion of *key-robustness*.[1] In a nutshell, key-robustness looks at a setting where multiple keys (possibly known and/or chosen by the adversary) are present in the system. When using strong encryption, like authenticated encryption, it might be tempting to assume that any given ciphertext would only be valid for a single secret key. As we shall see, this may or may not be the case depending on the context. We start with some motivating examples before discussing the details.

Example 1 – Storage Authenticity. In this application a user wants to encrypt some data which is stored on an untrusted storage provider. To ensure authenticity of the data, the user encrypts it using an AE scheme. Then the user stores the key on a different storage provider. What happens now if the second storage provider is corrupt? It might be tempting to think that, since the data is encrypted with AE, any tampering on the key will be detected when the user decrypts the data with the key. This in unfortunately not the case and as we discuss later, AE security alone does not guarantee authenticity of the original data against an adversary that can tamper with the stored key. Note that tampering with the key can be done with the knowledge of the original key.

Example 2 – Anonymous Communication. Most practical symmetric encryption schemes have ciphertexts that look random, which in particular implies a form of *key anonymity*: when given two ciphertexts c_0, c_1 it is hard to tell whether or not they were generated using the same (unknown) secret key. Imagine a protocol with one sender and several receivers, where each receiver shares a key k_i with the sender. Anonymity guarantees that if the sender broadcasts a ciphertext constructed using k_i, then a different receiver j should only learn that $i \neq j$ and nothing else. At the same time, such protocols often intuitively assume that at most one of the receivers will believe to be the intended receiver, i.e., decryption will fail for all but one of the users. This is however not covered by current security definitions. More generally, whenever user anonymity is a security goal, it is likely that some form of robustness is also needed in order to avoid undesired behavior [ABN10].

Example 3 – Oblivious Transfer. Consider the following protocol, for constructing a $\binom{3}{2}$-OT protocol using only $\binom{3}{1}$-OTs: the sender picks 3 random keys k_1, k_2, k_3 and inputs the message $x_1 = (k_2, k_3), x_2 = (k_1, k_3)$ and $x_3 = (k_1, k_2)$ to the OT. At the same time, the sender sends encryptions of his messages under these keys, i.e., sends $c_i = E(k_i, m_i)$ for $i = 1..3$. Now the receiver inputs the index of the message *he does not want to learn* to the $\binom{3}{1}$-OT and learns all keys except k_i. Intuitively the fact that the messages are sent only once (encrypted) should guarantee that the sender's choice of messages is uniquely defined. However, consider the following attack: the corrupt sender inputs $x_1^* = (k_2, k^*)$ (instead of x_1) such that $D(k^*, c_3) = m_3^*$ with $m_3^* \neq m_3$ and $m_3^* \neq \perp$. This means that the receiver will see two different versions of m_3 depending on whether the receiver asked for the pair $(2, 3)$ or $(1, 3)$. (This attack is an example of *input-dependence* and is a clear breach of security since it cannot be simulated in the ideal world.) The attack described here is a very simplified version on an actual attack described by [Lam16] on the private set-intersection protocol of [DCW13]. A strong form of key-robustness for symmetric encryption is also used to prove the security of the OT protocol presented in [CO15].

Our contributions. We give simple and strong definitions of key-robustness for a number of symmetric primitives of interest. Starting with the work of Abdalla et al. [ABN10] and Farshim et al. [FLPQ13] (which studied the notion of (key-)robustness in the public-key setting) we develop appropriate notions for symmetric encryption, MACs, and PRFs. To

[1] We refrain from *formally* referring to this notion as robustness in order to avoid confusion with robust AE schemes [HKR15]. In our discussions, however, we use robustness to ease readability.

the best of our knowledge this is the first attempt in this direction (we note that [Moh10] considers robustness and anonymity of hybrid encryption, but not for symmetric encryption directly). As briefly mentioned above, our notion also formalizes the non-existence of "unexpected collisions" in a cryptosystem over distinct keys, even when inputs (including keys) are maliciously generated.

We consider both notions where the adversary has control over the keys and notions where the keys are generated honestly. The strongest notion that we formulate is called *complete robustness* and allows an adversary to generate the keys used in the system. We show that whether the adversary is in control of the keys or not makes a significant difference, by giving separations between the notions. While previous work in the public-key setting also had to deal with adversarially generated keys that were also *invalid*, this is *not* an issue in our setting, since in the symmetric world keys are often bit-strings of some pre-specified length and can be easily checked for validity. By focusing on correctly formed keys we can show equivalence between *complete robustness* and a syntactically simpler notion, which we call *full robustness*.

By giving appropriate separating examples we show that AE security and strong unforgeability do not provide full robustness. Before building fully robust schemes, we first characterize the level of robustness that *is* enjoyed by AE-secure encryption and strongly unforgeable MACs. For MACs we prove that as long as the two keys are honestly generated and remain outside the view of the adversary the scheme is robust in the presence of tag-generation and verification routines. Interestingly, AE-secure encryption schemes achieve a higher level of robustness where both keys are honestly generated, but one is provided to the adversary. Intuitively, this gap arises from the fact that the adversary against the MAC can still choose a message with respect to which a common tag should verify under two distinct keys, but in the encryption setting such an adversary is bound to ciphertexts that are random and outside its control. Unfortunately these weaker notions of security provide guarantees only if the keys are honestly and independently generated. Therefore no guarantees are provided in applications where the adversary completely controls the keys in the system (like the in the OT example before), where encryption is performed using related keys, or when the scheme is used to encrypt key-dependent messages (KDM). Full robustness, on the other hand, would be sufficient in such settings.

We then show that full robustness composes well: any fully robust symmetric encryption when combined with a fully robust MAC results in a fully robust AE scheme. Analogous composition results also hold for MAC-then-Encrypt and Encrypt-and-MAC. In these transformations, however, the length of the key doubles (since independent keys are used for encryption and MAC), while in practical AE schemes it is desirable to use a single key for both tasks. Using a *single key* for both the encryption and MAC components not only reduces storage, it increases security by only relying on the robustness of *either* of its components. (We emphasize, however, that AE security of the generically composed scheme with key reuse, although provable for some schemes, does not always hold.) We show that this can be avoided by modifying the Encrypt-then-MAC transform to also *authenticate the encryption key*. As long as the MAC component is both pseudorandom and collision-resistant, we show this transform gives a robust and AE-secure scheme. Simultaneous pseudorandomness and collision-resistance is an expected property from standard hash functions (and is met by the random oracle). This provides the most practical route to generically build robust encryption schemes. We caution, however, that not all MACs would satisfy this requirement. In particular, we point out that CBC-MAC fails to be fully robust, even when one of two honestly generated keys is in adversary's view.

We then ask if feasibility results for robustness in the public-key setting can be translated to the symmetric setting. This turns out not to be the case. The main reason for this is that in the asymmetric setting the public key can be used as mechanism to *commit* to its

associated secret key. In the symmetric case, on the other hand, there is no such public information. It might be tempting to think that one can just commit to the secret key and append it to the ciphertext. Unfortunately this approach cannot be proven secure due to a circular *key-dependency* between the encryption and the commitment components. To give a provably secure construction, we construct appropriate commitments that can be used in this setting. This requires *a right-injective PRG*, that can be in turn based on one-way permutations. This result relies on the one-time security of the MAC and its collision-resistance, which once again we base on right-injective PRGs.

We finally study constructions of collision-resistant and robust PRFs (which can be immediately converted to collision-resistant and robust MACs). We first show that any robust PRF can be converted into a fully collision-resistant PRF using a right-injective PRG and a parallel application of a pseudorandom permutation. Next we identify of *left/right collision-resistant* (LRCR) PRGs as both a necessary and a sufficient assumption for building robust PRFs. In an LRCR PRG, no adversary should be able to find collisions over the left *or* right halves of the outputs of the PRG. We show that the GGM construction [Gol01] converts any LRCR PRG into a fully robust PRF. We then give an instantiation of left/right collision-resistant PRGs based on DDH to justify them in the standard model. The resulting PRG, however, is not GGM-friendly (as input and output spaces do not match). We show how to convert it into a GGM-friendly PRG via pairwise-independent permutations and regular collision-resistant hash functions. The first can be based on injective linear maps, while the latter can be based on claw-free permutations. Our work leaves open the task of constructions of LRCR PRGs from generic assumptions such as one-way functions/permutations or collision resistance.

2 Preliminaries

Notation. We denote the security parameter by $\lambda \in \mathbb{N}$ and assume it is implicitly given to all algorithms in the unary representation 1^λ. By an algorithm we mean a stateless Turing machine. Algorithms are randomized unless stated otherwise, and ppt as usual stands for "probabilistic polynomial-time," in the security parameter (rather than the total length of its inputs). Given a randomized algorithm $\mathcal{A}$ we denote the action of running $\mathcal{A}$ on input(s) $(1^\lambda, x_1, \ldots)$ with uniform random coins r and assigning the output(s) to $(y_1, \ldots)$ by $(y_1, \ldots) \twoheadleftarrow \mathcal{A}(1^\lambda, x_1, \ldots; r)$. For a finite set S, we denote its cardinality by $|S|$ and the action of sampling a uniformly at random element x from X by $x \twoheadleftarrow X$. We define $[k] \coloneqq \{1, \ldots, k\}$. A real-valued function $\varepsilon(\lambda)$ is negligible if $\varepsilon(\lambda) \in \mathcal{O}(\lambda^{-\omega(1)})$. We denote the set of all negligible functions by Negl. Throughout the paper $\perp$ stands for a special error symbol. We use $||$ to denote the concatenation of binary strings.

Pseudorandom generators. A pseudorandom generator $\mathbf{PRG}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ is a deterministic algorithm that on input a point $x \in \mathcal{D}$ outputs a value $y \in \mathcal{R}$. We define the advantage of an adversary $\mathcal{A}$ against $\mathbf{PRG}$ as

$$\mathbf{Adv}^{\mathrm{prg}}_{\mathbf{PRG},\mathcal{A}}(\lambda) \coloneqq 2 \cdot \Pr\left[\mathrm{PRG}^{\mathcal{A}}_{\mathbf{PRG}}(\lambda)\right] - 1 \ ,$$

where the game $\mathrm{PRG}^{\mathcal{A}}_{\mathbf{PRG}}(\lambda)$ is shown in Figure 1 (top left). A PRG is secure if the above advantage function is negligible for every ppt adversary $\mathcal{A}$. In what follows, we assume $\mathcal{D}$ and $\mathcal{R}$ come with algorithms for sampling elements, which by slight abuse of notation we denote by $\mathcal{D}(1^\lambda)$ and $\mathcal{R}(1^\lambda)$. We allow for arbitrary domain and range in this definition to allow for the analysis of our constructions later on.

Pseudorandom functions. A PRF is a pair of algorithms $(\mathbf{Gen}, \mathbf{PRF})$, where $\mathbf{Gen}$ is a randomized algorithm that on input the security parameter 1^λ generates a key K in some key space $\mathcal{K}$. We will assume that this algorithm simply outputs a random keys in

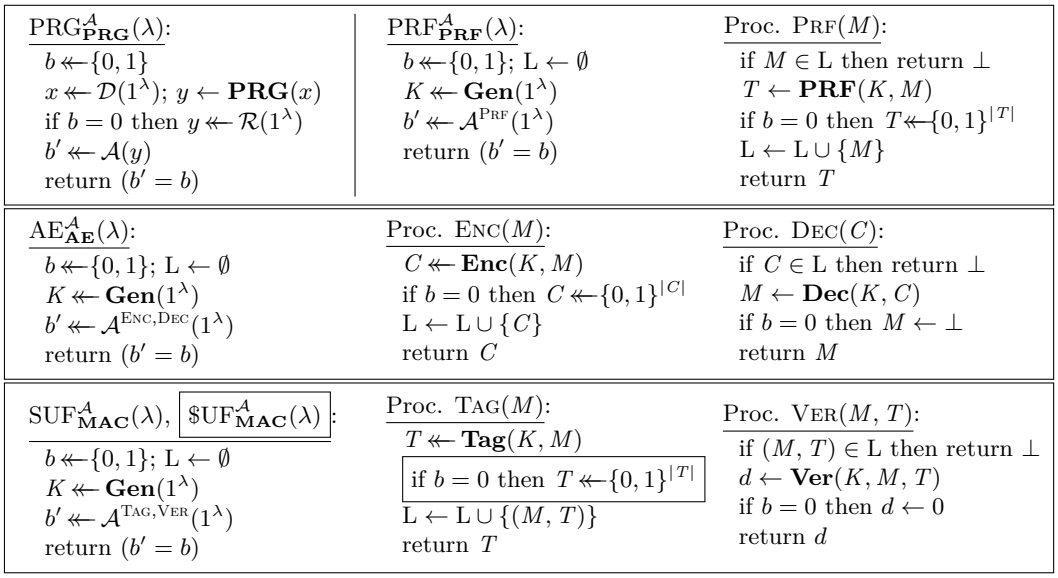


Figure 1: Games defining the security of pseudorandom generators (top left), pseudorandom functions (top right), authenticated encryption (middle), and pseudorandom and strongly unforgeable message authentication codes (down). The PRF, AE, and $UF notions entail strong notions of key anonymity for each primitive. IND$ security is a weakening of AE security where the adversary is not allowed to call the decryption oracle. The standard strong unforgeability game omits the boxed statement from the TAG procedure.

