A preliminary version of this paper appears in the proceedings of CRYPTO 2011. This is the full version.

# Authenticated and Misuse-Resistant Encryption of Key-Dependent Data

Mihir Bellare[1] Sriram Keelveedhi[2]

**Abstract**

This paper provides a comprehensive treatment of the security of authenticated encryption (AE) in the presence of key-dependent data, considering the four variants of the goal arising from the choice of universal nonce or random nonce security and presence or absence of a header. We present attacks showing that universal-nonce security for key-dependent messages is impossible, as is security for key-dependent headers, not only ruling out security for three of the four variants but showing that currently standarized and used schemes (all these target universal nonce security in the presence of headers) fail to provide security for key-dependent data. To complete the picture we show that the final variant (random-nonce security in the presence of key-dependent messages but key-independent headers) is efficiently achievable. Rather than a single dedicated scheme, we present a RO-based transform RHtE that endows *any* AE scheme with this security, so that existing implementations may be easily upgraded to have the best possible seurity in the presence of key-dependent data. RHtE is cheap, software-friendly, and continues to provide security when the key is a password, a setting in which key-dependent data is particularly likely. We go on to give a key-dependent data treatment of the goal of misuse resistant AE. Implementations are provided and show that RHtE has small overhead.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@cs.ucsd.edu`. URL: `http://www.cs.ucsd.edu/users/mihir`. Supported in part by NSF grants CNS-0904380 and CCF-0915675.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `sriramkr@cs.ucsd.edu`. URL: `http://www.cs.ucsd.edu/users/skeelvee/`. Supported in part by NSF grants CNS-0904380 and CCF-0915675.

## Contents

## 1 Introduction

The key used by BitLocker to encrypt your disk may reside on the disk. The key under which a secure filesystem is encrypted may itself be stored in a file on the same system. The result is encryption of key-dependent data.

There is growing recognition that security of key-dependent data, first defined to connect cryptography to formal methods [17] and provide anonymous credentials [23], is a more direct and widespread concern for secure systems. The problem is particularly acute when keys are passwords, for many of us store our passwords on our systems and systems store password hashes. If nothing else, one cannot expect applications to ensure or certify that their data is *not* key-dependent, making security for key-dependent data essential for easy-to-use, robust and misuse-resistant cryptography.

This paper provides a comprehensive treatment of security for key-dependent data for the central practical goal of symmetric cryptography, namely authenticated encryption. For each important variant of the goal we either show that it is impossible to achieve security or present an efficient solution. Our attacks rule out security for in-use and standardized schemes in their prescribed and common modes while our solutions show how to adapt them in minimal ways to achieve the best achievable security. Let us now look at all this more closely.

Background. The standard IND-CPA and IND-CCA goals that our encryption schemes are proven to meet do not guarantee security when the message being encrypted depends on the key. (In the symmetric setting, we mean the single key used for both encryption and decryption.) Black, Rogaway and Shrimpton (BRS) [17] extend IND-CPA to allow key-dependent messages (KDMs). The adversary provides its encryption oracle with a function $\phi$, called a message-deriving function, that the game applies to the target key $K$ to get a message $M$, and the adversary is returned either an encryption of $M$ under $K$ or the encryption of $0^{|M|}$, and must be unable to tell which. (They, and we, actually consider a multi-key setting, but the single-key setting will simplify the current discussion.) They present a simple random-oracle (RO) model solution.

Post-BRS work has aimed mainly at showing existence of schemes secure against as large as possible a class of message deriving functions without random oracles [18, 35, 4, 20, 22, 19, 7, 24, 16, 3, 37]. The schemes suffer from one or more of the following: they are in the asymmetric setting while data encryption in practice is largely symmetric; they are too complex to consider usage; or security is provided for a limited, mathematical class of message-deriving functions which does not cover all key-dependencies in systems.

Backes, Pfitzmann and Scedrov (BPS) [6] define KDM-security for a basic form of authenticated encryption and and show that Encrypt-then-MAC [11] achieves it if the encryption scheme is KDM secure and the MAC is strongly unforgeable (remarkably, no KDM security is required from the MAC), resulting in RO model solutions via [17]. In this paper we will extend their treatment of AE in several directions.

Setting. Privacy without authenticity, meaning plain (IND-CPA) encryption, is of limited utility. The most important symmetric primitive in practice is authenticated encryption (AE), which provides both privacy and integrity. This is evidenced by numerous standards and high usage: CCM [49, 48] is in IEEE 802.11, IEEE 802.15.4, IPSEC ESP and IKEv2; GCM [38] is standardized by NIST as SP 800-38D; EAX [15] is in ANSI C12.22 and ISO/IEC 19772; OCB 2.0 [44, 46] is in ISO 19772. Consideration of KDM security for these standards is compelling and urgent but has not been done. We seek to fill this gap.

Symmetric encryption schemes take as input a nonce, also called an IV. Classically — [9] following [29]— this was chosen at random by the encrypter. We call this random-nonce security (`r`). Later schemes targeted universal-nonce security (`u`) [43, 45, 47] where security must hold even when the adversary provides the nonce, as long as no nonce is re-used. This is adopted by the above-mentioned standards.

|   | (ki, ki) | (kd, kd) | (ki, kd) | (kd, ki) |
|---|---|---|---|---|
| u | Yes | No | No | No |
| r | Yes | No | No | Yes |

Figure 1: Each of message and header may be key-dependent (kd) or key-independent (ki), leading to the four choices naming the columns. Security could be universal-nonce (u) or random-nonce (r), leading to the two choices naming the rows. For each of the 8 possibilities, we indicate whether security is possible (Yes, meaning a secure scheme exists) or impossible (No, meaning there is an attack that breaks *any* scheme in this category). The first column reflects known results when inputs are not key-dependent.

Besides key, nonce and message, modern AE schemes, including the above standards, take input a header, or associated data [43]. The scheme must provide integrity but *not* privacy of the header. Thus we must consider that not just the message, but also the header, could be key-dependent.

Abbreviate key-dependent by kd and key-independent by ki. With two choices for nonce type — $\mathtt{nt} \in \{\mathtt{u}, \mathtt{r}\}$— two for message type —$\mathtt{mt} \in \{\mathtt{kd}, \mathtt{ki}\}$— and two for header type —$\mathtt{ht} \in \{\mathtt{kd}, \mathtt{ki}\}$— we have 8 variants of AE. The form of AE treated by Backes, Pfitzmann and Scedrov [6] is the special case of $(\mathtt{nt}, \mathtt{mt}, \mathtt{ht}) = (\mathtt{r}, \mathtt{kd}, \mathtt{ki})$ in which the header is absent.

Definition. Our first contribution is a definition of security for AE under key-dependent inputs that captures all these 8 variants in a unified way. The encryption oracle takes functions $\phi_m, \phi_h$, and applies them to the key to get message and header respectively, and the adversary gets back either an encryption of these under the game-chosen target key, or a random string of the same length. The decryption oracle takes a ciphertext and, importantly, not a header but a function $\phi_h$ to derive it from the key, and either says whether or not decryption under the key is valid, or always says it is invalid. Varying the way nonces are treated and from what spaces $\phi_m, \phi_h$ are drawn yields the different variants of the notion. A definition of MACs for key-dependent messages emerges as the special case of empty messages.

On a real system, the data may be a complex function of the key, such as a compressed (zipped) version of file containing, amongst other things, the key, or an error-corrected version of the key. If the key is a password the system will store its hash that will be encrypted as part of the disk, so common password-hashing functions must be included as message-deriving functions. All this argues for not restricting the types of message-deriving or header-deriving functions, and indeed, following [17, 6], we allow any functions in this role. These functions are even allowed to call the RO, a source of challenges in proofs.

Underlying the above definition is a new one of the standard AE goal that simplifies that of [47] by having the decryption oracle turn into a verification oracle, returning, not the full decryption, but only whether it succeeded or not, along the lines of [11]. When data is key-independent, these and prior formulations [11, 36] are equivalent, but the difference is important with key-dependent data.

Impossibility results. We present an attack that shows that *no* AE scheme can achieve universal-nonce security for key-dependent data. (Regardless of whether or not the header is key-dependent.) This explains the "No" entries in the first row of Figure 1. The attack requires only that the nonce is predictable. Thus it applies even when the nonce is a counter, ruling out KDM security for counter-based AE schemes and showing that the standardized schemes (CCM, GCM, EAX, OCB) are all insecure for key-dependent messages in this case. The attack does not use the decryption oracle, so rules out even KDM universal-nonce CPA secure encryption. Thus, the universal-nonce security proven for the standardized schemes for key-independent messages fails to extend to key-dependent ones, demonstrating that security for key-dependent messages is a fundamentally different and stronger security requirement.

An attack aiming to show that no stateful scheme is KDM-CPA secure was described in [17] but the message-deriving functions execute a search and it is not clear how long this will take to terminate or whether it will even succeed. (In asymptotic terms, the attack is not proven to terminate in polynomial time.) Our attack extends theirs to use pairwise independent hash functions, based on which we prove that it achieves a constant advantage in a bounded (polynomial) amount of time. Interestingly, as a corollary of the bound proven on our *modified* attack, we are able to also prove a bound on the running time of the attack of [17], although it was not clear to us how to do this directly.

We also present an attack that shows that *no* AE scheme can achieve security for key-dependent headers. (Even for random, rather than universal, nonce security, and even for key-independent messages.) This explains the "No" entries in columns 2 and 3 of Figure 1. This rules out security of the standardized schemes even with random nonces in a setting where headers may be key-dependent.

One might consider this trivial with the following reasoning: "Since the header is not kept private, the adversary sees it, and if it is key-dependent, it could for example just be the key, effectively giving the adversary the key." The fallacy is the assumption that the adversary sees the header. In our model, it is given a ciphertext but not directly given the header on which the ciphertext depends. This choice of model is not arbitrary but reflects applications, where a key-dependent header is present on the encrypting and decrypting systems (which may be the same system) but not visible to the adversary. Instead, the attack exploits the ability of the adversary to test validity of ciphertexts with implicitly specified headers.

$\mathsf{RHtE}$. We turn to achieving security in the only viable, but still important setting, namely $(\mathtt{nt}, \mathtt{mt}, \mathtt{ht}) = (\mathtt{r}, \mathtt{kd}, \mathtt{ki})$. As background, recall that to achieve KDM-CPA security, BRS [17] encrypt message $M$ by picking $R$ at random and returning $H(K\|R)\oplus M)$ where $H$ is a RO returning $|M|$ bits. (Here and below, it is assumed the decryptor and adversary also get the nonce $R$, but it is not formally part of the ciphertext.) We note that this is easily extended to achieve $(\mathtt{r}, \mathtt{kd}, \mathtt{ki})$-AE security. To encrypt header $H$ and message $M$ under key $K$, pick $R$ at random and return $(C, T)$ where $C = H_1(K\|R)\oplus M$ and $T = H_2(K\|R, H, C)$ and $H_1, H_2$ are ROs.

Randomized Hash then Encrypt ($\mathsf{RHtE}$) is more practical. Unlike the above, it is not a dedicated scheme but rather transforms a standard (secure only for key-independent data) base AE scheme into a $(\mathtt{r}, \mathtt{kd}, \mathtt{ki})$-secure AE scheme. $\mathsf{RHtE}$, given key $L$ and randomness $R$, derives subkey $K = H(R\|L)$ via RO $H$ and then runs the base scheme with key $K$ on the header and message to get the ciphertext $C$. Only one-time security of the base scheme is required, so it could even be deterministic. The software changes are non-intrusive since the code of the base scheme is used unchanged. Thus $\mathsf{RHtE}$ can easily be put on top of existing standards like CCM, GCM, EAX, OCB to add security in the presence of key-dependent messages. As long as these base schemes transmit their nonce, $\mathsf{RHtE}$ has *zero overhead in bandwidth* because it can use the base scheme with some fixed, known nonce and use the nonce space for $R$. (It is okay to re-use the base-scheme nonce because this scheme is only required to be one-time secure. Its key is changing with every encryption.) The computational overhead of $\mathsf{RHtE}$ is independent of the lengths of header and message and hence becomes negligible as these get longer.

The proof of security is surprisingly involved due to a combination of three factors. First is that the message-deriving functions are allowed to call the RO. Second, while the BRS scheme and its extension noted above are purely information theoretic, the security of $\mathsf{RHtE}$ is computational due to the base scheme, and must be proven by reduction. Third, unlike BRS, we must deal with decryption queries. To handle all this we will need to invoke the security of the base scheme in multiple, inter-related ways, leading to a proof with two, interleaved hybrids that go in opposite directions.

To illustrate the issues, let $L$ be the key and let $\mathcal{E}$ be the (deterministic) encryption function of the base scheme. Let $\phi_1, \ldots, \phi_{\mathsf{q_e}}$ be the message-deriving functions in $A$'s encryption queries. We begin with a natural hybrid in which the key $K = H(R_g\|L)$ underlying the $g$-th query, for random $g$, plays the role of a key for the base scheme. The reduction to the security of the latter fails if $A$ queries $R_g\|L$ to the RO. We must consider that it can do this indirectly, meaning the query is made via a message-deriving function, or directly. But once a reply is provided to the $g$-th query, $A$ gets $R_g$. But $\phi_i$ is given $L$ as input so *we cannot avoid* it querying $R_g\|L$ to the RO for $i > g$. We handle this by having the hybrid move from real to random replies rather than the other way round, so that $\phi_i$ does not even have to be computed by the reduction when $i > g$. One would imagine that $A$ cannot make the bad RO query directly because it does not know $L$. The subtle point is that the truth of this relies on the assumed security of the base scheme and must itself be proved by reduction. This reduction involves another hybrid that, to avoid the same issue arising in another place, goes in the opposite direction, first random then real. The second hybrid has the peculiar feature that the games in its constituent steps are differently weighted. On top of all this we must deal with decryption queries which are not present for BRS.

Some indication of the complexity of the proof is provided by the fact that the bound we finally achieve in Theorem 4.1 is weaker than we would like. It is an interesting open problem to either prove a better bound for $\mathsf{RHtE}$ or provide an alternative scheme with such a bound.

Extensions. In filesystem encryption, as with most applications, security is likely to stem from your password $\mathsf{pw}$. The system stores a hash $\overline{\mathsf{pw}} = h(\mathsf{pw})$ of it to authenticate you and an AE scheme must then encrypt or decrypt using $\mathsf{pw}$. Key dependent data is now an even greater concern. One reason is that users tend to write their passwords in files in their filesystems. The other reasons is that $\overline{\mathsf{pw}}$ is a function of $\mathsf{pw}$ that must be stored on the system and thus will be encrypted with disk encryption. To address this, we show that $\mathsf{RHtE}$ is secure even when its starting key $L$ is a password as long as the latter is drawn from a space that, asymptotically, has super-logarithmic min-entropy.

The security discussed so far relies crucially on using fresh randomness with each encryption. This is fine in theory but in real systems, failures of random-number generation (RNG) due to poorly gathered entropy or bugs are all too common and have lead to major security violations [28, 32, 21, 41, 40, 1, 50, 26]. Simply asking that system designers get their RNGs “right” is unrealistic. Misuse-resistant [17] or hedged [8] encryption take a different approach, mitigating the damage caused by RNG failures by providing as much security as possible when randomness fails.

We extend this to the key-dependent data setting in Section 5. A misuse resistant AE scheme for key-dependent data provides two things. First, it must continue to provide $(\mathtt{r}, \mathtt{kd}, \mathtt{kd})$-security when the nonce is random. Second, even for nonces that are entirely adversary controlled (and may repeat), the scheme must meet a second condition that we define to capture its providing the security of deterministic AE in the presence of key-dependent data. In the latter case it is impossible to protect against certain classes of message-deriving functions. We show however that $\mathsf{RHtE}$ provides security against any class of functions satisfying the output-unpredictability and collision-resistance conditions of [10]. This is a fairly significant class, containing functions of pragmatic interest.

Implementation. We implemented $\mathsf{RHtE}$ for base schemes CCM, EAX and GCM, with SHA256 instantiating the RO. The results, provided in Section 6, show for example that with CCM the slowdown is 11% for 5KB messages and only 1% for 50KB messages. The implementations use the crypto++ library on a Intel Core i5 M460 CPU running at 2.53 GHz with code compiled using g++ -O3 for data sizes small enough to fit in the level 2 cache.

Related work. The issue (key-dependent messages) was pointed out as early as Goldwasser and Micali [29], and asymmetric encryption of decryption keys was treated by Camenisch and Lysyanskaya [23], but a full treatment of key-dependent message (KDM) encryption awaited BRS [17], who provided RO model KDM-CPA secure schemes. Researchers then asked for what classes of message-deriving functions one could achieve KDM security in the standard model, providing results for both symmetric and asymmetric encryption under different assumptions [18, 35, 4, 20, 22, 19, 7, 24, 16, 3, 37]. On the more practical side, Backes, Dürmuth and Unruh [5] show that RSA-OAEP [12, 27] is KDM-secure in the RO model. Backes, Pfitzmann and Scedrov [6] treat active attacks and provide and relate a number of different notions of security.

