UNIVERSITY OF CALIFORNIA, SAN DIEGO

Encryption when messages and keys are related

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Sriram Keelveedhi Ravinarayanan

Committee in charge:

Professor Mihir Bellare, Chair
Professor Sam Buss
Professor Young-Han Kim
Professor Daniele Micciancio
Professor Hovav Shacham

2014

The Dissertation of Sriram Keelveedhi Ravinarayanan is approved and is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I wish to thank my advisor, Mihir Bellare, for all his support. His guidance and feedback have made me a better researcher and a better person. I admire his thoroughness and pursuit of perfection and I hope these attributes have become a part of me.

Chapter 2 is largely a reproduction of the material as it appears in Ciphers That Securely Encipher Their Own Keys, by Mihir Bellare, David Cash, and Sriram Keelveedhi from Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, 2011. I wish to thank David Cash for being an excellent co-author on this work and for being a knowledgeable yet friendly presence during my first couple of years.

Chapter 3 is largely a reproduction of the material as it appears in Authenticated and Misuse-resistant Encryption of Key-dependent Data, by Mihir Bellare and Sriram Keelveedhi from Proceedings of the 31st Annual Conference on Advances in Cryptology, CRYPTO 2011.

Chapter 4 is a reproduction of the material as it appears in Message-locked encryption and secure deduplication, by Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart from Advances in Cryptology, EUROCRYPT 2013. I wish to thank Thomas Ristenpart for several fruitful collaborations, and for guiding me towards improving my technical skill, writing style, and productivity.

Chapter 5 contains material from Instantiating random oracles via UCEs, by Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi from Advances in Cryptology, CRYPTO 2013. I wish to thank Viet Tung Hoang for being an excellent co-author in this work.

I would like to thank Sam Buss, Young-Han Kim, Daniele Micciancio, and Hovav Shacham for serving on my committee.

VITA

2009	Bachelor of Technology, Indian Institute of Technology, Madras
2011	Master of Science, Computer Science and Engineering University of California, San Diego
2014	Doctor of Philosophy, Computer Science and Engineering University of California, San Diego

PUBLICATIONS

Mihir Bellare and Sriram Keelveedhi. “Authenticated and Misuse-resistant Encryption of Key-dependent Data”. Proceedings of the 31st Annual Conference on Advances in Cryptology - CRYPTO 2011. 2011.

Mihir Bellare, David Cash, and Sriram Keelveedhi. “Ciphers That Securely Encipher Their Own Keys”. In: Proceedings of the 18th ACM Conference on Computer and Communications Security. CCS ‘11.

Keaton Mowery, Sriram Keelveedhi, and Hovav Shacham. “Are AES x86 Cache Timing Attacks Still Feasible?” Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop. CCSW ’12. 2012.

Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. “Message-locked encryption and secure deduplication”. Advances in Cryptology - EUROCRYPT 2013. 2013.

Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. “DupLESS: Server-aided Encryption for Deduplicated Storage”. Proceedings of the 22Nd USENIX Security Conference. USENIX Security ’13. 2013.

Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. “Efficient Garbling from a Fixed-Key Blockcipher”. Proceedings of the 2013 IEEE Symposium on Security and Privacy. SP ’13. 2013.

Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. “Instantiating random oracles via UCEs”. In: Advances in Cryptology - CRYPTO 2013.

ABSTRACT OF THE DISSERTATION

Encryption when messages and keys are related

by

Sriram Keelveedhi Ravinarayanan

Doctor of Philosophy in Computer Science

University of California, San Diego, 2014

Professor Mihir Bellare, Chair

We look at issues that arise when data being encrypted and keys used to encrypt the data are related. We look at the incidence of such relations in disk encryption and password-based encryption, and design practical solutions to address the issue. We explore problems in outsourced storage where keys and messages are related by design, and develop a new framework, message-locked encryption, to provide practical and theoretically interesting solutions. We look at a new framework for studying hash functions, called Universal Computational Extractors towards proving security for some of our constructions from cleaner assumptions.

Introduction

In this work, we look at issues that arise when data being encrypted and keys used to encrypt the data are related to each other. Such relations can occur in real systems as a consequence of inevitable operating conditions, or due to misuse or neglect in the parts of developers and users. Most cryptosystems are not built to deal with such relations and their security guarantees do not continue to hold in such settings. In some cases, the systems fail in obvious ways, while in other cases, the failures could be harder to detect, making them even more dangerous.

First, we look at the incidence of key message relations in the context of disk encryption and design practical solutions to address the issue. We then move on to study the problem in a more general setting, motivated by applications such as password-based encryption, and propose simple modifications to existing approaches to encryption that can secure systems in the presence of key-message relations.

There are scenarios where keys and messages might be related by design, with keys, instead of being picked at random, being generated from the messages themselves. We explore settings in outsourced storage where this occurs, and design and develop a new framework, message-locked encryption, to provide practical and theoretically interesting solutions. Some of our constructions are supported by proofs which make strong assumptions about the underlying primitives, particularly cryptographic hash functions. Specifically, the proofs view the hash functions as being instantiations of random functions. We look at a new framework for studying hash functions, called

Universal Computational Extractors, which enables us to circumvent viewing hash functions as instantiations of random functions. We now provide a high level overview of these results.

Key-enciphering security

Key-enciphering security requires that schemes can securely encrypt their own key as well messages containing the key as a substring. Encryption schemes have not been designed to work when encrypting their own keys in the past, based on the view that encryption of keys does not happen in practice.

However, there is a consensus among standardization groups, particularly among groups involved in disk encryption, that key enciphering security is an important property. Disk encryption must be length preserving, so we are talking about ciphers, not randomized encryption schemes. In email archives of discussions of candidate enciphering schemes for disk encryption, conducted by the Security in Storage Working Group (IEEE P1619), there is an intriguing comment "I have not received any meaningful response to the issue of grandma storing her keys on an encrypted drive. The WG must have considered it the first time it came up." Storing keys in the hard drive is safe only if the underlying scheme is key-enciphering secure. The group eventually concluded that key-enciphering security is important and desirable. The presence of key-enciphering attacks on one candidate (LRW) influenced its rejection in favor of others on which attacks were not found.

Meanwhile, the group did value security proofs. The encryption standard concerning encryption of disk sectors involves wideblock ciphers, which are block ciphers with input sizes much larger than conventional block ciphers such as AES. EME [79], a wideblock cipher, is proven to achieve PRP-CCA security. A variant EME2 [75] was standardized on the strength of its proofs, a host of efficiency properties, and the *presumed* security of enciphering the key. Given that key enciphering security is a requirement,

and that proofs are important, a discrepancy becomes obvious: while there is a proof of PRP-CCA security, there is no proof of key enciphering security, confidence in the latter resting only on the absence of discovered attacks.