$\{0,1\}^\lambda$. Algorithm **PRF** is deterministic and given K as input and a point $x \in \mathcal{D}$ outputs a value $y \in \mathcal{R}$. We define the advantage of an adversary $\mathcal{A}$ against **PRF** as

$$\mathbf{Adv}^{\text{prf}}_{\mathbf{PRF},\mathcal{A}}(\lambda) \coloneqq 2 \cdot \Pr\left[\text{PRF}^{\mathcal{A}}_{\mathbf{PRF}}(\lambda)\right] - 1 \ ,$$

where game PRF$^{\mathcal{A}}_{\mathbf{PRF}}(\lambda)$ is shown in Figure 1 (top right). A **PRF** is secure if the above advantage function in negligible for every PPT adversary $\mathcal{A}$.

AUTHENTICATED ENCRYPTION. An authenticated encryption scheme **AE** is a triple of algorithms **AE** := (**Gen**, **Enc**, **Dec**) such that: (1) $\mathbf{Gen}(1^\lambda)$ is the randomized key-generation algorithm that on input the security parameter 1^λ outputs a key K; (2) $\mathbf{Enc}(K, M; R)$ is the randomized encryption algorithm that on input a key K, a plaintext M and possibly random coins R outputs a ciphertext C; (3) $\mathbf{Dec}(K, C)$ is the deterministic decryption algorithm that on input a key K and a ciphertext C, outputs a plaintext M or the special error symbol $\perp$. We call a scheme **AE** (perfectly) correct (for message space $\{0,1\}^*$) if for all $\lambda \in \mathbb{N}$, all $K \twoheadleftarrow \mathbf{Gen}(1^\lambda)$, all $M \in \{0,1\}^*$ and all $C \twoheadleftarrow \mathbf{Enc}(K, M)$ we have that $\mathbf{Dec}(K, C) = M$. We define the advantage of an adversary $\mathcal{A}$ against **AE** as

$$\mathbf{Adv}^{\text{ae}}_{\mathbf{AE},\mathcal{A}}(\lambda) \coloneqq 2 \cdot \Pr\left[\text{AE}^{\mathcal{A}}_{\mathbf{AE}}(\lambda)\right] - 1 \ ,$$

where game AE$^{\mathcal{A}}_{\mathbf{AE}}(\lambda)$ is shown in Figure 1 (middle). An AE scheme is secure if the above advantage function in negligible for every PPT adversary $\mathcal{A}$. This is the standard definition of security for AE schemes [RS06, HK07]. An alternative security definition would come with a challenge oracle that on input two messages (M_0, M_1) of same length, returns an encryption of M_b. This definition is weaker that AE security as the latter already implies a strong form of *anonymity* due to the pseudorandomness of ciphertexts, whereas this is

not necessary the case for the left-right-based definition.[2]

Message-authentication codes. A message-authentication code (MAC) is a triple of algorithms $\mathbf{MAC} := (\mathbf{Gen}, \mathbf{Tag}, \mathbf{Ver})$ defined as follows: (1) $\mathbf{Gen}(1^\lambda)$ is the randomized key generation algorithm that on input the security parameter 1^λ outputs a key K. (2) $\mathbf{Tag}(K, M; R)$ is the randomized tagging algorithm that on input a key K, a plaintext M and possibly random coins R, outputs a tag T. (3) $\mathbf{Ver}(K, M, T)$ is the deterministic verification algorithm that on input a key K, a plaintext M and a tag T, outputs a bit. We call a **MAC** scheme (perfectly) correct (for message space $\{0,1\}^*$) if for all $\lambda \in \mathbb{N}$, all $K \twoheadleftarrow \mathbf{Gen}(1^\lambda)$, all $M \in \{0,1\}^*$ and all $T \twoheadleftarrow \mathbf{Tag}(K, M)$, the verification is successful: $\mathbf{Ver}(K, M, T) = 1$. We define the advantage of an adversary $\mathcal{A}$ against a **MAC** as

$$\mathbf{Adv}^{\$\mathrm{uf}}_{\mathbf{MAC},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\$\mathrm{UF}^{\mathcal{A}}_{\mathbf{MAC}}(\lambda)\right] - 1 \ ,$$

where game $\$\mathrm{UF}^{\mathcal{A}}_{\mathbf{MAC}}(\lambda)$ is shown in Figure 1. This game strengthens the standard strong unforgeability for MACs, which is shown in the same figure omitting the boxed statement, in a number of aspects. First, Ver outputs the error symbol only when the pair (M, T) was generated via the Tag procedure and therefore pseudorandom MACs are also strongly unforgeable. Second, the definition implies the tags are pseudorandom and hence they fully hide the messages and the *keys* that were used to generate them. Stated differently, pseudorandom MACs are both *confidential* and *anonymous* in the sense that they hide both the message and the key that is used to generate a tag. Finally, since Tag does not repeat T for repeated messages, it implies a notion of *unlinkability*.

Feasibility of pseudorandom MACs. Given a **PRF**, consider scheme **MAC** whose key-generation algorithm is identical to that of the **PRF** and whose tag-generation and verification algorithms operate as

$$\mathbf{Tag}(K, M; R) \coloneqq R || \mathbf{PRF}(K, M||R), \quad \mathbf{Ver}(K, M, (R||T)) \coloneqq (T \stackrel{?}{=} \mathbf{PRF}(K, M||R)) \ .$$

We call such message authentication codes *randomized* **MAC***s*. It is straightforward to prove that this **MAC** satisfies our strong security notion for **MAC**s given above.

Encrypt-then-MAC. Recall that in the Encrypt-then-MAC paradigm, one first encrypts a message M and finally authenticates the resulting ciphertext using a MAC. If the underlying encryption **AE** in this transform is AE-secure without access to decryption oracle (a.k.a. IND$ secure) and the **MAC** used is pseudorandom, the encryption scheme is AE secure [BN08].

3 Definitions

Informally, in a robust scheme no unexpected collisions in the input/output behavior of the system exists. For instance, in the case of encryption no adversary should be able to compute a ciphertext that decrypts correctly under two distinct keys. This notion was first formulated in the asymmetric setting [ABN10, FLPQ13] and we adapt it to symmetric encryption, MACs, and PRFs in this section.

The work of [FLPQ13] refines and strengthens the original definitions of robustness [ABN10]. The central security notion introduced in [FLPQ13] is *complete robustness* (CROB), a notion that contains three sub-notions of full robustness (FROB), key-less robustness (KROB) and mixed robustness (XROB). These, roughly speaking, correspond to three possible ways finding a colliding ciphertext using either the encryption or decryption

[2] The left-right-based definition can be modified to imply a left-right notion of key anonymity. The resulting game, however, is both more cumbersome to work with and weaker than standard AE security.

algorithms of the scheme. That is, for some $K_1, K_2, M_1, M_2, R_1, R_2, C_1, C_2$ at least one of the checks

$$\mathbf{Enc}(K_1, M_1; R_1) = \mathbf{Enc}(K_2, M_2; R_2), \quad \text{or} \quad \mathbf{Dec}(K_1, C_1) = \mathbf{Dec}(K_2, C_1), \quad \text{or}$$

$$\mathbf{Dec}(K_2, \mathbf{Enc}(K_1, M_1; R_1)) = M_2,$$

pass when $K_1 \neq K_2$. The last condition can be made stronger: one expects that $\mathbf{Enc}(K_1, M_1; R_1)$ would decrypt to $\perp$ under an unrelated key K_2. The middle check can be also made stronger by only checking that the outputs are both valid. We formalize the resulting notions next.

Robustness. We define the advantage of an adversary $\mathcal{A}$ in the CROB games against an encryption scheme $\mathbf{AE}$ as

$$\mathbf{Adv}^{\text{crob}}_{\mathbf{AE},\mathcal{A}}(\lambda) := \Pr\left[\text{CROB}^{\mathcal{A}}_{\mathbf{AE}}(\lambda)\right] ,$$

where game $\text{CROB}^{\mathcal{A}}_{\mathbf{AE}}(\lambda)$, is shown in Figure 2 (top). Similarly, for a message authentication code $\mathbf{MAC}$ we define

$$\mathbf{Adv}^{\text{crob}}_{\mathbf{MAC},\mathcal{A}}(\lambda) := \Pr\left[\text{CROB}^{\mathcal{A}}_{\mathbf{MAC}}(\lambda)\right] ,$$

where $\text{CROB}^{\mathcal{A}}_{\mathbf{MAC}}(\lambda)$ is shown in Figure 2 (bottom).

$\underline{\text{CROB}^{\mathcal{A}}_{\mathbf{AE}}(\lambda)}$:
$\text{L} \leftarrow \emptyset$
$\varepsilon \twoheadleftarrow \mathcal{A}^{\text{Enc},\text{Dec}}(1^\lambda)$
for $(K_1, M_1, C_1), (K_2, M_2, C_2) \in \text{L}$ do
  if $(C_1 = C_2 \neq \perp) \wedge (K_1 \neq K_2) \wedge (M_1 \neq \perp \wedge M_2 \neq \perp)$
    return 1
return 0

$\underline{\text{Proc. Enc}(K, M, R)}$:
$C \leftarrow \mathbf{Enc}(K, M; R)$
$\text{L} \leftarrow \text{L} \cup (K, M, C)$

$\underline{\text{Proc. Dec}(K, C)}$:
$M \leftarrow \mathbf{Dec}(K, C)$
$\text{L} \leftarrow \text{L} \cup (K, M, C)$

$\underline{\text{CROB}^{\mathcal{A}}_{\mathbf{MAC}}(\lambda)}$:
$\text{L} \leftarrow \emptyset$
$\varepsilon \twoheadleftarrow \mathcal{A}^{\text{Tag},\text{Ver}}(1^\lambda)$
for $(K_1, M_1, T_1), (K_2, M_2, T_2) \in \text{L}$ do
  if $(T_1 = T_2 \neq \perp) \wedge (K_1 \neq K_2)$ then
    return 1
return 0

$\underline{\text{Proc. Tag}(K, M, R)}$:
$T \leftarrow \mathbf{Tag}(K, M; R)$
$\text{L} \leftarrow \text{L} \cup (K, M, T)$

$\underline{\text{Proc. Ver}(K, M, T)}$:
$b \leftarrow \mathbf{Ver}(K, M, T)$
if $b = 1$ then $\text{L} \leftarrow \text{L} \cup (K, M, T)$

Figure 2: Complete robustness for symmetric encryption (top) and MAC (bottom).

