# On the Related-Key Attack Security of Authenticated Encryption Schemes

Sebastian Faust[1], Juliane Krämer[2], Maximilian Orlt[1], and Patrick Struck[2]

[1] Technische Universität Darmstadt, Germany
{sebastian.faust,maximilian.orlt}@tu-darmstadt.de
[2] Universität Regensburg, Germany
{juliane.kraemer,patrick.struck}@ur.de

**Abstract.** Related-key attacks (RKA) are powerful cryptanalytic attacks, where the adversary can tamper with the secret key of a cryptographic scheme. Since their invention, RKA security has been an important design goal in cryptography, and various works aim at designing cryptographic primitives that offer protection against related-key attacks. At EUROCRYPT'03, Bellare and Kohno introduced the first formal treatment of related-key attacks focusing on pseudorandom functions and permutations. This was later extended to cover other primitives such as signatures and public key encryption schemes, but until now, a comprehensive formal security analysis of authenticated encryption schemes with associated data (AEAD) in the RKA setting has been missing. The main contribution of our work is to close this gap for the relevant class of nonce-based AEAD schemes.
To this end, we revisit the common approach to construct AEAD from encryption and message authentication. We extend the traditional security notion of AEAD to the RKA setting and consider an adversary that can tamper with the key $K_e$ and $K_m$ of the underlying encryption and MAC, respectively. We study two security models. In our weak setting, we require that tampering will change both $K_e$ and $K_m$, while in our strong setting, tampering can be arbitrary, i.e., only one key might be affected. We then study the security of the standard composition methods by analysing the nonce-based AEAD schemes N1 (Encrypt-and-MAC), N2 (Encrypt-then-MAC), and N3 (MAC-then-Encrypt) due to Namprempre, Rogaway, and Shrimpton (EUROCRYPT'03). We show that these schemes are weakly RKA secure, while they can be broken under a strong related-key attack. Finally, based on the N3 construction, we give a novel AEAD scheme that achieves our stronger notion.

## 1 Introduction

The security of cryptographic schemes fundamentally relies on the secrecy of its keys. In particular, the secret key used by cryptographic algorithms must neither be revealed to the adversary nor must the adversary be able to change it. Unfortunately, countless advanced cryptanalytical attacks illustrate that the assumption on the secrecy of a key ceases to hold in practice. Prominent examples include side-channel attacks such as power analysis or timing attacks that partially reveal the secret key [26, 25]; or tampering and fault attacks [17], where the adversary can change the secret key and observe the effect of this change via the outputs. The latter type of attack often is referred to as a *related-key attack (RKA)*, and has been intensively studied by the research community over the last years [8, 6, 1, 5, 4, 12, 36, 28, 20, 16, 27, 19, 22, 13, 23]. But related keys may not only appear when the adversary actively tampers with the key. Another important setting where we have to deal with related keys is key updates. In this setting related-key cryptanalysis may exploit the relation of keys caused by bad key updates [14, 24, 16, 15]. Another scenario are devices with related keys. As a simple example consider a manufacturer that has some master key $K$. Rather than generating a fresh key for each device, it derives the key from the master key and some device id – for instance XORing the two.

The first work that provided a formal model for related-key attacks is the seminal work of Bellare and Kohno [8]. In this model, the related-key attacker can specify a related-key-deriving (RKD) function $\varphi$ (from some set $\Phi$) together with each black-box query to the cryptographic primitive, and observe the input/output behaviour for the primitive under the related key $\varphi(K)$. For instance, consider a PRF $\mathtt{F}(K, \cdot)$, that the adversary can query on some input $X$. As a result of a related-key attack the adversary receives $\mathtt{F}(\varphi(K), X)$, where $\varphi$ is the RKD function. Starting with [8], several works extend the notion of RKA security to a wide range of cryptographic primitives. This includes pseudorandom functions [6, 1], pseudorandom permutations [5], encryption schemes [4], and MACs [12, 36].

Somewhat surprisingly, RKA security has not been considered for the important case of authenticated encryption schemes with associated data[3] (AEAD). AEAD is a fundamental cryptographic primitive used, e.g., to secure communication in the Internet and is therefore ubiquitously deployed, especially in TLS 1.3 [31]. Lately, AEAD has received a lot of attention, for instance through the CAESAR competition [11] and the ongoing NIST standardization process on lightweight cryptography [30]. An important type of AEAD schemes, and simultaneously the focus of the NIST standardization process [30], are so-called nonce-based schemes [32]. These schemes have the advantage that they are deterministic, and hence their security does not rely on good quality randomness during encryption. Instead, they use nonces (e.g., a simple counter) and require that these nonces are never repeated to guarantee security [32].

### 1.1 Our Contribution

The main contribution of our work is to extend the notion of RKA security to nonce-based AEAD schemes. We study the common generic composition paradigms to construct AEAD from encryption schemes and MACs, and explore if RKA security of the underlying primitives carries over to the AEAD scheme.

[3] Associated data corresponds to header information that has to be authenticated but does not need to be confidential.

More concretely, let $K_e$ and $K_m$ be the keys of the encryption and MAC, respectively. Assuming that the encryption scheme is secure against the class $\Phi_e$ of related-key deriving functions and the MAC is secure against $\Phi_m$, then we ask if the AEAD scheme is secure with respect to related-key derivation functions from the Cartesian product $\Phi_e \times \Phi_m$.[4] In particular, we show that under certain restrictions of $\Phi_e \times \Phi_m$ the schemes N1, N2, and N3 by Namprempre et al. [29], falling into the composition paradigms E&M, EtM, and MtE, respectively, are secure under related-key attacks. By giving concrete attacks against all schemes in case the restrictions are dropped, we show further that these restrictions are necessary. Finally, on the positive side, we give a new construction for AEAD that is secure for the general case of functions from $\Phi_e \times \Phi_m$, i.e., without the aforementioned restrictions. We provide more details on our contribution below.

**RKA Security Notions for Nonce-based AEAD Schemes.** We give two RKA security notions s-RKA-AE and RKA-AE for nonce-based AEAD schemes. In our weaker notion (RKA-AE), we assume that the key is updated such that each underlying primitive never uses the same key twice.[5] This is modelled by imposing an additional restriction on the adversary, where the adversary is not allowed to make queries with RKD functions that would result in keys that have already appeared during earlier RKA queries. More precisely, let $K_e^i$ and $K_m^i$ the result of the $i$-th RKA query. We require that for all $i, j$, we have $K_e^i = K_e^j$ if and only if $K_m^i = K_m^j$. In our stronger notion (s-RKA-AE), the above restriction is not imposed on the adversary, i.e., it is allowed to make queries $i, j$ such that $K_e^i = K_e^j$ and $K_m^i \neq K_m^j$. Note that any adversary can trivially make such queries by repeating the RKD function for key $K_e$ while using two different RKD functions for $K_m$. One may object that our weaker security notion looks rather artificial for modelling tampering attacks. We believe however that it is interesting to study for what key relations state-of-the-art AEAD constructions that are widely deployed remain secure under related-key attacks. Moreover, we emphasize another setting where such weak key relations may occur naturally – so-called bad key updates. In this setting the RKA adversary may observe ciphertexts for different related keys, where the relation stems from the key updates described by the RKD functions. Since the users update the keys, the relation between the keys is in fact not chosen by the adversary. Hence, the weaker notion guarantees security if the users ensure that, after each update, both keys $K_e$ and $K_m$ are fresh. In contrast, the stronger notion guarantees security even in the case when the users might only update one of the keys. Further details on these two notions are given in Section 3.

**RKA Security of the N1, N2, and N3 Construction.** We study the security of the N1, N2, and N3 constructions for nonce-based AEAD schemes [29],

[4] A similar question using the Cartesian product of the related-key deriving functions from the underlying primitives is answered in [5] for Feistel constructions.

[5] Note that the adversary can still ask for several encryptions under each key.

which follow the Encrypt-and-MAC (E&M), Encrypt-then-MAC (EtM), and MAC-then-Encrypt (MtE) paradigm [9], respectively. These constructions build a nonce-based AEAD scheme from a nonce-based encryption scheme and a MAC. We show that all schemes achieve our weaker security notion, i.e., when it is ensured that both keys are updated properly. The overall proof approach is similar to the classical setting. The challenge lies in the analysis that all queries by the reduction are permitted due to the related keys. Regarding our stronger security notion, we show that all schemes have limitations. We show that N1 and N2 are insecure, irrespective of the underlying primitive, by giving concrete attacks. For N3, we show that the security crucially depends on the underlying encryption scheme, by giving an attack against any instantiation using a stream cipher. These results appear in Section 4.

**RKA-secure AEAD Scheme.** Finally, we give a new construction, called N*, of an AEAD scheme which is based on the N3 construction, and follows the MAC-then-Encrypt (MtE) paradigm. The underlying encryption scheme relies on an RKA-secure block cipher and a MAC. The resulting AEAD scheme achieves our stronger security notion s-RKA-AE, in fact, even in the case of nonce misuse. For simplicity we omit details regarding the nonce here, and discuss this setting more detail in Section 3. The construction and the proof is shown in Section 5.

**RKA-secure Encryption and MAC from Pseudorandom Functions.** We show that RKA-secure nonce-based encryption schemes and MACs can be built from RKA-secure pseudorandom functions. Combined with the results for the N1, N2, and N3 constructions, this reduces the task of constructing RKA-secure nonce-based AEAD schemes to the task of constructing RKA-secure pseudorandom functions which is a general goal in the RKA literature. More precisely, we show that the nonce-based encryption scheme and the MAC proposed by Degabriele et al. [18] in the setting of leakage-resilient cryptography achieve RKA security if the underlying pseudorandom function is RKA-secure. This is shown in Section 6.

### 1.2 Related Work

Based on the initial work by Biham [13] and Knudsen [23], the first formalisation of RKA security has been given by Bellare and Kohno [8]. They studied pseudorandom functions as well as pseudorandom permutations and showed an inherit limitation on the set of allowed RKD functions. Bellare and Cash [6] proposed RKA-secure pseudorandom functions based on the DDH assumption, which allowed a large class of RKD functions. Abdalla et al. [1] further increased the allowed class of RKD functions. Several other works study the RKA security for various primitives, e.g., pseudorandom permutations from Feistel networks [5], encryption schemes [4], and MACs [12, 36]. Harris [21], and later Albrecht et al. [3], showed inherent limitations of the Bellare-Kohno formalism by giving a generic attack against encryption schemes if the set of related-key deriving functions can depend on the primitive in question. The practical relevance of the alternative model by Harris has been questioned by Vaudenay [35].

Closer to our setting is the work by Lu et al. [28], who also study RKA security for authenticated encryption schemes. However, instead of nonce-based authenticated encryption schemes, they analyse probabilistic authenticated encryption schemes and only for the specific case of affine functions. Moreover, Han et al. [20] found their proof to be flawed, invalidating the results. To the best of our knowledge, these are the only works that consider RKA security for authenticated encryption schemes.

The practical relevance of RKA security has been shown by a number of works [16, 27, 19, 22] which present attacks against concrete primitives.

## 2 Preliminaries

In Section 2.1 we recall the used notation. The syntax of the cryptographic primitives and existing RKA security notions are given in Section 2.2 and Section 2.3, respectively. Additional background on security notions in the classical setting is given in Appendix A.

### 2.1 Notation

By $\{0,1\}^*$ and $\{0,1\}^x$ we denote the set of bit strings with arbitrary length and length $x$, respectively. We refer to probabilistic polynomial-time algorithms as adversaries if not otherwise specified, and use the code-based game-playing framework by Bellare and Rogaway [10]. For a game $\mathsf{G}$ and adversary $\mathcal{A}$, we write $\mathsf{G}^{\mathcal{A}} \Rightarrow y$ to indicate that the output of the game, when played by $\mathcal{A}$, is $y$. Likewise, $\mathcal{A}^{\mathsf{G}} \Rightarrow y$ indicates that $\mathcal{A}$ outputs $y$ when playing game $\mathsf{G}$. In case $\mathcal{A}$ has access to an oracle $\mathcal{O}$ we write $\mathcal{A}^{\mathcal{O}}$. We only use distinguishing games in which an adversary $\mathcal{A}$ tries to guess a secret bit $b$. The advantage of $\mathcal{A}$ in such a distinguishing game $\mathsf{G}$ is defined as $\mathbf{Adv}^{\mathsf{G}}(\mathcal{A}) := |2\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow \text{true}] - 1|$. Equivalent notions using adversarial advantages are $|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 0 \,|\, b = 1]|$ and $|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 0]|$. For sets $\mathcal{X}$ and $\mathcal{Y}$, the set of all functions mapping from $\mathcal{X}$ to $\mathcal{Y}$ is denoted by $\mathsf{Func}(\mathcal{X},\mathcal{Y})$ and the set of permutations over $\mathcal{X}$ by $\mathsf{Perm}(\mathcal{X})$. We write $\mathsf{Func}(\mathcal{K},\mathcal{X},\mathcal{Y})$ and $\mathsf{Perm}(\mathcal{K},\mathcal{X})$ for keyed functions in $\mathsf{Func}(\mathcal{X},\mathcal{Y})$ and $\mathsf{Perm}(\mathcal{X})$, respectively, where $\mathcal{K}$ denotes the key space. Tables $f$ are initialised with $\perp$ if not mentioned differently. For sets $\mathcal{S}$ and $\mathcal{T}$, we write $\mathcal{S} \xleftarrow{\cup} \mathcal{T}$ instead of $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{T}$. Our main focus lies in the RKA setting and we use the term *classical setting* whenever we refer to the setting which does not consider related-key attacks.