This leads to the important question of how to bridge this gap. We want efficient ciphers, both narrowblock (data is n bits long where n is the blocklength of the underlying blockcipher) and wideblock (data is mn bits long for $m \geq 2$). These ciphers should be *tweakable*, i.e., they should support an additional input called a tweak (usually the disk sector number) the same input produces completely independent outputs with different tweaks. They should be not only proven PRP-CCA secure but also proven to securely encipher data that contains their own keys. In the narrowblock case this means the message might equal the key, while in the wideblock case it means any block of the message might equal the key. We do not want to use random oracles (ROs) but rather want to prove all this assuming only what was assumed for EME, namely the standard PRP-CCA security of the underlying blockcipher.

Proving security of encryption of key-dependent messages is notoriously hard, as standard reduction techniques do not work. Random oracle based solutions [29] and sophisticated non-RO solutions based on novel techniques have been proposed [36, 5, 38, 9], but they are far from practical.

We show nonetheless how to achieve the stated objective. We first provide **StE**, a simple and efficient transform that, applied to any tweakable PRP-CCA blockcipher, results in a tweakable narrowblock cipher that is not only proven PRP-CCA but also proven to securely encipher its own key, assuming only PRP-CCA security of the starting tweakable blockcipher. We then use **StE** as the basis for **EtE**, a transform that, applied to any tweakable PRP-CCA wideblock cipher, results in another tweakable wideblock cipher that, again, is not only proven PRP-CCA but also proven to securely encipher data containing its own key in any block, assuming only PRP-CCA security of the starting

wideblock cipher and of the tweakable blockcipher underlying **StE**.

We instantiated **EtE** with AES as the base blockcipher, tweaked via XEX [101], and with EME as the base wideblock cipher. We implemented this with the AES-NI instruction set. We found that **EtE** is only 15% slower than plain EME.

Encrypting messages depending on the key

Key enciphering security seeks to provide security when the key appears as a message or as a substring of the message. We now move on to the more general problem of providing security when messages being encrypted depend on the key. This problem is referred to as Key dependent message security or KDM security, and was first studied by Black, Rogaway and Shrimpton [29]. There are several practical scenarios where such dependencies could arise. For example, Windows machines use a program called BitLocker to encrypt hard disks. The key used by BitLocker to encrypt your disk may reside on the disk. More generally, 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.

As with key enciphering security, there is growing recognition that security of key-dependent data, first defined to connect cryptography to formal methods [29] and provide anonymous credentials [43], 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 robust and misuse-resistant cryptography.

We provide a comprehensive treatment of security for key-dependent data for the central practical goal of symmetric cryptography, namely authenticated encryption. For each 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 modes while our solutions show how to adapt them in minimal ways to achieve the best achievable security.

Message-Locked Encryption

To save space, commercial cloud storage services such as Google Drive [69], Dropbox [56] and bitcasa [28] perform file-level deduplication across all their users. Say a user Alice stores a file M and Bob requests to store the same file M. Observing that M is already stored, the server, instead of storing a second copy of M, simply updates metadata associated to M to indicate that Bob and Alice both stored M. In this way, no file is stored more than once, moving storage costs for a file stored by u users from $\mathcal{O}(u \cdot |M|)$ to $\mathcal{O}(u + |M|)$.

However, as users we may want our files to be encrypted. We may not want the storage provider to see our data. Even if we did trust the provider, we may legitimately worry about errant employees or the risk of server compromise by an external adversary. When users themselves are corporations outsourcing their data storage, policy or government regulation may mandate encryption.

We introduce a new primitive that we call Message-Locked Encryption (MLE). An MLE scheme is a symmetric encryption scheme in which the key used for encryption and decryption is itself derived from the message. Instances of this primitive are seeing widespread deployment and application for the purpose of secure deduplication [28, 112, 49, 2, 51, 10, 50, 91, 106, 98, 3, 65, 58], but in the absence of a theoretical treatment, we have no precise indication of what these methods do or do not accomplish.

We provide definitions of privacy and integrity peculiar to this domain. A clear, strong target for design emerges, and we make practical and theoretical contributions.

We analyze existing schemes and new variants, breaking some and justifying others with proofs in the random-oracle-model. In terms of theoretical contributions, we address the challenging question of finding a standard-model MLE scheme, making

connections with deterministic public-key encryption [12], correlated-input-secure hash functions [70] and locally-computable extractors [8, 88, 107] to provide schemes exhibiting different trade-offs between assumptions made and supported message distributions.

UCE

The random-oracle paradigm of Bellare and Rogaway (BR93) [22] is often used to prove the security of systems which use cryptographic hash functions, by modelling those hash functions as random oracles, developing a proof in the presence of such random oracles (RO), and viewing the realized system as containing an instantiation of the RO in the form of the hash function. The central and justified critique of the paradigm [46] is that the instantiated scheme has only heuristic security.

We introduce Universal Computational Extractors (UCEs), an approach for standard model security of certain random-oracle model constructions. The basic intuition is that the output of a UCE-secure function looks random even given the key and some "leakage," as long as the leakage is appropriately restricted. The approach works by starting with a random oracle model construction, and instantiating the random oracle with a family of hash functions. The proof is based on the (standard-model) assumption that the instantiating function is UCE-secure. We show that UCEs can enable standard model instantiations of KDM secure encryption schemes and MLE schemes. UCE supports several applications beyond KDM secure encryption and MLE, as demonstrated in [18], including deterministic PKE, point-function obfuscation, encryption secure under related-key attack, and adaptively-secure garbled circuits. However, we do not look at these applications here, restricting attention to KDM and MLE as applications.

Chapter 1

Preliminaries

We denote the security parameter by $\lambda \in \mathbb{N}$. The length of a string $a \in \{0,1\}^*$ is denoted by $|a|$. We let ε denote the empty string. If s is a string we let $s[i]$ denote its i-th bit. We denote the concatenation of strings s and t by $s\|t$. If s is a string and $0 \leq \ell \leq |s|$, then $t\|_\ell u \leftarrow s$ denotes letting t and u be strings such that $|t| = \ell$ and $t\|y = s$. We let $a \leftarrow\!\!\$\, A$ denote picking an element of a finite set A uniformly at random and assigning it to a. For $a,b \in \mathbb{N}$ and $a \leq b$, we let $[a,b]$ denote the set $\{a,a+1,\ldots,b\}$. For $\ell \in \mathbb{N}$, the unary representation is given by 1^ℓ. The size of a finite set A is denoted by $|A|$. The number of coordinates of a vector $\mathbf{a}$ is denoted by $|\mathbf{a}|$. Algorithms are randomized unless otherwise indicated. An adversary is an algorithm or a tuple of algorithms. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every polynomial p, there exists $n_p \in \mathbb{N}$ such that $f(n) < 1/p(n)$ for all $n > n_p$. The guessing probability $g(X)$ and min-entropy $\mathrm{H}_\infty(X)$ of a random variable X are defined via $g(X) = \max_x \Pr[X = x] = 2^{-\mathrm{H}_\infty(X)}$. The conditional guessing probability $\mathbf{GP}(X\,|\,Y)$ and conditional min-entropy $\mathbf{H}_\infty(X\,|\,Y)$ of a random variable X given a random variable Y are defined via $\mathbf{GP}(X\,|\,Y) = \sum_y \Pr[Y = y] \cdot \max_x \Pr[X = x|Y = y] = 2^{-\mathbf{H}_\infty(X\,|\,Y)}$. By $\mathsf{SD}(X;Y)$ we denote the statistical distance between random variables X and Y.