Farshim et al. [FLPQ13] give pair-wise separations among the three sub-notions mentioned above, showing they are all incomparable and hence should be (implicitly) included in the CROB notion. Some of these separations use *invalid* keys, as key pairs in the public-key setting cannot be necessarily checked for validity. This issue disappears in our setting as the key space is $\{0,1\}^k$, a set which is trivially checkable for validity. This fact simplifies relations among notions (which we study in detail in Appendix A). Analogues of the FROB notion [FLPQ13] for AE and MAC turn out to be equivalent to our strongest notions above. We formalize FROB in Figure 3 (left) for AE and (middle) for MACs, and summarize this discussion is under Theorem 1. Figure 3 (right) also includes our definition of robustness for PRFs with advantage function

$$\mathbf{Adv}^{\text{frob}}_{\mathbf{PRF},\mathcal{A}}(\lambda) := \Pr\left[\text{FROB}^{\mathcal{A}}_{\mathbf{PRF}}(\lambda)\right] .$$

$\underline{\text{FROB}^{\mathcal{A}}_{\mathbf{AE}}(\lambda)}$:	$\underline{\text{FROB}^{\mathcal{A}}_{\mathbf{MAC}}(\lambda)}$:	$\underline{\text{FROB}^{\mathcal{A}}_{\mathbf{PRF}}(\lambda)}$:
$(C, K_1, K_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$	$(T, K_1, M_1, K_2, M_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$	$(K_1, M_1, K_2, M_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$
if $K_1 = K_2$ return 0	if $K_1 = K_2$ return 0	if $K_1 = K_2$ return 0
$M_1 \leftarrow \mathbf{Dec}(K_1, C)$	$d_1 \leftarrow \mathbf{Ver}(K_1, M_1, T)$	$T_1 \leftarrow \text{PRF}(K_1, M_1)$
$M_2 \leftarrow \mathbf{Dec}(K_2, C)$	$d_2 \leftarrow \mathbf{Ver}(K_2, M_2, T)$	$T_2 \leftarrow \text{PRF}(K_2, M_2)$
return $(M_1 \neq \perp \wedge M_2 \neq \perp)$	return $(d_1 = d_2 = 1)$	return $(T_1 = T_2)$

Figure 3: Games defining full robustness for a symmetric encryption scheme **AE** (left), a message authentication code **MAC** (middle) and a pseudorandom function **PRF** (right).

As we shall see, from a foundational perspective, robust **PRF**s underlie feasibility of robustness for many symmetric primitives.

Collision resistance. Complete robustness strengthens to *unkeyed* collision resistance when the case $K_1 = K_2$ is not ruled out. For MACs, (unkeyed) collision-resistance states that it should be hard to come up with $(K_1, M_1, R_1) \neq (K_2, M_2, R_1)$ such that $\mathbf{Tag}(K_1, M_1; R_1) = \mathbf{Tag}(K_2, M_2; R_1)$. Similarly, (unkeyed) collision resistance of **PRF**s requires that $\mathbf{PRF}(K_1, M_1) \neq \mathbf{PRF}(K_2, M_2)$ for $(K_1, M_1) \neq (K_2, M_2)$. The standard notion of keyed collision resistance, on the other hand, imposes that keys are equal, $K_1 = K_2$, and are honestly generated. (Note that unforgeable MACs and pseudorandom PRFs are always keyed collision-resistant.)

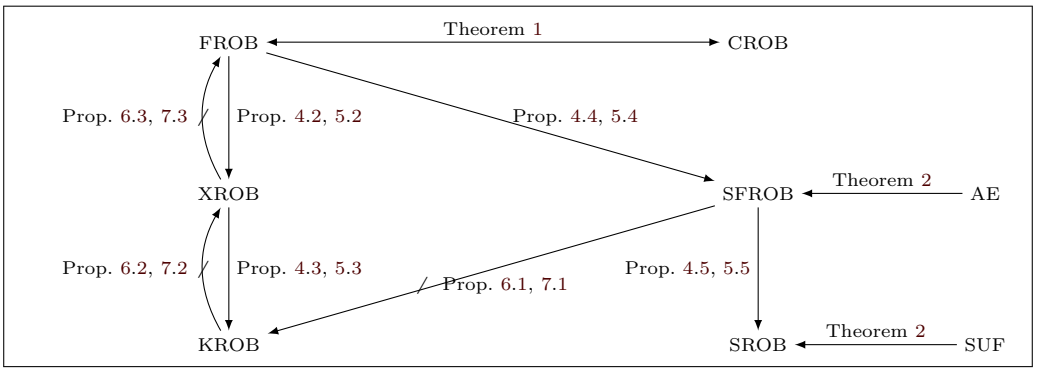


Figure 5: Relations among notions of robustness (with key validity) for AE and MAC schemes. XROB and KROB are formalized in Appendix A, and SFROB and SROB are formalized in Section 4. A crossed arrow entails a separation; see Appendix B.

Theorem 1 (Robustness with key validity)**.** *Let* **AE** *be a perfectly correct symmetric encryption scheme that checks keys for validity during encryption. Then* **AE** *is* CROB *secure if and only if it is* FROB *secure. Similarly, a perfectly correct message authentication code* **MAC** *whose tag-generation algorithm checks keys for validity is* CROB *secure if and only if it is* FROB *secure.*

Proof. The proof is simple and we give an example for one case. Suppose that adversary $\mathcal{A}$ wins the CROB game by finding a collision between the outputs of encryption. In other words $\mathcal{A}$ finds $(K_1, M_1, R_1, K_2, M_2, R_2)$ such that:

$$\mathbf{Enc}(K_1, M_1; R_1) = \mathbf{Enc}(K_2, M_2; R_2) \ .$$

This means that K_1 and K_2 are valid. Now consider an FROB adversary that computes $C := \mathbf{Enc}(K_1, M_1; R_1)$ and outputs (C, K_1, K_2). By the perfect correctness of the scheme

for *valid keys* it must be the case that $\mathbf{Dec}(K_1, C) = M_1$ and $\mathbf{Dec}(K_2, C) = M_2$, which wins the FROB game.

Other cases are dealt similarly, by either computing a colliding ciphertexts using **Enc** or a colliding tag using **Tag**. We provide the details in Appendix A. □

Throughout the paper we assume that keys are checkable for validity and that they are indeed checked for validity in all algorithms. Hence we will only use FROB security to establish CROB security in the subsequent sections. We limit our study to schemes that have *perfect* correctness (as defined under syntax). Correctness with all but negligible probability would allow for artificial attacks and separations. As an example, consider an encryption scheme that, when invoked with a special random tape computes the identity function – this is allowed since the probability of hitting that random tape is negligible and at the same time gives an easy way to break robustness.

4 Robustness, AE Security, and Unforgeability

We show that standard AE-secure encryption schemes offer a basic level of resilience against incorrect usage of keys. The level of robustness offered corresponds to a setting where the adversary does not get to choose any keys. Instead, two keys are honestly generated and the adversary is given oracle access to encryption and decryption algorithms under *both* keys. The notion for MACs is similar where oracle access to tag-generation and verification algorithms under honestly generated keys are provided to the adversary. This notions which we call *strong robustness* (SROB) are shown in Figure 6 (without boxed variables). This nomenclature follows the original notion of strong robustness by Abdalla et al. [ABN10]. We also define *semi-full robustness* (SFROB) as one where the adversary gets to see one of the keys (as shown in Figure 6 with boxed variables).

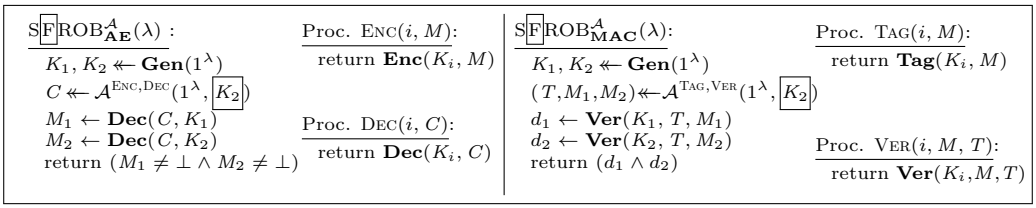


Figure 6: Games defining strong robustness (SROB) and semi-full robustness (SFROB) for symmetric encryption (left) and MACs (right). The boxed statements are included in the boxed games.

Theorem 2. *Any authentication scheme* **AE** *that is* AE*-secure is also* SFROB*-secure. Any strongly unforgeable (and in particular pseudorandom) scheme* **MAC***, on the other hand, is (only)* SROB*-secure. More precisely, for any adversary* $\mathcal{A}$ *against the* SFROB *security of the AE scheme, there is an adversary* $\mathcal{B}$ *against the* AE *security of the scheme such that*

$$\mathbf{Adv}^{\mathrm{sfrob}}_{\mathbf{AE},\mathcal{A}}(\lambda) \leq 3 \cdot \mathbf{Adv}^{\mathrm{ae}}_{\mathbf{AE},\mathcal{B}}(\lambda) \ .$$

Moreover, for any adversary $\mathcal{A}$ *against the* SROB*-security of the MAC there is an adversary* $\mathcal{B}$ *against the* SUF*-security of the scheme such that*

$$\mathbf{Adv}^{\mathrm{srob}}_{\mathbf{MAC},\mathcal{A}}(\lambda) \leq 3 \cdot \mathbf{Adv}^{\mathrm{suf}}_{\mathbf{AE},\mathcal{B}}(\lambda) \ .$$

Furthermore, there exist a pseudorandom MAC that is not SFROB*-secure.*

Proof. First, we prove the implication from AE security to SFROB. Let G_0 be the SFROB game. We assume without loss of generality that the adversary in G_0 never queries the $\textsc{Enc}(2,\cdot)$ and $\textsc{Dec}(2,\cdot)$ oracles as it has access to K_2, and that it never queries an output of $\textsc{Enc}(1,\cdot)$ to $\textsc{Dec}(1,\cdot)$ as it already knows the answer.

In G_1 we modify the winning condition of G_0 as follows. When the adversary returns a ciphertext C, instead of checking that $\mathbf{Dec}(C,K_1)\neq\perp$ and $\mathbf{Dec}(C,K_2)\neq\perp$, the game checks if C was one of the ciphertexts that was returned from the $\textsc{Enc}(1,\cdot)$ oracle and that $\mathbf{Dec}(C,K_2)\neq\perp$. The games G_0 and G_1 are identical unless $\mathcal{A}$ outputs a ciphertext C that was not obtained from the $\textsc{Enc}(1,\cdot)$ oracle, but decrypts correctly (call this event E). We bound the probability of E via the AE as follows. For any distinguishing $\mathcal{A}$, we define an algorithm $\mathcal{B}$ that picks a random key K_2, runs $\mathcal{A}(K_2)$, and answers its queries using its own equivalent pair of oracles. When $\mathcal{A}$ terminates with C, algorithm $\mathcal{B}$ queries C to its decryption oracle to get M_1 and also computes $M_2 \leftarrow \mathbf{Dec}(C,K_2)$. It returns $(M_1\neq\perp \wedge M_2\neq\perp)$. If $\mathcal{B}$'s decryption oracle is fake and implements $\perp$, algorithm $\mathcal{B}$ will always return 0. If $\mathcal{B}$'s decryption oracle is real, algorithm $\mathcal{B}$ runs $\mathcal{A}$ according to the environment of G_0 and G_1, and will output 1 whenever E happens. Hence $\Pr[\mathrm{G}_0^{\mathcal{A}}]-\Pr[\mathrm{G}_1^{\mathcal{A}}]\leq\Pr[E]=\mathbf{Adv}^{\mathrm{ae}}_{\mathbf{AE},\mathcal{B}}(\lambda)$.

In G_2 we replace $\textsc{Enc}(1,\cdot)$ and $\textsc{Dec}(1,\cdot)$ with the \$ and $\perp$ procedures respectively. (As in G_1 we still use the list of ciphertexts and K_2 for the winning condition.) The distance between G_1 and G_2 can be bounded via the AE game as follows. Consider an AE adversary $\mathcal{B}$ that generates an independent key K_2 and runs a distinguishing adversary $\mathcal{A}(K_2)$. Algorithm $\mathcal{B}$ answers $\mathcal{A}$'s oracle queries using the oracles provided to it. When $\mathcal{A}$ terminates with a ciphertext C, algorithm $\mathcal{B}$ performs the winning check and outputs its result. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ with respect to the real or replaced procedures according to the real or fake procedures that it gets. The output of $\mathcal{B}$ is identical to that of $\mathcal{A}$ in the two games. Hence $\Pr[\mathrm{G}_1^{\mathcal{A}}]-\Pr[\mathrm{G}_2^{\mathcal{A}}]\leq\mathbf{Adv}^{\mathrm{ae}}_{\mathbf{AE},\mathcal{B}}(\lambda)$.