### 2.2 Primitives

A nonce-based authenticated encryption scheme with associated data (AEAD), is a tuple of two deterministic algorithms $(\texttt{Enc}, \texttt{Dec})$. The encryption algorithm

$\mathtt{Enc}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C}$ maps a key $K$, a nonce $N$, associated data $A$, and a message $M$, to a ciphertext $C$. The decryption algorithm $\mathtt{Dec}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ maps a key $K$, a nonce $N$, associated data $A$, and a ciphertext $C$, to either a message or $\perp$ indicating an invalid ciphertext. The sets $\mathcal{K}$, $\mathcal{N}$, $\mathcal{A}$, $\mathcal{M}$, and $\mathcal{C}$, denote the key space, nonce space, associated data space, message space, and ciphertext space, respectively. An AEAD scheme is called *correct* if for any $K \in \mathcal{K}$, any $N \in \mathcal{N}$, any associated data $A \in \mathcal{A}$, and any $M \in \mathcal{M}$, it holds that $\mathtt{Dec}(K, N, A, \mathtt{Enc}(K, N, M, A)) = M$. It is called *tidy* if for any $K \in \mathcal{K}$, any $N \in \mathcal{N}$, any associated data $A \in \mathcal{A}$, any $M \in \mathcal{M}$, and any $C \in \mathcal{C}$ with $\mathtt{Dec}(K, N, A, C) = M$, it holds that $\mathtt{Enc}(K, N, A, M) = C$.

A nonce-based symmetric key encryption is similarly defined. The difference is that neither algorithm permits associated data as an input and only rejects ciphertext, i.e., outputs $\perp$, if computed on values outside the corresponding sets. For both primitives, we let $c$ denote the length of a ciphertext.

A message authentication code (MAC) is a tuple of two deterministic algorithms $(\mathtt{Tag}, \mathtt{Ver})$. The tagging algorithm $\mathtt{Tag}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^t$ maps a key $K$ and message $X$ to a tag $T$. The verification algorithm $\mathtt{Ver}\colon \mathcal{K} \times \mathcal{X} \times \{0,1\}^t \to \{\top, \perp\}$ takes as input a key $K$, a message $M$, and a tag $T$, and outputs either $\top$, indicating a valid tag, or $\perp$, indicating an invalid tag. Correctness requires that $\mathtt{Ver}(K, X, \mathtt{Tag}(K, X)) = \top$, for any $K \in \mathcal{K}$ and $X \in \mathcal{X}$. We denote the length of tags by $t$.

### 2.3 Security Notions against Related-Key Attacks

We recall some of the existing RKA security notions. All notions follow the style introduced by Bellare and Kohno [8]. That is, the set of admissible RKD functions is fixed at the start of the game. All our results, however, also apply to the alternative definition given by Harris [21], where the adversary first picks the set of RKD functions before the concrete scheme (from a family of primitives) is chosen by the game. This prevents an inherent limitation of the Bellare-Kohno formalism as the RKD function can not depend on the primitive.[6]

*$\Phi$-restricted Adversaries.* For RKA security notions, the adversary is typically restricted to a set of functions that it can query to its oracles. This restriction is necessary, as Bellare and Kohno [8] showed that RKA security is unachievable without such restrictions. Let $\mathcal{K}$ be the key space of some primitive, then the set of permitted RKD functions is $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. We call an adversary that only queries functions from the set $\Phi$ to its oracles, a *$\Phi$-restricted adversary*.

*Repeating Queries.* To avoid trivial wins certain queries must be excluded from the security games. In case of a MAC, we must forbid the adversary to query its challenge verification oracle on a tag it obtained from its tagging oracle. To do this, one can either adapt the game by keeping a list of such queries and let the

[6] It is questionable whether RKD functions that depend on the actual primitive are relevant in practice.

verification oracle check for such forbidden queries. The other option, would be to simply exclude adversaries that do such queries in the security definition. For ease of exposition, we use the latter approach.

```
Game rkaSUF
  $b \leftarrow_\$ \{0,1\}$
  $K \leftarrow_\$ \mathcal{K}$
  $b' \leftarrow \mathcal{A}^{\mathsf{Ver},\mathsf{Tag}}()$
  return $(b' = b)$

$\mathsf{Ver}(M, T, \varphi)$
  if $b = 0$
    return $\mathtt{Ver}(\varphi(K), M, T)$
  else
    return $\perp$

$\mathsf{Tag}(M, \varphi)$
  $T \leftarrow \mathtt{Tag}(\varphi(K), M)$
  return $T$
```

Fig. 1: Security game rkaSUF.

*RKA Security for MACs and Pseudorandom Functions/Permutations.* We give the definition of related-key attack security of MACs. Existing notions define it as an unforgeability game where the adversary finally outputs a forgery attempt [12, 36]. In this work, we define unforgeability of a MAC against RKA as a distinguishing game. Here the adversary aims to distinguish whether its challenge oracle implements the real verification algorithm or simply rejects any queried tag.

**Definition 1** (RKA-SUF **Security).** *Let $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC and $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. Let the game* rkaSUF *be defined as in Fig. 1. For a $\Phi$-restricted RKA adversary $\mathcal{A}$, that never forwards a query from its oracle* Tag*, we define its* RKA-SUF *advantage as*

$$\mathbf{Adv}_{\Gamma}^{\mathsf{rkaSUF}}(\mathcal{A}, \Phi) = 2\ \Pr[\mathsf{rkaSUF}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

```
Game rkaPRF, rkaPRP
  $b \leftarrow_\$ \{0,1\}$
  $K \leftarrow_\$ \mathcal{K}$
  $\mathtt{F}' \leftarrow_\$ \mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$
  $\mathtt{P}' \leftarrow_\$ \mathsf{Perm}(\mathcal{K}, \mathcal{X})$
  $b' \leftarrow \mathcal{A}^{\mathsf{F}}()$
  return $(b' = b)$

$\mathsf{F}(X, \varphi)$ in rkaPRF
  if $b = 0$
    $y \leftarrow F(\varphi(K), X)$
  else
    $y \leftarrow_\$ \mathtt{F}'(\varphi(K), X)$
  return $y$

$\mathsf{F}(X, \varphi)$ in rkaPRP
  if $b = 0$
    $y \leftarrow F(\varphi(K), X)$
  else
    $y \leftarrow \mathtt{P}'(\varphi(K), X)$
  return $y$
```

Fig. 2: Security game rkaPRF and rkaPRP.

RKA-Security for pseudorandom functions (PRFs) and pseudorandom permutations (PRPs) have been studied in many works, e.g., [8, 6, 7, 3], and are defined as the advantage in distinguishing the real function/permutation from a random function/permutation when having access to an oracle implementing either of these.

**Definition 2** (RKA-PRF **Security).** *Let* $\mathtt{F}\colon \mathcal{K}\times\mathcal{X}\to\mathcal{Y}$ *and* $\Phi\subset\mathsf{Func}(\mathcal{K},\mathcal{K})$*. Let the game* $\mathsf{rkaPRF}$ *be defined as in Fig. 2. For a* $\Phi$*-restricted RKA adversary adversary* $\mathcal{A}$*, that never repeats a query, we define its* RKA-PRF *advantage as*

$$\mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A},\Phi)=2\Pr[\mathsf{rkaPRF}^{\mathcal{A}}\Rightarrow\mathrm{true}]-1\,.$$

**Definition 3** (RKA-PRP **Security).** *Let* $\mathtt{F}\colon \mathcal{K}\times\mathcal{X}\to\mathcal{X}$ *and* $\Phi\subset\mathsf{Func}(\mathcal{K},\mathcal{K})$*. Let the game* $\mathsf{rkaPRP}$ *be defined as in Fig. 2. For a* $\Phi$*-restricted RKA adversary adversary* $\mathcal{A}$*, that never repeats a query, we define its* RKA-PRP *advantage as*

$$\mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRP}}(\mathcal{A},\Phi)=2\Pr[\mathsf{rkaPRP}^{\mathcal{A}}\Rightarrow\mathrm{true}]-1\,.$$

## 3 RKA Security Notions for Nonce-based AEAD

In this section, we define security for nonce-based encryption schemes and nonce-based AEAD schemes under related-key attacks. RKA security notions for encryption and authenticated encryption schemes have been proposed by Bellare et al. [7] and Lu et al. [28], respectively. However, neither notion considers nonce-based primitives and instead considers the case of probabilistic primitives. Furthermore, both works define indistinguishability in a left-or-right sense, while we follow the stronger IND\$ (indistinguishability from random bits) approach put forth by Rogaway [33]. For this notion, the adversary has to distinguish the encryption of a message from randomly chosen bits. We discuss how the classical property of nonce-respecting adversaries is extended to the RKA setting in Section 3.1 and provide two RKA security notions for nonce-based AEAD schemes in Section 3.2. In Section 3.3, we extend the notion to the nonce misuse case and Section 3.4 provides the RKA security notion for nonce-based encryption schemes.

### 3.1 Nonce Selection

Security notions in the classical setting are often restricted to adversaries which are *nonce-respecting*. These are adversaries that never repeat a nonce across their encryption queries. Hence, security proven against nonce-respecting adversaries guarantees security as long as the encrypting party never repeats a nonce. Below we argue why this adversarial restriction needs to be updated in the RKA setting.

Consider the following scenario. Alice and Bob communicate using an AEAD scheme across several sessions. In each session, Alice will send several encrypted messages to Bob, each time using a fresh nonce implemented as a counter. Instead of exchanging a fresh secret key for each session, they exchange a key for the first session and between two consecutive sessions, they update the key using some update function F. There is no guarantee that Alice does not reuse a nonce in different sessions. In fact, due to using a simple counter which might be reset between the sessions, this is likely to happen. This means that an adversary can observe encryptions using the same nonce under related keys, where the relation is given by the update function F.

The same applies to the scenario where different devices have related keys. Every user would only ensure unique nonces for the own device while there will be colliding nonces across related devices.

If we declare an RKA adversary to be nonce-respecting if and only if it never repeats a nonce, then a proof of security does not tell us anything for the scenarios depicted above. Instead, we define an RKA adversary to be *RKA-nonce-respecting* if it never repeats *the pair* of nonce and RKD function. An interpretation of this definition is that nonce-respecting is defined with respect to individual keys. Since in the classical setting there is only ever one key, this interpretation reflects this.

## 3.2 RKA-Security Notions for AEAD Schemes

We extend security for AEAD schemes to the RKA setting. Instead of the approach used in [28], which defines two separate RKA security notions for confidentiality and authenticity, we follow the unified security notion by Rogaway and Shrimpton [34]. That is, the adversary has access to two oracles Enc and Dec. The goal of the adversary is to distinguish the real world, in which the oracles implement the encryption and decryption algorithm, from the ideal world, where the first oracle returns random bits while the latter rejects any ciphertext. The adversary wins the game if it can distinguish in which world it is. To make our new RKA security notion achievable, we impose standard restrictions on the adversary. That is, first, the adversary is not allowed to forward the response of an encryption query to the decryption query and, second, the adversary must not repeat a query to its encryption oracle.[7] More precisely, we say that an adversary forwards a query from its encryption oracle, if it queries its decryption oracle on a ciphertext $C$ that it has obtained as a response from its encryption oracle, while the other queried values $N, A, \varphi$ are the same for both queries. We call the resulting notion s-RKA-AE, the "s" indicating strong. The reason for that is that we introduce a weaker notion below.

**Definition 4** (s-RKA-AE **Security**). *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an AEAD scheme and $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. Let the game* s-rka-AE *be defined as in Fig. 3. For an RKA-nonce-respecting and $\Phi$-restricted RKA adversary $\mathcal{A}$, that never repeats/forwards a query to/from* Enc*, we define its* RKA-AE *advantage as*

$$\mathbf{Adv}_{\Sigma}^{\mathsf{s\text{-}rka\text{-}AE}}(\mathcal{A}, \Phi) = 2 \Pr[\mathsf{s\text{-}rka\text{-}AE}^{\mathcal{A}} \Rightarrow \mathrm{true}] - 1 \,.$$

[7] The latter restriction can also be handled by letting the encryption oracle return the same response as it did when the query was made the first time. For ease of exposition, we simply forbid such queries to avoid additional bookkeeping in the security games.

```
Game s-rka-AE
    $b \leftarrow_{\$} \{0,1\}$
    $K \leftarrow_{\$} \mathcal{K}$
    $b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$
    return $(b' = b)$

$\mathsf{Enc}(N, A, M, \varphi)$
    if $b = 0$
        $C \leftarrow \mathtt{Enc}(\varphi(K), N, A, M)$
    else
        $C \leftarrow_{\$} \{0,1\}^c$
    return $C$

$\mathsf{Dec}(N, A, C, \varphi)$
    if $b = 0$
        $M \leftarrow \mathtt{Dec}(\varphi(K), N, A, C)$
    else
        $M \leftarrow \perp$
    return $M$
```

Fig. 3: Security game s-rka-AE.

The above definition treats the AEAD scheme to have a single key $K$. Such schemes, however, are often constructed from smaller building blocks which have individual keys. This encompasses the N constructions [29], on which we focus in the next section, but also all other constructions combining an encryption scheme and a MAC into an AEAD schemes. In this case, the set of RKD functions of the AEAD scheme is the Cartesian product of the set of RKD functions for the individual primitives. More precisely, let $E$ and $M$ be the underlying primitives and $\Phi_e$ and $\Phi_m$ be the respective sets of RKD functions. Then for the combined primitive $AE$, the set of RKD functions is $\Phi_{ae} = \Phi_e \times \Phi_m$. Thus the s-RKA-AE security game above allows the adversary to query the encryption oracle on $(N, \varphi_e, \varphi_m) \in \mathcal{N} \times \Phi_e \times \Phi_m$ and later querying it on $(N, \varphi_e, \varphi'_m) \in \mathcal{N} \times \Phi_e \times \Phi_m$, where $\varphi_m \neq \varphi'_m$. This essentially allows the adversary to bypass the nonce-respecting property of the underlying primitive.