Games

We use the code based game playing framework of [25, 99]. A game $G(\lambda)$ (for e.g. Fig. 2.1) consists of a MAIN procedure, and possibly other procedures, and begins by executing MAIN. The MAIN procedure in turn runs an adversary A after some initialization steps. The adversary A is executed with oracle access to certain procedures, and after it finishes executing, the game performs some more steps with the output of the adversary. Finally, the game exits with some output of its own. Boolean variables are initialized to false, integers are initialized to 0, strings are initialized to ε, and sets are initialized to $\emptyset$. We let $G^A \Rightarrow y$ denote the event that an execution of G with A outputs y. We abbreviate $G^A \Rightarrow \mathsf{true}$ as G^A.

Symmetric encryption

A symmetric encryption (SE) scheme $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is a triple of algorithms. Key generation K produces a key via $k \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$, which is a random string of length $\kappa(\lambda)$ bits. Encryption E encrypts message $m \in \{0,1\}^{\mu(\lambda)}$ under k to get a ciphertext $c \leftarrow\!\!\$\, \mathsf{E}(1^\lambda, k, m)$. Here, $c \in \{0,1\}^{\nu(\lambda)}$. Decryption D is deterministic and returns $m \leftarrow \mathsf{D}(1^\lambda, k, c)$ where $m \in \{0,1\}^{\mu(\lambda)} \cup \{\perp\}$. For correctness, we require that $\mathsf{D}(1^\lambda, k, \mathsf{E}(1^\lambda, k, m)) = m$ for all $\lambda \in \mathbb{N}$ for all $m \in \{0,1\}^{\mu(\lambda)}$ for all $k \in [\mathsf{K}(1^\lambda)]$ for all $c \in \mathsf{E}(1^\lambda, k, m)$. We say that SE is deterministic if E is deterministic.

The indistinguishability under chosen plaintext attck (IND-CPA) security notion for SE schemes is defined via game $\mathrm{CPA}^A_{\mathsf{SE}}(\lambda)$ which starts by picking key $k \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$ and a bit $b \leftarrow\!\!\$\, \{0,1\}$. The game then runs adversary A with access to an oracle ENC. When A makes a call to ENC with inputs $m_0, m_1 \in \{0,1\}^{\mu(\lambda)}$, the game computes $c \leftarrow\!\!\$\, \mathsf{E}(1^\lambda, k, m_b)$ and returns c. The adversary exits with output b', and the game returns $(b' = b)$. We define advantage as $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},A}(\lambda) = 2\Pr[\mathrm{CPA}^A_{\mathsf{SE}}(\lambda)] - 1$ and say that SE is CPA-secure if advantage is negligible for all PT A.

The real-or-random (ROR) security notion for SE schemes is defined via game $\mathrm{ROR}^A_{\mathsf{SE}}(\lambda)$ which starts by picking a bit $b \leftarrow\!\!\$\, \{0,1\}$. The adversary A is then executed

with access to an oracle ENC. When ENC is invoked with input $m \in \{0,1\}^{\mu(\lambda)}$, the game picks $k \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$, computes $c_1 \leftarrow\!\!\$\, \mathsf{E}(1^\lambda, k, m)$ and $c_0 \leftarrow\!\!\$\, \{0,1\}^{\nu(\lambda)}$, and returns c_b. The adversary finishes with output b', and the game returns $(b' = b)$. We define advantage as $\mathsf{Adv}^{\mathsf{ror}}_{\mathsf{SE},A}(\lambda) = 2\Pr[\mathrm{ROR}^A_{\mathsf{SE}}(\lambda)] - 1$ and say that SE is ROR-secure if advantage is negligible for all PT A.

Public key encryption

A public-key encryption (PKE) scheme $\mathsf{PKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is a triple of algorithms. Key generation K produces public key ek and secret key dk via $(ek, dk) \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$. Encryption E takes message $m \in \{0,1\}^{\mu(\lambda)}$ and ek to get a ciphertext $c \leftarrow\!\!\$\, \mathsf{E}(1^\lambda, k, m)$. Here, $c \in \{0,1\}^{\nu(\lambda)}$. Decryption D is deterministic and returns a message $m \leftarrow \mathsf{D}(1^\lambda, dk, c)$ where $m \in \{0,1\}^{\mu(\lambda)} \cup \{\perp\}$. For correctness, we require that $\mathsf{D}(1^\lambda, dk, \mathsf{E}(1^\lambda, ek, m)) = m$ for all $\lambda \in \mathbb{N}$ for all $m \in \{0,1\}^{\mu(\lambda)}$ for all $(ek, dk) \in [\mathsf{K}(1^\lambda)]$ for all $c \in [\mathsf{E}(1^\lambda, k, m)]$. We say that SE is a deterministic SE scheme (deterministic SE) scheme if E is deterministic.

The indistinguishability under chosen plaintext attck (IND-CPA) security notion for PKE schemes is defined via game $\mathrm{CPA}^A_{\mathsf{PKE}}(\lambda)$ which starts by picking keys $(ek, dk) \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$ and a bit $b \leftarrow\!\!\$\, \{0,1\}$. The game then runs adversary A with ek and access to an oracle ENC. When A makes a call to ENC with inputs $m_0, m_1 \in \{0,1\}^{\mu(\lambda)}$, the game computes $c \leftarrow\!\!\$\, \mathsf{E}(1^\lambda, ek, m_b)$ and returns c. The adversary exits with output b', and the game returns $(b' = b)$. We define advantage as $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},A}(\lambda) = 2\Pr[\mathrm{CPA}^A_{\mathsf{PKE}}(\lambda)] - 1$ and say that PKE is CPA-secure if advantage is negligible for all PT A. We let PKECPA denote the set of all CPA-secure PKE schemes.

Authenticated encryption

A symmetric authenticated encryption scheme $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is specified by a key generation algorithm K that returns κ-bit strings, and deterministic encryption and

decryption algorithms E and D. Inputs to E are a $\kappa(\lambda)$-bit key k, a $\rho(\lambda)$-bit nonce n, a header $h \in \{0,1\}^*$ and a message $m \in \{0,1\}^*$, and output is a ciphertext $c \in \{0,1\}^*$. Inputs to D are k,n,d,c, and output is a message $m \in \{0,1\}^* \cup \perp$. or $\perp$. We refer to κ as the keylength and $\rho(\lambda)$ as the noncelength. The way nonces are handled by the games defines two kinds of security, namely universal-nonce and random-nonce security. We require that $\mathsf{D}(K,n,d,\mathsf{E}(K,n,d,M)) = M$ for all values of the inputs shown. We also require that 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 ciphertextl ength such that $|c| = \mathsf{cl}(|m|,|d|)$ for any c that may be output by $\mathsf{E}(\cdot,\cdot,d,m)$.