In G_2, the adversary has essentially no control over C and we show its advantage is small. For this we will rely on the AE game once more (but now implicitly with respect to the second key). G_2 can only be won if $\mathbf{Dec}(K_2,C)\neq\perp$ for at least one of q distinct random strings C obtained from the \$ oracle. Consider an **AE** adversary $\mathcal{B}$ that generates q such random C and queries them to its $\textsc{Dec}$ oracle and outputs 1 if and only if one of the answers is non-$\perp$. Adversary $\mathcal{B}$ always outputs 0 when the oracle implements $\perp$. On the other hand, when the oracle implements the real decryption routine, the probability of $\mathcal{B}$ outputting 1 is exactly the probability that $\mathbf{Dec}(K_2,C)=\perp$ for one of the random C and key K_2. This means $\Pr[\mathrm{G}_2^{\mathcal{A}}]\leq\mathbf{Adv}^{\mathrm{ae}}_{\mathbf{AE},\mathcal{B}}(\lambda)$. The first part of the theorem follows from that last (in)equalities.

We now prove the second part of the Theorem 2. We first note that via a simple hybrid argument, unforgeability with respect to two keys reduces to unforgeability with respect to a single key with loss 2 in advantage. We also assume, without loss of generality, that an adversary in $\mathrm{G}_0 :=$ SROB does not query $\textsc{Ver}$ on any (i,M,T) where T is an output of $\textsc{Tag}(i,M)$; the answer is always 1 for such queries.

In G_1 we replace the $\textsc{Ver}(1,\cdot,\cdot)$ and $\textsc{Ver}(2,\cdot,\cdot)$ procedures with the $\perp$ procedure. We also replace the computation of $\mathbf{Ver}(K_i,T,M)$ for $i=1,2$ in the winning condition with 0 unless T was output by *both* $\textsc{Tag}(1,\cdot)$ and $\textsc{Tag}(2,\cdot)$ procedures. Hence G_0 and G_1 are identical unless $\mathcal{A}$ outputs a tag T that was not output by both tag-generation oracles and yet verifies under both keys. Call this event E. The probability of event E can be bounded via the (single-key) SUF game as follows. Algorithm $\mathcal{B}$ generates a key K_2. It uses its own oracles and K_2 to simulate the oracles for $\mathcal{A}$. When $\mathcal{A}$ terminates with a tag (T,M_1,M_2), algorithm $\mathcal{B}$ queries (T,M_1) to its verification oracle and returns 1 iff the result was not 0. Algorithm $\mathcal{B}$ will always output 0 when the oracle is 0 (i.e., when it is fake). If its oracles are real, $\mathcal{B}$ runs $\mathcal{A}$ according to the environments of G_0 and G_1, and whenever E happens it returns 1. Hence, $\Pr[\mathrm{G}_0^{\mathcal{A}}]-\Pr[\mathrm{G}_1^{\mathcal{A}}]\leq\Pr[E]=\mathbf{Adv}^{\mathrm{suf}}_{\mathbf{MAC},\mathcal{B}}(\lambda)$.

The advantage of any adversary $\mathcal{A}$ in G_1 can be bounded, once again, by the two-key SUF game as follows. Consider any adversary against the two-key SUF game as follows. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and answers its oracle queries using its own oracles. When $\mathcal{A}$ terminates with a tag (T, M_1, M_2), algorithm $\mathcal{B}$ checks for which i this tag was not obtained from $\textsc{Tag}(i, \cdot)$ (if both, it chooses either i). Algorithm $\mathcal{B}$ the queries $\textsc{Ver}(i, T, M_i)$ and returns 1 iff the result is not 0. Note that $\mathcal{B}$ never outputs 1 when its oracles are fake. However, when its oracles are real $\mathcal{B}$ runs $\mathcal{A}$ according to the rules of G_1 and it returns 1 whenever $\mathcal{A}$ wins. Hence, $\Pr[G_2^{\mathcal{A}}] \leq 2 \cdot \mathbf{Adv}^{\text{suf}}_{\mathbf{MAC},\mathcal{B}}(\lambda)$. The second part of the theorem follows.

Interestingly, MAC security (including pseudorandomness) does not imply SFROB security for MACs. (And the above theorem is, in a sense, "sharp.") Indeed, given a pseudorandom MAC consider a modified scheme whose verification procedure on input $M = K$ and *any* tag always passes. This MAC can be still shown to be pseudorandom (without access to K), but fails to be SFROB as any tag T obtained under K_1 for, say, message 0 would be also valid with respect to K_2 if message $M_2 \coloneqq K_2$. Note, however, that since any AE scheme is a pseudorandom MAC, the result for AE schemes shows SFROB-secure MACs can be built via authenticated encryption. □

In the above proof we showed that for MACs, SROB is strictly weaker than SFROB, and hence it also weaker than CROB. We next prove that SFROB is weaker than CROB for AE schemes. We show a stronger result that not all AE schemes, even those obtained via Encrypt-then-MAC, are CROB.

Proposition 1. *There exist an authenticated encryption scheme obtained via the Encrypt-then-MAC transform that is not* CROB *secure (but* SFROB *secure as shown in Theorem 2).*

Proof. Consider any symmetric encryption scheme whose decryption algorithm never outputs $\perp$. (A natural example is a scheme whose encryption algorithm evaluates a **PRF** at a random point and masks the message with the result: $\mathbf{Enc}(K_e, M; R) \coloneqq R||\text{PRF}(K_e, R) \oplus M$.) Then the **AE** scheme obtained by applying the EtM transform using such an encryption scheme and *any* **MAC** (even robust ones) will not be CROB secure. For a random **MAC** key K_m and random and distinct encryption keys K_{e_1}, K_{e_2} consider an attacker that computes $C \twoheadleftarrow \mathbf{Enc}(K_{e_1}, 0)$ and $T \twoheadleftarrow \mathbf{Tag}(K_m, C)$ and outputs $\big((C||T), (K_{e_1}||K_m), (K_{e_2}||K_m)\big)$. The ciphertext $(C||T)$ will decrypt to a valid message under the distinct keys $(K_{e_1}||K_m)$ and $(K_{e_2}||K_m)$ as the tag T is always checked against K_m and the base encryption scheme does not have invalid ciphertexts. □

The attack described above applies against authenticated encryption schemes that follow the EtM transform and use independent keys for the encryption and MAC components. If the same key is used for both the encryption and authentication components (and assuming the **AE** security of the composed construction), the above attack no longer works. Artificial counterexamples, however, still exist. As before, consider a MAC that verifies whenever $M = K$ irrespectively of its input tag. Such a MAC, when combined with any encryption scheme whose decryption never returns $\perp$ gives rise to a separating example between CROB and SFROB for AE schemes. Here the attacker gets K_2, sets $C := K_2$, computes a tag $T = \mathbf{Tag}(K_1, C)$ and outputs $((C||T), K_1, K_2)$. Now the verification of T for C with K_1 always passes. It also passes with respect to K_2 and $K_2 = C$. Since **Dec** never outputs $\perp$ in the base scheme, C also decrypts under both keys.

FROB INSECURITY OF CBC-MAC. We conclude this section showing that the popular CBC-MAC is not FROB (or even SFROB) secure as the block cipher used in CBC-MAC is invertible. In CBC-MAC, a tag is generated as $C_i = E(K, C_{i-1} \oplus M_i)$, with $C_0 := IV$ for some fixed IV. To attack the (semi-)full robustness of CBC-MAC, for two random keys K_1, K_2 take any plaintext M, generate $T \leftarrow E(K_1, M_1 \oplus IV)$, compute $M_2' \leftarrow D(K_2, T)$,

and set $M_2 := M_2' \oplus IV$. Now (T, K_1, M_1, K_2, M_2) constitutes a break against the (semi-)full robustness of CBC-MAC.

5 Constructions

We now prove two positive results for obtaining robust encryption through generic composition.

Theorem 3 (Robustness for generic composition)**.** *The* **AE** *schemes obtained through either Encrypt-then-Mac (*EtM*), Encrypt-and-MAC (*EaM*), or MAC-then-Encrypt (*MtE*) (with independent keys) are* CROB *secure as long as their encryption and MAC components are* CROB *secure. Moreover, the* **AE** *scheme obtained through* EtM*,* EaM *or* MtE *when reusing the same key for encryption and authentication is* CROB *secure as long as* either *the encryption or the MAC component is* CROB *secure.*

Proof. We provide the proofs for the three cases separately.

EtM composition. Suppose a CROB adversary $\mathcal{A}$ outputs a tuple $((C||T), (K_{e_1}||K_{m_1}), (K_{e_2}||K_{m_2}))$ winning the CROB game against the generically composed scheme with distinct keys. Since $(K_{e_1}, K_{m_1}) \neq (K_{e_2}, K_{m_2})$ there are two possibilities to consider:

Case $K_{e_1} \neq K_{e_2}$: then (C, K_{e_1}, K_{e_2}) wins the CROB game against encryption, as C would have decrypted correctly with respect to both keys for $\mathcal{A}$ to be successful.

Case $K_{m_1} \neq K_{m_2}$: then $(T, K_{m_1}, C, K_{m_2}, C)$ wins the CROB game for **MAC** as T would have to be a valid tag with respect to C and two distinct keys.

We conclude that for adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, $\mathbf{Adv}^{\text{crob}}_{\text{EtM},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{crob}}_{\mathbf{AE},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{crob}}_{\mathbf{MAC},\mathcal{B}_2}(\lambda)$. When the keys are reused, we can apply *both* branches of the reduction above. This proves CROB security of the composed scheme assuming CROB for *either* the **AE** or **MAC** component of the scheme and get $\mathbf{Adv}^{\text{crob}}_{\text{EtM},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{crob}}_{\mathbf{AE},\mathcal{B}_1}(\lambda)$ and $\mathbf{Adv}^{\text{crob}}_{\text{EtM},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{crob}}_{\mathbf{MAC},\mathcal{B}_2}(\lambda)$.

EaM composition. Suppose a CROB adversary $\mathcal{A}$ outputs a tuple $((C||T), (K_{e_1}||K_{m_1}), (K_{e_2}||K_{m_2}))$ winning the CROB game against the EaM generically composed scheme with distinct keys. Since $(K_{e_1}, K_{m_1}) \neq (K_{e_2}, K_{m_2})$, as for the EtM transform, if: (1) $K_{e_1} \neq K_{e_2}$, we have that (C, K_{e_1}, K_{e_2}) wins the CROB game against encryption, as C would have decrypted correctly with respect to both keys for $\mathcal{A}$ to be successful; (2) for the second case we let $M_1 \leftarrow \mathbf{Dec}(K_{e_1}, C)$ and $M_2 \leftarrow \mathbf{Dec}(K_{e_2}, C)$; when $K_{m_1} \neq K_{m_2}$, then $(T, K_{m_1}, M_1, K_{m_2}, M_2)$ wins the CROB game for **MAC** as T would have to be a valid tag with respect to M_1, M_2 and both keys for $\mathcal{A}$ to be successful. Thus for adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, the advantage of $\mathcal{A}$ can be bounded as follows $\mathbf{Adv}^{\text{crob}}_{\text{EaM},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{crob}}_{\mathbf{AE},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{crob}}_{\mathbf{MAC},\mathcal{B}_2}(\lambda)$. When the keys are reused, the same argument as in the previous case applies.

MtE composition. Let a CROB adversary $\mathcal{A}$ output a tuple $(C, (K_{e_1}||K_{m_1}), (K_{e_2}||K_{m_2}))$ winning the CROB game against the MtE generically composed scheme with distinct keys. Since $(K_{e_1}, K_{m_1}) \neq (K_{e_2}, K_{m_2})$, as for the EtM transform, if: (1) $K_{e_1} \neq K_{e_2}$, we have that (C, K_{e_1}, K_{e_2}) wins the CROB game against encryption, as C would have decrypted correctly with respect to both keys for $\mathcal{A}$ to be successful. Thus we assume $K_{e_1} = K_{e_2}$ and let $(M||T) \leftarrow \mathbf{Dec}(K_{e_1}, C)$; (2) when $K_{m_1} \neq K_{m_2}$ then $(T, K_{m_1}, M, K_{m_2}, M)$ wins the CROB game for **MAC** as T would have to be a valid tag with respect to M and both keys for $\mathcal{A}$ to be successful. (Note that the same tag is obtained after decryption).