Recall the key-update scenario described above. Allowing the adversary to query the same nonce while the queried RKD functions agree in exactly one part, models a scenario in which the key update either does not update one of the keys or later updates a key to a previously used key. We introduce a weaker security notion, in which these queries are forbidden. Security according to this notion then reflects security as long as *the pair of keys* is updated appropriately.

**Definition 5** (RKA-AE **Security).** *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an AEAD scheme and $\Phi = \Phi_e \times \Phi_m \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. Let the game* rka-AE *be defined as in Fig. 4. For an RKA-nonce-respecting and $\Phi$-restricted RKA adversary $\mathcal{A}$, that never repeats/forwards a query to/from* Enc*, we define its* RKA-AE *advantage as*

$$\mathbf{Adv}_{\Sigma}^{\mathsf{rka\text{-}AE}}(\mathcal{A}, \Phi) = 2 \Pr[\mathsf{rka\text{-}AE}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

```
Game rka-AE
  $b \leftarrow_\$ \{0,1\}$
  $(K_e \parallel K_m) \leftarrow_\$ \mathcal{K}$
  $\mathcal{S} \leftarrow \emptyset$
  $b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$
  return $(b' = b)$

Enc$(N, A, M, \varphi_e, \varphi_m)$
  if $\exists \varphi'_e \neq \varphi_e$ st $(N, \varphi'_e, \varphi_m) \in \mathcal{S}$
    return $\bot$
  if $\exists \varphi'_m \neq \varphi_m$ st $(N, \varphi_e, \varphi'_m) \in \mathcal{S}$
    return $\bot$
  $\mathcal{S} \leftarrow_\cup \{(N, \varphi_e, \varphi_m)\}$
  if $b = 0$
    $C \leftarrow \mathtt{Enc}(\varphi_e(K_e) \parallel \varphi_m(K_m), N, A, M)$
  else
    $C \leftarrow_\$ \{0,1\}^c$
  return $C$

Dec$(N, A, C, \varphi_e, \varphi_m)$
  if $\exists \varphi'_e \neq \varphi_e$ st $(N, \varphi'_e, \varphi_m) \in \mathcal{S}$
    return $\bot$
  if $\exists \varphi'_m \neq \varphi_m$ st $(N, \varphi_e, \varphi'_m) \in \mathcal{S}$
    return $\bot$
  $\mathcal{S} \leftarrow_\cup \{(N, \varphi_e, \varphi_m)\}$
  if $b = 0$
    $M \leftarrow \mathtt{Dec}(\varphi_e(K_e) \parallel \varphi_m(K_m), N, A, M)$
  else
    $M \leftarrow \bot$
  return $M$
```

Fig. 4: Security game rka-AE. The set $\mathcal{S}$ is used to detect forbidden queries, that is, queries where the triple of nonce and the two RKD functions differ in exactly one of the functions. Both oracles reject such queries by returning $\bot$.

The weaker security notion bears similarities to split-state non-malleable codes [2]. Here, the secret is encoded in such a way that it is secure against fault attacks as long as the left and right half of the code are tampered independently. In more detail, the decoding of such tampered codes is independent from the original secret and might be invalid. However, if we consider key-related devices or bad-key updates, non-malleable codes are not helpful any more since they are used for faults and not bad randomised keys. The reason for this is that after each key update we need to take care that the resulting key is still valid. Further, we do not want to update the keys independently but simultaneously such that all keys are fresh after the key update. So the requirement to the weaker notion is the opposite of that of non malleable codes. For key updates, it is a reasonable assumption to say that all underlying keys have to be updated for a new session.

## 3.3 RKA-Security against Nonce Misuse

Similar to the classical setting, we extend security to nonce misuse resistance. In this case, the adversary is allowed to repeat nonces to the encryption oracle. Below we define security in this stronger sense for s-RKA-AE security. Note that the game is the same as in Definition 4 (cf. Fig. 3), the sole difference is that the adversary is no longer restricted to be RKA-nonce-respecting.

**Definition 6** (mr-s-RKA-AE **Security).** *Let* $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ *be an AEAD scheme and* $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$*. Let the game* s-rka-AE *be defined as in Fig. 3. For an RKA-respecting and* $\Phi$*-restricted RKA adversary* $\mathcal{A}$*, that never repeats/forwards a query to/from* Enc*, we define its* mr-s-RKA-AE *advantage as*

$$\mathbf{Adv}_{\Sigma}^{\mathsf{mr\text{-}s\text{-}rka\text{-}AE}}(\mathcal{A}, \Phi) = 2 \Pr[\mathsf{s\text{-}rka\text{-}AE}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

In the same way, we can extend RKA-AE security to the nonce misuse scenario. However, we believe this notion not to be meaningful. The RKA-AE security notion already requires that keys are updated properly, i.e., they do not repeat. Since this task is way more complex than ensuring that nonces do not repeat, it seems strange to require this one while simultaneously dropping the simple requirement of unique nonces.

## 3.4 RKA-Security Notions for Encryption

The following definition extends the classical IND-CPA security notion for nonce-based encryption schemes to the RKA setting. The adversary has to tell apart the real encryption oracle from an idealised encryption oracle which returns random bits. The main distinction lies in the nonce selection of the adversary as it is allowed to repeat a nonce if the RKD functions are different.

**Definition 7** (RKA-IND **Security).** *Let* $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ *be an encryption scheme and* $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$*. Let the game* rkaIND *be defined as in Fig. 5. For an RKA-nonce-respecting and* $\Phi$*-restricted RKA adversary* $\mathcal{A}$*, that never repeats a query, we define its* RKA-IND *advantage as*

$$\mathbf{Adv}_{\Sigma}^{\mathsf{rkaIND}}(\mathcal{A}, \Phi) = 2 \Pr[\mathsf{rkaIND}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

```
Game rkaIND                      Enc(N, M, φ)
  $b \leftarrow_{\$} \{0,1\}$            if $b = 0$
  $K \leftarrow_{\$} \mathcal{K}$                $C \leftarrow \mathtt{Enc}(\varphi(K), N, M)$
  $b' \leftarrow \mathcal{A}^{\mathsf{Enc}}()$            else
  return $(b' = b)$                  $C \leftarrow_{\$} \{0,1\}^c$
                                   return $C$
```

Fig. 5: Security game rkaIND.

## 4 RKA Security of the N1, N2, and N3 Constructions

In this section we study the security of the nonce-based AEAD schemes N1, N2, and N3 [29], which fall into the generic composition paradigms Encrypt-and-MAC (E&M), Encrypt-then-MAC (EtM), MAC-then-Encrypt (MtE) [9]. We analyse each scheme with respect to the two security notions RKA-AE and s-RKA-AE defined above. The analysis reveals that all schemes achieve RKA-AE security if the underlying primitives are RKA-secure. Regarding the stronger s-RKA-AE security, the situation is more involved. We show that both N1 and N2 are insecure irrespective of the underlying primitives. For N3, we provide a concrete attack exploiting any instantiation using a stream cipher for the underlying encryption scheme.

Section 4.1 covers the analysis of the N1 construction. The N2 construction is analysed in Section 4.2 while we analyse the N3 construction in Section 4.3.

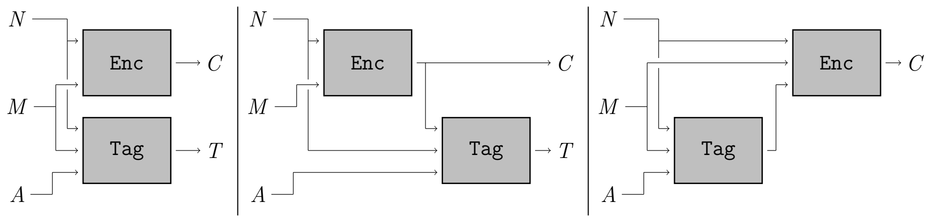

Fig. 6: The AEAD schemes N1 (left), N2 (middle), and N3 (right) [29].

### 4.1 N1 - Instantiation of Encrypt-and-MAC

The N1 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the E&M paradigm.The encryption algorithm is used to encrypt the message as is the MAC to compute a tag for the message. The ciphertext of the AEAD scheme consists of the ciphertext and the tag.

The following theorem shows that the N1 construction achieves RKA-AE security if the underlying primitives are RKA-secure. The overall proof approach is similar to the classical setting but needs some extra treatment when analysing that all queries of the reductions are permitted.

**Theorem 1.** *Let $\Sigma = (\texttt{Enc}, \texttt{Dec})$ be an encryption scheme and $\Gamma = (\texttt{Tag}, \texttt{Ver})$ be a MAC with RKA function sets $\Phi_e$ and $\Phi_m$, respectively. Further, let* $\mathrm{N1}$ *be the AEAD scheme built from $\Sigma$ and $\Gamma$ using the N1 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any RKA-nonce-respecting and $\Phi_{ae}$-restricted RKA adversary $\mathcal{A}$ against* $\mathrm{N1}$*, that never repeats/forwards a query to/from* $\mathsf{Enc}$*, there exists an RKA-nonce-respecting and $\Phi_e$-restricted RKA adversary $\mathcal{A}_{se}$, a $\Phi_m$-restricted RKA adversary $\mathcal{A}_{mac}$, and a $\Phi_m$-restricted RKA adversary $\mathcal{A}_{prf}$ such that*

$$\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}, \Phi_{ae}) \leq \mathbf{Adv}^{\mathsf{rkaIND}}_{\Sigma}(\mathcal{A}_{se}, \Phi_e) + \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m) + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{prf}, \Phi_m)\,.$$

*Proof (Sketch).* The proof consists of multiple game hops. In the first game hop, the decryption oracle is replaced by $\perp$ which is bound by the RKA security of $\Gamma$. In the subsequent game hops, first the tag and then the ciphertext are replaced by random values which is bound by the RKA security of $\mathsf{Tag}$ and $\Sigma$, respectively.

The following theorem shows that the N1 construction does not achieve the stronger s-RKA-AE security. The reason is that a ciphertext is the concatenation of a ciphertext from the underlying encryption scheme and tag from the underlying MAC. By making two queries which solely differ in one of the RKD functions, the adversary can easily distinguishing between the real and the ideal case.

**Theorem 2.** *Let $\Sigma = (\texttt{Enc}, \texttt{Dec})$ be an encryption scheme and $\Gamma = (\texttt{Tag}, \texttt{Ver})$ be a MAC with RKA function sets $\Phi_e$ and $\Phi_m$, respectively. Further, let* $\mathrm{N1}$ *be the AEAD scheme built from $\Sigma$ and $\Gamma$ using the N1 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then* $\mathrm{N1}$ *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and $\Phi_{ae}$-restricted RKA adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}) = 1\,.$$

*Proof.* We construct the following two adversaries $\mathcal{A}_e$ and $\mathcal{A}_m$. Adversary $\mathcal{A}_e$ picks a message $M$, a nonce $N$, and associated data $A$ at random from the respective sets. Furthermore, it picks $\varphi_e \in \Phi_e$ and $\varphi_m, \varphi'_m \in \Phi_m$ such that $\varphi_m \neq \varphi'_m$. Then the adversary makes two queries to its encryption oracle $\mathsf{Enc}$: (1) it queries $(N, A, M, (\varphi_e, \varphi_m))$ and (2) it queries $(N, A, M, (\varphi_e, \varphi'_m))$. These queries result in two ciphertexts $C_1 = C_{e,1} \parallel T_1$ and $C_2 = C_{e,2} \parallel T_2$. The adversary outputs $b' = 0$ if $C_{e,1} = C_{e,2}$, otherwise, it outputs $b' = 1$.

Note, first, that both queries are valid as they differ in the RKD function of the underlying MAC. If the bit $b$ of game $\mathsf{s\text{-}rka\text{-}AE}$ equals 1, the adversary will receive two randomly chosen bit strings as the ciphertexts $C_1$ and $C_2$. If the bit $b$ is 0, the adversary will receive $C_1 = C_{e,1} \parallel T_1$, where $C_{e,1} = \texttt{Enc}(\varphi_e(K_e), N, M)$, and $C_2 = C_{e,2} \parallel T_2$, where $C_{e,2} = \mathtt{Enc}(\varphi_e(K_e), N, M)$. Since $C_{e,1}$ equals $C_{e,2}$ this allows the adversary to distinguish the cases almost perfectly.

Adversary $\mathcal{A}_m$ picks a message $M$, a nonce $N$, and associated data $A$ at random from the respective sets. Furthermore, it picks $\varphi_m \in \Phi_m$ and $\varphi_e, \varphi'_e \in \Phi_e$ such that $\varphi_e \neq \varphi'_e$. Then the adversary makes two queries to its encryption oracle $\mathsf{Enc}$: (1) it queries $(N, A, M, (\varphi_e, \varphi_m))$ and (2) it queries $(N, A, M, (\varphi'_e, \varphi_m))$. These queries result in two ciphertexts $C_1 = C_{e,1} \parallel T_1$ and $C_2 = C_{e,2} \parallel T_2$. If $T_1 = T_2$, the adversary outputs $b' = 0$, otherwise, it outputs $b' = 1$.

Again, note that both queries are valid as they differ in the RKD function of the underlying encryption scheme. If the bit $b$ of game $\mathsf{s\text{-}rka\text{-}AE}$ equals 1, the adversary will receive two randomly chosen bit strings for $C_1$ and $C_2$. If the bit $b$ is 0, the adversary will receive $C_1 = C_{e,1} \parallel T_1$, where $T_1 = \mathtt{Tag}(\varphi_m(K_m), N, A, M)$, and $C_2 = C_{e,2} \parallel T_2$, where $T_2 = \mathtt{Tag}(\varphi_m(K_m), N, A, M)$. Since $T_1$ equals $T_2$ this allows the adversary to distinguish the cases almost perfectly.

Letting $\mathcal{A}$ be either of the adversaries $\mathcal{A}_e$ and $\mathcal{A}_m$ proves the claim.