By showing that IND-CPA security does not even imply security for the encryption of 2-cycles, Acar, Belenkiy, Bellare and Cash [2] and Green and Hohenberger [31] settled a basic question in this area and showed that achieving even weak KDM-security *requires* new schemes, validating previous efforts in that direction. Acar et. al. [2] also connect KDM secure encryption to cryptographic agility. Haitner and Holenstein [33] study the difficulty of proving KDM security by blackbox reduction to standard primitives.

Halevi and Krawczyk [34] consider blockciphers under key-dependent inputs. Muñiz and Steinwandt [39] study KDM secure signatures. González, in an unpublished thesis [30], studies KDM secure MACs.

Motivated by attacks on SSH, Paterson and Watson [42] consider notions of security (in the standard `ki`-data context) which allow the attacker to interact in a byte-by-byte manner with the decryption oracle. Our treatment does not encompass such attacks, and extending the model of [42] to allow key-dependent data is an interesting direction for future work.

## 2 Definitions

We provide a unified definition for universal and random nonce AE security and then extend this to definitions of universal and random nonce AE security in the presence of key-dependent messages and headers.

NOTATION. If $S$ is a (finite) set then $s \leftarrow\$ S$ denotes the operation of picking $s$ from $S$ at random and $|S|$ is the size of $S$. Read the term "efficient" as meaning "polynomial-time" in the natural asymptotic extension of our concrete framework. If $x$ is a string then $|x|$ denotes its length and $x[i]$ denotes its $i$-th bit. The empty string is denoted $\varepsilon$. By $a_1\| \ldots \|a_n$, we denote the concatenation of strings $a_1, \ldots, a_n$. Unless otherwise indicated, an algorithm may be randomized. We denote by $y \leftarrow\$ A(x_1, x_2, \ldots)$ the operation of running $A$ on the indicated inputs and fresh random coins to get an output denoted $y$. For integers $k, w$ let $\mathsf{Fun}(k, w)$ be the set of all functions $\phi$ for which there exists an integer $\mathsf{ol}(\phi)$, called the output length of $\phi$, such that $\phi$: $(\{0,1\}^k)^w \to \{0,1\}^{\mathsf{ol}(\phi)}$. Input-deriving functions will be drawn from this set. Let $\mathsf{Cns}(k, w)$ be the subset of $\mathsf{Fun}(k, w)$ consisting of constant functions, restricting attention to which drops KDI (key-dependent input) notions of security down to their standard, non-KDI counterparts.

GAMES. Some of our definitions and proofs are expressed via code-based games [14]. Such a game —see Figure 2 for an example— consists of procedures that respond to adversary oracle queries. In an execution of game $G$ with an adversary $A$, the latter must make exactly one INITIALIZE query, this being its first oracle query, and exactly one FINALIZE query, this being its last oracle query. In between, it can query other game procedures. Each time it makes a query, the corresponding game procedure executes, and what it returns, if anything, is the response to $A$'s query. The output of FINALIZE, denoted $G^A$, is called the output of the game, and we let "$\mathrm{G}^A \Rightarrow d$" denote the event that this game output takes value $d$. If FINALIZE is absent it is understood to be the identity function, so the game output is the adversary output. Boolean flags are assumed initialized to $\mathsf{false}$ and $\mathsf{BAD}(\mathrm{G}^A)$ is the event that the execution of game G with adversary $A$ sets flag $\mathsf{bad}$ to $\mathsf{true}$. The running time of an adversary by convention is the worst case time for the execution of the adversary with the game defining its security, so that the time of the called game procedures is included.

AE SYNTAX. A symmetric encryption scheme $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is specified by a key generation algorithm $\mathcal{K}$ that returns $k$-bit strings, an encryption function $\mathcal{E}$: $\{0,1\}^k \times \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ and decryption function $\mathcal{D}$: $\{0,1\}^k \times \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$. Inputs to $\mathcal{E}$ are key, nonce, header and message, and output is a ciphertext. Inputs to $\mathcal{D}$ are key, nonce, header and ciphertext, and output is a message or $\perp$. We refer to $k$ as the keylength and $n$ as the noncelength. Both $\mathcal{E}$ and $\mathcal{D}$ are deterministic, it being the way nonces are handled by the games defining security that will distinguish universal-nonce and random-nonce security. We require that $\mathcal{D}(K, N, H, \mathcal{E}(K, N, H, M)) = M$ for all values of the inputs shown. We also require that $\mathcal{E}$ is length respecting in the sense that the length of a ciphertext depends only on the length of the message and header. Formally, there is a function $\mathsf{cl}(\cdot, \cdot)$ called the ciphertextlength such that $|C| = \mathsf{cl}(|M|, |H|)$ for any $C$ that may be output by $\mathcal{E}(\cdot, \cdot, H, M)$.

As in [45, 47], $\mathcal{D}$ takes the nonce and header as an input. (In this view, the ciphertext in standard counter-mode encryption does not incude the counter. It is up to the application to transmit nonce

```
proc Initialize  // $\mathrm{KIAE}_{\mathsf{SE},\mathtt{nt}}$
$K \leftarrow\$ \mathcal{K}$
$S \leftarrow \emptyset$
$b \leftarrow\$ \{0,1\}$

proc Enc$(N, H, M)$  // $\mathrm{KIAE}_{\mathsf{SE},\mathtt{nt}}$
If $(\mathtt{nt} = \mathtt{r})$ then $N \leftarrow\$ \{0,1\}^n$
If $(b = 1)$ then $C \leftarrow \mathcal{E}(K, N, H, M)$
Else $c \leftarrow \mathsf{cl}(|M|, |H|)$ ; $C \leftarrow\$ \{0,1\}^c$
$S \leftarrow S \cup \{(N, H, C)\}$
Return $(N, C)$

proc Dec$(N, H, C)$  // $\mathrm{KIAE}_{\mathsf{SE},\mathtt{nt}}$
If $(N, H, C) \in S$ then return $\perp$
If $(b = 1)$ then $M \leftarrow \mathcal{D}(K, H, N, C)$
Else $M \leftarrow \perp$
If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
Return $V$

proc Finalize$(b')$  // $\mathrm{KIAE}_{\mathsf{SE},\mathtt{nt}}$
Return $(b' = b)$
```

```
proc Initialize$(w)$  // $\mathrm{KDAE}_{\mathsf{SE},\mathtt{nt}}$
For $j = 1, \ldots, w$ do
    $K_j \leftarrow\$ \mathcal{K}$ ; $S_j \leftarrow \emptyset$
$b \leftarrow\$ \{0,1\}$

proc Enc$(j, N, \phi_h, \phi_m)$  // $\mathrm{KDAE}_{\mathsf{SE},\mathtt{nt}}$
$M \leftarrow \phi_m(K_1, \ldots, K_w)$ ; $H \leftarrow \phi_h(K_1, \ldots, K_w)$
If $(\mathtt{nt} = \mathtt{r})$ then $N \leftarrow\$ \{0,1\}^n$
If $(b = 1)$ then $C \leftarrow \mathcal{E}(K_j, N, H, M)$
Else $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi_m), \mathsf{ol}(\phi_h))$ ; $C \leftarrow\$ \{0,1\}^c$
$S_j \leftarrow S_j \cup \{(N, H, C)\}$
Return $(N, C)$

proc Dec$(j, N, \phi_h, C)$  // $\mathrm{KDAE}_{\mathsf{SE},\mathtt{nt}}$
$H \leftarrow \phi_h(K_1, \ldots, K_w)$
If $(N, H, C) \in S_j$ then return $\perp$
If $(b = 1)$ then $M \leftarrow \mathcal{D}(K_j, N, H, C)$
Else $M \leftarrow \perp$
If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
Return $V$

proc Finalize$(b')$  // $\mathrm{KDAE}_{\mathsf{SE},\mathtt{nt}}$
Return $(b' = b)$
```

Figure 2: On the left is game $\mathrm{KIAE}_{\mathsf{SE},\mathtt{nt}}$ defining AE-security of encryption scheme $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where $\mathtt{nt} \in \{\mathtt{u}, \mathtt{r}\}$ indicates universal or random nonce. On the right is game $\mathrm{KDAE}_{\mathsf{SE},w,\mathtt{nt}}$ defining KDI AE-security of $\mathsf{SE}$.

and header if necessary, so the "ciphertext" in practice may be more than the output of $\mathcal{E}$, but in many settings the receiver gets nonce and header in out-of-band ways.) But our treatment differs from standard ones [9] in that the nonce must be explicitly provided to $\mathcal{D}$ even when it is random. This means that, for randomized schemes, we are limited to ones that make the randomness public, but this is typically true. The restriction is only to compact and unify the presentation. Otherwise we would have needed separate games to treat universal and random nonce security.

AE SECURITY. We now define standard (neither message nor header is key-dependent) AE security for $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Consider game $\mathrm{KIAE}_{\mathsf{SE},\mathtt{nt}}$ shown on the left side of Figure 2. Define the advantage of adversary $A$ via $\mathbf{Adv}^{\mathrm{ae}\text{-}\mathtt{nt}}_{\mathsf{SE}}(A) = 2\Pr[\mathrm{KIAE}^{A}_{\mathsf{SE},\mathtt{nt}} \Rightarrow \mathsf{true}] - 1$. When $\mathtt{nt} = \mathtt{u}$ the definition captures what we call universal-nonce security. (It is simply called nonce-based security in [45, 43, 47].) It is understood that in this case we only consider $A$ that is *unique-nonce*, meaning we have $N \neq N'$ for any two Enc queries $N, H, M$ and $N', H', M'$. Security is thus required even for adversary-chosen nonces as long as no nonce is used for more than one encryption. When $\mathtt{nt} = \mathtt{r}$, the adversary-provided nonce in Enc is ignored, a random value being substituted by the game, and we have random-nonce security, in the classical spirit of randomized encryption [29, 9]. The nonce returned by Enc is redundant in the $\mathtt{u}$ case but needed in the $\mathtt{r}$ case and thus always returned for uniformity.

Historically the first definitions of security for AE had separate privacy (IND-CPA) and integrity (INT-CTXT) requirements [11, 36, 13]. Our version is a blend of the single-game formulation of [47] and INT-CTXT. Privacy is in the strong sense of indistinguishability from random, meaning ciphertexts are indistinguishable from random strings, which implies the more common LR-style [9] privacy, namely that ciphertexts of different messages are indistinguishable from each other. (A subtle point is that the length-respecting property assumed of $\mathcal{E}$ is important for this implication.) The integrity is in the fact that the adversary can't create new ciphertexts with non-$\perp$ decryptions. ("New" means not output by Enc.) Unlike [47], oracle Dec does not return decryptions but only whether or not they succeed. This simpler version is nonetheless equivalent to the original. IND-CCA is implied by this definition of AE [11, 43].

KDI security of AE. We now extend the above along the lines of [17, 6] to provide our definition of security for AE in the presence of key-dependent inputs, considering both key-dependent messages and key-dependent headers. Consider game $\mathrm{KDAE}_{\mathsf{SE},\mathtt{nt}}$ shown on the right side of Figure 2. Define the advantage of adversary $A$ via $\mathbf{Adv}^{\mathrm{ae}\text{-}\mathtt{nt}}_{\mathsf{SE}}(A) = 2\Pr[\mathrm{KDAE}^{A}_{\mathsf{SE},\mathtt{nt}} \Rightarrow \mathsf{true}] - 1$. The argument $w$ to Initialize is the number of keys; arguments $\phi_m, \phi_h$ (message and header deriving functions, respectively) in the Enc, Dec queries must be functions in $\mathsf{Fun}(k, w)$; $\mathsf{ol}(\phi)$ is the output length of $\phi \in \mathsf{Fun}(k, w)$; and $\mathsf{cl}$ is the ciphertext length of $\mathsf{SE}$. When $\mathtt{nt} = \mathtt{u}$ the definition again captures universal-nonce security. That $A$ is unique-nonce (always assumed in this case) now means that for each $j \in [1..w]$ we have $N \neq N'$ for any two Enc queries $j, N, \phi_m, \phi_h$ and $j, N', \phi'_m, \phi'_h$. When $\mathtt{nt} = \mathtt{r}$ we have random-nonce security.

Messages could be key-dependent or not, and so could headers, giving rise to four settings of interest. These are best captured by considering different classes of adversaries. For $\Phi_m, \Phi_h \subseteq \mathsf{Fun}(k, w)$ let $\mathcal{A}[\Phi_m, \Phi_h]$ be the class of all adversaries $A$ for which $\phi_m$ in $A$'s Enc queries is in $\Phi_m$ and $\phi_h$ in its Enc, Dec queries is in $\Phi_h$. Let $\mathcal{A}[\mathtt{mt}, \mathtt{ht}] = \mathcal{A}[\Phi_m, \Phi_h]$ where the values of $(\Phi_m, \Phi_h)$ corresponding to $(\mathtt{mt}, \mathtt{ht}) = (\mathtt{kd}, \mathtt{kd}), (\mathtt{kd}, \mathtt{ki}), (\mathtt{ki}, \mathtt{kd}), (\mathtt{ki}, \mathtt{ki})$ are, respectively, $(\mathsf{Fun}(k, w), \mathsf{Fun}(k, w))$, $(\mathsf{Fun}(k, w), \mathsf{Cns}(k, w))$, $(\mathsf{Cns}(k, w), \mathsf{Fun}(k, w))$, $(\mathsf{Cns}(k, w), \mathsf{Cns}(k, w))$. Say that $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is $(\mathtt{nt}, \mathtt{mt}, \mathtt{ht})$-AE secure if $\mathbf{Adv}^{\mathrm{ae}\text{-}\mathtt{nt}}_{\mathsf{SE}}(A)$ is negligible for all efficient $A \in \mathcal{A}[\mathtt{mt}, \mathtt{ht}]$.

Now that the header may not be known to the adversary in a Dec query, it does not know in advance whether or not $(H, N, C) \in S_j$ and it deserves to know whether rejection took place due to this or due to unsuccessful decryption. This why we do not return $\perp$ for both but rather $\perp$ for one and 0 for the other. It was to disambiguate these that we found it convenient to modify the starting definition of AE. The issue is crucial when considering security with key-dependent headers.

In the RO model there is an additional procedure Hash representing the RO. As usual it may be invoked by the scheme algorithms and the adversary, but, importantly, also by the input-deriving functions $\phi_m, \phi_h$.

For input-deriving functions to be adversary queries it is assumed they are encoded in some way. Recall that, as per our convention, the running time of $A$ is that of the execution of $A$ with the game, so $A$ pays in run time if it uses functions whose description or evaluation time is too long. In asymptotic terms, $A$ is restricted to polynomial-time computable input-deriving functions, and their description could be set to the Turing-machine that computes them.

Passwords as keys. The key-generation algorithm $\mathcal{K}$ in our syntax $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ does not have to output random $k$-bit strings but could induce an arbitrary distribution, allowing us to capture passwords. The metric of interest in this case is the min-entropy $\mathbf{H}_\infty(\mathcal{K}) = -\log_2(\mathbf{GP}(\mathcal{K}))$, where the guessing probability $\mathbf{GP}(\mathcal{K})$ is defined as the maximum, over all $k$-bit strings $K$, of the probability that $K' = K$ when $K' \leftarrow_{\$} \mathcal{K}$. We aim to provide security as long as the min-entropy of the key-generator is not too small.

Providing security when keys are passwords is crucial because key-dependent data is more natural and prevalent in this case. In practice, our keys are largely passwords. They may be stored on disk. Their hashes are stored on the disk by the system.

## 3 Impossibility Results

We rule out universal-nonce security for key-dependent messages as well as security for key-dependent headers.

## 3.1 Universal-nonce insecurity

Standardized schemes all achieve universal-nonce security for ki-messages. This is convenient because an application-setting often provides for free something that can play the role of a nonce, like a counter. It also increases resistance to misuse. We would like to maintain this type of security in the presence of key-dependent data. Unfortunately we show that this is impossible. We show that no scheme is $(\mathtt{u}, \mathtt{kd}, \mathtt{ki})$-AE secure:

**Proposition 3.1** *Let* $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an encryption scheme. Then there is an efficient adversary* $A \in \mathcal{A}[\mathtt{kd}, \mathtt{ki}]$ *such that* $\mathbf{Adv}^{\text{ae-u}}_{\mathsf{SE}}(A) \geq 1/4$.

As the proof of the above will show, the attack we present is strong in that the adversary does not just distinguish real from random encryptions but recovers the key. (A simpler attack is possible if we only want to distinguish rather than recover the key.) Also the attack works even when the nonce is a counter rather than adversary controlled. And since the adversary does not use the decryption oracle we rule out even KDM-CPA security.

We begin with some background and an overview, then prove Proposition 3.1, and finally show how to apply an underlying lemma to provide the first analysis of an attack in BRS [17].