Decryption D takes the nonce and header as an input as in [102, 104]. In this view, the ciphertext in standard counter-mode encryption does not incude the counter. It is up to the application to transmit nonce and header if necessary, so the "ciphertext" in practice may be more than the output of E, but in many settings the receiver gets nonce and header in out-of-band ways. But our treatment differs from standard ones [15] in that the nonce must be explicitly provided to 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.

Hash functions

A hash function $\mathsf{H} = (\mathsf{K},\mathsf{H})$ is a pair of algorithms. Key generation K returns $hk \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$. Hashing is deterministic; it takes input $m \in \{0,1\}^{\mu(\lambda)}$ and key hk to return $h \leftarrow \mathsf{H}(1^\lambda, hk, m)$ where $h \in \{0,1\}^{\tau(\lambda)}$. We refer to μ and τ as the input and hash lengths of H. Collision resistance for hash functions is defined via game $\mathrm{CR}^A_{\mathsf{H}}(\lambda)$ which works by picking a key $hk \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$ and then running $A(1^\lambda, hk)$ to get m_0, m_1. The game then returns true if $(m_0 \neq m_1)$ and $(\mathsf{H}(1^\lambda, hk, m_0) = \mathsf{H}(1^\lambda, hk, m_1))$. We define advantage $\mathsf{Adv}^{\mathsf{cr}}_{\mathsf{SE},A}(\lambda) = \Pr[\mathrm{CR}^A_{\mathsf{SE}}(\lambda)]$ and say that H is collision resistant if advantage is negligible

for all PT A.

ROM

In the random oracle model (ROM), all procedures of the game and the adversary get access to a procedure called a random oracle (RO) [22] is a game procedure H that maintains a table $\mathtt{H}[\cdot,\cdot]$, initially everywhere $\perp$. Given a query x,k with $x \in \{0,1\}^*$ and $k \in \mathbb{N}$, it executes: If $\mathtt{H}[x,k] = \perp$ then $\mathtt{H}[x,k] \leftarrow\!\!\$\ \{0,1\}^k$. It then returns $\mathtt{H}[x,k]$. As this indicates, the RO can provide outputs of any desired length. We will at times omit the length when a single value is of import and that length is clear from context. In the ROM, both scheme algorithms and adversary algorithms will have access to H.

Chapter 2

Key enciphering security

2.1 Introduction

Traditionally, encryption schemes and ciphers have not been designed to remain secure when encrypting their own keys, and this has been justified based off the view that encryption of keys does not happen in practice. However, this view is increasingly seen as careless, and there is a broad consensus in standardization groups, particularly those involved in disk encryption that building schemes that remain secure when encrypting their own keys is an important goal.

Disks need length preserving encryption, achived through ciphers. Key enciphering security requires that such ciphers can securely encrypt their own key as well messages containing the key as a substring. While this is not a classical security goal for ciphers, the Security in Storage Working Group (IEEE P1619), when setting up disk encryption standards, considered it important and desirable enough that they rejected a candidate scheme, LRW, due to known key enciphering attacks. The group advocated key-enciphering security because users or the system may store the key on the disk but also because, if it would create a vulnerability, an attacker could attempt to manipulate the system so that it transfers the key from memory to the disk. Overall, they considered it important enough to reject candidates without the property.

Meanwhile, the group valued security proofs, and chose EME2 [75] to be standardized. EME2 is a variant of EME [79], a wideblock cipher proven to achieve PRP-CCA security and it was chosen based on the strength of its security proofs, a host of efficiency properties, and the *presumed* security of enciphering the key. If (as seems the consensus in this domain) key enciphering security is a requirement, and (as also seems the consensus) proofs are also important, a discrepancy becomes obvious, namely that, while there is a proof of PRP-CCA security, there is no proof of key enciphering security, confidence in the latter resting only on the absence of discovered attacks.

All this leads to the problem of finding ciphers to fill this gap, namely efficient tweakable ciphers, both narrowblock (data is n bits long where n is the blocklength of the underlying blockcipher) and wideblock (data is mn bits long for $m \geq 2$). The constructions should be not only proven PRP-CCA secure but also proven to securely encipher data that contains their own keys. In the narrowblock case this means the message might equal the key, while in the wideblock case it means any block of the message might equal the key. We explore the question of constructing ciphers with key enciphering security. We do not want to use random oracles (ROs) but rather want to prove all this assuming only what was assumed for EME, namely the standard PRP-CCA security of the underlying blockcipher.

Halevi and Krawczyk [77], also motivated by the concerns and needs of the Security in Storage Working Group that we have explained, introduced the notions of KDM secure PRFs and PRPs and asked whether there exist Φ-PRFs or Φ-PRPs for non-trivial Φ. (Meaning, Φ contains some interesting non-constant function such as the identity.) They were able to give a standard model construct in the PRF case, but it was not invertible and thus could not be used for disk encryption. They also gave a PRP but the proof is in the ideal cipher model. Left open by their work is to provide a Φ-PRP for non-trivial Φ in the standard model. We resolve this, providing not only constructs, but

efficient ones, with proofs not only in the standard model but assuming only standard PRP security of the underlying blockcipher, for non-trivial Φ of practical interest, with tweaks, and for both the narrow and wide block settings.

Key enciphering security is hard to achieve, as standard reduction techniques fail completely. The first solutions [29] used ROs. Sophisticated, non-RO solutions based on novel techniques have emerged [36, 5, 38, 9] but they are far from practical. Moreover, security for key-dependent data is usually considered for randomized IND-CPA encryption [29], so that solutions, for instance those provided in the RO model by [29] are necessarily length increasing.

Disk encryption needs length preserving encryption. The desired primitive is a tweakable cipher [87], E that deterministically maps an $\mu(\lambda)$-bit key k, a tweak $t \in \mathbf{T}(\lambda)$ and an $\omega(\lambda)$-block plaintext m to an $\omega(\lambda)$-block ciphertext $E(1^\lambda, k, t, m)$. So μ is both the keylength and the blocklength. It must be invertible, meaning $E(1^\lambda, k, t, \cdot)$ is, for every k, t, a permutation over $\{0,1\}^{\omega(\lambda)\mu(\lambda)}$ whose inverse we denote by $E^{-1}(1^\lambda, k, t, \cdot)$. This is a tweakable blockcipher, or a narrowblock design, if $\omega(\lambda) = 1$ for all $\lambda \in \mathbb{N}$, and a wideblock design otherwise. Both are of interest to the Security in Storage Working Group, the first under P1619 and the second under P1619.2. In disk encryption the tweak could be the sector number wideblock or the block index narrowblock and its use significantly increases security.

Attacks of Halevi and Krawczyk [77] show that, unlike for randomized encryption [29], there is no cipher that is Φ-PRP-CCA secure for the class Φ of all functions. To achieve security, we must restrict the class somehow. The restriction we make, namely to consider encrypting the key, results in a class capturing practical attacks for which we will show security to be achievable.