## 4.2 N2 - Instantiation of Encrypt-then-MAC

The N2 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the EtM paradigm and is displayed in Fig. 6. The scheme first encrypts the message using the encryption scheme. Subsequently, the MAC is used to compute a tag for the ciphertext. The ciphertext of the AEAD scheme consists of both the ciphertext and the tag.

The theorem below shows that the N2 construction achieves RKA-AE security if the underlying primitives are sound. The overall proof follows the classical one, except for a more complex analysis regarding the permitted queries.

**Theorem 3.** *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets $\Phi_e$ and $\Phi_m$, respectively. Further, let* N2 *be the AEAD scheme built from $\Sigma$ and $\Gamma$ using the N2 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any RKA-nonce-respecting and $\Phi_{ae}$-restricted RKA adversary $\mathcal{A}$ against* N2*, that never repeats/forwards a query to/from* $\mathsf{Enc}$*, there exists an RKA-nonce-respecting and $\Phi_e$-restricted RKA adversary $\mathcal{A}_{se}$, a $\Phi_m$-restricted RKA adversary $\mathcal{A}_{mac}$, and a $\Phi_m$-restricted RKA adversary $\mathcal{A}_{prf}$ such that*

$$\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N2}}(\mathcal{A}, \Phi_{ae}) \leq \mathbf{Adv}^{\mathsf{rkaIND}}_{\Sigma}(\mathcal{A}_{se}, \Phi_e) + \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m) + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathtt{Tag}}(\mathcal{A}_{prf}, \Phi_m)\,.$$

*Proof (Sketch).* The full proof is given in Appendix B.1. In the first game hop, the decryption oracle is replaced by $\perp$ which is bound by the RKA security of $\Gamma$. In the subsequent game hops, first the tag and then the ciphertext are replaced by random values which is bound by the RKA security of $\mathtt{Tag}$ and $\Sigma$, respectively.

Below we show that the N2 construction does not achieve s-RKA-AE security. It exhibits the same structure as the N1 construction, that is, a concatenation of a ciphertext and a tag from the underlying primitives. The difference is the tag is computed on the ciphertext rather than the message. Due to this one of the attacks against the N1 construction, the one exploiting the encryption scheme, also applies to the N2 construction, while the other, against the MAC, does not apply. Nevertheless, this shows that the N2 construction is insecure.

**Theorem 4.** *Let $\Sigma = (\texttt{Enc}, \texttt{Dec})$ be an encryption scheme and $\Gamma = (\texttt{Tag}, \texttt{Ver})$ be a MAC with RKA function sets $\Phi_e$ and $\Phi_m$, respectively. Further, let* N2 *be the AEAD scheme built from $\Sigma$ and $\Gamma$ using the N2 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then* N2 *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and $\Phi_{ae}$-restricted RKA adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\textsf{s-rka-AE}}_{\mathrm{N2}}(\mathcal{A}) = 1\,.$$

*Proof.* We construct the following adversary $\mathcal{A}$. It picks a message $M$, a nonce $N$, and associated data $A$ at random from the respective sets. Furthermore, it picks $\varphi_e \in \Phi_e$ and $\varphi_m, \varphi'_m \in \Phi_m$ such that $\varphi_m \neq \varphi'_m$. Then the adversary makes two queries to its encryption oracle $\textsf{Enc}$: (1) it queries $(N, A, M, (\varphi_e, \varphi_m))$ and (2) it queries $(N, A, M, (\varphi_e, \varphi'_m))$. These queries result in two ciphertexts $C_1 = C_{e,1} \parallel T_1$ and $C_2 = C_{e,2} \parallel T_2$. If $C_{e,1} = C_{e,2}$, the adversary outputs $b' = 0$, otherwise, it outputs $b' = 1$.

Note first that both queries are valid as they differ in the RKD function of the underlying MAC. If the bit $b$ of game $\textsf{s-rka-AE}$ equals 1, the adversary will receive two randomly chosen bit strings as the ciphertexts $C_1$ and $C_2$. If the bit $b$ is 0, the adversary will receive $C_1 = C_{e,1} \parallel T_1$, where $C_{e,1} = \texttt{Enc}(\varphi_e(K_e), N, M)$, and $C_2 = C_{e,2} \parallel T_2$, where $C_{e,2} = \texttt{Enc}(\varphi_e(K_e), N, M)$. Since $C_{e,1}$ equals $C_{e,2}$ this allows the adversary to distinguish the cases almost perfectly.

## 4.3 N3 - Instantiation of MAC-then-Encrypt

The N3 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the MtE paradigm and is displayed in Fig. 6. The message is first used as an input to the MAC and then both the message and the tag are encrypted. In contrast to the other compositions, the ciphertext of the AEAD scheme consists only of the ciphertext from the underlying encryption scheme.

In the theorem below, we show that the N3 construction is RKA-AE secure if both of the underlying primitives are secure. The overall proof follows the classical setting, except for the analysis that all queries by the reductions are indeed valid queries.

**Theorem 5.** *Let $\Sigma = (\texttt{Enc}, \texttt{Dec})$ be an encryption scheme and $\Gamma = (\texttt{Tag}, \texttt{Ver})$ be a MAC with RKA function sets $\Phi_e$ and $\Phi_m$, respectively. Further, let* N3 *be the AEAD scheme built from $\Sigma$ and $\Gamma$ using the N3 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any RKA-nonce-respecting and $\Phi_{ae}$-restricted RKA adversary $\mathcal{A}$ against* N3*, that never repeats/forwards a query to/from* $\textsf{Enc}$*, there* *exists an RKA-nonce-respecting and $\Phi_e$-restricted RKA adversary $\mathcal{A}_{se}$, a $\Phi_m$-restricted RKA adversary $\mathcal{A}_{mac}$, and a $\Phi_m$-restricted RKA adversary $\mathcal{A}_{prf}$ such that*

$$\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N3}}(\mathcal{A}, \Phi_{ae}) \leq \mathbf{Adv}^{\mathsf{rkaIND}}_{\Sigma}(\mathcal{A}_{se}, \Phi_e) + \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m) + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathtt{Tag}}(\mathcal{A}_{prf}, \Phi_m)\,.$$

```
Game $\mathsf{G}_i$
    $(K_e, K_m) \leftarrow_\$ \mathcal{K}$
    $b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_0$
    $M \parallel T \leftarrow \mathtt{Dec}(\varphi_e(K_e), N, C)$
    if $\mathtt{Ver}(\varphi_m(K_m), N, A, M, T) = \top$
        return $M$
    return $\bot$

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_1$, $\mathsf{G}_2$, $\mathsf{G}_3$
    return $\bot$
```

```
$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_0$, $\mathsf{G}_1$
    $T \leftarrow \mathtt{Tag}(\varphi_m(K_m), N, A, M)$
    $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$
    return $C$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_2$
    $T \leftarrow_\$ \{0,1\}^t$
    $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$
    return $C$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_3$
    $C \leftarrow_\$ \{0,1\}^c$
    return $C$
```

Fig. 7: Hybrid games $\mathsf{G}_i$ used to prove Theorem 5 (RKA-AE security of N3).

*Proof.* We prove the theorem using the hybrid games $\mathsf{G}_0$, $\mathsf{G}_1$, $\mathsf{G}_2$, and $\mathsf{G}_3$ displayed in Fig. 7. For sake of simplicity, the games do not contain the set $\mathcal{S}$ to detect invalid queries. Instead, we assume that the adversary does not make such queries, which the reduction can simply answer with $\bot$. Game $\mathsf{G}_0$ is $\mathsf{rka\text{-}AE}$ instantiated with N3 and secret bit $b$ fixed to 0. In $\mathsf{G}_1$, the decryption oracle is modified to reject any ciphertext. In $\mathsf{G}_2$, encryption oracle computes a random tag which is then encrypted along with the message. Game $\mathsf{G}_3$ equals $\mathsf{rka\text{-}AE}$ with secret bit $b$ fixed to 1, where the encryption oracle outputs random ciphertexts and the decryption oracle rejects any ciphertext. We have

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N3}}(\mathcal{A})\\
&= \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 0 \mid b = 0] - \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 0 \mid b = 1]\\
&= \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0]\\
&= \sum_{i=1}^{3} \Pr[\mathcal{A}^{\mathsf{G}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_i} \Rightarrow 0]\,.
\end{aligned}$$

To bound the term $\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0]$ we construct the following adversary $\mathcal{A}_{mac}$ against the RKA-SUF security of $\Gamma$. It chooses a random key $K_e$ for the encryption scheme $\Sigma$ and then runs $\mathcal{A}$. When $\mathcal{A}$ makes a query $(N, A, M, (\varphi_e, \varphi_m))$ to $\mathsf{Enc}$, $\mathcal{A}_{mac}$ proceeds as follows. It queries its oracle $\mathsf{Tag}$ on $(N, A, M, \varphi_m)$ to obtain a tag $T$. Then it locally computes $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$ and sends $C$ back to $\mathcal{A}$. For queries $(N, A, C, (\varphi_e, \varphi_m))$ to $\mathsf{Dec}$ by $\mathcal{A}$, $\mathcal{A}_{mac}$ locally computes $M \parallel T \leftarrow \mathtt{Dec}(\varphi_e(K_e), N, C)$ and queries $(N, A, M, T, \varphi_m)$ to its challenge oracle $\mathsf{Ver}$. If the response is $\perp$, it forwards it to $\mathcal{A}$, otherwise, it sends $M$ to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit $b'$, $\mathcal{A}_{mac}$ outputs the same bit.

Ir remains to argue that $\mathcal{A}_{mac}$ never makes a forbidden query (forwarding from $\mathsf{Tag}$ to $\mathsf{Ver}$) conditioned on $\mathcal{A}$ making only permitted queries. Assume, for sake of contradiction, that $\mathcal{A}$ makes a valid query $(N, A, C, \varphi_e, \varphi_m)$ to $\mathsf{Dec}$ for which $\mathcal{A}_{mac}$ makes a forbidden query. By construction $\mathcal{A}_{mac}$ computes $M \parallel T \leftarrow \mathtt{Dec}(\varphi_e(K_e), N, C)$ and queries $\mathsf{Ver}$ on $(N, A, M, T, \varphi_m)$. This query is forbidden if $\mathcal{A}_{mac}$ has queried $(N, A, M, \varphi_m)$ to $\mathsf{Tag}$ which resulted in $T$. This happens if $\mathcal{A}$ has made a query $(N, A, M, \varphi_e', \varphi_m)$ to $\mathsf{Enc}$. We need to distinguish between the case $\varphi_e' = \varphi_e$ and $\varphi_e' \neq \varphi_e$. The former is forbidden as this means that $\mathcal{A}$ forwards a query from $\mathsf{Enc}$ to $\mathsf{Dec}$. The latter is forbidden since game $\mathsf{rka\text{-}AE}$ forbids queries that agree on the nonce and exactly one of the RKD functions while disagreeing on the other RKD function. Hence $\mathcal{A}_{mac}$ only makes permitted queries.

By construction, $\mathcal{A}_{mac}$ simulates either $\mathsf{G}_0$ or $\mathsf{G}_1$ for $\mathcal{A}$, depending on its secret bit $b$ from game $\mathsf{rkaSUF}$. More precisely, it simulates $\mathsf{G}_0$ and $\mathsf{G}_1$ if its own challenge is 0 and 1, respectively. This gives us

$$\begin{aligned}
&\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] \\
\leq &\Pr[\mathcal{A}_{mac}^{\mathsf{rkaSUF}} \Rightarrow 0 \mid b = 0] - \Pr[\mathcal{A}_{mac}^{\mathsf{rkaSUF}} \Rightarrow 0 \mid b = 1] \\
\leq &\mathbf{Adv}_{\Gamma}^{\mathsf{rkaSUF}}(\mathcal{A}_{mac}, \Phi_m)\,.
\end{aligned}$$

For the term $\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0]$, we construct an adversary $\mathcal{A}_{prf}$ against the RKA-PRF security of the tagging algorithm $\mathtt{Tag}$. First, $\mathcal{A}_{prf}$ chooses a random key $K_e$ to simulate all encryption related functionalities. Queries to $\mathtt{Dec}$ by $\mathcal{A}$ are answered with $\perp$. Queries $(N, A, M, (\varphi_e, \varphi_m))$ to $\mathsf{Enc}$, are processed as follows. The reduction $\mathcal{A}_{prf}$ invokes its own oracle $\mathsf{F}$ on $(N, A, M, \varphi_m)$ to obtain $T$, locally computes $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$, and sends $C$ back to $\mathcal{A}$. When $\mathcal{A}$ terminates, $\mathcal{A}_{prf}$ also terminates and outputs whatever $\mathcal{A}$ does.

We briefly argue that $\mathcal{A}_{prf}$ never repeats a query to $\mathsf{F}$. By construction, every query $(N, A, M, \varphi_m)$ by $\mathcal{A}_{prf}$ stems from a query $(N, A, M, \varphi_e, \varphi_m)$ by $\mathcal{A}$. The only cases that result in a repeating query are (1) $\mathcal{A}$ repeats a query and (2) $\mathcal{A}$ makes two queries which only differ in $\varphi_e$. However, both cases are forbidden queries for $\mathcal{A}$. This yields that every output of $\mathsf{Tag}$ is a random value.

The adversary $\mathcal{A}_{prf}$ simulates game $\mathsf{G}_1$ for $\mathcal{A}$ if its own challenge bit $b$ equals 0, while it simulates $\mathsf{G}_2$ for $\mathcal{A}$ if $b$ equals 1. Thus it holds that

$$\begin{aligned}
&\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] \\
&\leq \Pr[\mathcal{A}_{prf}^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{prf}^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 1] \\
&\leq \mathbf{Adv}_{\Gamma}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}, \Phi_m)\,.
\end{aligned}$$

We bound the final term $\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0]$ by constructing an adversary $\mathcal{A}_{se}$ against the RKA-IND security of the underlying encryption scheme $\Sigma$. At the start, $\mathcal{A}_{se}$ chooses a random key $K_m$. Any query to $\mathsf{Dec}$ is answered with $\perp$. When $\mathcal{A}$ queries its oracle $\mathsf{Enc}$ on $(N, A, M, (\varphi_e, \varphi_m))$, $\mathcal{A}_{se}$ chooses a random tag $T$ of length $t$, invokes its oracle $\mathsf{Enc}$ on $(N, M \parallel T, \varphi_e)$ to obtain $C$, and sends $C$ to $\mathcal{A}$. At the end, $\mathcal{A}_{se}$ outputs whatever $\mathcal{A}$ outputs.