Background and overview. BRS [17, Section 6] suggest an attack aimed at showing that no stateful symmetric encryption scheme is KDM-secure. For the purpose of our discussion we adapt it to an attack on universal-nonce security of an AE scheme $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Let $k$ be the keylength of the scheme. We will use messages of length $m$. Let $c$ denote the length of the resulting ciphertexts. Let $H_{\text{ip}}(V, C) = V[1]C[1] + \cdots + V[c]C[c] \bmod 2$ denote the inner product modulo two of $c$-bit strings $V, C$. Let $\phi_{V,i}$ denote the message-deriving function that on input a key $K$ returns the first $m$-bit message $M$ such that $H_{\text{ip}}(V, \mathcal{E}(K, i, \varepsilon, M)) = K[i]$, or $0^m$ if there is no such message. (Here we use $i$ as the nonce and $\varepsilon$ as the header.) The adversary can pick $V$ (BRS do not say how, but the natural choice is at random), query $\phi_{V,i}$ to get $(i, C)$, and then recover $K[i]$ as $H_{\text{ip}}(V, C)$, repeating for $i = 1, \ldots, k$ to get $K$.

The difficulty is that $\phi_{V,i}$ must search the message space until it finds a message satisfying the condition, and it is unclear how long this will take. In asymptotic terms, this means there is no proof that the attack runs in polynomial time, meaning is a legitimate attack at all. This issue does not appear to be recognized by BRS, who provide no analysis or formal claims relating to the attack.

In order to have a polynomial time attack where the key-recovery probability is, say, a constant, one would need to show that there is a polynomial number $l$ of trials in which the failure probability to recover a particular bit $K[i]$ of the key is $O(1/k)$. (A union bound will then give the desired result.) We did not see a direct way to show this. Certainly, for a particular $i$, the probability that the first message $M$ fails to satisfy $H_{\text{ip}}(V, \mathcal{E}(K, i, \varepsilon, M)) = K[i]$ is at most $1/2$, but it is not clear what is the failure probability in multiple trials because they all use the same $V$. The first thought that comes to mind is to modify the attack so that $\phi_{V_1,\ldots,V_l,i}$ now depends on a sequence $V_1, \ldots, V_l$ of strings, chosen independently at random by the adversary. On input the key $K$, the function computes the smallest $j$ such that $H_{\text{ip}}(V_j, \mathcal{E}(K, i, \varepsilon, M_j)) = K[i]$, where $M_1, M_2, \ldots, M_l$ is a fixed sequence of messages, and returns $M_j$. Although one can prove that this "successful" $j$ is quickly found, the attack fails to work, since, to recover $K[i] = H_{\text{ip}}(V_j, C)$ from the ciphertext $C = \mathcal{E}(K, i, \varepsilon, M_j)$, the adversary needs to know $j$, and it is not clear how the ciphertext is to "communicate" the value of $j$ to the adversary.

We propose a different modification, namely to replace the inner product function with a family $H\colon \{0,1\}^s \times \{0,1\}^c \to \{0,1\}$ of pairwise independent functions. The message-deriving function $\phi_{S,i}$, on input $K$, will now search for $M$ such that $H(S, \mathcal{E}(K, i, \varepsilon, M)) = K[i]$. The adversary can pick $S$ at random, query $\phi_{S,i}$ to get $(i, C)$, and then recover $K[i]$ as $H(S, C)$, repeating for $i = 1, \ldots, k$ to get $K$. We will prove that $O(k)$ trials suffice for the search to have failure probability at most $O(1/k)$ for each $i$, and thus that the adversary gets a constant advantage in a linear number of trials.

This strategy can be instantiated by the pairwise independent family of functions $H\colon \{0,1\}^{c+1} \times \{0,1\}^c \to \{0,1\}$ defined by $H(S, C) = H_{\text{ip}}(S[1] \ldots S[c], C) + S[c+1] \bmod 2$ to get a concrete attack that is only a slight modification of the BRS one but is proven to work. Given this, the question of whether the original attack can be proven to work is perhaps moot, but we find it interesting for historical reasons. Our results would not at first appear to help to answer this because the inner product function is not pairwise independent. (For example, $0^c$ is mapped to 0 by all functions in the family.) But curiously, as a corollary of our proof that the attack works for the *particular* family $H$ we just defined, we get a proof that the BRS attack works as well. This is because we show that the attack using $H$ works for an overwhelming fraction of functions from $H$, and thus, with sufficient probability, even for functions drawn only from the subspace of inner-product functions. Let us now proceed to the details.

Attack and analysis. We begin with a general lemma.

**Lemma 3.2** *Let* $H\colon \{0,1\}^s \times \{0,1\}^c \to \{0,1\}$ *be a family of pairwise independent hash functions. Let* $C_1, \ldots, C_l \in \{0,1\}^c$ *be distinct and let* $T \in \{0,1\}$*. Then*

$$\Pr\left[\,\forall j \,:\, H(S, C_j) \neq T\,\right] \;\leq\; \frac{1}{l}$$

*where the probability is over a random choice of* $S$ *from* $\{0,1\}^s$*.*

**Proof of Lemma 3.2:** For each $j \in \{1, \ldots, l\}$ define $X_j\colon \{0,1\}^s \to \{0,1\}$ to take value 1 on input $S$ if $H(S, C_j) = T$ and 0 otherwise. Regard $X_1, \ldots, X_j$ as random variables over the random choice of $S$ from $\{0,1\}^s$. Let $X = X_1 + \cdots + X_l$ and let $\mu = \mathbf{E}[X]$. By Chebyshev's inequality, the probability above is

$$\Pr\left[\,X = 0\,\right] \;\leq\; \Pr\left[\,|X - \mu| \geq \mu\,\right] \;\leq\; \frac{\mathbf{Var}[X]}{\mu^2}\ .$$

Since $H$ is pairwise independent, so are $X_1, \ldots, X_l$ and hence $\mathbf{Var}[X] = \mathbf{Var}[X_1] + \cdots + \mathbf{Var}[X_l]$. But for each $j$ we have $\mathbf{E}[X_j] = 1/2$ and $\mathbf{Var}[X_j] = 1/4$, so $\mu = l/2$ and $\mathbf{Var}[X] = l/4$. Thus the above is at most $(l/4)/(l/2)^2 = 1/l$ as desired. ▌

We now use this to prove Proposition 3.1.

**Proof of Proposition 3.1:** Let $k$ be the keylength, $n$ the noncelength and $\mathsf{cl}$ the ciphertextlength of $\mathsf{SE}$. Let $l = 4k$. Let $\mathsf{NumToStr}(j)$ denote a representation of integer $j \in \{0, \ldots, l\}$ as a string of length exactly $m = \lceil \log_2(l+1) \rceil$ bits. Let $H\colon \{0,1\}^s \times \{0,1\}^{\mathsf{cl}(m,0)} \to \{0,1\}$ denote a family of pairwise independent hash functions with $s$-bit keys. We construct an adversary $B$ that recovers the target key with probability at least 3/4 when playing the real game, meaning game $\mathrm{KDAE}_{\mathsf{SE},\mathsf{u}}$ with challenge bit $b = 1$. From $B$ it is easy to build $A$ achieving advantage at least 1/4. Below we depict $B$ and also define the message-deriving functions it uses. Nonces are given as integers and assumed encoded as $n$-bit strings:

```
Adversary $B$
Initialize(1)
For $j = 1, \ldots, l$ do
    $\boldsymbol{m}[j] \leftarrow \mathsf{NumToStr}(j)$
$S \leftarrow\!\!\$\ \{0,1\}^s$
For $i = 1, \ldots, k$ do
    $(i, C) \leftarrow\!\!\$\ \mathrm{Enc}(1, i, \phi_\varepsilon, \phi_{\boldsymbol{m},S,i})$ ; $L[i] \leftarrow H(S, C)$
Return $L$
```

```
Function $\phi_{\boldsymbol{m},S,i}(K)$
$M \leftarrow \mathsf{NumToStr}(0)$
For $j = 1, \ldots, l$ do
    $C_j \leftarrow \mathcal{E}(K, i, \varepsilon, \boldsymbol{m}[j])$
    If $H(S, C_j) = K[i]$ then
        $M \leftarrow \boldsymbol{m}[j]$
Return $M$
```

Above $\boldsymbol{m}$ is a $l$-vector over $\{0,1\}^m$ and $\phi_\varepsilon$ is the constant function that returns the empty string on every input. In its first step, $B$ initializes the game to play with $w = 1$, meaning a single target key. Function $\phi_{\boldsymbol{m},S,i}(K)$ returns a message from whose encryption under nonce $i$ and empty header one can recover bit $i$ of the key by encoding this bit as the result of $H(S, \cdot)$ on the ciphertext. For the analysis, Lemma 3.2 says that for each $i$, adversary $B$ fails to recover $K[i]$ with probability at most $1/4k$. By the union bound $B$ fails to recover $K$ with probability at most $1/4$. ▌

Analysis of the BRS attack. As a corollary of Lemma 3.2 we not only show that the inner-product function works but that it is worse only by a factor of two:

**Lemma 3.3** *Let $H_{\mathrm{ip}}\colon \{0,1\}^c \times \{0,1\}^c \to \{0,1\}$ be defined by $H_{\mathrm{ip}}(V,C) = V[1]C[1] + \cdots + V[c]C[c] \bmod 2$. Let $C_1, \ldots, C_l \in \{0,1\}^c$ be distinct and let $T \in \{0,1\}$. Then*

$$\Pr\left[\forall j \,:\, H_{\mathrm{ip}}(V,C_j) \neq T\right] \;\leq\; \frac{2}{l} \tag{1}$$

*where the probability is over a random choice of $V$ from $\{0,1\}^c$.*

**Proof of Lemma 3.3:** Define $H\colon \{0,1\}^{c+1} \times \{0,1\}^c \to \{0,1\}$ by

$$H(S,C) = H_{\mathrm{ip}}(S[1]\ldots S[c], C) + S[c+1] \bmod 2\ .$$

This family of functions is pairwise independent. Let $G$ be the set of all $S \in \{0,1\}^{c+1}$ such that $H(S,C_j) = T$ for some $j$. For $b \in \{0,1\}$ let $G_b$ be the set of all $S \in G$ with $S[c+1] = b$. Let $\epsilon = 1/l$. Lemma 3.2 says that $|G| \geq (1-\epsilon)2^{c+1}$. But $G = G_0 \cup G_1$ and $G_0, G_1$ are disjoint so

$$|G_0| = |G| - |G_1| \;\geq\; |G| - 2^c \;\geq\; (1-\epsilon)2^{c+1} - 2^c \;=\; (1-2\epsilon)2^c\ .$$

To conclude we note that the probability on the left of Equation (1) equals $1 - |G_0|/2^c$. ▌

With this in hand, one can substitute $H$ by $H_{\mathrm{ip}}$ in the proof of Lemma 3.1. By also doubling the value of $l$, the analysis goes through and shows that the BRS attack terminates in a linear number of trials and achieves a constant advantage.

## 3.2 Header insecurity

We would like to use schemes in such a way that headers are not key-dependent but it may not be under our control. Applications may create headers based on data present on the system in a way that results in their depending on the key. We would thus prefer to maintain security in the presence of key-dependent headers. We show that this, too, is impossible, even when messages are key-independent. For both $\mathtt{nt} = \mathtt{u}$ and $\mathtt{nt} = \mathtt{r}$, we present attacks showing no scheme is $(\mathtt{nt}, \mathtt{ki}, \mathtt{kd})$-secure.

**Proposition 3.4** *Let $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Then for any $\mathtt{nt} \in \{\mathtt{u}, \mathtt{r}\}$ there is an efficient adversary $A \in \mathcal{A}[\mathtt{ki}, \mathtt{kd}]$ such that $\mathbf{Adv}^{\mathrm{ae}\text{-}\mathtt{nt}}_{\mathsf{SE}}(A) \geq 1/2$.*

**Proof of Proposition 3.4:** Let $k$ be the keylength of $\mathsf{SE}$. Again, we present an adversary $B$ that recovers the key with probability 1, from which $A$ is easily built. Below we depict $B$ and also define the message-deriving functions it uses. Nonces are given as integers and assumed encoded as $n$-bit strings:

```
Adversary B
INITIALIZE(1)
For $i = 1, \ldots, k$ do
    $(N_i, C_i) \leftarrow\$ \textsc{Enc}(1, i, \mathsf{bit}_i, \phi_0)$ ; $V_i \leftarrow \textsc{Dec}(1, N_i, \phi_0, C_i)$
    If $V_i = \perp$ then $L[i] \leftarrow 0$ else $L[i] \leftarrow 1$
Return $L$

Function $\mathsf{bit}_i(K)$
Return $K[i]$
```

Here $\phi_c$ denotes the constant function that returns $c \in \{0,1\}$. The header computed and used by the game in response to the $i$-th Enc query is $K[i]$. The header computed and used by the game in response to the $i$-th Dec query is 0. Thus, Dec will return $\perp$ if $K[i] = 0$. Otherwise, it will most likely return

0 because the headers don't match, although it might return 1, but in either case we have learned that $K[i] = 1$.

The attack has been written so that it applies in both the universal and random nonce cases. In the first case we will have $N_i = i$. In the second case, $N_i$ will be a random number independent of $i$ chosen by the game. ▮

## 3.3 Remarks

The message-deriving functions used by the adversary in the proof of Proposition 3.1 invoke the encryption algorithm, which is legitimate since any efficient function is allowed. Having encryption depend on a RO will not avoid the attack because the message-deriving functions are allowed to call the RO and can continue to compute encryptions. (In an instantiation the RO will be a hash function and the system may apply it to the key to get data that is later encrypted.)

We do not suggest that precisely these attacks may be mounted in practice. (The message-deriving functions in our attacks are contrived.) However, our attacks rule out the possibility of a proof of security and thus there may exist other, more practical attacks. Indeed, the history of cryptography shows that once an attack is uncovered, better and more practical ones often follow.

## 4 The RHtE transform and its security

We describe our RHtE (Randomized Hash then Encrypt) transform and prove that it endows the base scheme to which it is applied with $(\mathtt{r}, \mathtt{kd}, \mathtt{ki})$-AE security.

### 4.1 The transform

Given a base symmetric encryption scheme $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, a key-generation algorithm $\mathcal{L}$ returning $l$-bit strings, and an integer parameter $r$ representing the length of the random seed used in the key-hashing, the RHtE transform returns a new symmetric encryption scheme $\overline{\mathsf{SE}} = \mathsf{RHtE}[\mathsf{SE}, \mathcal{L}, r] = (\mathcal{L}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$. It has $\mathcal{L}$ as its key-generation algorithm, keylength $l$, noncelength $r$ and the same ciphertextlength as the base scheme. Its encryption and decryption algorithms are defined as follows, where $\textsc{Hash}\colon \{0,1\}^{r+l} \to \{0,1\}^k$ is a RO, $L \in \{0,1\}^l$ is the key, $R \in \{0,1\}^r$ is the nonce (which in the security game will be random), $H$ is the header and $M$ is the message:

```
Algorithm $\overline{\mathcal{E}}(L, R, H, M)$
$K \leftarrow \textsc{Hash}(R \,\|\, L)$ ; $C \leftarrow \mathcal{E}(K, H, M)$
Return $C$

Algorithm $\overline{\mathcal{D}}(L, R, H, C)$
$K \leftarrow \textsc{Hash}(R \,\|\, L)$ ; $M \leftarrow \mathcal{D}(K, H, C)$
Return $M$
```

The base scheme $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is assumed to achieve standard $(\mathtt{nt}, \mathtt{ki}, \mathtt{ki})$-AE security, with $\mathtt{nt}$ being either $\mathtt{u}$ or $\mathtt{r}$. It is assumed to be a standard (as opposed to RO) model scheme. This is not a restriction because for the type of security we assume of it (no key-dependent data) there is no need to use a RO and none of the standardized, in use schemes do, and in any case the assumption is only for simplicity. We are not concerned with keys of the base scheme being passwords because, in standard schemes, they aren't. (Most of the time the key is an AES key.) So it is assumed that $\mathcal{K}$ returns random strings of length $k$. We only require one-time security of the base scheme. Accordingly we assume it is nonceless and deterministic and drop the nonce input above for both encryption and decryption. One can obtain such a scheme from standard ones by fixing a single, public nonce and hardwiring it into the algorithm. The repeated use of the nonce causes no problems since the key $K$ is different on each encryption.