We first provide **StE**, a simple and efficient transform that, applied to any tweakable PRP-CCA blockcipher, results in a tweakable narrowblock cipher that is not only

proven PRP-CCA but also proven to securely encipher its own key, assuming only PRP-CCA security of the starting tweakable blockcipher. We then use **StE** as the basis for **EtE**, a transform that, applied to any tweakable PRP-CCA wideblock cipher, results in another tweakable wideblock cipher that, again, is not only proven PRP-CCA but also proven to securely encipher data containing its own key in any block, assuming only PRP-CCA security of the starting wideblock tweakable cipher and of the tweakable blockcipher underlying **StE**.

In terms of concrete instantiations, we implemented **EtE** with AES as the base blockcipher, tweaked via XEX [101], and with EME as the base wideblock cipher. We implemented this with the AES-NI instruction set. We found that **EtE** is only 15% slower than plain EME.

Defining key enciphering security

The standard security notion for a (tweakable) cipher is to be PRP-CCA secure [87]. This is sometimes called strong PRP security following the terminology of [89]. We extend this to key dependent messages (KDM), defining what it means for E to be Φ-PRP-CCA secure where Φ is a class of functions that map $\mu(\lambda)$-bit keys to $\omega(\lambda)\mu(\lambda)$-bit inputs, following [29, 77]. The game picks a random challenge bit b, a random key k and a random permutation $\pi(t,\cdot)$: $\{0,1\}^{\omega(\lambda)\mu(\lambda)} \to \{0,1\}^{\omega(\lambda)\mu(\lambda)}$ for each t. It gives the adversary the standard oracles $\mathrm{FN}, \mathrm{FN}^{-1}$ from the PRP-CCA game and a new oracle KDFN. Oracle FN, on input t,m, returns $E(1^\lambda,k,t,m)$ if $b=1$ and $\pi(t,m)$ otherwise; oracle FN^{-1}, on input t,c, returns $E^{-1}(1^\lambda,k,t,c)$ if $b=1$ and $\pi^{-1}(t,c)$ otherwise; oracle KDFN, on input ϕ,t, where ϕ is required to be in Φ, returns $E(k,t,\phi(k))$ if $b=1$ and $\pi(t,\phi(k))$ otherwise. To ensure non-triviality, the adversary is required to be legitimate, meaning does not query $\mathrm{FN}^{-1}(t,c)$ for a c previously received as a response to a $\mathrm{KDFN}(t,\cdot)$ query. The adversary advantage is $2\Pr[b=b']-1$ where b' is its output bit. Φ-PRP-CCA security obviously implies PRP-CCA security for all Φ,

and coincides with it when $\Phi = \emptyset$.

What we mean by "encrypting the key" depends on whether the design is narrowblock or wideblock. In the narrowblock case, we are concerned with the obvious, namely that the message equals the key, captured formally by letting Φ consist of the identity function id. For wideblock designs, we seek security when *any block* of the message equals the key. This reflects P1619.2 requirements. We clarify that the key may occur in multiple blocks but may not overlap between blocks. Thus if $m = m[1] \ldots m[\omega(\lambda)]$ is the message then $m[i]$ may equal the key k for any i but we do not allow k to start somewhere in one block and end somewhere in the next block. In Section 2.4 we specify a Φ, that we denote id_ω, that captures this formally.

Approach

First we address the narrowblock case, providing a transform **StE** that turns a given PRP-CCA tweakable blockcipher into a $\{\mathrm{id}\}$-PRP-CCA tweakable blockcipher, meaning the constructed cipher not only preserves the PRP-CCA security of the base one but, assuming only PRP-CCA security of the latter, is proven to securely encrypt its key. **StE** is of interest in its own right as the first (tweakable) blockcipher that can provably encipher its own key under standard and minimal assumptions.

We would have liked to show that instantiating the base blockcipher of E of EME [79] with $F = \mathbf{StE}[E]$ results in a wideblock cipher that can provably encipher its own key, but this does not work. Instead, we show how to add a pre-processing step to EME, or any other PRP-CCA secure wideblock cipher, to get another wideblock cipher that now, when instantiated with F as the underlying blockcipher, is id_ω-PRP-CCA secure, meaning not only PRP-CCA secure but able to securely encipher messages that contain the key in any block. We now look at these two contributions closely.

Narrowblock design

Our starting point is a suggestion of [36] to securely encrypt keys with a *randomized* encryption scheme by exchanging the key with another point. To make this work for deterministic encryption we swap with a "hidden" point, the latter determined by encryption under the key of a constant. A crucial idea is to use tweaks, defining the hidden point via a tweak not used for anything else.

We take a tweakable blockcipher E with block and key length μ, and tweakspace $\mathbf{T}$, assumed to have normal, meaning PRP-CCA, security, but not necessarily able to securely encrypt its key. We pick an arbitrary tweak $\gamma_\lambda \in \mathbf{T}(\lambda)$ for E and also an arbitrary point $\alpha \in \{0,1\}^{\mu(\lambda)}$. We usually drop the subscript λ in γ_λ and refer to it as γ. Both α and γ are public and known to the adversary. Our **StE** (Swap-then-Encipher) transform now turns E into another tweakable blockcipher F with the same block and key lengths, and a modified tweakspace $\mathbf{T}'$ where $\mathbf{T}'(\lambda) = \mathbf{T}(\lambda) \setminus \{\gamma_\lambda\}$. We define F via

$\underline{\mathsf{F}(1^\lambda, k, t, m)}$

$h \leftarrow E(1^\lambda, k, \gamma, \alpha)$

If $m = k$ then $y \leftarrow E(1^\lambda, k, t, h)$

Else If $m = h$ then $y \leftarrow E(1^\lambda, k, t, k)$

Else $y \leftarrow E(1^\lambda, k, t, m)$

Return y

In Section 2.3 we show that F is invertible, as required to be a cipher. A curious aspect of the design is that at the third line we actually encrypt the key k with the original blockcipher E. Given that the latter may not necessarily be able to securely encrypt its key, why would this work? The answer is that we only do this if $m = h$ and we ensure the latter is very unlikely. Thus h is our "hidden" point.

Making sure h is hidden takes some care. It is not so merely because its computation depends on k, for the adversary will have an oracle that (at least in the real game)

allow it to compute F under k for any tweak of the adversary's choice and this might be used to extract information about h. The crucial point is that γ, the tweak used in computing h, was removed from the tweak space of F so the adversary cannot use it.

The proof that F securely encrypts its key is done by reduction to the assumed PRP-CCA security of E and is delicate due to the cyclic nature of the construction. It is crucial for disk encryption that F is invertible. It is to ensure this that we need the swap that effectively exchanges the roles of k and h.

At first glance it would appear that **StE** doubles the cost since it requires two invocations of E, one to get h and the other to get y. This, however, is also true of XEX-AES [101]: presented as a fast tweakable blockcipher, XEX actually requires two invocations of the underlying blockcipher. In both cases, however, the answer is the same, namely that the ciphers will be used in a mode where the extra blockcipher computation is done once and its cost is thus amortized out so that the effective cost of the cipher is one blockcipher call. Specifically if F, like XEX-AES, is used as the base blockcipher in a wideblock design to encipher a sector consisting of F blocks (eg. $F = 32$), we can compute h just once across the F encipherings so **StE** effectively involves just a single blockcipher invocation.