It holds that $\mathcal{A}_{se}$ is RKA-nonce-respecting as any query $(N, M \parallel T, \varphi_e)$ stems from a query $(N, A, M, \varphi_e, \varphi_m)$ by $\mathcal{A}$. This means that $\mathcal{A}_{se}$ repeats a pair of nonce $N$ and RKD function $\varphi_e$ if $\mathcal{A}$ makes two queries using $(N, \varphi_e, \varphi_m)$ and $(N, \varphi_e, \varphi'_m)$. We can distinguish between the cases (1) $\varphi_m = \varphi'_m$ and (2) $\varphi_m \neq \varphi'_m$. Case (1) does not occur, as $\mathcal{A}$ is RKA-nonce-respecting and case (2) is forbidden in game $\mathsf{rka\text{-}AE}$. The other option would be that $\mathcal{A}$ makes two queries differing only in the associated data $A$. This turns out not to be an issue, as the tag $T$ that $\mathcal{A}_{se}$ queries along with the message depends on $A$, i.e., different $A$ results in a different message queries by $\mathcal{A}_{se}$.

The adversary $\mathcal{A}_{se}$ perfectly simulates games $\mathsf{G}_2$ or $\mathsf{G}_3$ for $\mathcal{A}$ depending on its own challenge from $\mathsf{rkaIND}$. Hence we have

$$\begin{aligned}
&\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] \\
&\leq \Pr[\mathcal{A}_{se}^{\mathsf{rkaIND}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{se}^{\mathsf{rkaIND}} \Rightarrow 0 \,|\, b = 1] \\
&\leq \mathbf{Adv}_{\Gamma}^{\mathsf{rkaIND}}(\mathcal{A}_{se}, \Phi_e)\,.
\end{aligned}$$

Collecting the bounds above proves the claim.

Unlike for the N1 and N2 construction, the s-RKA-AE security of the N3 construction is more subtle. The difference is that the tag is appended to the ciphertext for both the N1 and N2 construction while it is encrypted alongside the message for the N3 construction. The attacks against the N1 and N2 construction rely on the property that the ciphertext consists of two parts which can be manipulated separately. Due to the construction such attacks do not work against the N3 construction.

It turns out that the s-RKA-AE security of the N3 construction crucially depend on the used encryption scheme. Namely, if the underlying encryption scheme is a stream cipher, then the N3 construction is s-RKA-AE insecure. Below we show an attack against any instantiation using a stream cipher. For such ciphers the ciphertext is the XOR of the message and a keystream derived from the key and the nonce.

**Theorem 6.** *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be a stream cipher and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets $\Phi_e$ and $\Phi_m$, respectively. Further, let* N3 *be the AEAD scheme built from $\Sigma$ and $\Gamma$ using the N2 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then* N3 *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and $\Phi_{ae}$-restricted RKA adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N3}}(\mathcal{A}) = 1\,.$$

*Proof.* Adversary $\mathcal{A}$ chooses a nonce $N$, associated data $A$, a message $M$, RKD functions $\varphi_e$, $\varphi_m$, and $\varphi'_m$ from the respective sets such that $\varphi_m \neq \varphi'_m$. Then it queries its encryption oracle $\mathsf{Enc}$ on $(N, A, M, (\varphi_e, \varphi_m))$ and $(N, A, M, (\varphi_e, \varphi'_m))$ to obtain ciphertext $C_1$ and $C_2$. If the first $|M|$ bits of $C_1$ and $C_2$ are equal, $\mathcal{A}$ outputs 0, otherwise, it outputs 1.

In case $b = 0$, it holds thatwe have $C_1 = \mathtt{Enc}(\varphi_e(K_e), N, M \parallel \mathtt{Tag}(\varphi_m(K_m), N, A, M))$ and $C_2 = \mathtt{Enc}(\varphi_e(K_e), N, M \parallel \mathtt{Tag}(\varphi'_m(K_m),\ N, A, M))$. Since the encryption uses the same nonce and the same key, the same keystream for the stream cipher will be used. Together with the fact that the first $|M|$ bits are identical as the same message is encrypted, this yields that $C_1$ and $C_2$ agree on the first bits. In case $b = 1$, both $C_1$ and $C_2$ are chosen at random, hence they will not agree on the first $|M|$ bits.

In the attack above, the RKA-nonce-respecting adversary essentially bypasses the nonce-respecting property of the underlying encryption scheme by repeating the nonce $N$ and the RKD function $\varphi_e$ for the encryption scheme. Then it exploits the fact that the underlying stream cipher is secure only against nonce-respecting adversaries. We conjecture that any instantiation using an encryption scheme that can be broken in the nonce-misuse case results in an s-RKA-AE insecure instantiation of the N3 construction. The problematic part is that both the message and the tag are encrypted. While the adversary has full control over the former, it can not choose the latter at will. This seems to thwart a simple proof showing that any nonce-misuse adversary against the underlying encryption scheme can be turned into an s-RKA-AE adversary against N3.

## 5 RKA Nonce Misuse-resistent AEAD

As described in Section 4, N1, N2, and N3 are not secure in the strong RKA setting.[8] In this section we give a new AE scheme, N*, that achieves mr-s-RKA-AE security and hence also s-RKA-AE security. The construction, following the N3 construction, is displayed in Fig. 8. The message, nonce, and associated data are first used as an input to the MAC, and then both the message and the tag are encrypted. The difference to the N3 construction is that the encryption scheme no longer takes the nonce as input. Instead, the (pseudorandom) tag ensures that the encryption is randomised.

[8] One solution would be to use the key derivation technique proposed in [7]. However, this requires the usage of an additional PRF on top of the existing AE scheme.

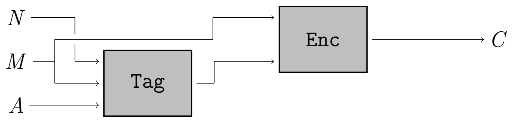

Fig. 8: The AEAD scheme N* [This work].

The theorem below shows that the new construction achieves our strong RKA security notion conditioned on the encryption scheme being an RKA-secure block cipher (pseudorandom permutation).

**Theorem 7.** *Let $\Sigma = (\texttt{Enc}, \texttt{Dec})$ be an encryption scheme and $\Gamma = (\texttt{Tag}, \texttt{Ver})$ be a MAC with RKA function sets $\Phi_e$ and $\Phi_m$, respectively. Further, let* N* *be the AEAD scheme built from $\Sigma$ and $\Gamma$ using the* N* *construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any $\Phi_{ae}$-restricted RKA adversary $\mathcal{A}$ against* N* *with $q$ queries to the encryption and decryption oracle, that never repeats/forwards a query to/from* Enc*, there exists $\Phi_e$-restricted RKA adversaries $\mathcal{A}_{prp}$, and $\Phi_m$-restricted RKA adversaries $\mathcal{A}_{mac}$ and $\mathcal{A}_{prf}$ such that*

$$\begin{aligned}&\mathbf{Adv}^{\textsf{mr-s-rka-AE}}_{\mathrm{N}^*}(\mathcal{A}, \Phi_{ae})\\ &\le \mathbf{Adv}^{\textsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m) + \mathbf{Adv}^{\textsf{rkaPRP}}_{\Sigma}(\mathcal{A}_{prp}, \Phi_e)\\ &\quad + \mathbf{Adv}^{\textsf{rkaPRF}}_{\texttt{Tag}}(\mathcal{A}_{prf}, \Phi_m) + \frac{2q^2}{2^c}\,.\end{aligned}$$

*Proof.* Game $\mathsf{G}_0$ in Fig. 9 is the $\mathsf{mr\text{-}s\text{-}rka\text{-}AE}$ security game instantiated with N* and secret bit $b = 0$ and game $\mathsf{G}_5$ is the $\mathsf{mr\text{-}s\text{-}rka\text{-}AE}$ security game with $b = 1$. To estimate the security of N*, four additional games $\mathsf{G}_1$, $\mathsf{G}_2$, $\mathsf{G}_3$, and $\mathsf{G}_4$ are needed. Starting with $\mathsf{mr\text{-}s\text{-}rka\text{-}AE}$ with $b = 0$ ($\mathsf{G}_0$), we modify the intermediate games as follows: In game $\mathsf{G}_1$ the decryption always outputs $\perp$ except the resulting message was sent to the encryption oracle with the same $N$, $A$, and $\varphi_m$ before. In $\mathsf{G}_2$, the underlying encryption scheme is replaced by a random permutation. In $\mathsf{G}_3$, the decryption oracle always outputs $\perp$. In $\mathsf{G}_4$, the Tag algorithm is replaced by a random function. Finally, in game $\mathsf{G}_5$, the encryption oracle ignores the input, and outputs a uniform random cipher $C$ as in $\mathsf{mr\text{-}s\text{-}rka\text{-}AE}$ with $b = 1$. With $\mathbf{Adv}^{\textsf{mr-s-rka-AE}}_{\mathrm{N}^*}(\mathcal{A}, \Phi_{ae}) \le \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_5})$ and $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_5}) \le \sum_{i=0}^{4} \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_i}, \mathcal{A}^{\mathsf{G}_{i+1}})$, Claim 1 - 5 conclude the proof.

**Claim 1** *For any $\Phi_{ae}$-restricted RKA distinguisher $\mathcal{A}$ between game $\mathsf{G}_0$ and $\mathsf{G}_1$ defined in Fig. 9, there exists a $\Phi_m$-restricted RKA adversary $\mathcal{A}_{mac}$ such that*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_1}) \le \mathbf{Adv}^{\textsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m)\,.$$

*Proof.* In the following, an adversary $\mathcal{A}_{mac}$ is given that wins the game with the advantage of $\mathcal{A}$. $\mathcal{A}_{mac}$ simulates the game by using the oracles of the security game $\mathsf{rkaSUF}$ to get the tags $T$ for the encryption and to verify $T$ for the decryption of the N* scheme. Further, $\mathcal{A}_{mac}$ computes the encryption

```
Game $\mathsf{G}_i$
    $(K_e, K_m) \leftarrow_\$ \mathcal{K}$
    $\mathcal{T} \leftarrow \emptyset$
    $\mathtt{F} \leftarrow_\$ \mathsf{Func}(\mathcal{K}_m, \mathcal{N} \times \mathcal{A} \times \mathcal{M}, \{0,1\}^t)$
    $\mathtt{P} \leftarrow_\$ \mathsf{Perm}(\mathcal{K}_e, \{0,1\}^c)$
    $b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_0$, $\mathsf{G}_1$
    $T \leftarrow \mathtt{Tag}(\varphi_m(K_m), N, A, M)$
    $\mathcal{T} \leftarrow_\cup \{(N, A, C, (\varphi_e, \varphi_m))\}$
    return $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), M \parallel T)$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_2$, $\mathsf{G}_3$
    $T \leftarrow \mathtt{Tag}(\varphi_m(K_m), N, A, M)$
    $f[N, A, \varphi_m] \leftarrow_\cup \{(M, T)\}$
    $\mathcal{T} \leftarrow_\cup \{(N, A, C, (\varphi_e, \varphi_m))\}$
    return $C \leftarrow \mathtt{P}(\varphi_e(K_e), M \parallel T)$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_4$
    $T \leftarrow \mathtt{F}(\varphi_m(K_m), N, A, M)$
    return $C \leftarrow \mathtt{P}(\varphi_e(K_e), M \parallel T)$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_5$
    return $C \leftarrow_\$ \{0,1\}^c$

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_0$
    if $(N, A, C, (\varphi_e, \varphi_m)) \in \mathcal{T}$
        return $\perp$
    $M \parallel T \leftarrow \mathtt{Dec}(\varphi_e(K_e), C)$
    $V \leftarrow \mathtt{Ver}(\varphi_m(K_m), N, A, M, T)$
    if $V = \perp$
        return $\perp$
    return $M$

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_1$
    if $(N, A, C', (\varphi'_e, \varphi_m)) \in \mathcal{T}$ with $\varphi_e \neq \varphi'_e$
        if $\mathtt{Dec}(\varphi_e(K_e), C) = \mathtt{Dec}(\varphi'_e(K_e), C')$
            $(M \parallel T) \leftarrow \mathtt{Dec}(\varphi'_e(K_e), C')$
            return $M$
    return $\perp$

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_2$
    if $(N, A, C', (\varphi'_e, \varphi_m)) \in \mathcal{T}$ with $\varphi_e \neq \varphi'_e$
        for $(M, T) \in f[N, A, \varphi_m]$
            if $C = \mathtt{P}(\varphi_e(K_e), M \parallel T))$
            return $M$
    return $\perp$

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_3$, $\mathsf{G}_4$, $\mathsf{G}_5$
    return $\perp$
```

Fig. 9: Hybrid games $\mathsf{G}_i$ used to prove Theorem 7.

scheme $\Sigma$ locally with a key $K_e$ chosen uniform at random to simulate the encryption oracle of game $\mathsf{G}_0$ and $\mathsf{G}_1$. $\mathcal{A}_{mac}$ simulates both games perfectly and $\mathcal{A}$ can only distinguish both games if it requests a decryption of a valid ciphertext $(N, A, C, (\varphi_e, \varphi_m))$ with $(M \parallel T) = \mathtt{Dec}(\varphi_e(K_e), N, C)$, such that $(N, A, M, T, \varphi_m)$ is new and the Tag $T$ is valid. Hence, $(N, A, M, \varphi_m)$ was not forwarded to the Tag oracle of rkaSUF and $\mathcal{A}_{mac}$ can use this request to win the game rkaSUF by forwarding $(N, A, M, T, \varphi_m)$ to the verification oracle Ver of rkaSUF, since it was not sent to the oracle Tag of rkaSUF before. Hence, $\mathcal{A}_{mac}$ is a $\Phi_m$-restricted RKA adversary because $\mathcal{A}$ is $\Phi_{ae}$-restricted, and it holds $\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] \leq \Pr[\mathcal{A}_{mac}^{\mathsf{rkaSUF}} \Rightarrow 0 \mid b = 0] - \Pr[\mathcal{A}_{mac}^{\mathsf{rkaSUF}} \Rightarrow 0 \mid b = 1]$, and therefore $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_1}) \leq \mathbf{Adv}_{\Gamma}^{\mathsf{rkaSUF}}(\mathcal{A}_{mac}, \Phi_m)$.