We want the constructed scheme $\overline{\mathsf{SE}}$ to provide security not only when its keys are full-fledged cryptographic ones but also when they are passwords. Hence we view as given an (arbitrary) key-generation algorithm $\mathcal{L}$ returning $l$-bit strings under some arbitrary distribution, and design $\overline{\mathsf{SE}}$ to have $\mathcal{L}$ as its key-generation algorithm.

```
proc $\text{INITIALIZE}(w)$  // All
For $j = 1, \ldots, w$ do $L_j \leftarrow_\$ \mathcal{L}$ ; $S_j \leftarrow \emptyset$

proc $\text{DEC}(j, R, H, C)$  // $\text{F}_{1,1}, \text{F}_{0,1}$
If $(R, H, C) \in S_j$ then return $\perp$
$M \leftarrow \overline{\mathcal{D}}(L_j, R, H, C))$
If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
Return $V$

proc $\text{DEC}(j, R, H, C)$  // $\text{F}_{1,0}, \text{F}_{0,0}$
If $(R, H, C) \in S_j$ then return $\perp$
Return 0

proc $\text{ENC}(j, N, H, \phi)$  // $\text{F}_{1,1}, \text{F}_{1,0}$
$R \leftarrow_\$ \{0,1\}^r$ ; $C \leftarrow_\$ \overline{\mathcal{E}}(L_j, R, H, \phi(L_1, \ldots, L_w))$
$S_j \leftarrow S_j \cup \{(R, H, C)\}$
Return $(R, C)$

proc $\text{ENC}(j, N, H, \phi)$  // $\text{F}_{0,1}, \text{F}_{0,0}$
$c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$ ; $C \leftarrow_\$ \{0,1\}^c$
$R \leftarrow_\$ \{0,1\}^r$ ; $S_j \leftarrow S_j \cup \{(R, H, C)\}$
Return $(R, C)$

proc $\text{FINALIZE}(b')$  // All
Return $(b' = 1)$
```

Figure 3: Games $\text{F}_{\alpha,\beta}$ for $\alpha, \beta \in \{0,1\}$. Next to each procedure we write the games in which it occurs.

The ciphertext returned is a ciphertext of the base scheme but this is deceptive since in practice $R$ will have to be transmitted too to enable decryption. Nonetheless, in common usage, there will be no bandwidth overhead. This is because we must compare to a standard use of the base scheme where it too uses and transmits a nonce. We have saved this space by fixing this nonce and can use it for $R$. However, if we are in a mode where the base scheme gets the nonce out-of-band, we have $r$ bits of bandwidth overhead. The computational overhead is independent of the message size. Implementations with base schemes CCM, EAX and GCM (see Section 6) show that for the first the slowdown is 11% for 5KB messaegs and only 1% for 50KB messages.

The BRS scheme [17] is purely RO-based, and one needs ROs with outputs of length equal to the length of the message. In our scheme the RO is used only for key-derivation and its output length is independent of the length of the message to be encrypted. In this sense, the reliance on ROs is reduced.

## 4.2 Security of $\mathsf{RHtE}$

The following theorem says that if the base scheme is secure for key-independent headers and messages then the constructed scheme is random-nonce secure for key-dependent messages and key-independent headers.

**Theorem 4.1** *Let $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a base symmetric encryption scheme as above. Let $\mathcal{L}$ be a key-generation algorithm with keylength $l$ and let $r$ be a positive integer. Let $\overline{\mathsf{SE}} = \mathsf{RHtE}[\mathsf{SE}, \mathcal{L}, r]$ be the RO model symmetric encryption scheme associated to $\mathsf{SE}, \mathcal{L}, r$ as above. Let $A \in \mathcal{A}[\mathtt{kd}, \mathtt{ki}]$ be an adversary making $\mathsf{q_e}$ ENC queries, $\mathsf{q_d}$ DEC queries and $\mathsf{q_h}$ HASH queries, and let $w \leq 2^{\mathbf{H}_\infty(\mathcal{L})-1}$ be the number of keys, meaning the argument of $A$'s INITIALIZE query. Then there is an adversary $D$ such that*

$$\mathbf{Adv}^{\text{ae-r}}_{\overline{\mathsf{SE}}}(A) \quad \leq \quad (24\mathsf{q_e}^2 + 2\mathsf{q_d}) \cdot \mathbf{Adv}^{\text{ae}}_{\mathsf{SE}}(D) + \frac{8w\mathsf{q_e}\mathsf{q_h} + 2w(w-1)\mathsf{q_e}}{2^{\mathbf{H}_\infty(\mathcal{L})}} + \frac{2\mathsf{q_e}(\mathsf{q_h} + 2\mathsf{q_e}w)}{2^r} \ . \qquad (2)$$

*Adversary $D$ makes only one ENC query and has the same number of DEC queries and the same time complexity as $A$.* ▌

We have omitted the $\mathtt{nt}$ superscript in the advantage of $D$ because $\mathsf{SE}$ is nonceless. That only one-time security is required of $\mathsf{SE}$ is reflected in the fact that $D$ makes only one ENC query. We remark that the bound in Theorem 4.1 does not appear to be tight. It is an interesting open problem to either provide a proof with a better bound or an alternative scheme for which a tight bound can be proved.

## 4.3 Proof overview

As we noted in Section 1 the proof is surprisingly involved because message-deriving functions are allowed to query the RO and because the assumed security of the base scheme must be invoked in multiple, inter-related ways in different parts of the argument, leading to two hybrids in opposite directions, one, unusually, with steps that are differently weighted.

Assume for simplicity that $w = 1$, meaning there is a single target key, denoted $L$. Also assume $A$ makes no DEC queries. Denote by $\phi_1, \ldots, \phi_{\mathsf{q_e}}$ the message-deriving functions in its ENC queries and ignore the corresponding headers. Picking index $g$ at random we set up a hybrid in which the $i$-th ENC query $\phi_i$ is answered by encrypting message $\phi_i(L)$ under $L$ as in the real game if $i < g$ and answered at random if $i > g$, the $g$-th query toggling between real and random to play the role of the challenge for an adversary $B$ against the base scheme. Let $R_1, \ldots, R_{\mathsf{q_e}}$ denote the random nonces chosen by the game. The reduction $B$ cannot answer hash oracle query $R_g \| L$ because the reply is its target key so a bad event is flagged if $A$ either makes this query directly, or indirectly via a message-deriving function. But once query $g$ has been answered, $A$ has $R_g$ and thus for queries $i > g$, nothing can prevent $\phi_i$ from querying $R_g \| L$ to the RO, and how are these queries to be answered by $B$? Crucial to this was doing the hybrid top to bottom, meaning first real then random rather than the other way round. This enables us to avoid evaluating $\phi_i$ on $L$ for post-challenge queries, so that its RO queries do not need to be answered at all. This leaves the possibility that $A$ directly makes hash query $R_g \| L$ after it gets $R_g$. Intuitively this is unlikely because $A$ does not know $L$. The subtle point is that this relies on the assumed security of the base scheme and hence must be proven by reduction. However, doing such a reduction means another hybrid and seems to simply shunt the difficulty to another query. To get around this circularity, we do the second hybrid in the opposite direction and also with different "weights" on the different steps.

## 4.4 Proof of Theorem 4.1

Consider executing $A$ with the games of Figure 3. For simplicity we have not displayed procedure HASH, which implements the RO and is called by $\overline{\mathcal{E}}, \overline{\mathcal{D}}$. We need to bound

$$\Pr[\mathrm{F}_{1,1}^A] - \Pr[\mathrm{F}_{0,0}^A] \quad = \quad \left(\Pr[\mathrm{F}_{1,1}^A] - \Pr[\mathrm{F}_{0,1}^A]\right) + \left(\Pr[\mathrm{F}_{0,1}^A] - \Pr[\mathrm{F}_{0,0}^A]\right) \tag{3}$$

and will bound the two terms in turn, meaning that, leaving the decryption oracle fixed to the real one, we will first turn answers to encryption queries from real to random and then, leaving them at random, flip the decryption oracle to $\perp$.

For the first step we consider games G, H of Figure 4. Game G sets up a hybrid in which the first $g-1$ ENC queries are answered by real encryption, the $g$-th toggles between real and random depending on bit $b$, and the rest are answered by random ciphertexts. This aims to set up a reduction to the assumed security of $\mathsf{SE}$ for the $g$-th encryption with $K$ playing the role of the key underlying the game of the adversary attacking $\mathsf{SE}$. This adversary would not know $K$ and, to allow it to simulate $A$, hash queries that would need to return $K$ are flagged and removed from the now simulatable game H at the cost of the probability of setting $\mathsf{bad}$. The decryption oracle is the real one, in both games. Procedure IHASH has been introduced to answer indirect hash queries, namely those made by $\phi$, ENC or DEC, because they will have to be treated differently from direct hash queries of $A$, still answered by HASH. The proof of the following is in Section 4.5.

**Lemma 4.2** *There is an adversary $B_1$ so that*

$$\frac{1}{\mathsf{q_e}}\left(\Pr[\mathrm{F}_{1,1}^A] - \Pr[\mathrm{F}_{0,1}^A]\right) \quad \leq \quad \Pr[\mathrm{E}_{1,1}^{B_1}] - \Pr[\mathrm{E}_{0,1}^{B_1}] + 2\Pr[\mathsf{BAD}(\mathrm{H}^A)] \quad \blacksquare$$

The games $\mathrm{E}_{\alpha,\beta}$ referred to here are in Figure 5. We must now bound the probability that $\mathrm{H}^A$ sets $\mathsf{bad}$. The proof of the following is in Section 4.6.

**Lemma 4.3** *The probability of* $\mathsf{BAD}(\mathrm{H}^A)$ *is at most*

$$\Pr[\mathsf{BAD}(\mathrm{H}^A, 303)] + \frac{\mathsf{q_h} + 2\mathsf{q_e}w}{2^r} \,. \tag{4}$$

```
proc Initialize  // G, H
000 For $j = 1, \ldots, w$ do
001     $L_j \leftarrow_{\$} \mathcal{L}$ ; $S_j \leftarrow \emptyset$
002     For $i = 1, \ldots, \mathsf{q_e}$ do $R_{i,j} \leftarrow_{\$} \{0,1\}^r$
003 $i \leftarrow 0$ ; $b \leftarrow_{\$} \{0,1\}$ ; $K \leftarrow_{\$} \{0,1\}^k$
004 $g \leftarrow_{\$} \{1, \ldots, \mathsf{q_e}\}$

proc Enc($j, N, H, \phi$)  // $\boxed{\text{G}}$, H
101 $i \leftarrow i + 1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$
102 If $(i \leq g)$ then
103     $M \leftarrow \text{Eval}^{\text{IHash}}(\phi, L_1, \ldots, L_w)$
104 If $i < g$ then
105     $C_i \leftarrow \mathcal{E}(\text{IHash}(R_{i,j} \,\|\, L_j), H, M)$
106 If $(i = g$ and $b = 1)$ then
107     If $H[R_{i,j} \,\|\, L_j]$ then
108         bad $\leftarrow$ true ; $\boxed{K \leftarrow H[R_{i,j} \,\|\, L_j]}$
109     $s \leftarrow j$ ; $C_i \leftarrow \mathcal{E}(K, H, M)$
110 If $(i = g$ and $b = 0)$ then
111     $s \leftarrow j$ ; $C_i \leftarrow_{\$} \{0,1\}^c$
112 If $i > g$ then $C_i \leftarrow_{\$} \{0,1\}^c$
113 $S_j \leftarrow S_j \cup \{(R_{i,j}, H, C_i)\}$
114 Return $(R_{i,j}, C_i)$

proc IHash($R \,\|\, L$)  // $\boxed{\text{G}}$, H
200 If not $H[R \,\|\, L]$ then
201     $H[R \,\|\, L] \leftarrow_{\$} \{0,1\}^k$
202     If $(R \,\|\, L = R_{g,s} \,\|\, L_s)$ then
203         bad $\leftarrow$ true ; $\boxed{H[R \,\|\, L] \leftarrow K}$
204 Return $H[R \,\|\, L]$

proc Hash($R \,\|\, L$)  // $\boxed{\text{G}}$, H
300 If not $H[R \,\|\, L]$ then
301     $H[R \,\|\, L] \leftarrow_{\$} \{0,1\}^k$
302     If $(R \,\|\, L = R_{g,s} \,\|\, L_s)$ then
303         bad $\leftarrow$ true ; $\boxed{H[R \,\|\, L] \leftarrow K}$
304 Return $H[R \,\|\, L]$

proc Dec($j, R, H, C$)  // G, H
400 If $(R, H, C) \in S_j$ then return $\perp$
401 If $(R \,\|\, L_j = R_{g,s} \,\|\, L_s)$ then
402     $M \leftarrow \mathcal{D}(K, H, C)$
403 Else $M \leftarrow \mathcal{D}(\text{IHash}(R \,\|\, L_j), H, C)$
404 If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
405 Return $V$

proc Finalize($b'$)  // G, H
500 Return $(b' = b)$
```

Figure 4: Games G, H for the proof of Theorem 4.1. A box around the name of a game next to a procedure means the boxed code of that procedure is included in the game.

Here $\mathsf{BAD}(\text{H}^A, x)$ is the event that bad is set at line $x$. The argument makes crucial use of the fact that the games provide random answers to post-challenge Enc queries. This is what allows us to not evaluate $\phi$ post-challenge. Had we done the hybrids in the opposite direction, beginning with random answers and then providing real ones, the probability that IHash sets bad could be large. We now need to bound the first term above where subtle issues arise.

Post-challenge, $A$ has $R_{g,s}$. If the probability $\Pr[\mathsf{BAD}(\text{H}^A, 303)]$ that $\text{H}^A$ sets bad at line 303 is to be small, we expect that it is because $A$ is unlikely to know $L_s$ and thus unlikely to query $R_{g,s} \,\|\, L_s$. We might think, accordingly, that $\Pr[\mathsf{BAD}(\text{H}^A, 303)]$ is unconditionally bounded by $\mathsf{q_h}/2^{\mathbf{H}_\infty(\mathcal{L})}$, but this turns out to be wrong. In fact, the argument is necessarily computational, relying on the assumed security of $\mathsf{SE}$. To see this, suppose $\mathcal{E}(K, M) = M$ for all $M$ (no header). Let $A$'s first Enc query be $1, \phi$ where $\phi(L_1, \ldots, L_n) = L_1$, and suppose $g > 1$. In response, $A$ gets $(R_{1,1}, \mathcal{E}(\text{IHash}(R_{1,1} \,\|\, L_1), L_1)) = (R_{1,1}, L_1)$, and thus has $L_1$. It uses $j = 1$ in all subsequent Enc queries as well (it does not matter what is the corresponding $\phi$) so that $s$ is set to 1 at line 107 or 109. After it receives $R_{g,1}$ from the response to its $g$-th Enc query, it can make hash query $R_{g,1} \,\|\, L_1$ since it knows $L_1$ and thus sets bad whenever $g > 1$, meaning with (high) probability $1 - 1/\mathsf{q_e}$.

Of course, if $\mathsf{SE}$ is secure it will not be that $\mathcal{E}(K, M) = M$ for all $M$. But what this means is that bounding $\Pr[\mathsf{BAD}(\text{H}^A, 303)]$ *must rely on the assumed security of* $\mathsf{SE}$. Towards obtaining this bound, consider game $\text{I}_{g,h}$ of Figure 6, which is parameterized by $g, h$. It provides random responses to the first $h$ Enc queries, then provides real responses until it gets to the $g$-th query, to which it responds as does H. Subsequent queries get random responses. The game returns true when bad is set in procedure Hash. We have

$$\Pr[\mathsf{BAD}(\text{H}^A, 303)] \;=\; \tfrac{1}{\mathsf{q_e}} \textstyle\sum_{g=1}^{\mathsf{q_e}} \Pr[\text{I}_{g,0}^A] \; . \tag{5}$$

```
proc INITIALIZE($w$)  // All
    $K \leftarrow_{\$} \{0,1\}^k$ ; $S \leftarrow \emptyset$

proc DEC($R,H,C$)  // $E_{1,1}, E_{0,1}$
    If $(R,H,C) \in S$ then return $\perp$
    $M \leftarrow \mathcal{D}(K,R,H,C)$
    If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
    Return $V$

proc DEC($R,H,C$)  // $E_{1,0}, E_{0,0}$
    If $(R,H,C) \in S$ then return $\perp$
    Return 0

proc ENC($N,H,M$)  // $E_{1,1}, E_{1,0}$
    $R \leftarrow_{\$} \{0,1\}^n$ ; $C \leftarrow \mathcal{E}(K,R,H,M)$
    $S \leftarrow S_j \cup \{(H,C)\}$
    Return $(R,C)$

proc ENC($N,H,M$)  // $E_{0,1}, E_{0,0}$
    $c \leftarrow \mathsf{cl}(|M|,|H|)$ ; $R \leftarrow_{\$} \{0,1\}^n$
    $C \leftarrow_{\$} \{0,1\}^c$ ; $S \leftarrow S \cup \{(H,C)\}$
    Return $(R,C)$

proc FINALIZE($b'$)  // All
    Return $(b' = 1)$
```

Figure 5: Games $E_{\alpha,\beta}$ for $\alpha,\beta \in \{0,1\}$.

Towards bounding this we begin by considering game $I_{g,g-1}$. In the case $b = 0$, all ENC queries receive random answers and thus ENC does not leak any information about $L_s$. We would like to say that this leads to an unconditional bound on the probability that bad is set at line 303 but this fails to consider DEC queries whose answers can still depend on $L_s$. Using the fact that these answers, however, depend only on entries $H[\cdot \,\|\, L_s]$ we do succeed in unconditionally bounding the probability that bad is set in game $I_{g,g-1}$ when $b = 0$. The assumed security of SE can then be used to say that bad is set not much more often when $b = 1$. All this is captured by the following whose proof is in Section 4.7.