The IEEE 1619 standard for narrowblock ciphers is based on XEX-AES [101]. **StE** is easily applied here, yielding an alternative narrowblock candidate that is as efficient as the current one when the cost of computing H is amortized out, but also has provable security for enciphering the key.

Wideblock design

EME [79] is a wideblock tweakable cipher that is proven PRP-CCA secure assuming only PRP-CCA security of the underlying blockcipher. It makes two passes through the data and runs at 2 blockcipher invocations per message block. One of its attractions, and its advantage over the earlier CMC [78], is that it is parallelizable. No

attack is known when one encrypts messages containing the key in any block, but nor is there any proof that such an attack is absent.

We wish to enhance designs like EME so that PRP-CCA security is preserved but provable key enciphering security is also added, without assuming any more of the underlying blockcipher than PRP-CCA security. Our **EtE** transform does just this. It makes an ECB pass through the data using $\mathsf{F} = \mathbf{StE}[E]$ under one tweak and then applies any wideblock PRP-CCA cipher (EME would be one possibility) using F with another tweak. We prove that the resulting cipher is id_ω-PRP-CCA secure assuming only that tweakable blockcipher E is PRP-CCA secure. Thus **EtE** provides a generic way to upgrade any PRP-CCA wideblock cipher to also be able to securely encipher messages containing the key in any block, at the cost of one extra blockcipher operation per block. The design preserves parallelizability, so that when instantiated with a parallelizable wideblock cipher like EME, the resulting cipher is also parallelizable.

Instantiation and implementation

Getting the best out of **EtE** requires good choices for the underlying components and some optimizations. We use AES as the base blockcipher and XEX [101] to tweak it. We use EME as the wideblock cipher. Let J denote the resulting wideblock cipher as produced by **EtE**. We implemented J, as well as plain EME for comparison. Our experiments show that J is only 15% slower than EME.

Recall that EME uses about two blockcipher calls per message block. Since J would thus use three blockcipher calls per message block, one might naively expect a 50% slowdown. However, EME computes various offsets that are added to the blockcipher inputs and outputs, and these computations are quite costly, so our simple ECB pass is less expensive compared to EME than one might imagine. We remark that AES-NI is now widely available in Intel and AMD processors on modern machines, and disk controllers might potentially have hardware AES support as well, so using AES-NI as a

starting point is realistic.

2.2 Preliminaries

Tweakable blockciphers

A *tweakable blockcipher* [87] with block length μ, key length κ and tweak space $\mathbf{T}$ is a map $E(1^\lambda,\cdot,\cdot,\cdot)$: $\{0,1\}^{\kappa(\lambda)} \times \mathbf{T}(\lambda) \times \{0,1\}^{\mu(\lambda)} \to \{0,1\}^{\mu(\lambda)}$ that takes input a $\kappa(\lambda)$-bit key k, a *tweak* t drawn from the tweakspace $\mathbf{T}(\lambda)$ and a $\mu(\lambda)$-bit message m to return an $\mu(\lambda)$-bit output $E(1^\lambda,k,t,m)$. The map $E(1^\lambda,k,t,\cdot)$: $\{0,1\}^{\mu(\lambda)} \to \{0,1\}^{\mu(\lambda)}$ that on input $m \in \{0,1\}^{\mu(\lambda)}$ returns $E(1^\lambda,k,t,m)$ is required to be a permutation and its inverse is denoted $E^{-1}(1^\lambda,k,t,\cdot)$.

The standard notion of security for a tweakable blockcipher is to be a tweakable pseudorandom permutation [87]. This can be considered under either chosen-plaintext attack (usually called PRP security) or chosen-ciphertext attack (usually called strong PRP security) [87, 89], but we will use the terms PRP-CPA and PRP-CCA as more indicative of the models and more consistent with notation for other primitives. We will be working with PRP-CCA. To define it consider games Real_E and Rand_E of Fig. 2.1. In Rand, we are denoting by $\mathsf{TwPm}(\mathbf{T},\{0,1\}^{\mu(\lambda)})$ the set of all π: $\mathbf{T} \times \{0,1\}^{\mu(\lambda)} \to \{0,1\}^{\mu(\lambda)}$ such that $\pi(t,\cdot)$ is a permutation on $\{0,1\}^{\mu(\lambda)}$ for each $t \in \mathbf{T}(\lambda)$. In this case, $\pi^{-1}(t,\cdot)$: $\{0,1\}^{\mu(\lambda)} \to \{0,1\}^{\mu(\lambda)}$ is the inverse of $\pi(t,\cdot)$. We define the prp-cca advantage of A as

$$\mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{E,A}(\lambda) = \Pr[\mathrm{Real}^A_E(\lambda)] - \Pr[\mathrm{Rand}^A_E(\lambda)].$$

We extend PRP-CCA to allow for the encryption of key-dependent messages via games $\mathrm{Real}_{E,\Phi}$ and $\mathrm{Rand}_{E,\Phi}$ of Figure 2.1. Oracle KDFN takes input a tweak t and a function $\phi \in \mathsf{Maps}(\{0,1\}^{\kappa(\lambda)},\{0,1\}^{\mu(\lambda)})$ and derives m as $\phi(K)$. If $\Phi \subseteq$

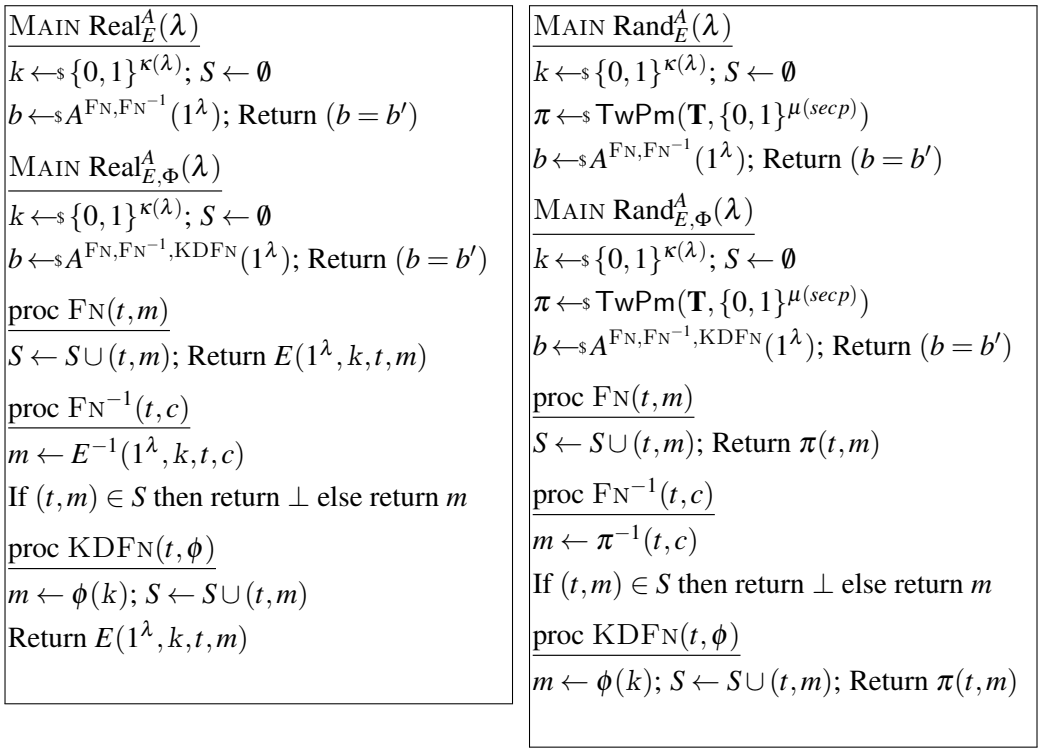


Figure 2.1. Games defining PRP-CCA and KDM PRP-CCA security of tweakable blockcipher E with key length κ and block length μ.