**Claim 2** *For any $\Phi_{ae}$-restricted RKA distinguisher $\mathcal{A}$ between game $\mathsf{G}_1$ and $\mathsf{G}_2$ defined in Fig. 9, there exists an $\Phi_e$-restricted RKA adversary $\mathcal{A}_{prp}$ such that*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) \leq \mathbf{Adv}_{\Sigma}^{\mathsf{rkaPRP}}(\mathcal{A}_{prp}, \Phi_e)\,.$$

*Proof.* Before we describe the simulator, we transform $\mathsf{G}_1$ to $\mathsf{G}_1'$ to avoid that we need to query the inverse of the oracle F in rkaPRP. In $\mathsf{G}_1'$[9] we replace the underling decryption function with the encryption function in such a way that the input/output behaviour of $\mathsf{G}_1$ and $\mathsf{G}_1'$ is still the same.

```
Dec$(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_1'$
  if $(N, A, C', (\varphi_e', \varphi_m)) \in \mathcal{T}$ with $\varphi_e \neq \varphi_e'$
  for $(M, T) \in f[N, A, \varphi_m]$
  if $C = \mathtt{Enc}(\varphi_e'(K_e), N, M \parallel T)$
    return $M$
  return $\bot$
```

```
Enc$(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_1'$
  $T \leftarrow \mathtt{Tag}(\varphi_m(K_m), N, A, M)$
  $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$
  $f[N, A, \varphi_m] \leftarrow\cup \{(M, T)\}$
  $\mathcal{T} \leftarrow\cup \{(N, A, C, (\varphi_e, \varphi_m))\}$
  return $C$
```

In $\mathsf{G}_1$ the decryption oracle only returns the decrypted message $M$ if $M$ was sent to the encryption oracle before. Since the underlying Enc is deterministic, we can also save the queries to the encryption oracles in $f$ and test if it encrypts to $C$ as we do in the decryption oracle of $\mathsf{G}_1'$. Hence, it holds that the games are identical and it is enough to show that $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1'}, \mathcal{A}^{\mathsf{G}_2}) \leq \mathbf{Adv}_{\Sigma}^{\mathsf{rkaPRP}}(\mathcal{A}_{prp}, \Phi_e)$. We construct an adversary $\mathcal{A}_{prp}$ simulating $\mathsf{G}_1'$ and $\mathsf{G}_2$ by computing the MAC locally with a uniform random key $K_m$ and using oracle F of rkaPRP for the encryption. $\mathcal{A}_{prp}$ is $\Phi_e$-restricted because $\mathcal{A}$ is $\Phi_{ae}$-restricted. Hence, $\mathcal{A}_{prp}$ perfectly simulates $\mathsf{G}_1'$ if the challenge bit of rkaPRP is 0, and $\mathsf{G}_2$ if the challenge bit is 1. It holds $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) = \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1'}, \mathcal{A}^{\mathsf{G}_2}) \leq \mathbf{Adv}_{\Sigma}^{\mathsf{rkaPRP}}(\mathcal{A}_{prp}, \Phi_e)\,.$

**Claim 3** *For any $\Phi_{ae}$-restricted RKA distinguisher $\mathcal{A}$ with $q$ queries between game $\mathsf{G}_2$ and $\mathsf{G}_3$ defined in Fig. 9, it holds*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) \leq \frac{q^2}{2^c}\,.$$

[9] This transformation allows us to use a normal PRP for the simulation, and not a strong PRP

*Proof.* Both games only differ if $\mathcal{A}$ asks for a decryption of a cipher text $C$ which maps to a message which was already encrypted with the same $(N, A, \varphi_m)$. Since the underlying encryption is a random permutation this collision happens with probability less then $\frac{q^2-q}{2^c}$. In detail $\mathcal{A}$ makes $q_e$ queries to the encryption oracle, and $q_d$ queries to the decryption oracle with $q_e + q_d = q$. The collision probability is less then $\frac{q_e}{2^c}$ for each query to the decryption oracle. Hence, the probability to get at least one collision is less then $\frac{q_e q_d}{2^c}$ with $q_d$ queries to the decryption oracle. Hence, $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) \leq \frac{q^2}{2^c}$.

**Claim 4** *For any $\Phi_{ae}$-restricted RKA distinguisher $\mathcal{A}$ between game $\mathsf{G}_3$ and $\mathsf{G}_4$ defined in Fig. 9, there exists a $\Phi_m$-restricted RKA adversary $\mathcal{A}_{prf}$ such that*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_3}, \mathcal{A}^{\mathsf{G}_4}) \leq \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{prf}, \Phi_m) .$$

*Proof.* $\mathcal{A}_{prf}$ simulates $\mathsf{G}_3$ and $\mathsf{G}_4$ for $\mathcal{A}$ with the oracles of the security game $\mathsf{rkaPRF}$. For any request $(N, M, A, (\varphi_e, \varphi_m))$ to the oracle $\mathsf{Enc}$, $\mathcal{A}_{prf}$ forwards $(N, A, M, \varphi_m)$ to the $\mathsf{rkaPRF}$ game's oracle $\mathsf{F}$ to get the tag $T$, computes the ciphertext $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$ locally with a random key $K_e$, and sends the ciphertext $C$ to $\mathcal{A}$. Since $\mathcal{A}$ is $\Phi_{ae}$-restricted, $\mathcal{A}_{prf}$ is $\Phi_m$-restricted and $\mathcal{A}_{prf}$ perfectly simulates $\mathsf{G}_{b+3}$ where $b$ is the challenge bit of game $\mathsf{rkaPRF}$ and outputs $b'$ if $\mathcal{A}$ does. It holds that $\Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_4} \Rightarrow 0] \leq \Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{prf} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{prf} \Rightarrow 0 \,|\, b = 1]$. Hence, $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_3}, \mathcal{A}^{\mathsf{G}_4}) \leq \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{prf}, \Phi_m)$.

**Claim 5** *For any $\Phi_{ae}$-restricted RKA distinguisher $\mathcal{A}$ between game $\mathsf{G}_4$ and $\mathsf{G}_5$ with $q$ queries to the encryption oracle defined in Fig. 9, it holds*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_4}, \mathcal{A}^{\mathsf{G}_5}) \leq \frac{q^2}{2^c} .$$

*Proof.* Both games only differ from the choice of the underlying encryption. Game $\mathsf{G}_4$ uses a random permutation and $\mathsf{G}_5$ generates randomly chosen ciphertexts. Since the adversary is not allowed to query the same tuple $(N, A, M, (\varphi_e, \varphi_m))$ and $\mathtt{F}$ is a real random function, it follows $T$ is fresh and uniform distributed or $\varphi_e$ is fresh. In case of a fresh $\varphi_e$, it follows directly that $C$ is chosen uniformly at random. If $\varphi_e$ was already used, an adversary can only distinguish both games if it finds a collision in $\mathsf{G}_5$ since $\mathsf{G}_4$ uses a permutation it is not possible to get the same $C$ twice with the same $\varphi_e$. The probability for such a collision is less then $\frac{q^2-q}{2^c}$. In detail we know that the probability to get a collision for the $i^{th}$ query is less then $\frac{i-1}{2^c}$ and hence $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_4}, \mathcal{A}^{\mathsf{G}_5}) \leq \sum_{i=1}^{q} \frac{i-1}{2^c} \leq \frac{q^2}{2^c}$. This proves the claim.

Similar to the N* construction, we can also initialize the N3 construction with a block cipher (PRP) to achieve security in the stronger security model. As discussed in the previous section this only works for N3 since the attack on N1 and N2 work independent of the underlying encryption scheme. Further, the security proof for N3 is similar to the proof of Theorem 7, we only have to adapt the block cipher that it also takes the nonce as input. We emphasize that the new construction N* is more efficient then N3, since the block cipher does not receive the nonce as an input. For the instantiation of the block cipher we refer to [5], where the authors construct an RKA-secure PRP using a three-round Feistel construction. The construction contains three RKA-secure PRFs, where the last two PRFs are initialized with the same key. Hence, with Theorem 7, we can build an mr-s-RKA-AE-secure AE scheme out of four RKA-secure PRFs, three to instantiate the block cipher and one for the MAC (cf. Section 6.2).

## 6 RKA-Secure Encryption and MAC

In this section we show that RKA-secure encryption and RKA-secure MACs can be built from RKA-secure pseudorandom functions. We investigate the FGHF' construction by Degabriele et al. [18] and show that the underlying encryption scheme and MAC are RKA-secure if the pseudorandom function is RKA-secure.

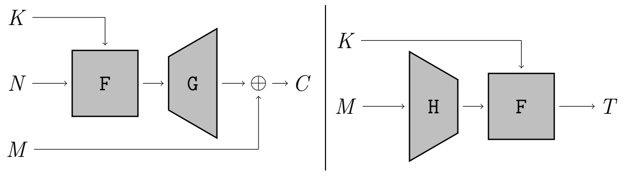

Fig. 10: Encryption scheme FG (left) and message authentication code HF (right) [18].

### 6.1 Related-Key Attack Secure Encryption

In this section we show that the encryption scheme FG proposed by Degabriele et al. [18] is RKA-secure. The scheme comprises a function $\mathtt{F}$ and a pseudorandom generator $\mathtt{G}$ and is displayed in Fig. 10. The ciphertext is computed by feeding the nonce $N$ into the function $\mathtt{F}$, expanding the output using the PRG $\mathtt{G}$, and XORing the output to the message. The scheme achieves RKA-IND security against RKA-nonce-respecting and $\Phi$-restricted adversaries, where $\Phi$ is specified from the RKA-PRF security of the underlying function $\mathtt{F}$. The result is given in the following theorem.

**Theorem 8.** *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme built from a function $\mathtt{F} : \mathcal{K} \times \{0,1\}^k \to \{0,1\}^m$ and a pseudorandom generator $\mathtt{G} : \{0,1\}^m \to \{0,1\}^m$ as displayed in Fig. 10. Then for any RKA-nonce-respecting and $\Phi$-restricted RKA adversary $\mathcal{A}$, making $q$ queries to $\mathsf{Enc}$, that never repeats a query, there exists a $\Phi$-restricted RKA adversary $\mathcal{A}_{prf}$ and an adversary $\mathcal{A}_{prg}$ such that*

$$\mathbf{Adv}_{\Sigma}^{\mathsf{rkaIND}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}) + q\,\mathbf{Adv}_{\mathtt{G}}^{\mathsf{PRG}}(\mathcal{A}_{prg})\,.$$

*Proof (Sketch).* The full proof is given in Appendix B.2. It consists of two game hops. In the first, the function $\mathtt{F}$ is replaced by a random function, in the second, the output of the PRG is replaced by random. The game hops are bound by the RKA security of the function $\mathtt{F}$ and the security of the PRG, respectively.

## 6.2 Related-Key Attack Secure Message Authentication

In this section we show that the message authentication code HF proposed by Degabriele et al. [18] is RKA-secure. The scheme comprises a hash function $\mathtt{H}$ and a function $\mathtt{F}$ and is displayed in Fig. 10. The tag is obtained by evaluating the hash function $\mathtt{H}$ on the message $M$ and evaluating the function $\mathtt{F}$ on the hash value.

**RKA-Secure MAC from RKA-secure PRF.** The theorem below shows that an RKA-secure pseudorandom function yields an RKA-secure MAC via the canonical MAC construction. That is, the tag is computed by evaluating the function on the message and verification works by recomputing the tag and comparison with the candidate tag.

**Theorem 9 (RKA-PRF yields RKA-MAC).** *Let $\mathtt{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function and $\Gamma$ be the canonical MAC built from $\mathtt{F}$. Then for any $\Phi$-restricted RKA adversary $\mathcal{A}$ against $\Gamma$ playing $\mathsf{rkaSUF}$, making $q$ queries to $\mathsf{Ver}$, there exists a $\Phi$-restricted RKA adversary $\mathcal{A}_{prf}$ against $\mathtt{F}$ playing $\mathsf{rkaPRF}$ such that*

$$\mathbf{Adv}_{\Gamma}^{\mathsf{rkaSUF}}(\mathcal{A}, \Phi) \leq 2\,\mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}, \Phi) + \frac{q}{|\mathcal{Y}|}\,.$$

*Proof (Sketch).* The full proof is given in Appendix B.3. It consists of three game hops. In the first hop, which is bound by the RKA security of $\mathtt{F}$, the function is replaced by a random function in both oracles. In the second hop, the verification oracle always rejects, which is bound by a simple combinatorial argument. In the third hop, we switch back to the real function $\mathtt{F}$ for the tagging oracle, which is again bound by the RKA security of $\mathtt{F}$.