**Lemma 4.4** *There is an adversary $B_2$ so that*

$$\tfrac{1}{\mathsf{q_e}} \textstyle\sum_{g=1}^{\mathsf{q_e}} \Pr\left[\, I^A_{g,g-1} \,\right] \;\leq\; \frac{1}{2}\Pr[E^{B_2}_{1,1}] - \frac{1}{2}\Pr[E^{B_2}_{0,1}] + \frac{4w\mathsf{q_h} + w(w-1)}{2^{\mathbf{H}_\infty(\mathcal{L})+1}} \quad \blacksquare$$

The next lemma bridges the gap from $I_{g,g-1}$ to $I_{g,0}$. An unusual aspect of the proof of the following, which appears in Section 4.8, is that adversary $B_3$ will need to assign different weights to the different hybrids.

**Lemma 4.5** *There is an adversary $B_3$ so that*

$$\tfrac{1}{\mathsf{q_e^2}} \textstyle\sum_{g=1}^{\mathsf{q_e}} \left(\Pr\left[\, I^A_{g,0} \,\right] - \Pr\left[\, I^A_{g,g-1} \,\right]\right) \;\leq\; \Pr[E^{B_3}_{1,1}] - \Pr[E^{B_3}_{0,1}] \quad \blacksquare$$

From Lemma 4.5 and Lemma 4.4 we have the following, which bounds the first term of Equation (3).

**Lemma 4.6** *There is an adversary $D_1$ such that*

$$\Pr[F^A_{1,1}] - \Pr[F^A_{0,1}] \;\leq\; 24\mathsf{q_e^2} \cdot \mathbf{Adv}^{\mathrm{ae}}_{\mathsf{SE}}(D_1) + \frac{4w\mathsf{q_e}\mathsf{q_h} + w(w-1)\mathsf{q_e}}{2^{\mathbf{H}_\infty(\mathcal{L})}} + \frac{2\mathsf{q_e}(\mathsf{q_h} + 2\mathsf{q_e}w)}{2^r} \quad \blacksquare . \tag{6}$$

The proof is in Section 4.9. We proceed to bound the second term of Equation (3). Game J of Figure 7 responds to the first $g-1$ DEC queries correctly, to the $g$-th either correctly or by $\perp$ depending on whether $b = 1$ or $b = 0$, and to the rest by $\perp$. ENC queries all receive random answers. This aims to set up a reduction to the assumed security of SE with $K[g]$ playing the role of the key underlying the game of the adversary attacking SE. To allow such an adversary to simulate $A$, hash queries that would need to return any of the keys are flagged and removed from the now simulatable game L at the cost of the probability of setting bad. The proof of the following is in Section 4.10.

**Lemma 4.7** *There is an adversary $B_4$ so that*

$$\frac{1}{\mathsf{q_d}}\left(\Pr[F^A_{0,1}] - \Pr[F^A_{0,0}]\right) \;\leq\; \Pr[E^{B_4}_{\mathsf{SE},0,1}] - \Pr[E^{B_4}_{\mathsf{SE},0,0}] + 2\Pr[\mathsf{BAD}(L^A)] \quad \blacksquare$$

```
proc $\text{Initialize}$ // $\mathrm{I}_{g,h}$
000 For $j = 1, \ldots, w$ do
001     $L_j \leftarrow\!\$\, \mathcal{L}$ ; $S_j \leftarrow \emptyset$
002     For $i = 1, \ldots, \mathsf{q_e}$ do $R_{i,j} \leftarrow\!\$\, \{0,1\}^r$
003 $i \leftarrow 0$ ; $b \leftarrow\!\$\, \{0,1\}$ ; $K \leftarrow\!\$\, \{0,1\}^k$

proc $\text{Enc}(j, N, H, \phi)$ // $\mathrm{I}_{g,h}$
100 $i \leftarrow i + 1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$
101 If $(i \leq g)$ then
102     $M \leftarrow \text{Eval}^{\text{IHash}}(\phi, L_1, \ldots, L_w)$
103 If $(i \leq h)$ then $C_i \leftarrow\!\$\, \{0,1\}^c$
104 If $(h < i < g)$ then
105     $C_i \leftarrow \mathcal{E}(\text{IHash}(R_{i,j} \,\|\, L_j), H, M)$
106 If $(i = g$ and $b = 1)$ then
107     $s \leftarrow j$ ; $C_i \leftarrow \mathcal{E}(K, H, M)$
108 If $(i = g$ and $b = 0)$ then
109     $s \leftarrow j$ ; $C_i \leftarrow\!\$\, \{0,1\}^c$
110 If $i > g$ then $C_i \leftarrow\!\$\, \{0,1\}^c$
111 $S_j \leftarrow S_j \cup \{(R_{i,j}, H, C_i)\}$
112 Return $(R_{i,j}, C_i)$

proc $\text{IHash}(R \,\|\, L)$ // $\mathrm{I}_{g,h}$
200 If not $H[R \,\|\, L]$ then
201     $H[R \,\|\, L] \leftarrow\!\$\, \{0,1\}^k$
202 Return $H[R \,\|\, L]$

proc $\text{Hash}(R \,\|\, L)$ // $\mathrm{I}_{g,h}$
300 If not $H[R \,\|\, L]$ then
301     $H[R \,\|\, L] \leftarrow\!\$\, \{0,1\}^k$
302     If $(R \,\|\, L = R_{g,s} \,\|\, L_s)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
304 Return $H[R \,\|\, L]$

proc $\text{Dec}(j, R, H, C)$ // $\mathrm{I}_{g,h}$
400 If $(R, H, C) \in S_j$ then return $\perp$
401 If $(R \,\|\, L_j = R_{g,s} \,\|\, L_s)$ then
402     $M \leftarrow \mathcal{D}(K, H, C)$
403 Else $M \leftarrow \mathcal{D}(\text{IHash}(R \,\|\, L_j), H, C)$
404 If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
405 Return $V$

proc $\text{Finalize}(b')$ // $\mathrm{I}_{g,h}$
500 Return $\mathsf{bad}$
```

Figure 6: Games $\mathrm{I}_{g,h}$ for $0 \leq h \leq g - 1 \leq \mathsf{q_e} - 1$.

```
proc $\text{Initialize}$ // $\mathrm{J}, \mathrm{L}$
000 For $j = 1, \ldots, w$ do
001     $L_j \leftarrow\!\$\, \mathcal{L}$ ; $S_j \leftarrow \emptyset$
002     For $i = 1, \ldots, \mathsf{q_e}$ do $R_{i,j} \leftarrow\!\$\, \{0,1\}^r$
003 $g \leftarrow\!\$\, \{1, \ldots, \mathsf{q_d}\}$
004 For $d = 1, \ldots, \mathsf{q_d}$ do $K[d] \leftarrow\!\$\, \{0,1\}^k$
005 $i, d \leftarrow 0$ ; $b \leftarrow\!\$\, \{0,1\}$

proc $\text{Enc}(j, N, H, \phi)$ // $\mathrm{J}, \mathrm{L}$
100 $i \leftarrow i + 1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$
101 $C_i \leftarrow\!\$\, \{0,1\}^c$ ; $S_j \leftarrow S_j \cup \{(R_{i,j}, H, C_i)\}$
102 Return $(R_{i,j}, C_i)$

proc $\text{Hash}(R \,\|\, L)$ // $\boxed{\mathrm{J}}$, $\mathrm{L}$
200 If not $H[R \,\|\, L]$ then
201     $H[R \,\|\, L] \leftarrow\!\$\, \{0,1\}^k$
202     If $\text{Ind}[R \,\|\, L]$ then
203         $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{H[R \,\|\, L] \leftarrow K[\text{Ind}[R \,\|\, L]]}$
204 Return $H[R \,\|\, L]$

proc $\text{Dec}(j, R, H, C))$ // $\mathrm{J}$
300 If $(R, H, C) \in S_j$ then return $\perp$
301 If not $\text{Ind}[R \,\|\, L_j]$ then
302     $d \leftarrow d + 1$ ; $\text{Ind}[R \,\|\, L_j] \leftarrow d$
303 $e \leftarrow \text{Ind}[R \,\|\, L_j]$
304 If $H[R \,\|\, L_j]$ then
305     $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{K[e] \leftarrow H[R \,\|\, L_j]}$
307 If $(e < g)$ then $M \leftarrow \mathcal{D}(K[e], H, C)$
308 If $((e = g)$ and $(b = 1))$ then
309     $M \leftarrow \mathcal{D}(K[e], H, C)$
310 If $((e = g)$ and $(b = 0))$ then return 0
311 If $(e > g)$ then return 0
312 If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
313 Return $V$

proc $\text{Finalize}(b')$ // $\mathrm{J}, \mathrm{L}$
400 Return $(b = b')$
```

Figure 7: Games $\mathrm{J}, \mathrm{L}$ to bound second term in Equation (3).

We must now bound the probability that $\mathrm{L}^A$ sets $\mathsf{bad}$. Assume $L_1, \ldots, L_w$ are distinct, which happens except with probability $w(w-1)/2^{\mathbf{H}_\infty(\mathcal{L})+1}$. Now the role of $L_j$ as an index to $H$ can be played by $j$. Since the boxed code is omitted in game L, the tests of lines 202 and 304 need not be perfomed there. Rather, the queries can be recorded and the tests to set $\mathsf{bad}$ done in FINALIZE. At this point $L_1, \ldots, L_w$ are not referred to by the oracles and may also be chosen in FINALIZE. The flag $\mathsf{bad}$ is set only when a query to HASH involves $L_j$ for some $j$ hence is set with probability at most $w\mathsf{q_h}/2^{\mathbf{H}_\infty(\mathcal{L})}$. We conclude that

$$\Pr[\mathsf{BAD}(\mathrm{L}^A)] \;\leq\; \frac{w(w-1)}{2^{\mathbf{H}_\infty(\mathcal{L})+1}} + \sum_{j=1}^{w} \frac{\mathsf{q_h}}{2^{\mathbf{H}_\infty(\mathcal{L})} - (j-1)} \;\leq\; \frac{w(w-1)}{2^{\mathbf{H}_\infty(\mathcal{L})+1}} + \frac{2w\mathsf{q_h}}{2^{\mathbf{H}_\infty(\mathcal{L})}} \;=\; \frac{4w\mathsf{q_h} + w(w-1)}{2^{\mathbf{H}_\infty(\mathcal{L})+1}} \;. \quad (7)$$

The second inequality above used the assumption, made in the theorem statement, that $w \leq 2^{\mathbf{H}_\infty(\mathcal{L})-1}$. From Lemma 4.7 and Equation (7) we have

$$\Pr[\mathrm{F}^A_{0,1}] - \Pr[\mathrm{F}^A_{0,0}] \;\leq\; \mathsf{q_d}(\Pr[\mathrm{E}^{B_4}_{\mathsf{SE},0,1}] - \Pr[\mathrm{E}^{B_4}_{\mathsf{SE},0,0}]) + \frac{4w\mathsf{q_d}\mathsf{q_h} + w(w-1)\mathsf{q_d}}{2^{\mathbf{H}_\infty(\mathcal{L})}} \;.$$

Apply Lemma 4.8 (Section 4.11) to $B_4$. This yields adversary $D_2$ such that

$$\Pr[\mathrm{F}^A_{0,1}] - \Pr[\mathrm{F}^A_{0,0}] \;\leq\; 2\mathsf{q_d} \cdot \mathbf{Adv}^{\mathrm{ae}}_{\mathsf{SE}}(D_2) + \frac{4w\mathsf{q_d}\mathsf{q_h} + w(w-1)\mathsf{q_d}}{2^{\mathbf{H}_\infty(\mathcal{L})}} \;. \quad (8)$$

Finally let adversary $D$ run $D_1$ with probability $24\mathsf{q_e}^2/(24\mathsf{q_e}^2+2\mathsf{q_d})$ and $D_2$ otherwise. From Equation (3), Equation (6) and Equation (8) we have Equation (2), concluding the proof of Theorem 4.1.

## 4.5 Proof of Lemma 4.2

The proof refers to the games of Figure 3 and Figure 5. We have

$$\begin{aligned} \frac{1}{\mathsf{q_e}}\left(\Pr[\mathrm{F}^A_{1,1}] - \Pr[\mathrm{F}^A_{0,1}]\right) &= 2\Pr[\mathrm{G}^A] - 1 & (9)\\ &= 2(\Pr[\mathrm{H}^A] + \Pr[\mathrm{G}^A] - \Pr[\mathrm{H}^A]) - 1 & \\ &\leq 2\Pr[\mathrm{H}^A] - 1 + 2\Pr[\mathsf{BAD}(\mathrm{H}^A)] \;. & (10) \end{aligned}$$

Equation (10) follows from the Fundamental Lemma of Game Playing [14] because games G, H are identical until $\mathsf{bad}$, meaning differ only in code following the setting of $\mathsf{bad}$ to $\mathsf{true}$. To justify Equation (9) note that when $b = 1$ the first $g$ replies are real and the rest random, and when $b = 0$ the first $g - 1$ replies are real and the rest random.

Adversary $B_1$ begins with initializations

```
For $j = 1, \ldots, w$ do
    $L_j \leftarrow_\$ \mathcal{L}$ ; $S_j \leftarrow \emptyset$
    For $i = 1, \ldots, \mathsf{q_e}$ do $R_{i,j} \leftarrow_\$ \{0,1\}^r$
$i \leftarrow 0$ ; $g \leftarrow_\$ \{1, \ldots, \mathsf{q_e}\}$
```

Now $B_1$ runs $A$. It replies to ENC query $j, H, \phi$ of $A$ via

```
$i \leftarrow i + 1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi, |H|))$
If $(i \leq g)$ then
    $M \leftarrow \mathrm{EVAL}^{\mathrm{IHASH}}(\phi, L_1, \ldots, L_w)$
If $i < g$ then $C_i \leftarrow \mathcal{E}(\mathrm{IHASH}(R_{i,j} \,\|\, L_j), H, M)$
If $(i = g)$ then $s \leftarrow j$ ; $C_i \leftarrow_\$ \mathrm{ENC}(H, M)$
```

```
        If $i > g$ then $C_i \leftarrow\$ \{0,1\}^c$
        $S_j \leftarrow S_j \cup \{(R_{i,j}, H, C_i)\}$
        Return $(R_{i,j}, C_i)$
```

In this code, ENC is $B_1$'s own encryption oracle which it calls with message $M$. $B_1$ replies to IHASH or HASH query $R \,\|\, L$ via

```
        If not $H[R \,\|\, L]$ then $H[R \,\|\, L] \leftarrow\$ \{0,1\}^k$
        Return $H[R \,\|\, L]$
```

It replies to DEC query $j, R, H, C$ via

```
        If $(R, H, C) \in S_j$ then return $\perp$
        If $(R \,\|\, L_j = R_{g,s} \,\|\, L_s)$ then $M \leftarrow$ DEC$(H, C)$
        Else $M \leftarrow \mathcal{D}(\text{IHASH}(R \,\|\, L_j), H, C)$
        If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
        Return $V$
```

where DEC called in this code is $B_1$'s own decryption oracle. When $A$ halts with output $b'$, so does adversary $B_1$. Think of the key $K$ of game H as being the one underlying games $\text{KDAE}_{\mathsf{SE},1,1}, \text{KDAE}_{\mathsf{SE},0,1}$ for $B_1$. Thus

$$2\Pr[\text{H}^A] - 1 \quad = \quad \Pr[\text{E}_{1,1}^{B_1}] - \Pr[\text{E}_{0,1}^{B_1}] \,.$$

## 4.6 Proof of Lemma 4.3

No information about $R_{g,s}$ is provided to $A$ until the $g$-th ENC query is answered, so the probability that $\mathsf{bad}$ is set at line 106 is at most $\mathsf{q_h}/2^r$. Once $R_{g,s}$ is released, however, $A$ *could* in fact specify a $\phi$ whose evaluation would result in hash query $R_{g,s} \,\|\, L$. This difficulty is circumvented by the If statement on line 102, which performs the evaluation of $\phi$ only if $i \leq g$. As a result procedure IHASH is not called by ENC after the challenge. It might be called post-challenge by DEC but due to line 401 this will not set $\mathsf{bad}$. Thus IHASH sets $\mathsf{bad}$ only with the probability that some $R_{i,j}$ equals $R_{g,s}$ which is at most $\mathsf{q_e}w/2^{r+1}$.