Therefore for adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ the advantage of $\mathcal{A}$ is bounded as $\mathbf{Adv}^{\text{crob}}_{\text{MtE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{crob}}_{\mathbf{AE},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{crob}}_{\mathbf{MAC},\mathcal{B}_2}(\lambda)$. When the keys are reused, the same argument as in the first case applies. □

Some CAESAR candidates follow the generic composition paradigm but incorporate various optimizations to reduce computation, bandwidth and keying material. This means that a strategy similar to Theorem 3 can be used to carry out robustness proofs for them. We leave a provable security treatment of the robustness of the CAESAR candidates to future work.

To instantiate the components in Theorem 3, we start by observing that randomizing a CROB-secure **PRF** gives a pseudorandom **MAC** that is CROB secure. Indeed, a successful CROB adversary against this randomized **PRF** outputs a tuple (T, K_1, M_1, K_2, M_2) with $T = (R, Y)$ such that $\mathbf{PRF}(K_1, M_1||R) = Y = \mathbf{PRF}(K_2, M_2||R)$, which means $(Y, K_1, M_1||R, K_2, M_2||R)$ wins the CROB game against **PRF**.

An analogous route for directly building a CROB secure encryption scheme from a CROB secure **PRF** does *not* go through as the decryption algorithm of such schemes would never return $\perp$. However, by using a *common* **PRF** in both the encryption and MAC components we safely reuse the keys across encryption and MAC. More precisely, given a CROB-secure **PRF**, the following scheme is both CROB and AE secure

$$\begin{aligned}\mathbf{Enc}(K, M; R) &:= (R, \mathbf{PRF}(K, R) \oplus M, \mathbf{PRF}(K, \mathbf{PRF}(K, R) \oplus M)))\\ \mathbf{Dec}(K, (R, C, T)) &:= \text{ if } \mathbf{PRF}(K, C) = T \text{ return } \mathbf{PRF}(K, R) \oplus C \text{ else return } \perp .\end{aligned}$$

By our theorem above, this scheme is CROB as long as the **PRF** is CROB. An alternative and practical route for achieving robustness makes use of a random oracle to instantiate the **MAC** as it can be easily shown to be CROB and also allows secure reuse of keys with any scheme.

The above raises the question if robustness can be achieved *without key reuse* or random oracles. Such an approach is sometimes recommended as it allows for modular proofs of **AE** security. Below we give a transform akin to EtM that also *authenticates the encryption key* and which results in a scheme that is both AE and CROB secure. We give the details of the transform in Figure 7.

$\overline{\mathbf{Gen}}(1^\lambda)$:	$\overline{\mathbf{Enc}}((K_e, K_m), M)$:	$\overline{\mathbf{Dec}}((K_e, K_m), (C\|\|T))$:
$K_e \twoheadleftarrow \mathbf{Gen}_e(1^\lambda)$	$C \twoheadleftarrow \mathbf{Enc}(K_e, M)$	if $\mathbf{Ver}(K_m, (C\|\|K_e), T) = 0$ return $\perp$
$K_m \twoheadleftarrow \mathbf{Gen}_m(1^\lambda)$	$T \twoheadleftarrow \mathbf{Tag}(K_m, (C\|\|K_e))$	$M \leftarrow \mathbf{Dec}(K_e, C)$
return (K_e, K_m)	return (C, T)	return M

Figure 7: The modified EtM transform that authenticated the encryption key via a collision-resistant MAC.

Theorem 4. *Suppose* $\mathbf{AE} = (\mathbf{Gen}_e, \mathbf{Enc}, \mathbf{Dec})$ *is* IND$ *secure (see Figure 1) and* $\mathbf{MAC} = (\mathbf{Gen}_m, \mathbf{Tag}, \mathbf{Ver})$ *is pseudorandom. Then the scheme* $\overline{\mathbf{AE}} = (\overline{\mathbf{Gen}}, \overline{\mathbf{Enc}}, \overline{\mathbf{Dec}})$ *in Figure 7 is* AE *secure. Furthermore, this scheme is* CROB *secure if* **MAC** *is collision resistant.*

Proof. For CROB, consider an adversary that outputs $((C||T), (K_e||K_m), (K'_e||K'_m))$ such that $(C||T)$ decrypts to valid messages under both keys. Then the tag T must also verify under both K_m and K'_m. This however constitutes an attack on the collision resistance of **MAC** unless $K_m = K'_m$ and $K_e = K'_e$.

For AE security, we follow the standard path as follows. Let G_0 be the AE with real procedure. In G_1 we compute T in the Enc procedure by replacing T with random bit strings, and also replace the Dec procedure with the $\perp$ procedure. We can bound the

difference between G_0 and G_1 using a direct reduction to the pseudorandomness of **MAC**: $\Pr[G_0^{\mathcal{A}}] - \Pr[G_1^{\mathcal{A}}] \leq \mathbf{Adv}^{\$\text{uf}}_{\mathbf{MAC},\mathcal{B}_1}(\lambda)$. In G_2 we replace the ciphertext components in the outputs of the ENC procedure with random strings. Again, using a reduction to the IND\$ security of **AE** we can bound the difference between games G_1 and G_2: $\Pr[G_2^{\mathcal{A}}] - \Pr[G_3^{\mathcal{A}}] \leq = \mathbf{Adv}^{\text{ind-\$}}_{\mathbf{AE},\mathcal{B}_2}(\lambda)$. Finally note that G_2 is the AE with fake procedures which translates to: $\mathbf{Adv}^{\text{ae}}_{\overline{\mathbf{AE}},\mathcal{A}}(\lambda) = \Pr[G_0^{\mathcal{A}}] - \Pr[G_2^{\mathcal{A}}] \leq \mathbf{Adv}^{\$\text{uf}}_{\mathbf{MAC},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{ind-\$}}_{\mathbf{SE},\mathcal{B}_2}(\lambda)$. □

5.1 The symmetric ABN transform

The starting point for our second construction is the transform introduced by Abdalla et al. [ABN10] (henceforth, the ABN transform) to convert any PKE scheme into one that is also completely robust as shown in [FLPQ13]. Roughly speaking in the ABN transform one commits to the public key during encryption, encrypts the decommitment along with the plaintext, and includes the commitment as part of the ciphertext. The commitment is then checked against the public key in the decryption algorithm. The transform is shown in Figure 8. ABN relies on a commitment scheme $(\mathbf{CPG}, \mathbf{Com}, \mathbf{Ver})$ and operates in the CRS model via a common parameter-generation algorithm **CPG**.

$\overline{\mathbf{PKSetup}}(1^\lambda)$:
$\mathbf{crs} \twoheadleftarrow \mathbf{CPG}(1^\lambda)$
return $\mathbf{crs}$

$\overline{\mathbf{PKGen}}(1^\lambda)$:
$(pk, sk) \twoheadleftarrow \mathbf{PKGen}(1^\lambda)$
return (pk, sk)

$\overline{\mathbf{PKEnc}}(\mathbf{crs}, pk, M)$:
$(\mathbf{com}, \mathbf{dec}) \twoheadleftarrow \mathbf{Com}(\mathbf{crs}, pk)$
$C \twoheadleftarrow \mathbf{PKEnc}(pk, (M, \mathbf{dec}))$
return $(C, \mathbf{com})$

$\overline{\mathbf{PKDec}}(\mathbf{crs}, pk, sk, (C, \mathbf{com}))$:
$(M, \mathbf{dec}) \leftarrow \mathbf{PKDec}(K, C)$
if $\mathbf{Ver}(\mathbf{crs}, pk, \mathbf{com}, \mathbf{dec})$ then
  return M
return $\perp$

Figure 8: The ABN transform [ABN10] for public-key encryption.

We ask if an analogue of ABN, perhaps in the CRS model, can be also formulated for symmetric encryption. In this setting there is no public key and a natural alternative would be to commit to the secret key instead. This however results in a *key-dependent message* being encrypted as the decommitment **dec** is computed based on the encryption key K. Furthermore, the commitment string **com** must be pseudorandom to accomplish AE security.

One can attempt to repair the ABN transform as follows. First, use a commitment scheme with pseudorandom commitments. Any *collision-resistant* **PRF** is equivalent to such a commitment scheme, where $\mathbf{crs} = \varepsilon$ (assuming the **PRF** does not use a CRS) and $\mathbf{Com}(M||K)$ outputs $(\mathbf{PRF}(K, M), K)$ as the $(\mathbf{com}, \mathbf{dec})$ pair. The verification algorithm simply checks the commitment by recomputing the **PRF** using K and M. This scheme is computationally hiding down to the pseudorandomness of **PRF**. Furthermore, it is computationally binding down to its collision resistance. This technique still does not resolve the key-dependency issue. Although in this scheme the decommitment string is simply a random PRF key independent of the encryption key, a *circular* dependency between the encryption key and the **PRF** key exists which prevents a proof to go through. (Recall that in the public-key setting this issue does not arise as the public key is a key-dependent value that is available "for free.")

To fix these issues we compute a string that acts as a "public labeling" of the encryption key, and which does not hurt the security of the scheme. We first expand K using a **PRG**, use its left-half in encryption, and commit to its right-half as the public labeling. For this, we must however ensure that different keys give always rise to different public labellings. This can be achieved if the **PRG** is collision resistant (for example injective) on the right-half of outputs. Such PRGs can be based on one-way permutations via Yao's

transform [Yao82]. Indeed, assuming π is a one-way permutation and $\mathbf{HC}$ is a hardcore predicate for it [GL89], we get a right-injective PRG via

$$\mathbf{PRG}(x) := \mathbf{HC}(x)||\mathbf{HC}(\pi(x))||\ldots||\mathbf{HC}(\pi^{|x|-1}(x))||\pi^{|x|}(x) \ .$$

Observe the last part in this $\mathbf{PRG}$ is a permutation, which provides the required injectivity. This results in the transform shown in Figure 9.

$\overline{\mathbf{Gen}}(1^\lambda)$:	$\overline{\mathbf{Enc}}(K_e, M)$:	$\overline{\mathbf{Dec}}(K_e, (C\|\|T))$:
$K_e \twoheadleftarrow \mathbf{Gen}_e(1^\lambda)$	$K_m \twoheadleftarrow \mathbf{Gen}_m(1^\lambda)$	$(K_e^1\|\|K_e^2) \leftarrow \mathbf{PRG}(K_e)$
return K_e	$(K_e^1\|\|K_e^2) \leftarrow \mathbf{PRG}(K_e)$	$(M\|\|K_m) \leftarrow \mathbf{Dec}(K_e^1, C)$
	$C \twoheadleftarrow \mathbf{Enc}(K_e^1, (M\|\|K_m))$	if $\mathbf{Ver}(K_m, (C\|\|K_e^2), T) = 0$
	$T \twoheadleftarrow \mathbf{Tag}(K_m, (C\|\|K_e^2))$	return $\perp$
	return $(C\|\|T)$	return M

Figure 9: The modified EtM transform for obtaining CROB security.

Theorem 5. *Suppose* $\mathbf{AE} = (\mathbf{Gen}_e, \mathbf{Enc}, \mathbf{Dec})$ *is* IND$ *secure,* $\mathbf{MAC} = (\mathbf{Gen}_m, \mathbf{Tag}, \mathbf{Ver})$ *is pseudorandom, and* $\mathbf{PRG}$ *is secure. Then the scheme* $\overline{\mathbf{AE}} = (\overline{\mathbf{Gen}}, \overline{\mathbf{Enc}}, \overline{\mathbf{Dec}})$ *in Figure 9 is* $\mathbf{AE}$ *secure. Furthermore, this scheme is* CROB *secure as long as* $\mathbf{MAC}$ *is collision resistant and* $\mathbf{PRG}$ *is right collision resistant.*

Proof. Suppose that an adversary computes a ciphertext $(C||T)$ that decrypts correctly under two keys $K_e \neq K'_e$. The fact that $K_e \neq K'_e$ together with the right collision resistance of $\mathbf{PRG}$ implies that $K_e^2 \neq K_e'^2$. This then can be used to break the collision resistance of $\mathbf{MAC}$ using the pair $(K_m, (C||K_e^2))$ and $(K'_m, (C||K_e'^2))$ where K_m and K'_m are computed by decrypting C using the left halves K_e^1 and $K_e'^1$ of the $\mathbf{PRG}$ output, respectively.