$\mathsf{Maps}(\{0,1\}^k,\{0,1\}^{\mu(\lambda)})$ we say that adversary A is Φ-restricted if the argument ϕ in its KDFN queries is always from $\Phi(\lambda)$. We define the Φ-kdm-prp-cca advantage of such an A to be

$$\mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{E,\Phi,A}(\lambda) = \Pr[\mathrm{Real}^A_{E,\Phi}(\lambda)] - \Pr[\mathrm{Rand}^A_{E,\Phi}(\lambda)].$$

We continue, thus, to denote the notion via prp-cca, the key-dependent messages indicated by the extra subscript Φ in the advantage function.

Consider the following strategy for A. It makes KDFN query t,id to get back a ciphertext c and then queries $\mathrm{FN}^{-1}(t,c)$. The response will be $m = \mathrm{id}(k) = k$, and now A has the key and can easily win. (That is, get a high advantage, for example by returning

1 if $E(k,t,m') = \mathrm{FN}(t,m')$ for some t and some $m' \neq m$, and 0 otherwise.) To preclude this, we require that A is *legitimate*, meaning that, for all t, ϕ, it never makes a $\mathrm{FN}^{-1}(t,c)$ query for c previously received in response to a $\mathrm{KDFN}(t,\phi)$ query. The analogy is the definition of IND-CCA secure encryption where the adversary is not allowed to query to the decryption oracle a ciphertext it previously received as a challenge. The (necessary) assumption that adversaries are legitimate is made throughout and is implicit in all our results. We remark that we do not need to prohibit A from query $\mathrm{FN}^{-1}(t,c)$ for c previously returned in response to query $\mathrm{FN}(t,m)$, because A already knows the message M it would get as response. So no restriction is present in the plain prp-cca notion. But when key-dependent messages are introduced, we must require legitimacy.

KDM-secure IND-CPA encryption was defined by [29]. Subsequently, KDM security of tweakable blockciphers was defined by Halevi and Krawczyk [77]. Because blockciphers, unlike encryption schemes, are deterministic, KDM security is not achievable when arbitrary functions ϕ are allowed in KDFN queries. This is analogous to what happens with related-key attacks (RKAs) and thus, as with the formalization of RKA security from [20], Halevi and Krawczyk [77] parameterize the advantage and definition of KDM security for tweakable blockciphers by a class Φ of functions. Our narrow-block design **StE** will achieve security when $\Phi = \{\mathrm{id}\}$, corresponding to encrypting the key. Our extended cipher **GStE** will achieve security for a broader class Φ and our wideblock design for Φ corresponding to any block of the message equaling the key.

Halevi and Krawczyk [77] do not explicitly state the legitimacy condition. They also allow the possibility of decryption of key-dependent ciphertexts, via an extra oracle. We have not considered this capability because, while plaintexts may be key dependent, we did not see why ciphertexts would be key dependent. Our schemes do not aim to achieve security in the presence of decryption of key-dependent ciphertexts.

2.3 Narrowblock Cipher

In this section we show how to construct a narrowblock tweakable blockcipher that can securely encipher its own key.

Construction

Let E be a tweakable blockcipher with key length and block length $\mu : \mathbb{N} \to \mathbb{N}$ and tweakspace $\mathbf{T}$. We fix an arbitrary tweak $\gamma_\lambda \in \mathbf{T}(\lambda)$ as well as an arbitrary message $\alpha_\lambda \in \{0,1\}^{\mu(\lambda)}$. Both γ_λ and α_λ are public parameters of the system known to the adversary. Our Swap-then-Encipher transform $\mathbf{StE}_{\gamma,\alpha}$ associates to E another tweakable blockcipher $F = \mathbf{StE}_{\gamma,\alpha}[E]$ with the key length and block length μ, with tweakspace $\mathbf{T}'$ such that $\mathbf{T}'(\lambda) = \mathbf{T}'(\lambda) \setminus \{\gamma_\lambda\}$. The function F is defined as follows:

$\underline{F(1^\lambda, k, t, m)}$

01 $h \leftarrow E(1^\lambda, k, \gamma_\lambda, \alpha_\lambda)$

02 If $m = k$ then $y \leftarrow E(1^\lambda, k, t, h)$

03 Else If $m = h$ then $y \leftarrow E(1^\lambda, k, t, k)$

04 Else $y \leftarrow E(1^\lambda, k, t, m)$

05 Return y

Before considering security, We establish that F is invertible as required to be a tweakable blockcipher. Indeed, the inverse F^{-1} is given by

$\underline{F^{-1}(1^\lambda, k, t, c)}$

01 $y \leftarrow E^{-1}(1^\lambda, k, t, c)$

02 $h \leftarrow E(1^\lambda, k, \gamma_\lambda, \alpha_\lambda)$

03 If $y = k$ then $m \leftarrow h$

04 Else If $y = h$ then $m \leftarrow k$

05 Return m

Efficiency

In the form it is described above, evaluating F requires two calls to the underlying cipher E. But in practice we are enciphering a message consisting of multiple blocks. In this case the value $E(1^\lambda, k, \gamma_\lambda, \alpha_\lambda)$ can be computed just once and then cached, which means each evaluation of $F(1^\lambda, k, t, m)$ requires one call to E with the same key and tweak value. Similarly, $E(1^\lambda, k, \gamma_\lambda, \alpha_\lambda)$ can also be cached for inversion. Thus, the amortized cost of $\mathbf{StE}_{\gamma,\alpha}$ is the same as that of E. We remark that the situation here is analogous to that of XEX [101]. XEX too needs two applications of the underlying blockcipher, but one can usually be amortized out. **StE** is no worse.

Security

The following theorem says that $F = \mathbf{StE}_{\gamma,\alpha}[E]$ is a PRP-CCA that can securely encipher its own key, or more formally, that it is $\{\mathrm{id}\}$-kdm-prp-cca secure, assuming only that E meets the standard PRP-CCA notion of security.