## 4.7 Proof of Lemma 4.4

For any $g \in \{1, \ldots, \mathsf{q_e}\}$ we claim that

$$\Pr\left[\,\text{I}_{g,g-1}^A \mid b = 0\,\right] \quad \leq \quad \frac{2w\mathsf{q_h} + w(w-1)}{2^{\mathbf{H}_\infty(\mathcal{L})+1}} \,. \tag{11}$$

Towards justifying this the first observation is that in $\text{I}_{g,g-1}$ all ENC queries receive random responses when $b = 0$ and thus provide the adversary no information about $L_s$. We would like thence to conclude that the probability of setting bad is at most $\mathsf{q_h}/2^l$. This is true if there are no DEC queries. To obtain a bound in the presence of the latter we claim

$$\Pr\left[\,\text{I}_{g,g-1}^A \mid b = 0 \text{ and } L_1, \ldots, L_w \text{ are distinct}\,\right] \;\leq\; \Pr[\text{K}_g^A] \;\leq\; \frac{2w\mathsf{q_h}}{2^{\mathbf{H}_\infty(\mathcal{L})}}$$

where game $\text{K}_g$ is in Figure 8. Briefly, when $L_1, \ldots, L_w$ are distinct, the role of $L_j$ can be played by $j$ as long as queries to the two hash oracles stay different, so that $L_j$ is no longer referred to by the oracles, allowing us to move the setting of $\mathsf{bad}$ to FINALIZE. The probability that $\mathsf{bad}$ is set is then at most

$$\frac{\mathsf{q_h}}{2^{\mathbf{H}_\infty(\mathcal{L})} - 1} + \cdots + \frac{\mathsf{q_h}}{2^{\mathbf{H}_\infty(\mathcal{L})} - (w-1)} \;\leq\; \frac{2w\mathsf{q_h}}{2^{\mathbf{H}_\infty(\mathcal{L})}} \,,$$

```
proc Initialize // $\mathrm{K}_g$
000 For $j = 1, \ldots, w$ do
001     $S_j \leftarrow \emptyset$
002     For $i = 1, \ldots, \mathsf{q_e}$ do $R_{i,j} \leftarrow\!\!\$\ \{0,1\}^r$
003 $i \leftarrow 0$ ; $K \leftarrow\!\!\$\ \{0,1\}^k$ ; $T \leftarrow \emptyset$

proc Enc$(j, N, H, \phi)$ // $\mathrm{K}_g$
100 $i \leftarrow i+1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$
101 $C_i \leftarrow\!\!\$\ \{0,1\}^c$ ; $S_j \leftarrow S_j \cup \{(R_{i,j}, H, C_i)\}$
102 If $(i = g)$ then $s \leftarrow j$
103 Return $(R_{i,j}, C_i)$

proc Finalize$(b')$ // $\mathrm{K}_g$
500 For $j = 1, \ldots, w$ do $L_j \leftarrow 0^l$
501 While $|\{L_1, \ldots L_w\}| < w$
502     For $j = 1, \ldots, w$ do $L_j \leftarrow\!\!\$\ \mathcal{L}$
503 If $(T \cap \{L_1, \ldots, L_w\} \neq \emptyset)$ then
504     $\mathsf{bad} \leftarrow \mathsf{true}$
505 Return $\mathsf{bad}$

proc IHash$(R \,\|\, j)$ // $\mathrm{K}_g$
200 If not $H'[R \,\|\, j]$ then
201     $H'[R \,\|\, j] \leftarrow\!\!\$\ \{0,1\}^k$
202 Return $H'[R \,\|\, j]$

proc Hash$(R \,\|\, L)$ // $\mathrm{K}_g$
300 If not $H[R \,\|\, L]$ then
301     $H[R \,\|\, L] \leftarrow\!\!\$\ \{0,1\}^k$
302     $T \leftarrow T \cup \{L\}$
303 Return $H[R \,\|\, L]$

proc Dec$(j, R, H, C)$ // $\mathrm{K}_g$
400 If $(R, H, C) \in S_j$ then return $\perp$
401 If $(R \,\|\, j = R_{g,s} \,\|\, s)$ then $M \leftarrow \mathcal{D}(K, H, C)$
402 Else $M \leftarrow \mathcal{D}(\mathrm{IHash}(R \,\|\, j), H, C)$
403 If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
404 Return $V$
```

Figure 8: Game $\mathrm{K}_g$ for proof of Lemma 4.4 where $1 \le g \le \mathsf{q_e}$.

the last because we assumed $w \le 2^{\mathbf{H}_\infty(\mathcal{L})-1}$. Since $L_1, \ldots, L_w$ are distinct except with probability less than $w(w-1)/2^{\mathbf{H}_\infty(\mathcal{L})}$ we have Equation (11).

Below we will construct $B_2$ so that

$$\frac{1}{\mathsf{q_e}} \sum_{g=1}^{\mathsf{q_e}} \left( \Pr\left[ \mathrm{I}^A_{g,g-1} \mid b=1 \right] - \Pr\left[ \mathrm{I}^A_{g,g-1} \mid b=0 \right] \right) \quad \le \quad \Pr[\mathrm{E}^{B_2}_{1,1}] - \Pr[\mathrm{E}^{B_2}_{0,1}] \,. \tag{12}$$

Assuming this we have

$$\begin{aligned}
&\frac{1}{\mathsf{q_e}} \sum_{g=1}^{\mathsf{q_e}} \Pr\left[ \mathrm{I}^A_{g,g-1} \right] \\
&\quad = \frac{1}{2\mathsf{q_e}} \sum_{g=1}^{\mathsf{q_e}} \left( \Pr\left[ \mathrm{I}^A_{g,g-1} \mid b=1 \right] + \Pr\left[ \mathrm{I}^A_{g,g-1} \mid b=0 \right] \right) \\
&\quad = \frac{1}{2\mathsf{q_e}} \sum_{g=1}^{\mathsf{q_e}} \left( \Pr\left[ \mathrm{I}^A_{g,g-1} \mid b=1 \right] - \Pr\left[ \mathrm{I}^A_{g,g-1} \mid b=0 \right] + 2\Pr\left[ \mathrm{I}^A_{g,g-1} \mid b=0 \right] \right) \\
&\quad \le \frac{1}{2} \Pr[\mathrm{E}^{B_2}_{1,1}] - \frac{1}{2} \Pr[\mathrm{E}^{B_2}_{0,1}] + \frac{1}{\mathsf{q_e}} \sum_{g=1}^{\mathsf{q_e}} \frac{4w\mathsf{q_h} + w(w-1)}{2^{\mathbf{H}_\infty(\mathcal{L})+1}} \\
&\quad = \frac{1}{2} \Pr[\mathrm{E}^{B_2}_{1,1}] - \frac{1}{2} \Pr[\mathrm{E}^{B_2}_{0,1}] + \frac{4w\mathsf{q_h} + w(w-1)}{2^{\mathbf{H}_\infty(\mathcal{L})+1}}
\end{aligned}$$

which proves the lemma.

We now construct $B_2$ so that Equation (12) is true. Adversary $B_2$ begins with initializations

```
For $j = 1, \ldots, w$ do
    $L_j \leftarrow\!\!\$\ \mathcal{L}$ ; $S_j \leftarrow \emptyset$
```

```
        For $i = 1, \ldots, \mathsf{q_e}$ do $R_{i,j} \leftarrow\!\!\$\, \{0,1\}^r$
    $i \leftarrow 0$ ; $g \leftarrow\!\!\$\, \{1, \ldots, \mathsf{q_e}\}$
```

Now $B_2$ runs $A$. It replies to Enc query of $A$ via

```
    $i \leftarrow i + 1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$
    If $(i \le g)$ then
        $M \leftarrow \text{EVAL}^{\text{IHASH}}(\phi, L_1, \ldots, L_w)$
    If $(i \le g - 1)$ then $C_i \leftarrow\!\!\$\, \{0,1\}^c$
    If $(i = g)$ then $s \leftarrow j$ ; $C_i \leftarrow\!\!\$\, \text{ENC}(H, M)$
    If $i > g$ then $C_i \leftarrow\!\!\$\, \{0,1\}^c$
    $S_j \leftarrow S_j \cup \{(R_{i,j}, H, C_i)\}$
    Return $(R_{i,j}, C_i)$
```

where Enc in this code is $B_2$'s own encryption oracle. It replies to IHash, Hash queries as in the games of Figure 6. It replies to Dec query $j, R, H, C$ via

```
    If $(R, H, C) \in S_j$ then return $\perp$
    If $(R \,\|\, L_j = R_{g,s} \,\|\, L_s)$ then
        $V \leftarrow \text{DEC}(H, C)$
        Return $V$
    Else $M \leftarrow \mathcal{D}(\text{IHASH}(R \,\|\, L_j), H, C)$
    If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
    Return $V$
```

where Dec called in this code is $B_2$'s own decryption oracle. When $A$ halts, adversary $B_2$ outputs 1 if $\mathsf{bad}$ was set and 0 otherwise.

## 4.8 Proof of Lemma 4.5

Adversary $B_3$ will perform a hybrid based on the games of Figure 6 but with the twist that different hybrids are differently weighted. It begins with initializations

```
    For $j = 1, \ldots, w$ do
        $L_j \leftarrow\!\!\$\, \mathcal{L}$ ; $S_j \leftarrow \emptyset$
        For $i = 1, \ldots, \mathsf{q_e}$ do $R_{i,j} \leftarrow\!\!\$\, \{0,1\}^r$
    $i \leftarrow 0$ ; $b \leftarrow\!\!\$\, \{0,1\}$ ; $g \leftarrow\!\!\$\, \{1, \ldots, \mathsf{q_e}\}$ ; $m \leftarrow\!\!\$\, \{1, \ldots, \mathsf{q_e}\}$
```

If $g = 1$ it outputs 0 and halts. Else it picks $h$ at random from $\{1, \ldots, g-1\}$. (For this to make sense the set must be non-empty which is why we only do it if $g > 1$.)

Now if $m \ge g$ then $B_3$ outputs 0 and halts. Else —this happens when $1 \le m \le g - 1$ hence with probability $(g-1)/\mathsf{q_e}$— it runs $A$. It replies to Enc query $(j, H, \phi)$ of $A$ via

```
    $i \leftarrow i + 1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$
    If $(i \le g)$ then
        $M \leftarrow \text{EVAL}^{\text{IHASH}}(\phi, L_1, \ldots, L_w)$
    If $(i \le h - 1)$ then $C_i \leftarrow\!\!\$\, \{0,1\}^c$
    If $(i = h)$ then $C_i \leftarrow\!\!\$\, \text{ENC}(H, M)$
    If $(h < i < g)$ then $C_i \leftarrow\!\!\$\, \mathcal{E}(\text{IHASH}(R_{i,j} \,\|\, L_j), H, M)$
    If $(i = g$ and $b = 1)$ then $s \leftarrow j$ ; $C_i \leftarrow\!\!\$\, \mathcal{E}(K, H, M)$
    If $(i = g$ and $b = 0)$ then $s \leftarrow j$ ; $C_i \leftarrow\!\!\$\, \{0,1\}^c$
```

```
If $i > g$ then $C_i \leftarrow\!\!\$\, \{0,1\}^c$
$S_j \leftarrow S_j \cup \{(R_{i,j}, H, C_i)\}$
Return $(R_{i,j}, C_i)$
```

where ENC in this code is $B_3$'s own encryption oracle. It replies to IHASH, HASH queries as in the games of Figure 6. It replies to DEC query $j, R, H, C$ via

```
If $(R, H, C) \in S_j$ then return $\perp$
If $(R \,\|\, L_j = R_{g,s} \,\|\, L_s)$ then
    $V \leftarrow \text{DEC}(H, C)$
    Return $V$
Else $M \leftarrow \mathcal{D}(\text{IHASH}(R \,\|\, L_j), H, C)$
If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
Return $V$
```

where DEC called in this code is $B_3$'s own decryption oracle. When $A$ halts, adversary $B_3$ outputs 1 if bad was set and 0 otherwise. We have

$$\Pr[\mathrm{E}_{1,1}^{B_3}] \;=\; \frac{1}{\mathsf{q_e}}\sum_{g=2}^{\mathsf{q_e}} \frac{1}{g-1}\frac{g-1}{\mathsf{q_e}}\sum_{h=1}^{g-1} \Pr\left[\,\mathrm{I}_{g,h-1}^{A}\,\right] \;=\; \frac{1}{\mathsf{q_e}^2}\sum_{g=2}^{\mathsf{q_e}}\sum_{h=1}^{g-1} \Pr\left[\,\mathrm{I}_{g,h-1}^{A}\,\right]$$

and

$$\Pr[\mathrm{E}_{0,1}^{B_3}] \;=\; \frac{1}{\mathsf{q_e}}\sum_{g=2}^{\mathsf{q_e}} \frac{1}{g-1}\frac{g-1}{\mathsf{q_e}}\sum_{h=1}^{g-1} \Pr\left[\,\mathrm{I}_{g,h}^{A}\,\right] \;=\; \frac{1}{\mathsf{q_e}^2}\sum_{g=2}^{\mathsf{q_e}}\sum_{h=1}^{g-1} \Pr\left[\,\mathrm{I}_{g,h}^{A}\,\right] .$$

Subtracting we get

$$\Pr[\mathrm{E}_{1,1}^{B_3}] - \Pr[\mathrm{E}_{0,1}^{B_3}] \;=\; \frac{1}{\mathsf{q_e}^2}\sum_{g=2}^{\mathsf{q_e}} \left(\Pr\left[\,\mathrm{I}_{g,0}^{A}\,\right] - \Pr\left[\,\mathrm{I}_{g,g-1}^{A}\,\right]\right) \;=\; \frac{1}{\mathsf{q_e}^2}\sum_{g=1}^{\mathsf{q_e}} \left(\Pr\left[\,\mathrm{I}_{g,0}^{A}\,\right] - \Pr\left[\,\mathrm{I}_{g,g-1}^{A}\,\right]\right) .$$

In the last step we were able to start the summation index at 1 rather than 2 because if $g = 1$ then games $\mathrm{I}_{g,0}$ and $\mathrm{I}_{g,g-1}$ are the same.

## 4.9 Proof of Lemma 4.6

We have

$$\begin{aligned}
&\frac{1}{\mathsf{q_e}}\sum_{g=1}^{\mathsf{q_e}} \Pr\left[\,\mathrm{I}_{g,0}^{A}\,\right] \\
&\quad\le\; \mathsf{q_e}\cdot\left(\Pr[\mathrm{E}_{1,1}^{B_3}] - \Pr[\mathrm{E}_{0,1}^{B_3}]\right) + \frac{1}{\mathsf{q_e}}\sum_{g=1}^{\mathsf{q_e}} \Pr\left[\,\mathrm{I}_{g,g-1}^{A}\,\right] \\
&\quad\le\; \mathsf{q_e}\cdot\left(\Pr[\mathrm{E}_{1,1}^{B_3}] - \Pr[\mathrm{E}_{0,1}^{B_3}]\right) + \frac{1}{2}\left(\Pr[\mathrm{E}_{1,1}^{B_2}] - \Pr[\mathrm{E}_{0,1}^{B_2}]\right) + \frac{4w\mathsf{q_h} + w(w-1)}{2^{\mathbf{H}_\infty(\mathcal{L})+1}} . \qquad (13)
\end{aligned}$$

From Lemma 4.2, Equation (4), Equation (5) and Equation (13) we have

$$\begin{aligned}
&\Pr[\mathrm{F}_{1,1}^{A}] - \Pr[\mathrm{F}_{0,1}^{A}] \\
&\quad\le\; 4\mathsf{q_e}^2\cdot\sum_{j=1}^{3}\left(\Pr[\mathrm{E}_{1,1}^{B_j}] - \Pr[\mathrm{E}_{0,1}^{B_j}]\right) + \frac{4w\mathsf{q_e}\mathsf{q_h} + w(w-1)\mathsf{q_e}}{2^{\mathbf{H}_\infty(\mathcal{L})}} + \frac{2\mathsf{q_e}(\mathsf{q_h} + 2\mathsf{q_e}w)}{2^r} .
\end{aligned}$$

Let adversary $B_1'$ pick $j$ at random in $\{1,2,3\}$ and run $B_j$. Now apply Lemma 4.8 (Appendix 4.11) to $B_1'$.

## 4.10 Proof of Lemma 4.7

We have

$$\begin{aligned}
\frac{1}{\mathsf{q_d}}\left(\Pr[\mathrm{F}_{0,1}^A] - \Pr[\mathrm{F}_{0,0}^A]\right) &= 2\Pr[\mathrm{J}^A] - 1 && (14)\\
&= 2(\Pr[\mathrm{L}^A] + \Pr[\mathrm{J}^A] - \Pr[\mathrm{L}^A]) - 1 \\
&\le 2\Pr[\mathrm{L}^A] - 1 + 2\Pr[\mathsf{BAD}(\mathrm{L}^A)] \,. && (15)
\end{aligned}$$

Equation (15) follows from the Fundamental Lemma of Game Playing [14] because games $\mathrm{J}, \mathrm{L}$ are identical until $\mathsf{bad}$, meaning differ only in code following the setting of $\mathsf{bad}$ to $\mathsf{true}$. To justify Equation (14) note that when $b = 1$, decryption under the first $g$ keys yields real responses, the rest $\perp$, and when $b = 0$ decryption under the first $g - 1$ keys yields real responses, the rest $\perp$. The boxed code ensures that key assignments remain consistent with responses to HASH queries.