**RKA-Secure PRF from RKA-secure PRF and Hash.** Theorem 9 shows that we can construct a RKA-secure MAC from a RKA-secure pseudorandom function. The main restriction is that the message space of the MAC equals the message space of the function. The following theorem shows that this is not a hindrance. It shows that the message space of a RKA-secure pseudorandom function can be extended to arbitrary long messages by first hashing the input and then applying the function to the hash value.

**Theorem 10 (Hash and RKA-PRF yields RKA-PRF).** *Let $\mathtt{F}\colon \mathcal{K}\times\mathcal{X} \to \mathcal{Y}$ be a function, $\mathtt{H}\colon \{0,1\}^* \to \mathcal{X}$ be a hash function and $\mathtt{F}'\colon \{0,1\}^* \to \mathcal{Y}, \mathtt{F}'(X) \mapsto \mathtt{F}(\mathtt{H}(X))$ be a function. Then for any $\Phi$-restricted RKA adversary $\mathcal{A}$ against $\mathtt{F}'$ playing $\mathsf{rkaPRF}$, there exists a $\Phi$-restricted RKA adversary $\mathcal{A}_{prf}$ against $\mathtt{F}$ playing $\mathsf{rkaPRF}$ and an adversary $\mathcal{A}_{hash}$ against $\mathtt{H}$ playing $\mathsf{CR}$ such that*

$$\mathbf{Adv}_{\mathtt{F}'}^{\mathsf{rkaPRF}}(\mathcal{A}, \Phi) \leq 2\,\mathbf{Adv}_{\mathtt{H}}^{\mathsf{CR}}(\mathcal{A}_{hash}) + \mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}, \Phi)\,.$$

*Proof (Sketch).* The proof consists of two hops which are bound by the collision resistance of the hash function. This allows us bound the resulting games by a straightforward reduction from the RKA security of $\mathtt{F}$.

**Acknowledgements**

This work was funded by the German Research Foundation (DFG) – SFB 1119 – 236615297 and the Emmy Noether Program FA 1320/1-1 of the German Research Foundation (DFG).

# Bibliography

[1] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. Related-key security for pseudorandom functions beyond the linear barrier. *Journal of Cryptology*, 31(4):917–964, October 2018.

[2] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 393–417. Springer, Heidelberg, January 2016.

[3] Martin R. Albrecht, Pooya Farshim, Kenneth G. Paterson, and Gaven J. Watson. On cipher-dependent related-key attacks in the ideal-cipher model. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 128–145. Springer, Heidelberg, February 2011.

[4] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *ICS 2011*.

[5] Manuel Barbosa and Pooya Farshim. The related-key analysis of Feistel constructions. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 265–284. Springer, Heidelberg, March 2015.

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

[7] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 486–503. Springer, Heidelberg, December 2011.

[8] 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.

[9] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, December 2000.

[10] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[11] Daniel J. Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2014.

[12] Rishiraj Bhattacharyya and Arnab Roy. Secure message authentication against related-key attack. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 305–324. Springer, Heidelberg, March 2014.

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

[14] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 507–525. Springer, Heidelberg, May 2005.

[15] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18. Springer, Heidelberg, December 2009.

[16] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, Heidelberg, August 2009.

[17] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer, Heidelberg, May 1997.

[18] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 209–240. Springer, Heidelberg, December 2019.

[19] Orr Dunkelman, Nathan Keller, and Jongsung Kim. Related-key rectangle attack on the full SHACAL-1. In Eli Biham and Amr M. Youssef, editors, *SAC 2006*, volume 4356 of *LNCS*, pages 28–44. Springer, Heidelberg, August 2007.

[20] Shuai Han, Shengli Liu, and Lin Lyu. Efficient KDM-CCA secure public-key encryption for polynomial functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 307–338. Springer, Heidelberg, December 2016.

[21] David G. Harris. Critique of the related-key attack concept. *Des. Codes Cryptogr.*, 2011.

[22] Takanori Isobe. A single-key attack on the full GOST block cipher. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 290–305. Springer, Heidelberg, February 2011.

[23] Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 196–208. Springer, Heidelberg, December 1993.

[24] Lars R. Knudsen and Tadayoshi Kohno. Analysis of RMAC. In Thomas Johansson, editor, *FSE 2003*, volume 2887 of *LNCS*, pages 182–191. Springer, Heidelberg, February 2003.

[25] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, August 1996.

[26] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999.

[27] Bonwook Koo, Deukjo Hong, and Daesung Kwon. Related-key attack on the full HIGHT. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC 10*, volume 6829 of *LNCS*, pages 49–67. Springer, Heidelberg, December 2011.

[28] Xianhui Lu, Bao Li, and Dingding Jia. KDM-CCA security from RKA secure authenticated encryption. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 559–583. Springer, Heidelberg, April 2015.

[29] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014.

[30] National Institute of Standards and Technology. Lightweight cryptography standardization process, 2015.

[31] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.

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

[33] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, Heidelberg, February 2004.

[34] 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.

[35] Serge Vaudenay. Clever arbiters versus malicious adversaries - on the gap between known-input security and chosen-input security. In Peter Y. A. Ryan, David Naccache, and Jean-Jacques Quisquater, editors, *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, Lecture Notes in Computer Science. Springer, 2016.

[36] Keita Xagawa. Message authentication codes secure against additively related-key attacks. Cryptology ePrint Archive, Report 2013/111, 2013. http://eprint.iacr.org/2013/111.

## A Additional Preliminaries

We recall the classical security notion for AEAD schemes given by Rogaway and Shrimpton [34]. The adversary gets access to two oracles: an encryption oracle $\mathsf{Enc}$ and a decryption oracle $\mathsf{Dec}$. Depending on a randomly chosen bit $b$, $\mathsf{Enc}$ and $\mathsf{Dec}$ implement either the real encryption and decryption algorithm (if $b = 0$) or an idealised version, i.e., $\mathsf{Enc}$ returns random bit and $\mathsf{Dec}$ always returns $\perp$, (if $b = 1$). The adversary wins the game if it guesses the bit $b$ correctly. For nonce-based schemes, the adversary can specify the nonce for each of its queries. The standard restriction imposed on the adversary is called *nonce-respecting* and forbids the adversary to query a nonce more than once to its encryption oracle. Note that there is no restriction on the queried nonces to the decryption oracle. Below we define security for nonce-based AEAD schemes.

```
Game AE
  $b \leftarrow_{\$} \{0,1\}$
  $K \leftarrow_{\$} \mathcal{K}$
  $b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$
  return $(b' = b)$

oracle $\mathsf{Enc}(N, A, M)$
  if $b = 0$
    $C \leftarrow \mathtt{Enc}(K, N, A, M)$
  else
    $C \leftarrow_{\$} \{0,1\}^c$
  return $C$

oracle $\mathsf{Dec}(N, A, C)$
  if $b = 0$
    $M \leftarrow \mathtt{Dec}(K, N, A, C)$
  else
    $M \leftarrow \perp$
  return $M$
```

Fig. 11: Security game AE.

**Definition 8 (AE Security).** *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an AEAD scheme and the game* AE *be defined as in Fig. 11. For a nonce-respecting adversary $\mathcal{A}$, that never repeats/forwards a query to/from* Enc*, we define its* AE *advantage as*

$$\mathbf{Adv}_{\Sigma}^{\mathsf{AE}}(\mathcal{A}) = 2 \Pr[\mathsf{AE}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

For pseudorandom generators and hash functions we use the standard security notions. For a pseudorandom generator the adversary has to distinguish the outputs of a pseudorandom generator on a randomly chosen seed from a random output. For a hash function the adversary has to find two distinct inputs that yield the same hash value.

## B Full Proofs

In this section we give the full proofs for the sketches given in Section 4.1, Section 6.1, and Section 6.2.

### B.1 Proof of Theorem 3

*Proof.* We prove the theorem using the hybrid games $\mathsf{G}_0$, $\mathsf{G}_1$, $\mathsf{G}_2$, and $\mathsf{G}_3$ displayed in Fig. 12. For ease of exposition, the games do not contain the set $\mathcal{S}$ to detect forbidden queries. Instead, we assume that the adversary does not make such a queries, in which case the reduction would simply return $\perp$. Game $\mathsf{G}_0$ is rka-AE instantiated with N2 and secret bit $b$ fixed to 0. In $\mathsf{G}_1$, the decryption oracle is modified to reject any ciphertext. In $\mathsf{G}_2$, the tag appended to the ciphertext is chosen at random. Game $\mathsf{G}_3$ equals rka-AE with secret bit $b$ fixed to 1, where the encryption oracle outputs random ciphertexts and the decryption

```
Game $\mathsf{G}_i$
    $(K_e, K_m) \leftarrow\$ \mathcal{K}$
    $b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_0$
    $C_e \parallel T \leftarrow C$
    if $\mathtt{Ver}(\varphi_m(K_m), N, A, C_e, T) = \top$
        $M \leftarrow \mathtt{Dec}(\varphi_e(K_e), N, C_e)$
        return $M$
    return $\bot$

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in $\mathsf{G}_1$, $\mathsf{G}_2$, $\mathsf{G}_3$
    return $\bot$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_0$, $\mathsf{G}_1$
    $C_e \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M)$
    $T \leftarrow \mathtt{Tag}(\varphi_m(K_m), N, A, C_e)$
    $C \leftarrow C_e \parallel T$
    return $C$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_2$
    $C_e \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M)$
    $T \leftarrow\$ \{0,1\}^t$
    $C \leftarrow C_e \parallel T$
    return $C$

$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in $\mathsf{G}_3$
    $C_e \leftarrow\$ \{0,1\}^c$
    $T \leftarrow\$ \{0,1\}^t$
    $C \leftarrow C_e \parallel T$
    return $C$
```

Fig. 12: Hybrid games $\mathsf{G}_i$ used to prove Theorem 3 (RKA-AE security of N2).

oracle rejects any ciphertext. We have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N2}}(\mathcal{A}) &= \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 0 \mid b = 0] - \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 0 \mid b = 1] \\ &= \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] \\ &= \sum_{i=1}^{3} \Pr[\mathcal{A}^{\mathsf{G}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_i} \Rightarrow 0]\,.\end{aligned}$$

To bound the first game hop, we construct an adversary $\mathcal{A}_{mac}$ playing $\mathsf{rkaSUF}$. It picks a key $K_e$ for the encryption scheme at random and runs adversary $\mathcal{A}$ answering queries as follows. Encryption queries of the form $(N, A, M, (\varphi_e, \varphi_m))$ are processed by computing $C_e \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M)$, querying $(N, A, C_e, \varphi_m)$ to $\mathsf{Tag}$ to obtain $T$, and sending $C \leftarrow C_e \parallel T$ back to $\mathcal{A}$. For decryption queries $(N, A, C, (\varphi_e, \varphi_m))$, $\mathcal{A}_{mac}$ parses $C$ as $C_e \parallel T$ queries $\mathsf{Ver}$ on $(N, A, C_e, T, \varphi_m)$ to get $V$. If $V = \top$, $\mathcal{A}_{mac}$ computes $M \leftarrow \mathtt{Dec}(\varphi_e(K_e), N, C_e)$ and sends $M$ back to $\mathcal{A}$. If $V = \bot$, $\mathcal{A}_{mac}$ sends $\bot$ back to $\mathcal{A}$. When $\mathcal{A}$ terminates by outputting a bit $b'$, $\mathcal{A}_{mac}$ outputs the same bit.

It holds that $\mathcal{A}_{mac}$ perfectly simulates game $\mathsf{G}_0$ and $\mathsf{G}_1$ based on its own challenge bit from game $\mathsf{rkaSUF}$ being 0 and 1, respectively. Let $(N, A, C, (\varphi_e, \varphi_m))$, with $C = C_e \parallel T$, be the decryption query that allows $\mathcal{A}$ to distinguish. For this query $\mathcal{A}_{mac}$ invokes its oracle $\mathsf{Ver}$ on $(N, A, C_e, T, \varphi_m)$. It remains to argue that this query is not forbidden, i.e., $\mathcal{A}_{mac}$ did not query its oracle $\mathsf{Tag}$ on $(N, A, C_e, \varphi_m)$ resulting in $T$. Assume for sake of contradiction, that $\mathcal{A}_{mac}$ did query $(N, A, C_e, \varphi_m)$ to $\mathsf{Tag}$. By construction, this means that $\mathcal{A}$ has made a query $(N, A, M, (\varphi'_e, \varphi_m))$ to $\mathsf{Enc}$, such that $C_e = \mathtt{Enc}(\varphi'_e(K_e, N, M))$ for which the response was $C_e \parallel T$. Note that both cases $\varphi'_e = \varphi_e$ and $\varphi'_e \neq \varphi_e$ are forbidden queries by $\mathcal{A}$. Thus we conclude with

$$\begin{aligned}
&\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] \\
&\leq \Pr[\mathcal{A}_{mac}^{\mathsf{rkaSUF}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{mac}^{\mathsf{rkaSUF}} \Rightarrow 0 \,|\, b = 1] \\
&\leq \mathbf{Adv}_{\Gamma}^{\mathsf{rkaSUF}}(\mathcal{A}_{mac}, \Phi_m)\,.
\end{aligned}$$

For the remaining to game hops, we omit the decryption oracle since any reduction merely needs to respond with $\perp$. To bound the advantage between game $\mathsf{G}_1$ and $\mathsf{G}_2$ we construct an adversary $\mathcal{A}_{prf}$ against $\mathtt{Tag}$. It picks a key $K_e$ for the encryption scheme and runs $\mathcal{A}$. For each query $(N, A, M, (\varphi_e, \varphi_m))$ that $\mathcal{A}$ makes to $\mathsf{Enc}$, $\mathcal{A}_{prf}$ locally computes $C_e \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M)$, and queries $(N, A, C_e, \varphi_m)$ to $\mathsf{F}$ to get $T$. Then it sends $C \leftarrow C_e \parallel T$ back to $\mathcal{A}$. Finally, $\mathcal{A}_{prf}$ outputs whatever $\mathcal{A}$ outputs.