AE security can be proven in the standard way as follows. Let G_0 be the AE game with respect to the real encryption and decryption oracles. In G_1 we replace the outputs of the $\mathbf{PRG}$ with truly random bit strings. This transition can be justified using the security of $\mathbf{PRG}$: $\Pr[\mathrm{G}_0^{\mathcal{A}}] - \Pr[\mathrm{G}_1^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{prg}}_{\mathbf{PRG},\mathcal{B}_1}(\lambda)$. In G_2 we replace T with random tags and decryption with the $\perp$ oracle. A direction reduction to $UF security the $\mathbf{MAC}$ can be used to bound this transition: $\Pr[\mathrm{G}_1^{\mathcal{A}}] - \Pr[\mathrm{G}_2^{\mathcal{A}}] \leq \mathbf{Adv}^{\$\mathrm{uf}}_{\mathbf{MAC},\mathcal{B}_2}(\lambda)$. In G_3 we replace C with random strings via the IND$ security of the $\mathbf{AE}$. Now note that G_3 corresponds to the AE game with respect to the fake encryption and decryption oracles: $\Pr[\mathrm{G}_2^{\mathcal{A}}] - \Pr[\mathrm{G}_3^{\mathcal{A}}] \leq \mathbf{Adv}^{\mathrm{ind\$}}_{\mathbf{AE},\mathcal{B}_3}(\lambda)$. □

One advantage of the second transform is that it only relies on the pseudorandomness of $\mathbf{MAC}$ with freshly generated keys. This in turns allows for simple instantiation of it. For a right collision-resistant $\mathbf{PRG}$, let

$$\mathbf{PRG}(K) = \mathbf{PRG}_0(K)||\mathbf{PRG}_1(K) \text{ with } (\ell_0, \ell_1) := (|\mathbf{PRG}_0(K)|, |\mathbf{PRG}_1(K)|) \ .$$

Then we compute a MAC on a (hashed) message M with $|M| = \ell_1$ as:

$$\mathbf{Tag}(K, M) := (M||0^{\ell_0}) \oplus \big(\mathbf{PRG}_0(K)||\mathbf{PRG}_1(K)\big) \ .$$

The collision resistance of this MAC follows from the fact that the right (and collision-resistant) half of $\mathbf{PRG}$ is output in the clear.

6 Robust and Collision-Resistant PRFs

We now turn to the problem of constructing robust and collision-resistant PRFs. For practical purposes, it is a reasonable assumption that a keyed hash function acts as a $\mathbf{PRF}$

when used with a random and unknown key, and is also an unkeyed collision-resistant function.[3] Hence, a practical hash function can be used to instantiate the transformations in the previous section.

We ask if collision-resistant PRFs can be based on simpler assumptions in the standard model. One method to immediately obtain collision-resistant PRFs would be to use a *combiners*. Roughly speaking, a hash function combiner is a transform that takes two (ore more) hash functions as input and outputs a hash function that is secure if either hash function is secure. For example, concatenation is a combiner for collision resistance. Fischlin et al. [FLP14] give a multi-property combiner for hash function that is above to *simultaneously* preserve multiple security properties of input hash functions, including collision-resistance and pseudorandomness. This raises an alternative route to obtain collision-resistant/robust PRFs based on multi-property hash combiners. The construction of Fischlin et al. [FLP14], however, considers keyed collision resistance which is not sufficient for our purposes. Furthermore, a modification to unkeyed hash functions results in key dependency issues (somewhat similarly to the ABN transform) which then prevents a security proof.

$\overline{\mathbf{Gen}}(1^\lambda)$:	$\overline{\mathbf{PRF}}(K, M)$:
$K \twoheadleftarrow \{0,1\}^n$	$(K_1 \| K_2) \leftarrow \mathbf{PRG}(K)$
return K	$C_1 \leftarrow \mathbf{PRP}(K_1, M)$
	$C_2 \leftarrow \mathbf{PRF}(K_2, C_1)$
	return $(C_1 \| C_2)$

Figure 10: Collision-resistant **PRF** from a key-injective **PRF**. Keys are derived via a right-injective length-doubling **PRG**.

Our first result is a simple transform that converts any CROB-secure **PRF** into a fully collision-resistant **PRF**. In this transform, which is shown in Figure 10, we use a length-doubling **PRG** that is collision resistant on the right half of its output. We expand a key K to $(K_1||K_2)$ via a **PRG**, use K_2 in a *key-injective* **PRF** and K_1 in a pseudorandom *permutation* to guarantee collision resistance over both keys and inputs. *Key-injective* **PRF** [CMR98, Fis99] is a weakening of FROB where it is required that $M_1 = M_2$, i.e., it should be infeasible to find $K_1 \neq K_2$ such that $\mathbf{PRF}(K_1, M) = \mathbf{PRF}(K_2, M)$. We will also use a pseudorandom permutation **PRP** to ensure injectivity over messages.

Proposition 2. *The PRF construction in Figure 10 is collision-resistant (and in particular* CROB*) if the underlying* **PRF** *is key-injective and the* **PRG** *is right collision-resistant. Furthermore, the construction is* **PRF** *secure if the* **PRG**, **PRF**, *and* **PRP** *are secure.*

Proof. We first prove collision resistance. Suppose an adversary outputs $(K, M) \neq (K', M')$ such that $\overline{\mathbf{PRF}}(K, M) = \overline{\mathbf{PRF}}(K', M')$. Let $(K_1, K_2) \leftarrow \mathbf{PRG}(K)$ and $(K_1', K_2') \leftarrow \mathbf{PRG}(K')$. Then by construction:

$$\mathbf{PRF}(K_2, C) = \mathbf{PRF}(K_2', C) \quad \text{where} \quad C = \mathbf{PRP}(K_1, M) = \mathbf{PRP}(K_1', M')$$

This means that the adversary breaks the assumed key-injectivity property of the **PRF** unless $K_2 = K_2'$ (note that the **PRF** is run on the same input). But $K_2 = K_2'$ implies that we also have $K = K'$ as otherwise the adversary would break the right collision-resistance property of the **PRG**. This however means that $K_1 = K_1'$. Now since **PRP** is a permutation over this key, collisions can only occur if $M = M'$. This, however, contradicts the assumption that $(K, M) \neq (K', M')$.

The proof of PRF security is standard and proceeds as follows.

G_1 : This is the PRF experiment with $b = 0$, where the outputs are computed using the $\overline{\mathbf{PRF}}$.

G_2 : In this game, instead of outputs of **PRG** we use random and independent K_1 and K_2. The distance to the previous game can be bounded via the security of **PRG**. This step decouples the two keys.

[3] And indeed the random oracle meets this simultaneous security requirement.

G_3 : In this game we replace the outputs of the **PRF** with random strings. The distance to the previous game can be bounded via the PRF security of the **PRF**.

G_4 : In this game we replace the outputs of the **PRP** with random strings. The distance to the previous game can be bounded via the security of the **PRP**. This game corresponds to PRF experiment with $b = 1$.

Therefore, for any $\mathcal{A}$ there are $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ such that

$$\mathbf{Adv}^{\mathrm{prf}}_{\overline{\mathbf{PRF}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathrm{prg}}_{\mathbf{PRG},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\mathrm{prf}}_{\mathbf{PRF},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathrm{prf}}_{\mathbf{PRP},\mathcal{B}_3}(\lambda) \ .$$

□

We now prove that the key-injective **PRF** used above can be based on length-doubling PRGs that achieve collision-resistance both on the left and the right halves of their outputs. That is, when for any efficient $\mathcal{A}$ the probability

$$\Pr\left[(K^1, K^2) \twoheadleftarrow \mathcal{A}(1^\lambda); (K_0^i, K_1^i) \leftarrow \mathbf{PRG}(K^i);\ \text{return}\ (K_0^1 = K_0^2 \vee K_1^1 = K_1^2) \wedge K^1 \neq K^2\right]$$

is negligible. We call such a PRG *left-right collision-resistant* (LRCR). The next lemma build on results from [CMR98, Fis99] shows that the GGM construction [GGM86] when instantiated with an LRCR-secure PRG is key-injective. Recall that the GGM construction defines a **PRF** as

$$\mathbf{PRF}(K, [M_0, \ldots, M_n]) \coloneqq \mathbf{PRG}_{M_n}(\mathbf{PRG}_{M_{n-1}}(\ldots \mathbf{PRG}_{M_1}(K)\ldots)) \ ,$$

where M_i denotes the i-th bit of M, $\mathbf{PRG}_0(X)$ the left half of the output of $\mathbf{PRG}(K)$ and $\mathbf{PRG}_1(K)$ its right half. The difference with [CMR98, Fis99] is that we do not rely on a CRS (a.k.a. tribe-key) but rely on the stronger LRCR security of the PRG.

Proposition 3. *The GGM construction when instantiated with a left/right collision-resistant* **PRG** *results in a key-injective pseudorandom function.*

Proof. The pseudorandomness proof is identical to that of the GGM. We prove key-injectivity. Let

$$y_j^i = \mathbf{PRG}_{M_i}(\mathbf{PRG}_{M_{i-1}}(\ldots \mathbf{PRG}_{M_1}(K_j)\ldots))$$

be the i-th intermediate value for key j. Suppose an adversary finds $(K_1, K_2, M = [M_1, \ldots, M_n])$ with $K_1 \neq K_2$ such that $y_1^n = y_2^n$. Now either $y_1^{n-1} \neq y_2^{n-1}$ or $y_1^{n-1} = y_2^{n-1}$. In the first case a collision is found and we are done. In the second case we look at y_1^{n-2} and y_2^{n-2} and so on. If we reach y_1^1 and y_2^1 and a collision is yet to be found then, since $K_1 \neq K_2$, this is the collision for the PRG. □

Finally, we show that left/right collision-resistant **PRG**s can be built in the standard model (without the use of ROs). Consider the function $G : \mathbb{Z}_p^3 \longrightarrow \mathbb{G}^6$ for a group $\mathbb{G}$ of order p generated by g [BCP02]:

$$G(x_1, x_2, x_3) := (g^{x_1}, g^{x_1 x_2}, g^{x_2 x_3}, g^{x_2}, g^{x_1 x_3}, g^{x_3}) \ .$$

We start by observing that this function is indeed injective on its left and right halves of output. Suppose there exists $(x_1, x_2, x_3) \neq (y_1, y_2, y_3)$ such that $(g^{x_1}, g^{x_1 x_2}, g^{x_2 x_3}) = (g^{y_1}, g^{y_1 y_2}, g^{y_2 y_3})$. Then by comparing the first elements, we must have $x_1 = y_1$, which in conjunction with the equality of second components implies $x_2 = y_2$. This together with the equality of third components implies $x_3 = y_3$. Injectivity for the right half of the outputs is shown similarly.

The outputs of G when run on random inputs are indistinguishable from a random element of $\mathbb{G}^6$ under the DDH assumption. To see this, we start with $(g^{x_1}, g^{x_1 x_2}, g^{x_2 x_3}, g^{x_2}, g^{x_1 x_3}, g^{x_3})$ and replace $g^{x_1 x_2}$ with g^{z_1} using DDH applied to $(g^{x_1}, g^{x_2}, g^{x_1 x_2})$ and

$\underline{\overline{\mathbf{PRG}}(s)}$:
$(a_0, b_0, a_1, b_1, x) \leftarrow \mathbf{PRG}_0(s)$
if $(a_0 = 0 \vee a_1 = 0)$ then return $\perp$
$(x_0, x_1) \leftarrow G(x)$
$s_0 \leftarrow \mathbf{H}((a_0, b_0)||\pi((a_0, b_0), x_0))$; $s_1 \leftarrow \mathbf{H}((a_1, b_1)||\pi((a_1, b_1), x_1))$
return $s_0||s_1$

Figure 11: A length-doubling left/right collision resistant $\overline{\mathbf{PRG}} : \{0,1\}^n \to \{0,1\}^{2n}$, based on a regular collision-resistant hash $\mathbf{H} : \{0,1\}^{3\cdot 3(n+l)} \to \{0,1\}^n$, a pairwise-independent permutation $\pi : \{0,1\}^{3(n+l)} \to \{0,1\}^{3(n+l)}$, a LRCR-secure PRG $G : \{0,1\}^{3n} \to \{0,1\}^{2\cdot 3(n+l)}$ and a $\mathbf{PRG}_0 : \{0,1\}^n \to \{0,1\}^{4\cdot 3(n+l)+3n}$ right injective over the last $3n$ bits.

generating an x_3 to simulate the remaining elements. Next we replace $g^{x_2 x_3}$ with g^{z_2} via DDH applied to $(g^{x_2}, g^{x_3}, g^{x_2 x_3})$ and generate an x_1 to simulate the remaining elements. We finally replace $g^{x_1 x_3}$ with g^{z_2} using DDH.