Adversary $B_4$ begins with initializations

```
For $j = 1, \ldots, w$ do
    $L_j \leftarrow_\$ \mathcal{L}$ ; $S_j \leftarrow \emptyset$
    For $i = 1, \ldots, \mathsf{q_e}$ do $R_{i,j} \leftarrow_\$ \{0,1\}^r$
$g \leftarrow_\$ \{1, \ldots, \mathsf{q_d}\}$
For $d = 1, \ldots, \mathsf{q_d}$ do
    If $(d \neq g)$ then $K[d] \leftarrow_\$ \mathcal{K}$
$i, d \leftarrow 0$
```

Now $B_4$ runs $A$. It replies to ENC query $j, H, \phi$ of $A$ via

```
$i \leftarrow i + 1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$
$C_i \leftarrow_\$ \{0,1\}^c$ ; $S_j \leftarrow S_j \cup \{(R_{i,j}, H, C_i)\}$
Return $(R_{i,j}, C_i)$
```

$B_4$ replies to HASH query $R \,\|\, L$ via

```
If not $H[R \,\|\, L]$ then $H[R \,\|\, L] \leftarrow_\$ \{0,1\}^k$
Return $H[R \,\|\, L]$
```

It replies to DEC query $j, R, H, C$ via

```
If $(R, H, C) \in S_j$ then return $\perp$
If not $\mathrm{Ind}[R \,\|\, L_j]$ then $d \leftarrow d + 1$ ; $\mathrm{Ind}[R \,\|\, L_j] \leftarrow d$
$e \leftarrow \mathrm{Ind}[R \,\|\, L_j]$
If $(e < g)$ then $M \leftarrow \mathcal{D}(K[e], H, C)$
If $(e = g)$ then
    $V \leftarrow \mathrm{DEC}(H, C)$
    Return $V$
If $(e > g)$ then $M \leftarrow \perp$
If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
Return $V$
```

where DEC called in this code is $B_4$'s own decryption oracle. When $A$ halts with output $b'$, so does adversary $B_4$. Think of the key $K[g]$ of game L as being the one underlying games $\mathrm{E}_{0,1}, \mathrm{E}_{0,0}$ for $B_4$. Thus

$$2\Pr[\mathrm{L}^A] - 1 = \Pr[\mathrm{E}_{0,1}^{B_4}] - \Pr[\mathrm{E}_{0,0}^{B_4}] \,.$$

## 4.11 Splitting lemma

**Lemma 4.8** *Let* $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a symmetric encryption scheme. Let* $B_1, B_2$ *be adversaries. Then there are adversaries* $D_1, D_2$ *such that*

$$\Pr[\mathrm{E}_{1,1}^{B_1}] - \Pr[\mathrm{E}_{0,1}^{B_1}] \leq 2 \cdot \mathbf{Adv}_{\mathsf{SE}}^{\mathrm{ae}}(D_1) \tag{16}$$

$$\Pr[\mathrm{E}_{0,1}^{B_2}] - \Pr[\mathrm{E}_{0,0}^{B_2}] \leq 2 \cdot \mathbf{Adv}_{\mathsf{SE}}^{\mathrm{ae}}(D_2)\ . \tag{17}$$

*The running times and number of oracle queries of* $D_1, D_2$ *equal those of* $B_1, B_2$ *respectively.* ▌

**Proof:** We will construct $D_{1,0}, D_{1,1}$ such that

$$\Pr[\mathrm{E}_{1,1}^{B_1}] - \Pr[\mathrm{E}_{0,0}^{B_1}] \leq \mathbf{Adv}_{\mathsf{SE}}^{\mathrm{ae}}(D_{1,0})$$

$$\Pr[\mathrm{E}_{0,0}^{B_1}] - \Pr[\mathrm{E}_{0,1}^{B_1}] \leq \mathbf{Adv}_{\mathsf{SE}}^{\mathrm{ae}}(D_{1,1})\ .$$

Let $D_1$ pick $d \in \{0, 1\}$ at random and run $D_{1,d}$ and Equation (16) follows by adding the equations above. The construction of $D_{1,0}$ is straightforward. Adversary $D_{1,1}$ runs $B_1$, replying to Enc queries by random strings and to Dec queries via its own Dec oracle, and returns $1 - b'$ where $b'$ is the output of $B_1$. The proof of Equation (17) is similar and is omitted. ▌

## 5 Misuse Resistant Authenticated Encryption

r nonces are to be randomly chosen, with the adversary having no control over the process. But, in real systems, if the PRNG used is broken or if the adversary gains control over the system, the nonce can no longer be assumed to be random. To capture these conditions, we look at two misuse resistance scenarios: MR1 security, where the nonce is not random but guaranteed to be unique each time, say due to a bad PRNG, and MR2 security, where we make no assumptions on the nonce: the adversary may provide any nonce it feels like. The security guarantees that can be provided under such conditions are weaker, but we show that our scheme degrades gracefully, providing security against a large class of functions. We note that the MR1 setting is the same as universal nonces but we make the distinction to emphasize that we are aiming for a weaker kind of security not ruled out by our impossibility results. Misuse resistant authenticated encryption was first considered by Rogaway and Shrimpton in [47].

Definitions. AE-security of a scheme $\mathsf{SE}$ under MR1 and MR2 can be defined via the same game $\mathrm{KIAE}_{\mathsf{SE}}$ and corresponding advantage statement we used above in the randomized case, with appropriate relaxations on the nonces. When $\mathtt{nt} = \mathtt{mr1}$, $\mathrm{KIAE}_{\mathsf{SE}}$ captures MR1 security, the adversary being constrained not to repeat nonces. Setting $\mathtt{nt} = \mathtt{mr2}$ captures MR2 security, which means that the nonces may be chosen arbitrarily by the adversary. We base the security on the type of adversary: a MR1 adversary always chooses a fresh nonce and a MR2 adversary need not. KDM security under MR1 and MR2 is defined analogously for an adversary $A \in \mathcal{A}[\Phi, \mathtt{ki}]$.

Restrictions on $\Phi$. Having seen that it is not possible to achieve full KDM security under MR1 or MR2 (MR1 security is the same as universal nonce security, as mentioned earlier), we now look at classes of functions $\Phi$ that RHtE can support under MR1 and MR2. In MR1, $\Phi$ cannot contain Hash. Otherwise, the adversary can choose $\phi = \textsc{Hash}(K \,\|\, N)$ and set the nonce to be $N$, forcing the encryption of the key under itself in the base scheme. We note that deterministic AE is a special case of MR2, one where the adversary chooses to use a fixed nonce. In deterministic encryption schemes, KDM security is not meaningful for all classes of functions. The adversary can learn some properties of the key by checking the equality of returned ciphertexts either between KDM queries or between KDM and normal queries. For example, the adversary can learn the most significant bit of the ciphertext by comparing its encryption to the encryption of 0 and 1. Such issues do not occur in randomized KDM encryption, but are unavoidable when only one ciphertext is possible for a given plaintext. As a result, we would require that $\Phi$ only contain functions whose value cannot be guessed by the adversary (unless they are constant) on a random key with non-negligible probability and that it be hard to pick collisions between functions in $\Phi$.

To formalize this requirement on $\Phi$, we recall the notions of output unpredictability and collision resistance for a class of functions from [10]. Let $\Phi$ consist of functions $\phi : (\{0,1\}^l)^w \to \{0,1\}^*$. We devise a game $\mathrm{OPCR}_{\Phi,w,r,t,\mathcal{L}}$, which consists of a single FINALIZE routine to which an adversary playing the game submits two sets $(P, X)$, $P \subset \Phi$ and $X \subset \{0,1\}^*$. The game generates $L_j \leftarrow_\$ \mathcal{L}, j \in \{1, 2, \ldots, w\}$. The adversary wins if $\{\phi(L_1, L_2, \ldots, L_w) : \phi \in P\} \cap X \neq \emptyset$ or $|\{\phi(L_1, L_2, \ldots, L_w) : \phi \in P\}| < |P|$, with $|P| < r$ and $|X| < t$. In other words, the adversary wins if two functions in $P$ have the same value or if one of the functions in $\Phi$ evaluates to one of the elements of $X$. We denote $\Pr[\mathrm{OPCR}^A_{\Phi,w,r,t,\mathcal{L}} \Rightarrow \mathsf{true}]$ by $\mathbf{Adv}^{\mathrm{opcr}}_{\Phi,w,r,t,\mathcal{L}}(A)$. If the keyspace is viewed as the finite field $\mathbb{F}$ of size $2^{\mathbf{H}_\infty(\mathcal{L})}$, the set of polynomials over $\mathbb{F}^n$ with non zero degree is an example of a $\Phi$ with good collision resistance and output unpredictability properties.

We can now state the security of $\overline{\mathsf{SE}} = \mathsf{RHtE}[\mathsf{SE}, \mathcal{L}, r]$ under MR1 and MR2. Let $A$ be an adversary that makes message queries from a class $\Phi$ that does not contain HASH. We releate the security of $\overline{\mathsf{SE}}$ to the security of $\mathsf{SE}$ with the same kind of nonces.

**Theorem 5.1** *For every MR1 adversary $A \in \mathcal{A}[\Phi, \mathtt{ki}]$, there exists an adversary $B_1$ such that*

$$\mathbf{Adv}^{\text{ae-mr1}}_{\overline{\mathsf{SE}}}(A) \leq 2(\mathsf{q_e} + \mathsf{q_d}) \cdot \mathbf{Adv}^{\text{ae-mr1}}_{\mathsf{SE}}(B_1) + \frac{2w \cdot q_h}{2^{\mathbf{H}_\infty(\mathcal{L})}},$$

*where $\mathsf{q_e}, \mathsf{q_d}$ and $\mathsf{q_h}$ are the number of encryption, decryption and random oracle queries made by $A$ respectively and $w$ is the number of keys, provided by $A$ as the argument to* INITIALIZE.

**Theorem 5.2** *For every MR2 adversary $A \in \mathcal{A}[\Phi, \mathtt{ki}]$, there exist adversaries $B_1$ and $B_2$ such that*

$$\mathbf{Adv}^{\text{ae-mr2}}_{\overline{\mathsf{SE}}}(A) \leq 2(\mathsf{q_e} + \mathsf{q_d}) \cdot \mathbf{Adv}^{\text{ae-mr2}}_{\mathsf{SE}}(B_1) + 2 \cdot \mathbf{Adv}^{\mathrm{opcr}}_{\Phi',w,\mathsf{q_{kd}},\mathsf{q_{ki}},\mathcal{L}}(B_2) + \frac{2w \cdot q_h}{2^{\mathbf{H}_\infty(\mathcal{L})}},$$

*where $\mathsf{q_e}, \mathsf{q_d}$ and $\mathsf{q_h}$ are the number of encryption, decryption and random oracle queries, $w$ is the number of keys, provided by $A$ as the argument to* INITIALIZE *and $\mathsf{q_{ki}}$ and $\mathsf{q_{kd}}$ are the numbers of normal and key-dependent encryption queries made by $A$ respectively. $\Phi' = \Phi - \mathsf{Cns}(l, w)$.*

The proofs proceed similar to the proof in the random nonce case, but are simpler since HASH is not a part of $\Phi$. Towards proving these two theorems, we introduce two games, G and H (Figure 9). G is similar to $\mathrm{KDAE}_{\overline{\mathsf{SE}}}$ (Figure 2), except that keys $K_i$, which would have been generated during the encryption and decryption routines of $\overline{\mathsf{SE}}$ are drawn explicitly during initialization. This would not change the game's behavior, as long as the simulated random oracle is maintained correctly: if $R[L_j||N]$ is already defined during the $i$-th encryption query $\mathrm{ENC}(j, N, \cdot, \cdot)$, $K_i$ is updated to this value. Every time the adversary queries ENC with a fresh nonce-key pair, a new $K_i$ is used for encryption and $R$ is updated to reflect this change. DEC uses a new $K_i$ or the appropriate existing one depending on the nonce and key provided by the adversary. We have

$$\Pr[\mathrm{KDAE}^A_{\overline{\mathsf{SE}}}] = \Pr[\mathrm{G}^A]. \tag{18}$$

H and G are identical-until-bad games. To see this, we note that the difference between the two games is in how $R$ is maintained. Specifically, in H, $R$ is not updated when a new $K_i$ is used after the adversary makes an encryption or decryption query with a new key-nonce pair. The adversary cannot notice this unless it makes a query to HASH with a key-nonce pair that was queried already. From the code of HASH, such a query would set $\mathsf{bad}$. H does not revert $K_i$ when $R[L_j \,\|\, N]$ is already defined at line 106, but this follows after $\mathsf{bad}$ is set. Moreover, it can be shown that

$$\Pr[\mathsf{BAD}(\mathrm{H}^A, 106)] = 0.$$

```
proc INITIALIZE  // $\boxed{G}$,H
000 For $j = 1, \ldots, w$ do
001     $L_j \leftarrow\!\!\$\ \mathcal{L}$ ; $S_j \leftarrow \emptyset$
002 For $i = 1, \ldots, \mathsf{q_e} + \mathsf{q_d}$ do
003     $K_i \leftarrow\!\!\$\ \{0,1\}^k$ ; $T_i \leftarrow \perp$
004 For $x \in \{0,1\}^*$ do $R[x] \leftarrow \perp$
005 $i \leftarrow 0$ ; $b \leftarrow\!\!\$\ \{0,1\}$ ; $\mathsf{bad} \leftarrow \mathsf{true}$

proc ENC$(j, N, H, \phi)$  // $\boxed{G}$, H
100 $i \leftarrow i + 1$ ; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |H|)$ ; $p = i$
101 $M \leftarrow \mathrm{EVAL}(\phi, L_1, \ldots, L_n)$
102 For $t \in 1, \ldots i - 1$ do
103     If $T_t = (N, j)$ then $p = t$
104 If $p = i$ and $R[L_j \,\|\, N] \neq \perp$ then
105     $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{K_i \leftarrow R[L_j \,\|\, N]}$
106 $C_1 \leftarrow \mathcal{E}(K_p, M, H, N)$ ; $C_0 \leftarrow\!\!\$\ \{0,1\}^c$
107 $T_p \leftarrow (N, j)$ ; $\boxed{R[L_j \,\|\, N] \leftarrow K_p}$
108 $S_j \leftarrow S_j \cup \{(N, C, H)\}$
109 Return $(N, C_b)$

proc HASH$(L \,\|\, N)$  // $\boxed{G}$, H
200 For $j \in 1, \ldots, w$ do
201     If $(L = L_j)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
202 If $R[L \,\|\, N] = \perp$ then $R[L \,\|\, N] \leftarrow\!\!\$\ \{0,1\}^k$
203 Return $R[L \,\|\, N]$

proc DEC$(j, N, H, C)$  // $\boxed{G}$,H
300 If $(N, H, C) \in S_j$ then return $\perp$
301 For $t \in 1, \ldots i - 1$ do
302     If $T_t = (N, j)$ then
303         $M \leftarrow \mathcal{D}(K_t, N, H, C)$
304         If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
305         Return $V$
306 $i \leftarrow i + 1$ ; $T_i \leftarrow (N, j)$
307 $\boxed{R[L_j \,\|\, N] \leftarrow K_p}$
308 $M \leftarrow \mathcal{D}(K_i, C, N, H)$
309 If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
310 Return $V$

proc FINALIZE$(b')$  // $\boxed{G}$,H
400 Return $(b' = b)$
```

Figure 9: Games $\boxed{\mathrm{G}}$,H for the proofs of Theorem 5.1 and Theorem 5.2. A box around the name of a game next to a procedure means the boxed code of that procedure is included in the game.