It holds that $\mathcal{A}_{prf}$ perfectly simulates $\mathsf{G}_1$ for $\mathcal{A}$ it its own challenge oracle is initialised with $b = 0$. Likewise, $\mathcal{A}_{prf}$ perfectly simulates $\mathsf{G}_2$ if its own challenge oracle is initialised with $b = 1$. This yields

$$\begin{aligned}
&\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] \\
&\leq \Pr[\mathcal{A}_{prf}^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{prf}^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 1] \\
&\leq \mathbf{Adv}_{\mathtt{Tag}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}, \Phi_m)\,.
\end{aligned}$$

To bound the last game hop, we construct the following adversary $\mathcal{A}_{se}$ against $\Sigma$ playing $\mathsf{rkaIND}$. It runs $\mathcal{A}$ and answers its queries $(N, A, M, (\varphi_e, \varphi_m))$ to $\mathsf{Enc}$ as follows. It picks $T$ at random of appropriate length, queries its own oracle $\mathsf{Enc}$ on $(N, M, \varphi_e)$ to obtain $C_e$ and sends $C \leftarrow C_e \parallel T$ to $\mathcal{A}$. At the end, $\mathcal{A}_{se}$ outputs whatever $\mathcal{A}$ outputs.

If its own challenge bit $b$ equals 0, $\mathcal{A}_{se}$ simulates game $\mathsf{G}_2$ for $\mathcal{A}$, if the bit $b$ equals 1, it simulates game $\mathsf{G}_3$ for $\mathcal{A}$. Thus we get

$$\begin{aligned}
&\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] \\
&\leq \Pr[\mathcal{A}_{se}^{\mathsf{rkaIND}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{se}^{\mathsf{rkaIND}} \Rightarrow 0 \,|\, b = 1] \\
&\leq \mathbf{Adv}_{\Sigma}^{\mathsf{rkaIND}}(\mathcal{A}_{se}, \Phi_e)\,.
\end{aligned}$$

Collecting the bounds from the individual game hops proves the claim.

### B.2 Proof of Theorem 8

*Proof.* The theorem is proven via the games $\mathsf{G}_0$, $\mathsf{G}_1$, and $\mathsf{G}_2$ displayed in Fig. 13. Game $\mathsf{G}_0$ equals game $\mathsf{rkaIND}$ instantiated with $\Sigma$ and secret bit fixed to 0. In game $\mathsf{G}_1$, the output of $\mathtt{F}$ is chosen at random and in game $\mathsf{G}_2$ the output of $\mathtt{G}$ is chosen at random. While $\mathsf{G}_2$ is not exactly $\mathsf{rkaIND}$ with secret bit fixed to 1, it is easy to see that the games are perfectly indistinguishable as they output identically distributed ciphertexts. Hence, Claim 6 and 7 conclude the proof. It holds that

$$\begin{aligned}\mathbf{Adv}_{\Sigma}^{\mathsf{rkaIND}}(\mathcal{A}) &= \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_2}) \\ &\leq \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_1}) + \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) \\ &\leq \mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}) + q\,\mathbf{Adv}_{\mathtt{G}}^{\mathsf{PRG}}(\mathcal{A}_{prg})\,.\end{aligned}$$

Distinguishing games $\mathsf{G}_0$ and $\mathsf{G}_1$ is bound the the RKA-PRF security of the function $\mathtt{F}$.

**Claim 6** *For any $\Phi_e$-restricted RKA distinguisher $\mathcal{A}$ between game $\mathsf{G}_0$ and $\mathsf{G}_1$ defined in Fig. 13, there exists a $\Phi_e$-restricted RKA adversary $\mathcal{A}_{prf}$ such that*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_1}) \leq \mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf})\,.$$

*Proof.* We give an adversary $\mathcal{A}_{prf}$ who simulates $\mathsf{G}_0$ or $\mathsf{G}_1$ to win game $\mathsf{rkaPRF}$ with the help of $\mathcal{A}$. For any encryption query $(N, M, \varphi)$ by $\mathcal{A}$, $\mathcal{A}_{prf}$ forwards $(N, \varphi)$ to the oracle $\mathsf{F}$ of the security game $\mathsf{rkaPRF}$ to get $y$. Then, $\mathcal{A}_{prf}$ locally computes the ciphertext $C = \mathtt{G}(y) \oplus M$ sends it to $\mathcal{A}$. Whenever $\mathcal{A}$ outputs its guess, $\mathcal{A}_{prf}$ outputs the same. by construction, $\mathcal{A}_{prf}$ perfectly simulates game $\mathsf{G}_b$ for $\mathcal{A}$ where $b$ is the challenge bit in game $\mathsf{rkaPRF}$. Hence we have

$$\begin{aligned}&\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_1}) \\ &= \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] \\ &= \Pr[\mathcal{A}_{prf}^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{prf}^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 1] \\ &= \mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}, \Phi)\,.\end{aligned}$$

The advantage in distinguishing games $\mathsf{G}_1$ and $\mathsf{G}_2$ is bound by the security of the pseudorandom generator $\mathtt{G}$.

**Claim 7** *For any RKA-nonce-respecting distinguisher $\mathcal{A}$ between game $\mathsf{G}_1$ and $\mathsf{G}_2$ defined in Fig. 13, there exists an adversary $\mathcal{A}_{prg}$ such that*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) \leq q\,\mathbf{Adv}_{\mathtt{G}}^{\mathsf{PRG}}(\mathcal{A}_{prg})\,,$$

*where $q$ is the number of queries that $\mathcal{A}$ makes to oracle $\mathsf{Enc}$.*

*Proof.* The security game $\mathsf{PRG}$ provides only one PRG output, therefore the proof is done by $q+1$ hybrid games $\mathsf{H}_0, \ldots, \mathsf{H}_q$. In hybrid game $\mathsf{H}_i$, the first $i$ queries to $\mathsf{Enc}$ are answered as in game $\mathsf{G}_2$ while the remaining $q - i$ queries are answered as in game $\mathsf{G}_1$. Hence $\mathsf{H}_0$ equals $\mathsf{G}_1$ while $\mathsf{H}_q$ equals $\mathsf{G}_2$. Via a standard hybrid argument, we can construct an adversary $\mathcal{A}_{prg}$ that picks a query $i$ at random for which it will use its own challenge from the PRG security game to compute the ciphertext. The inputs of the PRG can be chosen uniformly at random since it is the output of the random function $\mathtt{F}'$ and each input of $\mathtt{F}'$ is only used once because $\mathcal{A}$ is RKA-nonce-respecting. This yields

$$\begin{aligned}\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) &= \mathbf{Adv}(\mathcal{A}^{\mathsf{H}_0}, \mathcal{A}^{\mathsf{H}_q}) \\ &\leq q\,\mathbf{Adv}_{\mathtt{G}}^{\mathsf{PRG}}(\mathcal{A}_{prg})\,.\end{aligned}$$

The last inequality concludes the proof.

```
Game $\mathsf{G}_i$
    $K \leftarrow_{\$} \mathcal{K}$
    $\mathtt{F}' \leftarrow_{\$} \mathsf{Func}(\mathcal{K}, \mathcal{N}, \{0,1\}^m)$
    $b' \leftarrow \mathcal{A}^{\mathsf{Enc}}()$

$\mathsf{Enc}(N, M, \varphi)$ in $\mathsf{G}_0$
    $C \leftarrow \mathtt{G}(\mathtt{F}(\varphi(K), N)) \oplus M$
    return $C$

$\mathsf{Enc}(N, M, \varphi)$ in $\mathsf{G}_1$
    $C \leftarrow \mathtt{G}(\mathtt{F}'(\varphi(K), N)) \oplus M$
    return $C$

$\mathsf{Enc}(N, M, \varphi)$ in $\mathsf{G}_2$
    $x \leftarrow_{\$} \{0,1\}^{|M|}$
    $C \leftarrow x \oplus M$
    return $C$
```

Fig. 13: Games used to prove the rkaIND security of Theorem 8.

## B.3 Proof of Theorem 9

```
Games $\mathsf{G}_0$, $\mathsf{G}_1$, $\mathsf{G}_2$, $\mathsf{G}_3$
    $K \leftarrow_{\$} \mathcal{K}$
    $\mathtt{F}' \leftarrow_{\$} \mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$
    $b' \leftarrow \mathcal{A}^{\mathsf{Tag},\mathsf{Ver}}()$
    return $(b' = b)$

$\mathsf{Ver}(M, T, \varphi)$ in $\mathsf{G}_2$, $\mathsf{G}_3$
    return $\perp$

$\mathsf{Ver}(M, T, \varphi)$ in $\mathsf{G}_0$
    $T' \leftarrow \mathtt{F}(\varphi(K), M)$
    return $(T' = T)$

$\mathsf{Ver}(M, T, \varphi)$ in $\mathsf{G}_1$
    $T' \leftarrow_{\$} \mathtt{F}'(\varphi(K), M)$
    return $(T' = T)$

$\mathsf{Tag}(M, \varphi)$ in $\mathsf{G}_0$, $\mathsf{G}_3$
    $T \leftarrow \mathtt{F}(\varphi(K), M)$
    return $T$

$\mathsf{Tag}(M, \varphi)$ in $\mathsf{G}_1$, $\mathsf{G}_2$
    $T \leftarrow_{\$} \mathtt{F}'(\varphi(K), M)$
    return $T$
```

Fig. 14: Hybrid games used in the proof of Theorem 9.

*Proof.* We prove the theorem using the hybrid games $\mathsf{G}_0$, $\mathsf{G}_1$, $\mathsf{G}_2$, and $\mathsf{G}_3$ displayed in Fig. 14. Game $\mathsf{G}_0$ is rkaSUF instantiated with $\mathtt{F}$ and secret bit $b = 0$. In game $\mathsf{G}_1$, both oracles Tag and Ver use a random function to generate the tags. In game $\mathsf{G}_2$, oracle Ver rejects any queried tag. In game $\mathsf{G}_3$, oracle Tag uses again $\mathtt{F}$ instead of choosing the tags at random, thus it corresponds to rkaSUF with secret bit $b = 1$. It holds that

$$\begin{aligned}
&\mathbf{Adv}_{\Gamma}^{\mathsf{rkaSUF}}(\mathcal{A}, \Phi) \\
&= \Pr[\mathcal{A}^{\mathsf{rkaSUF}} \Rightarrow 0 \mid b = 0] - \Pr[\mathcal{A}^{\mathsf{rkaSUF}} \Rightarrow 0 \mid b = 1] \\
&= \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] \\
&= \sum_{i=1}^{3} \left( \Pr[\mathcal{A}^{\mathsf{G}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_i} \Rightarrow 0] \right) .
\end{aligned}$$

We transform any adversary $\mathcal{A}$ that distinguishes between $\mathsf{G}_0$ and $\mathsf{G}_1$ into an adversary $\mathcal{A}_0$ against $\mathtt{F}$. For each query $(X, \varphi)$ to $\mathsf{Tag}$, $\mathcal{A}_0$ queries its oracle $\mathsf{F}$ on $(X, \varphi)$ and sends the response $T$ back to $\mathcal{A}$. For each query $(X, T, \varphi)$ to $\mathsf{Ver}$, $\mathcal{A}_0$ queries its oracle $\mathsf{F}$ on $(X, \varphi)$ to get $T'$. If $T = T'$ $\mathcal{A}_0$ sends $\top$ back to $\mathcal{A}$, otherwise, it sends $\bot$ back. When $\mathcal{A}$ eventually outputs a bit $b'$, $\mathcal{A}_0$ outputs the same.

It holds that $\mathcal{A}_0$ perfectly simulates game $\mathsf{G}_0$ and game $\mathsf{G}_1$ when its own challenge bit $b$, from game $\mathsf{PRF}$, is 0 and 1, respectively. We have

$$\begin{aligned}
&\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] \\
&\leq \Pr[\mathcal{A}_0^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_0^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 1] \\
&\leq \mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A}_0, \Phi)\,.
\end{aligned}$$

The advantage in distinguishing between $\mathsf{G}_1$ and $\mathsf{G}_2$ is a simple counting argument. The adversary can only distinguish if it guesses the output of the random function, in which case $\mathsf{Ver}$ returns $\top$ in $\mathsf{G}_1$ and $\bot$ in $\mathsf{G}_2$. Since $\mathcal{A}$ makes $q$ queries to $\mathsf{Ver}$ we get

$$\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] \leq \frac{q}{|\mathcal{Y}|}\,.$$

Distinguishing $\mathsf{G}_2$ and $\mathsf{G}_3$ essentially asks to distinguish where oracle $\mathsf{Tag}$ is implemented using $\mathtt{F}$ or a random function. We construct the following adversary $\mathcal{A}_2$. It returns $\bot$ for every query that $\mathcal{A}$ makes to $\mathsf{Ver}$. For queries $(X, \varphi)$ to $\mathsf{Tag}$, $\mathcal{A}_2$ forwards the query to its own oracle $\mathsf{F}$ to obtain $T$ which it sends to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit $b'$, $\mathcal{A}_2$ outputs $1 - b'$.

It holds that $\mathcal{A}_2$ perfectly simulates game $\mathsf{G}_2$ if its own challenge bit $b$ equals 1 while it simulates $\mathsf{G}_3$ if $b$ equals 0. Thus we have

$$\begin{aligned}
&\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] \\
&\leq \Pr[\mathcal{A}_2^{\mathsf{rkaPRF}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}_2^{\mathsf{rkaPRF}} \Rightarrow 1 \,|\, b = 0] \\
&\leq \mathbf{Adv}_{\mathtt{F}}^{\mathsf{rkaPRF}}(\mathcal{A}_2, \Phi)\,.
\end{aligned}$$

Collecting the bounds and defining $\mathcal{A}_{prf}$ to be the adversary with the higher advantage among $\mathcal{A}_0$ and $\mathcal{A}_2$ gives the desired result.