The outputs of G, however, are not GGM-friendly as they lie in $\mathbb{G}$ which may be encoded as strings that are longer than $2 \cdot |(x_1, x_2, x_3)|$. Furthermore these outputs are not uniformly distributed. Rather the outputs are indistinguishable from some distribution $\mathcal{D} \times \mathcal{D}$ on $\{0,1\}^{3(n+l)} \times \{0,1\}^{3(n+l)}$, where l is the length of the bits needed to represent the group elements.

Following [Dod05, DS05], we address these issues by applying in parallel a *collision-resistant extractor* to the outputs of G in two steps: (1) we apply a pairwise-independent permutation to bring the output distribution close to uniform; (2) we then use a collision-resistant, regular hash function to compress the result down to n bits without losing uniformity of the outputs. A pairwise-independent permutation π can be instantiated as

$$\pi((a, b), X) := a \cdot X + b \quad \text{where} \quad a, b \twoheadleftarrow \{0,1\}^{3(n+l)}, a \neq 0$$

(where the $\cdot$ and $+$ operations are defined over an extension field). A function $\mathbf{H} : D \longrightarrow R$ is regular if its outputs are uniformly distributed over R for uniform inputs in D, equivalently for all $y \in R$ it holds $|\mathbf{H}^{-1}(y)| = |D|/|R|$. Regular, collision-resistant hash functions can be obtained from claw-free permutations [CMR98].

We define the required LRCR-secure and GGM-friendly $\overline{\mathbf{PRG}}$ in Figure 11, where $\mathbf{PRG}_0$ is a right-injective PRG and $G(x) = (x_0, x_1)$ is a LRCR-secure PRG (for example the one above obtained from DDH).

Theorem 6. *The $\overline{\mathbf{PRG}}$ in Figure 11 is* LRCR*-secure and a secure PRG if $\mathbf{PRG}_0$ is secure, G is secure with respect to the output distribution of $\mathcal{D}$ with min-entropy at least $3n$, $\mathbf{H}$ is a regular and collision-resistant hash function, and π is a pairwise-independent permutation.*

Proof. We first show $\overline{\mathbf{PRG}}$ is LRCR secure. Let $\overline{\mathbf{PRG}}(s) = s_0||s_1$. Suppose that an adversary outputs $s \neq s'$ such that $s_d = s'_d$ for some $d = 0, 1$. Let $d = 0$. So either the adversary can be used to break the collision resistance of $\mathbf{H}$ or $((a_0, b_0), \pi((a_0, b_0), x_0)) = ((a'_0, b'_0), \pi((a'_0, b'_0), x'_0))$. Therefore $(a_0, b_0) = (a'_0, b'_0)$ and $\pi((a_0, b_0), x_0) = \pi((a'_0, b'_0), x'_0)$. Since $\pi((a_0, b_0), \cdot)$ is a permutation we must have that $x_0 = x'_0$. This contradicts the LRCR security of G unless $x = x'$. This in turns means that a collision on the right side (corresponding to x) of the output of $\mathbf{PRG}_0$ is found unless $s = s'$. The case $d = 1$ is dealt with similarly. This concludes the proof of LRCR security.

We now turn to the pseudorandomness of the $\overline{\mathbf{PRG}}$. If $\mathbf{H}$ is regular, its outputs are uniform when fed with uniform inputs. Hence, we show the outputs of π are uniform. We prove this by first replacing the key (a_0, b_0) (and respectively, (a_1, b_1)) of π with truly random keys using the security of $\mathbf{PRG}_0$. We then replace x_0 (and respectively x_1) with random strings sampled according to the distribution $\mathcal{D}$ on $\{0,1\}^{3(n+l)}$. This follows

from the security of G. Note that the distribution $\mathcal{D}$ has min-entropy at least $3n$ by the injectivity of group exponentiation.

Dodis and Smith [DS05, Prop. 11] show a left-over hash lemma for composition with functions: for $\mathbf{H}$ a regular collision-resistant hash function with output length $\ell \leq t - 2\log(\frac{1}{\epsilon})$, where t is the min-entropy of the input source $\mathcal{D}$ to a pairwise-independent permutation π, the statistical distance between $\mathbf{H}(\pi(\mathcal{D}))$ and $\mathbf{H}(\mathcal{U})$ is at most ϵ. Applying this result to our setting with $\epsilon := 2^{-n}$, we get that setting $\ell \leq 3n - 2\log(\frac{1}{\epsilon}) = n$ would result in uniform outputs. This matches the output length of $\mathbf{H}$ and concludes the proof of security of $\overline{\mathbf{PRG}}$. □

Remark. We note that LRCR security is also necessary for building key-injective PRFs as any key-injective PRF would immediately give rise to an LRCR-secure PRG by setting the seed to the PRF key and the outputs of the PRG to those of the PRF evaluated at two points. We leave the possibility of basing LRCR-secure PRGs on generic assumptions, such as one-way functions/permutations or collision-resistance, to future work. We, however, observe that collision resistant does not seem to be a necessary condition as the left or right halves of the PRG do not need to be compressing.

Acknowledgments. Farshim was supported by grant ANR-14-CE28-0003 (Project EnBid). Orlandi was supported by the Danish Independent Research Council and COST Action IC1306. Roşie was supported by European Union's Horizon 2020 research and innovation programme under grant agreement No H2020-MSCA-ITN-2014-643161 ECRYPT-NET.

References

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Heidelberg, February 2010.

[BCP02] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 321–336. Springer, Heidelberg, April / May 2002.

[Bih94] Eli Biham. New types of cryptoanalytic attacks using related keys (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 398–409. Springer, Heidelberg, May 1994.

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

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 320–335. Springer, Heidelberg, August 2002.

[CMR98] Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *30th ACM STOC*, pages 131–140. ACM Press, May 1998.

[CO15] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 40–58. Springer, Heidelberg, August 2015.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 789–800. ACM Press, November 2013.

[Dod05] Yevgeniy Dodis. On extractors, error-correction and hiding all partial information. In *IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security, 2005.*, pages 74–79. IEEE, 2005.

[DS05] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 654–663. ACM Press, May 2005.

[Fis99] Marc Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 432–445. Springer, Heidelberg, May 1999.

[FLP14] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust multi-property combiners for hash functions. *Journal of Cryptology*, 27(3):397–428, July 2014.

[FLPQ13] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 352–368. Springer, Heidelberg, February / March 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 553–571. Springer, Heidelberg, August 2007.

[HKR15] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 15–44. Springer, Heidelberg, April 2015.

[Lam16] Mikkel Lambæk. Breaking and fixing private set intersection protocols. Cryptology ePrint Archive, Report 2016/665, 2016. http://eprint.iacr.org/2016/665.

[Moh10] Payman Mohassel. A closer look at anonymity and robustness in encryption schemes. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 501–518. Springer, Heidelberg, December 2010.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 02*, pages 98–107. ACM Press, November 2002.

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.

Appendices

A Relations among Notions of Robustness

For completeness and comparison with prior work we introduce symmetric analogues of mixed-robustness (XROB) and keyless-robustness (KROB) for AE schemes in Figure 12 below. This follow the definitions of [FLPQ13] in the context of public-key encryption.

$\underline{\text{XROB}^{\mathcal{A}}_{\mathbf{AE}}(\lambda)}$:	$\underline{\text{KROB}^{\mathcal{A}}_{\mathbf{AE}}(\lambda)}$:
$(M_1, K_1, R_1, C_2, K_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$	$(M_1, K_1, R_1, M_2, K_2, R_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$
$C_1 \leftarrow \mathbf{Enc}(K_1, M_1; R_1)$	$C_1 \leftarrow \mathbf{Enc}(K_1, M_1; R_1)$
$M_2 \leftarrow \mathbf{Dec}(K_2, C_2)$	$C_2 \leftarrow \mathbf{Enc}(K_2, M_2; R_2)$
if $M_1 \neq \bot \wedge M_2 \neq \bot \wedge K_1 \neq K_2 \wedge$	if $M_1 \neq \bot \wedge M_2 \neq \bot \wedge K_1 \neq K_2 \wedge$
$\quad C_1 = C_2 \neq \bot$ return 1	$\quad C_1 = C_2 \neq \bot$ return 1
return 0	return 0

Figure 12: Mixed robustness (XROB) and key-less robustness (KROB) for **AE**.

We study relations among notions of robustness for AE schemes below.

Proposition 4. *Let* **AE** *be an encryption scheme.*

1. **AE** *is* FROB *secure if and only if it is* CROB *secure.*
2. *If* **AE** *is* FROB *secure, then it is also* XROB *secure.*
3. *If* **AE** *is* XROB *secure, then it is also* KROB *secure.*
4. *If* **AE** *is* FROB *secure, then it is also* SFROB *secure.*
5. *If* **AE** *is* SFROB *secure, then it is also* SROB *secure.*

Proof. (1) FROB $\iff$ CROB**.** ("$\Leftarrow$") Assume the existence of an adversary that wins the FROB game. Then this adversary also wins the CROB game by querying the FROB winning tuples to the **Dec** oracle. ("$\Rightarrow$") First, from (2) and (3) we have that a FROB scheme is also KROB and XROB. Then, note that a pair of winning tuples for the CROB game can arise in one of three possible ways: (1) Both tuples were added to the list through decryption queries. This directly translates into a winning output for a FROB adversary; (2) Both tuples were added to the list through encryption queries. This translates into a winning output for a KROB adversary; (3) One tuple was added to the list through an encryption query and the other through a decryption query. This translates into a winning output for an XROB adversary.

(2) FROB $\implies$ XROB**.** We proceed as in the previous case. We build an adversary $\mathcal{B}$ that wins the FROB game in Figure 13. $\mathcal{B}$ runs $\mathcal{A}$ to obtain an XROB winning tuple $(M_1, K_1, R_1, C_2, K_2)$ that fulfills the XROB constraints: $C_1 = \mathbf{Enc}(K_1, M_1; R_1) = C_2 \wedge \mathbf{Dec}(K_2, C_2) \neq \perp$. Then $\mathcal{B}$ computes $C_1 \leftarrow \mathbf{Enc}(K_1, M_1; R_1)$ and uses the tuple (C_1, K_1, K_2) to win the FROB game: both tuples $\mathbf{Dec}(K_1, C_1)$ and $\mathbf{Dec}(K_2, C_2)$ will return $\neq \perp$, given that C is a valid ciphertext. Therefore $\mathbf{Adv}^{\text{frob}}_{\mathbf{AE},\mathcal{B}}(\lambda) = \mathbf{Adv}^{\text{xrob}}_{\mathbf{AE},\mathcal{A}}(\lambda)$.

Algorithm $\mathcal{B}(1^\lambda)$:
1. $(M_1, K_1, R_1, C_2, K_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$
2. $C_1 \leftarrow \mathbf{Enc}(K_1, M_1; R_1)$
3. return (C_1, K_1, K_2)

Figure 13: FROB $\Rightarrow$ XROB.

(3) XROB $\implies$ KROB**.** The intuition behind the proof is that an adversary breaking KROB can be used to construct an XROB winning tuple simply by encrypting part of the output obtained from the KROB adversary. The reduction is shown in Figure 14. Let $\mathcal{A}$ be an adversary having a non-negligible advantage against the KROB game. We build an adversary $\mathcal{B}$ that wins the XROB game as follows: $\mathcal{B}$ begins by running $\mathcal{A}$ to obtain a KROB winning tuple $(M_1, K_1, R_1, M_2, K_2, R_2)$ that fulfills the KROB constraint: $C_1 \leftarrow \mathbf{Enc}(K_1, M_1; R_1) \wedge C_2 \leftarrow \mathbf{Enc}(K_2, M_2; R_2) \wedge C_1 = C_2$. Next, $\mathcal{B}$ computes $C_2 \leftarrow \mathbf{Enc}(K_2, M_2; R_2)$ and creates the tuple $(M_1, K_1, R_1, C_2, K_2)$ to win the XROB game; we state that $C_1 \leftarrow \mathbf{Enc}(K_1, M_1; R_1) \neq \perp$ because it is part of a KROB tuple while $\mathbf{Dec}(K_2, C_2) \neq \perp$ returns a valid message with non-negligible probability. We conclude that $\mathbf{Adv}^{\text{xrob}}_{\mathbf{AE},\mathcal{B}}(\lambda) = \mathbf{Adv}^{\text{krob}}_{\mathbf{AE},\mathcal{A}}(\lambda)$.