Line 106 checks if R at a fresh nonce key pair had been already defined. The only way this could have happened is that $A$ queried HASH at this point. But such a query would have already set bad in HASH. This is why we require HASH to be excluded from $\Phi$; otherwise, setting bad at 106 would be trivial. In H, the keys $K_i$ are not used anywhere else except in the $\mathcal{E}$ and $\mathcal{D}$ calls, which makes it possible to simulate H if access to an encryption and decryption oracle is provided. We show that if $A$ is a MR1 adversary, there exists a MR1 adversary for $\mathsf{SE}$, $B_1$, running INITIALIZE with $\mathsf{q_e} + \mathsf{q_d}$ keys such that

$$\mathbf{Adv}^{\text{ae-mr1}}_{\mathsf{SE}}(B_1) \geq \Pr[\mathrm{H}^A | b = 1] - \Pr[\mathrm{H}^A | b = 0]. \tag{19}$$

We note that $B_1$ is a KIAE adversary, but with multiple keys. The same equation holds in the MR2 case. Given an adversary $A$ playing H, an adversary $B_1$ can be constructed that simulates $A$ to play $\mathrm{KIAE}_{\mathsf{SE},\mathsf{q_e}+\mathsf{q_d}}$. $B_1$ chooses $w$ $l$-bit strings $L_1, L_2, \ldots L_w$ randomly and runs $A$ with these as the keys. The keys $K_1, \ldots K_{\mathsf{q_e}+\mathsf{q_d}}$ in H correspond to the keys in the $\mathrm{KIAE}_{\mathsf{SE}}$ game with $\mathsf{q_e}+\mathsf{q_d}$ keys. $B_1$ implements a random oracle HASH that $A$ can query. To handle an encryption query with $L_j$, $B_1$ maps $\mathrm{HASH}(L_j \,\|\, N_i)$ to key $i$ in $\mathrm{KIAE}_{\mathsf{SE}}$ (although $B_1$ does not know those keys' values) where $N_i$ is the $i^{th}$ unique nonce. This way, $B_1$ can forward all the encryption and decryption queries with $K_i$ in H to the ENC and DEC oracles with key $i$ in $\mathrm{KIAE}_{\mathsf{SE}}$, after evaluating the function. Finally, $B_1$ outputs 1 if $A$ outputs 1. $B_1$ does not have to do anything else with $K_i$ and can simulate H correctly. Furthermore, $B_1$ uses the same nonces as those provided by $A$. $B_2$ is similar to $B_1$ but also keeps track of bad. From the previous arguments, to do so, $B_2$ only needs to monitor queries to HASH and check if any such query contains one of $L_i$ as a prefix. It outputs 1 if bad was set.

$$\mathbf{Adv}^{\text{ae-mr1}}_{\mathsf{SE}}(B_2) \leq \Pr[\mathsf{BAD}(\mathrm{H}^A) | b = 1] - \Pr[\mathsf{BAD}(\mathrm{H}^A) | b = 0]. \tag{20}$$

Now we bound the probability of bad getting set in H when $b = 0$. In the MR1 case, each encryption query gets a fresh nonce and hence returns a randomly generated ciphertext. $A$ learns nothing about the keys $L_i$ during the interaction. It follows that

$$\Pr[\mathsf{BAD}(\mathrm{H}^A)|b=0] \leq \frac{n \cdot q_h}{2^{\mathbf{H}_\infty(\mathcal{L})}}. \tag{21}$$

This argument would not hold in the MR2 case, however. The adversary could repeat the same nonce and the previous discussion about deterministic encryption would apply. To handle this, let us divide $\Phi$ into $\mathsf{Const} = \mathsf{Cns}(l, w)$, the set of constant functions and $\Phi' = \Phi - \mathsf{Const}$. Rather than bound the probability of setting $\mathsf{bad}$ combinatorially, we are required to relate it to $\Phi$. We show that there exists an adversary $B_3$ for every MR2 adversary $A$ such that

$$\Pr[\mathsf{BAD}(\mathrm{H}^A)|b=0] \leq \mathbf{Adv}^{\mathrm{opcr}}_{\Phi',w,\mathsf{q_{kd}},\mathsf{q_{ki}},\mathcal{L}}(B_3) + \frac{n \cdot q_h}{2^{\mathbf{H}_\infty(\mathcal{L})}}. \tag{22}$$

where $\mathsf{q_{ki}}$ and $\mathsf{q_{kd}}$ are the numbers of normal and key-dependent encryption queries and $\mathsf{q_h}$ is the number of random oracle queries made by $A$ respectively. Let $E$ denote the event that two of $A$'s queries produce the same ciphertext. $A$ does not learn anything about any of the keys unless two of its queries produce the same ciphertext since all the ciphertexts are random. Consequently, $\Pr[\mathsf{BAD}(\mathrm{H}^A)|b=0 \wedge \neg E] = \frac{n \cdot q_h}{2^{\mathbf{H}_\infty(\mathcal{L})}}$. Let $B_3$ be an adversary playing $\mathrm{OPCR}_{\Phi',w,\mathsf{q_{kd}},\mathsf{q_{ki}},\mathcal{L}}$. $B_3$ simulates a random oracle, returns random ciphertexts for any $\textsc{Enc}$ queries made by $A$ (while tracking them) and $\perp$ for any $\textsc{Dec}$ queries. This is an exact simulation of a H execution when $b = 0$ till $A$ makes two $\textsc{Enc}$ queries that would have returned the same ciphertext ($A$ would have caused $E$ ). But such a pair of queries would mean that either two queries $A$ made with functions from $\Phi'$ have collided or $A$ guessed the value of a function in $\Phi'$. Either way, this would correspond to a win for $B_3$ in its $\mathrm{OPCR}_{\Phi',n}$ game.

$$\begin{aligned}
\Pr[E] &\leq \mathbf{Adv}^{\mathrm{opcr}}_{\Phi',w,\mathsf{q_{kd}},\mathsf{q_{ki}},\mathcal{L}}(B_3)\\
\Pr[\mathsf{BAD}(\mathrm{H}^A)|b=0] &\leq \Pr[E] + \Pr[\mathsf{BAD}(\mathrm{H}^A)|b=0 \ \wedge\ \neg E]\\
&\leq \mathbf{Adv}^{\mathrm{opcr}}_{\Phi',w,\mathsf{q_{kd}},\mathsf{q_{ki}},\mathcal{L}}(B_3) + \frac{n \cdot q_h}{2^{\mathbf{H}_\infty(\mathcal{L})}}.
\end{aligned}$$

$B_3$ runs in time comparable to $A$. Returning to the proof of Theorem 5.1 and Theorem 5.2, we have shown that G and H are identical-until-bad, H can be simulated and bounded $\Pr[\mathsf{BAD}(\mathrm{H}^A)|b=0]$. We can construct a $B$ ( Lemma 5.3) running in time comparable to $B_1$ and invoking $\textsc{Initialize}$ with just one key instead of $\mathsf{q_e} + \mathsf{q_d}$ keys, such that

$$\mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE}}(B_1) \leq (\mathsf{q_e} + \mathsf{q_d})\mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE}}(B).$$

Combining this with equations (18, 19, 20, 21 and 22) proves the statement of the theorem.

$$\begin{aligned}
\mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE}}(A) &= \Pr[\mathrm{G}^A|b=1] - \Pr[\mathrm{G}^A|b=0]\\
&\leq \Pr[\mathrm{H}^A|b=1] - \Pr[\mathrm{H}^A|b=0] + \Pr[\mathsf{BAD}(\mathrm{H}^A)]\\
&\leq \Pr[\mathrm{H}^A|b=1] - \Pr[\mathrm{H}^A|b=0] + 2\Pr[\mathsf{BAD}(\mathrm{H}^A)|b=0]\\
&\quad + \Pr[\mathsf{BAD}(\mathrm{H}^A)|b=1] - \Pr[\mathsf{BAD}(\mathrm{H}^A)|b=0]\\
&\leq \mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE},\mathsf{q_e}+\mathsf{q_d}}(B_1) + \mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE},\mathsf{q_e}+\mathsf{q_d}}(B_2) + \frac{2n \cdot \mathsf{q_h}}{2^{\mathbf{H}_\infty(\mathcal{L})}}\\
&\leq 2(\mathsf{q_e} + \mathsf{q_d})\mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE}}(B) + \frac{2w \cdot \mathsf{q_h}}{2^{\mathbf{H}_\infty(\mathcal{L})}}.
\end{aligned}$$

The MR2 case follows in a similar manner, with the additional $B_3$ term. We note that the running time of $B_1$ depends on the key dependent queries made by $A$ since $B_1$ has to evaluate them.

```
proc INITIALIZE($w$)
For $i = 1, \ldots, w$ do $K_i \leftarrow_{\$} \mathcal{K}$ ; $S_i \leftarrow \emptyset$
$s \leftarrow_{\$} \{1, 2, \ldots, w\}$ ; $b \leftarrow_{\$} \{0, 1\}$

proc ENC($i, N, H, M$)
If $(s + b \leq i)$ then $C \leftarrow \mathcal{E}(K_i, N, H, M)$
Else $c \leftarrow \mathsf{cl}(|M|, |H|)$ ; $C \leftarrow_{\$} \{0, 1\}^c$
$S_i \leftarrow S_i \cup \{(H, C)\}$ ; Return $C$

proc DEC($i, H, C$)
If $(H, C) \in S_i$ then Return $\perp$
If $(s + b \leq i)$ then $M \leftarrow \mathcal{D}(K_i, H, C)$
Else $M \leftarrow \perp$
If $M = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
Return $V$

proc FINALIZE($b'$)
Return $(b' = b)$
```

Figure 10: Game G for Lemma 5.3.

| Hash | Scheme | RHtE Relative Running Time | | | |
|---|---|---|---|---|---|
| | | KeySetup | 5KB | 50KB | 500KB |
| SHA256 | CCM | 2.73 | 1.11 | 1.01 | 1.00 |
| | EAX | 1.94 | 1.10 | 1.01 | 1.00 |
| | GCM-2k | 1.66 | 1.10 | 1.02 | 1.00 |
| | GCM-64k | 1.19 | 1.09 | 1.02 | 1.00 |

Figure 11: Table showing relative slowdown of RHtE with SHA256 in Crypto++ for common AE schemes and different message sizes. KeySetup is the relative slowdown in the keysetup phase alone. GCM-2k and GCM-64k correspond to GCM implemented with tables of corresponding size.

**Lemma 5.3** *For every adversary $A$, there exists $B$ running in comparable time such that*

$$\mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE}}(A) \leq w \cdot \mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE}}(B).$$

*where $w$ is the number of keys, provided by $A$ to* INITIALIZE.

Proof: We consider a game G (Figure 10) which runs a hybrid between KIAE with $b = 0$ and $b = 1$. We have

$$\frac{1}{w}\mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE}}(A) = 2\Pr[\mathrm{G}^A] - 1.$$

A KIAE$_{\mathsf{SE}}$ adversary $B$ can be constructed as follows: $B$ picks a random $s < n$, generates $s$ keys to handle the encryption and decryption queries with the first $s$ keys, forwards queries with key $s + 1$ to its own ENC and DEC oracles and returns returns random answers and $\perp$ to encryption and decryption queries with other keys. Clearly, $2\Pr[\mathrm{G}^A] - 1 \leq \mathbf{Adv}^{\text{ae-}\mathtt{mr1}}_{\mathsf{SE}}(B)$, proving the claim. The proof for the MR2 case is similar.

## 6 Implementation results

We recall that RHtE works on an existing AE scheme and a hash function. We ran RHtE with common AE schemes like CCM, EAX and GCM (with tables of 2k and 64k entries) to measure the slowdown relative to the original schemes, using a truncated version of SHA256 as the hash function and setting $l = r = k = 128$. We ran these tests using Crypto++ [25], a standard cryptography library. The measurements in Figure 11 correspond to a Intel Core i5 M460 64-bit CPU running at 2.53 GHz with code compiled using g++ -O3 for data sizes small enough to fit in the level 2 cache. For our purposes, the relative performance of these routines is of more importance. From Figure 11, we can observe that even at modest message sizes of around 50KB, the slowdown due to RHtE is no more than 1%. Futhermore, if algorithms like GCM are implemented with large tables and in turn a lot of precomputation in the key-setup phase, the RHtE overhead is even less noticeable.

## References

[1] P. Abeni, L. Bello, and M. Bertacchini. Exploiting DSA-1571: How to break PFS in SSL with EDH, July 2008. http://www.lucianobello.com.ar/exploiting_DSA-1571/index.html. (Cited on page 4.)

[2] T. Acar, M. Belenkiy, M. Bellare, and D. Cash. Cryptographic agility and its relation to circular encryption. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 403–422. Springer, May 2010. (Cited on page 4.)

[3] B. Applebaum. Key-dependent message security: Generic amplification and completeness. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 527–546. Springer, May 2011. (Cited on page 1, 4.)

[4] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Aug. 2009. (Cited on page 1, 4.)

[5] M. Backes, M. Dürmuth, and D. Unruh. OAEP is secure under key-dependent messages. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 506–523. Springer, Dec. 2008. (Cited on page 4.)

[6] M. Backes, B. Pfitzmann, and A. Scedrov. Key-dependent message security under active attacks - brsim/uc-soundness of dolev-yao-style encryption with key cycles. *Journal of Computer Security*, 16(5):497–530, 2008. (Cited on page 1, 2, 4, 7.)

[7] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 423–444. Springer, May 2010. (Cited on page 1, 4.)

[8] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 232–249. Springer, Dec. 2009. (Cited on page 4.)

[9] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, Oct. 1997. (Cited on page 1, 6.)

[10] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, May 2003. (Cited on page 4, 24.)

[11] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Dec. 2000. (Cited on page 1, 2, 6, 7.)

[12] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, May 1994. (Cited on page 4.)

[13] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer, Dec. 2000. (Cited on page 6.)

[14] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. (Cited on page 5, 17, 22.)

[15] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 389–407. Springer, Feb. 2004. (Cited on page 1.)

[16] N. Bitansky and R. Canetti. On strong simulation and composable point obfuscation. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 520–537. Springer, Aug. 2010. (Cited on page 1, 4.)

[17] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In K. Nyberg and H. M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Aug. 2003. (Cited on page 1, 2, 3, 4, 7, 8, 12.)

[18] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision diffie-hellman. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, Aug. 2008. (Cited on page 1, 4.)

[19] Z. Brakerski and S. Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 1–20. Springer, Aug. 2010. (Cited on page 1, 4.)

[20] Z. Brakerski, S. Goldwasser, and Y. T. Kalai. Black-box circular-secure encryption beyond affine functions. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 201–218. Springer, Mar. 2011. (Cited on page 1, 4.)

[21] D. R. Brown. A weak randomizer attack on RSA-OAEP with e=3. IACR ePrint Archive, 2005. (Cited on page 4.)

[22] J. Camenisch, N. Chandran, and V. Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 351–368. Springer, Apr. 2009. (Cited on page 1, 4.)

[23] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001. (Cited on page 1, 4.)

[24] R. Canetti, Y. T. Kalai, M. Varia, and D. Wichs. On symmetric encryption and point obfuscation. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 52–71. Springer, Feb. 2010. (Cited on page 1, 4.)

[25] W. Dai. Crypto++ library. `http://www.cryptopp.com`. (Cited on page 27.)

[26] L. Dorrendorf, Z. Gutterman, and B. Pinkas. Cryptanalysis of the windows random number generator. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 476–485. ACM Press, Oct. 2007. (Cited on page 4.)

[27] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. *Journal of Cryptology*, 17(2):81–104, Mar. 2004. (Cited on page 4.)

[28] I. Goldberg and D. Wagner. Randomness in the Netscape browser. Dr. Dobb's Journal, January 1996. (Cited on page 4.)

[29] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. (Cited on page 1, 4, 6.)

[30] M. González. *Cryptography in the Presence of Key Dependent Messages.* PhD thesis, Florida Atlantic University, 2009. (Cited on page 5.)

[31] M. Green and S. Hohenberger. CPA and CCA-secure encryption systems that are not 2-circular secure. Cryptology ePrint Archive, Report 2010/144, 2010. `http://eprint.iacr.org/`. (Cited on page 4.)

[32] Z. Gutterman and D. Malkhi. Hold your sessions: An attack on Java session-id generation. In A. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 44–57. Springer, Feb. 2005. (Cited on page 4.)

[33] I. Haitner and T. Holenstein. On the (im)possibility of key dependent encryption. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 202–219. Springer, Mar. 2009. (Cited on page 4.)

[34] S. Halevi and H. Krawczyk. Security under key-dependent inputs. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 466–475. ACM Press, Oct. 2007. (Cited on page 5.)

[35] D. Hofheinz and D. Unruh. Towards key-dependent message security in the standard model. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 108–126. Springer, Apr. 2008. (Cited on page 1, 4.)

[36] J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In B. Schneier, editor, *FSE 2000*, volume 1978 of *LNCS*, pages 284–299. Springer, Apr. 2000. (Cited on page 2, 6.)

[37] T. Malkin, I. Teranishi, and M. Yung. Efficient circuit-size independent public key encryption with KDM security. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 507–526. Springer, May 2011. (Cited on page 1, 4.)

[38] D. A. McGrew and J. Viega. The security and performance of the Galois/counter mode (gcm) of operation. In A. Canteaut and K. Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, Dec. 2004. (Cited on page 1.)

[39] M. G. Muñiz and R. Steinwandt. Security of signature schemes in the presence of key-dependent messages. *Tatra Mt. Math. Publ.*, 47:15–29, 2010. (Cited on page 5.)

[40] M. Mueller. Debian OpenSSL predictable PRNG bruteforce SSH exploit, May 2008. `http://milw0rm.com/exploits/5622`. (Cited on page 4.)

[41] K. Ouafi and S. Vaudenay. Smashing SQUASH-0. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 300–312. Springer, Apr. 2009. (Cited on page 4.)

[42] K. G. Paterson and G. J. Watson. Plaintext-dependent decryption: A formal security treatment of SSH-CTR. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 345–361. Springer, May 2010. (Cited on page 5.)

[43] P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *ACM CCS 02*, pages 98–107. ACM Press, Nov. 2002. (Cited on page 1, 2, 6, 7.)

[44] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Dec. 2004. (Cited on page 1.)

[45] P. Rogaway. Nonce-based symmetric encryption. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, Feb. 2004. (Cited on page 1, 5, 6.)

[46] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 01*, pages 196–205. ACM Press, Nov. 2001. (Cited on page 1.)

[47] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, May / June 2006. (Cited on page 1, 2, 5, 6, 7, 23.)

[48] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). Undated manuscript. Submission to NIST, available from their web page, June 2002. (Cited on page 1.)

[49] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610 (Informational), Sept. 2003. (Cited on page 1.)

[50] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When private keys are public: Results from the 2008 Debian OpenSSL vulnerability. In *IMC*. ACM, 2009. (Cited on page 4.)