Algorithm $\mathcal{B}(1^\lambda)$:
1. $(M_1, K_1, R_1, M_2, K_2, R_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$
2. $C_2 \leftarrow \mathbf{Enc}(K_2, M_2; R_2)$
3. return $(M_1, K_1, R_1, C_2, K_2)$

Figure 14: XROB $\Rightarrow$ KROB.

(4) FROB $\implies$ SFROB**.** As in the previous cases, we build an adversary $\mathcal{B}$ that wins the FROB game in Figure 15. $\mathcal{B}$ samples K_1, K_2 uniformly at random and runs $\mathcal{A}$ and answers its oracle queries using the keys. When $\mathcal{A}$ returns C; then, $\mathcal{B}$ constructs an FROB winning tuple (C, K_1, K_2) that fulfills the constraints: $M_1 \leftarrow \mathbf{Dec}(K_1, C) \wedge M_2 \leftarrow \mathbf{Dec}(K_2, C) \wedge M_1 \neq \perp \wedge M_2 \neq \perp$. $\mathcal{B}$ simply returns (C, K_1, K_2) to win the FROB game. Therefore $\mathbf{Adv}^{\text{frob}}_{\mathbf{AE},\mathcal{B}}(\lambda) = \mathbf{Adv}^{\text{sfrob}}_{\mathbf{AE},\mathcal{A}}(\lambda)$.

Algorithm $\mathcal{B}(1^\lambda)$:
1. $(K_1, K_2) \twoheadleftarrow \mathbf{Gen}(1^\lambda)$
2. $C \twoheadleftarrow \mathcal{A}^{\text{Enc},\text{Dec}}(1^\lambda, K_2)$
3. return (C_1, K_1, K_2)

Figure 15: FROB $\Rightarrow$ SFROB.

(5) SFROB $\implies$ SROB**.** This follows from a trivial reduction as the games are identical except that an SROB adversary does not get to see K_2.

We define mixed-robustness (XROB) and keyless-robustness (KROB) for MACs in Figure 16 below.

$\underline{\text{XROB}^{\mathcal{A}}_{\mathbf{MAC}}(\lambda)}$:
$(M_1, K_1, R_1, M_2, K_2, T_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$
$T_1 \leftarrow \mathbf{Tag}(K_1, M_1; R_1)$
$b_2 \leftarrow \mathbf{Ver}(K_2, M_2, T_2)$
if $M_1 \neq \perp \wedge M_2 \neq \perp \wedge K_1 \neq K_2 \wedge$
  $b_2 = 1 \wedge T_1 = T_2 \neq \perp$ return 1
return 0

$\underline{\text{KROB}^{\mathcal{A}}_{\mathbf{MAC}}(\lambda)}$:
$(M_1, K_1, R_1, M_2, K_2, R_2) \twoheadleftarrow \mathcal{A}(1^\lambda)$
$T_1 \leftarrow \mathbf{Tag}(K_1, M_1; R_1)$
$T_2 \leftarrow \mathbf{Tag}(K_2, M_2; R_2)$
if $M_1 \neq \perp \wedge M_2 \neq \perp \wedge K_1 \neq K_2 \wedge$
  $T_1 = T_2 \neq \perp$ return 1
return 0

Figure 16: Mixed robustness (XROB) and key-less robustness (KROB) for **MAC**.

Proposition 5. *Let* **MAC** *be a message authentication code.*
1. *A* **MAC** *is* FROB *secure if an only if it is* CROB *secure.*
2. *If* **MAC** *is* FROB *secure, then it is also* XROB *secure.*
3. *If* **MAC** *is* XROB *secure, then it is also* KROB *secure.*
4. *If* **MAC** *is* FROB *secure, then it is also* SFROB *secure.*
5. *If* **MAC** *is* SFROB *secure, then it is also* SROB *secure.*

The proofs are omitted as they are virtually identical to those of Proposition 4.

B Separations among Notions of Robustness

We show separating examples to further clarify the level of robustness offered by various notions. In particular we show that FROB is a strict strengthening of XROB, which itself is a strict strengthening of KROB. We also show SFROB is incomparable to KROB and that SROB is incomparable to XROB. Hence FROB is the only robustness notion that encompasses all these notions.

Consistently with our previous discussions, all our counterexamples scheme will have valid keys and preserve full correctness. Although not formally required, we also give separating examples that do not affect AE security for AE scheme or SUF security for MACs.

Proposition 6. *Suppose there is a scheme* **AE** *which is both* AE *and* FROB *secure.*
1. *There exists a scheme* $\overline{\mathbf{AE}}$ *that is* AE *and* SFROB *secure but not* KROB *secure.*
2. *There exists a scheme* $\overline{\mathbf{AE}}$ *that is* AE *and* KROB *secure but not* XROB *secure.*
3. *There exists a scheme* $\overline{\mathbf{AE}}$ *that is* AE *and* XROB *secure but not* FROB *secure.*

Proof. (1) SFROB $\not\Rightarrow$ KROB**.** Consider scheme $\overline{\mathbf{AE}}$ whose encryption on keys 0^k and 1^k always uses the key 1^k. Decryption also always uses 1^k on these two keys. This scheme is not KROB as ciphertexts for the same plaintext and randomness on these two keys collide. But it remains SUF and SFROB as 0^k or 1^k will be used in these games with negligible probability.

(2) KROB $\not\Rightarrow$ XROB**.** Let **PRF** be a PRF with outputs in $\{0,1\}^k \setminus \{0^k, 1^k\}$. Such PRFs can be easily constructed by modifying outputs whenever they happen to be 0^k or 1^k to, say, 01^{k-1}. Consider scheme $\overline{\mathbf{AE}}$ that operates as follows. The modified encryption algorithm first expands K via a right-injective PRG to (K_f, K_e) and then computes

$$\overline{\mathbf{Enc}}(K, M; R) \coloneqq B || \mathbf{Enc}(K_e, M; R) \ ,$$

where $B := K$ if $K \in \{0^k, 1^k\}$ and $B := \mathbf{PRF}(K_f, \mathbf{Enc}(K_e, M; R))$ otherwise. The modified decryption returns message 0 on the special keys whenever the output is $\perp$. This scheme is not XROB as any correctly generated ciphertext in the range of key 0^k will also decrypt to a valid message under key 1^k.

But the scheme is still KROB. Indeed, since encryption has disjoint ranges for keys 0^k and 1^k and $K \neq 0^k, 1^k$ (due to B and range of PRF), a KROB attack must exploit two keys that are both not 0^k or 1^k. However such an attack would translate to one on the base scheme as **PRG** is right injective. This scheme is also still AE secure, as it is essentially an "Encrypt-then-MAC" construction. (Note that we do not need to rely on the **PRG** for a separating example that does not need to preserve AE security.)

(3) XROB $\not\Rightarrow$ FROB**.** Let **PRF** be a PRF with outputs in $\{0,1\}^k \setminus \{0^k\}$ as above. Consider scheme $\overline{\mathbf{AE}}$ that operates as follows. The modified encryption algorithm first expands K via a right-injective PRG to (K_f, K_e) and then

$$\overline{\mathbf{Enc}}(K, M; R) \coloneqq B || \mathbf{Enc}(K_e, M; R) \ ,$$

where $B := \mathbf{PRF}(K_f, \mathbf{Enc}(K_e, M; R))$ otherwise. The modified decryption on keys 0^k and 1^k, decrypts the second part of the ciphertext, and if it is $\perp$ and $B = 0^k$, it returns 0. This scheme is not FROB as any ciphertext beginning with $B = 0^k$ will decrypt to a valid message under keys 0^k and 1^k.

This scheme, however, is still XROB secure. Encryption under any key will not result in a ciphertext that begins with $B = 0^k$. Hence an XROB must have $B \neq 0^k$ in its ciphertext. Such an XROB attack must have the second components of the ciphertext matching too. This then translates to an XROB on the base scheme **AE** as modified decryption is equivalent to the original one when $B \neq 0^k$. This scheme is also AE secure, as it is an Encrypt-then-MAC construction and a random key will hit 0^k and 1^k with negligible probability.

Note that only valid keys are used in the above attacks, and that the modified schemes all remain perfectly correct. □

We do not prove that there exists an **AE** scheme that is XROB secure but not SROB secure as SROB security is already implied by AE security. Without AE security, counterexamples similar to those given above exist.

We give similar separations for MACs. We recall that (by Theorem 2) a KROB secure MAC is SROB secure.

Proposition 7. *Let* **MAC** *be a MAC that is* SUF *and* FROB *secure.*

1. *There exists a scheme* $\overline{\mathbf{MAC}}$ *that is* SUF *and* SFROB *secure but not* KROB *secure.*
2. *There exists a scheme* $\overline{\mathbf{MAC}}$ *that is* SUF *and* KROB *secure but not* XROB *secure.*
3. *There exists a scheme* $\overline{\mathbf{MAC}}$ *that is* SUF *and* XROB *secure but not* FROB *secure.*

Proof. (1) SFROB $\not\Rightarrow$ KROB**.** Consider scheme $\overline{\mathbf{MAC}}$ whose verification always passes on keys 0^k and 1^k and whose tag generation on these keys always returns 0. This scheme is trivially not KROB secure as tags on 0^k and 1^k always collide. But it remains SUF and SFROB as a random key will be 0^k or 1^k with only a negligible probability.

(2) KROB $\not\Rightarrow$ XROB**.** Consider scheme $\overline{\mathbf{MAC}}$ that operates as

$$\overline{\mathbf{Tag}}(K, M; R) \coloneqq B||\mathbf{Tag}(K, M; R) \ ,$$

where $B := 00$ if $K = 0^k$, $B := 01$ if $K = 1^k$, and $B := 10$ otherwise. The modified verification algorithm first checks that $B = 10$ if $K \notin \{0^k, 1^k\}$ and that $B = 00$ if $K = 0^k$, and the verifies the rest of the tag. However, when $K = 1^k$ it accepts any tag value that starts with $B = 01$ or $B = 00$.

This scheme is not XROB. An adversary can win this game by outputting

$$(M_1, K_1, R_1, M_2, K_2, T_2) := (0, 0^k, 0^{|R_1|}, 1^k, 00|\, T_1) \ ,$$

where $T_1 := \mathbf{Tag}(K_1, M_1; R_1)$. Indeed, $00|\, T_1$ will verify under $K_1 = 0^k$ and $M_1 = 0$ as it is a correctly generated tag value for these inputs. It will also verify under $K_2 = 1^k$ as the leading bits are ignored for this key upon verification.

This scheme is still KROB. To see this note that the scheme has disjoint tag ranges for keys 0^k and 1^k and that KROB security at all other keys are not affected. The SUF security of the scheme is also not affected as the scheme checks the attached bits upon verification for random keys (keys $0^k, 1^k$ are used in the SUF game with only a negligible probability).

(3) XROB $\not\Rightarrow$ FROB**.** Consider scheme $\overline{\mathbf{MAC}}$ that operates as

$$\overline{\mathbf{Tag}}(K, M; R) \coloneqq 0||\mathbf{Tag}(K, M; R) \ .$$

The modified verification algorithm passes if $B = 0$ and the rest of the tag verifies or if $B = 1$. This scheme is clearly not FROB as any tag starting with a 1 will always verify

under any key. However, The modified scheme is still XROB. Indeed a tag that verifies under two keys, one of which is in the range of $\overline{\mathbf{Tag}}$, must begin with 0. For tag values that start with a 0, the scheme is FROB and hence also XROB. The modified scheme is also SUF as keys 0^k and 1^k will be used in SUF with negligible probability.

Note that only valid keys are used in the above counterexamples, and the modified schemes are also perfectly correct. □

Using similar ideas, it can be also shown that there is a MAC which is SUF and XROB secure but not SFROB secure. Without SUF security a counterexample between XROB and SROB can be also given.