Theorem 2.3.1 *Let E be a tweakable blockcipher with key length and block length μ and tweak space $\mathbf{T}$. Let γ and α be a sequence of points such that $\gamma_\lambda \in \mathbf{T}(\lambda)$ and $\alpha_\lambda \in \{0,1\}^{\mu(\lambda)}$. Let $F = \mathbf{StE}_{\gamma,\alpha}[E]$ be the tweakable blockcipher associated to E via* **StE** *as defined above. Let $\Phi = \{\mathrm{id}\}$ consist of the identity function. Let A be an adversary making at most $Q : \mathbb{N} \to \mathbb{N}$ oracle queries. Then there is an adversary B such that*

$$\mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{F,\Phi,A}(\lambda) < 2 \cdot \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{E,B}(\lambda) + \frac{3Q(\lambda)}{2^{\mu(\lambda)} - 1} .$$

Moreover, the running time of B is about equal to the running time of A.

The intuition is that, during the $\mathrm{Real}_{F,\{\mathrm{id}\}}$ and $\mathrm{Rand}_{F,\{\mathrm{id}\}}$ games, an adversary is unlikely to trigger the test in lines 02 and 03 in the computation of F, except by submitting $\phi = \mathrm{id}$ to its oracle. Once this established, we can give a reduction to the prp-cca security of E.

For the first part, we show that k and h are hard to guess using the prp-cca security of E. If the adversary guesses k, then we can show how to break the prp-cca security of E, a contradiction. We can show the same if the adversary guesses h. This is where we use the fact that the tweak γ_λ is never allowed elsewhere, effectively making h look random. The proof implements this approach and deals with other complications.

MAIN // G_0, G_1	MAIN // G_2
$k \leftarrow\$ \{0,1\}^{\kappa(\lambda)}$; $h \leftarrow E(1^\lambda, k, \gamma_\lambda, \alpha_\lambda)$	$\pi \leftarrow\$ \mathsf{TwPm}(\mathbf{T}, \{0,1\}^{\mu(\lambda)})$
$b' \leftarrow\$ A^{\mathrm{FN},\mathrm{FN}^{-1},\mathrm{KDFN}}(1^\lambda)$; Return $(b'=1)$	$h \leftarrow \pi(\gamma_\lambda, \alpha_\lambda)$
proc $\mathrm{FN}(t,x)$ // $\boxed{G_0}$, G_1	$b' \leftarrow\$ A^{\mathrm{FN},\mathrm{FN}^{-1},\mathrm{KDFN}}(1^\lambda)$
If $m = k$ then bad $\leftarrow$ true; $\boxed{x \leftarrow h}$	Return $(b'=1)$
Else If $m = h$ then bad $\leftarrow$ true; $\boxed{x \leftarrow k}$	proc $\mathrm{FN}(t,m)$ // G_2
Return $E(1^\lambda, k, t, x)$	Return $\pi(t,m)$
proc $\mathrm{KDFN}(t,\phi)$ // G_0, G_1	proc $\mathrm{KDFN}(t,\phi)$ // G_2, G_4
Return $E(1^\lambda, k, t, h)$	Return $\pi(t,h)$
proc $\mathrm{FN}^{-1}(t,c)$ // $\boxed{G_0}$, G_1	proc $\mathrm{FN}^{-1}(t,c)$ // G_2
$X \leftarrow E^{-1}(1^\lambda, k, t, c)$; $m \leftarrow X$	Return $\pi^{-1}(t,c)$
If $X = k$ then bad $\leftarrow$ true; $\boxed{m \leftarrow h}$	
Else If $X = h$ then bad $\leftarrow$ true; $\boxed{m \leftarrow k}$	
Return m	

Figure 2.2. Games used in the proof of Theorem 2.3.1.

Proof: We begin with the games in figures Fig. 2.2 and Fig. 2.2. Recall that here $\Phi = \{\mathrm{id}\}$, so it is assumed that $\phi = \mathrm{id}$ in any KDFN query. Game G_0 includes the boxed code and hence is the same as Real_F^A, and thus we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{F,\Phi,A}(\lambda) &= \Pr[G_0^A(\lambda)] - \Pr[G_2^A(\lambda)] \\
&= \Pr[G_1^A(\lambda)] - \Pr[G_2^A(\lambda)] + \Pr[G_0^A(\lambda)] - \Pr[G_1^A(\lambda)] \\
&\le \Pr[G_1^A(\lambda)] - \Pr[G_2^A(\lambda)] + \Pr[\mathrm{BAD}(G_1^A(\lambda))] \,.
\end{aligned}$$

$\underline{\text{MAIN}}$ / G_3
$k \leftarrow\$ \{0,1\}^{\mu(\lambda)}$
$h \leftarrow E(1^\lambda, k, \gamma_\lambda, \alpha_\lambda)$
$h' \leftarrow E(1^\lambda, k, \gamma_\lambda, \alpha'_\lambda)$
$b' \leftarrow\$ A^{\text{FN},\text{FN}^{-1},\text{KDFN}}(1^\lambda)$
Return $(\mathsf{bad}_1 \vee \mathsf{bad}_2)$

$\underline{\text{MAIN}}$ / G_4
$\pi \leftarrow\$ \mathsf{TwPm}(\mathbf{T}, \{0,1\}^{\mu(\lambda)})$
$h \leftarrow \pi(\gamma_\lambda, \alpha_\lambda)$
$h' \leftarrow \pi(\gamma_\lambda, \alpha'_\lambda)$
$b' \leftarrow\$ A^{\text{FN},\text{FN}^{-1},\text{KDFN}}(1^\lambda)$
Return $(\mathsf{bad}_1 \vee \mathsf{bad}_2)$

$\underline{\text{proc KDFN}(t,\phi)}$ / G_3
Return $E(1^\lambda, k, t, h)$

$\underline{\text{proc FN}(t,m)}$ / G_3
If $E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda) = h'$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
Else If $m = h$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
Return $E(1^\lambda, k, t, m)$

$\underline{\text{proc FN}^{-1}(t,c)}$ / G_3
$m \leftarrow E^{-1}(1^\lambda, k, t, c)$
If $E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda) = h'$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
Else If $m = h$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
Return m

$\underline{\text{proc FN}(t,m)}$ / G_4
If $E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda) = h'$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
Else If $m = h$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
Return $\pi(t,m)$

$\underline{\text{proc FN}^{-1}(t,c)}$ / G_4
$m \leftarrow \pi^{-1}(t,c)$
If $E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda) = h'$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
Else If $m = h$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
Return m

Figure 2.3. Games G_3 and G_4 used in the proof of Theorem 2.3.1.

The inequality is by the Fundamental Lemma of Game Playing [25] since G_0, G_1 are identical until bad. We design adversary B_1 so that

$$\Pr[\text{G}_1^A(\lambda)] - \Pr[\text{G}_2^A(\lambda)] \leq \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{E,B_1}(\lambda)\ . \tag{2.1}$$

The simulation is straightforward since G_1 does not include the boxed code. B_1 has oracles $\text{FN}, \text{FN}^{-1}$. It starts by letting $h \leftarrow \text{FN}(\gamma_\lambda, \alpha_\lambda)$. It then runs A. When A makes a query (t,m) to its FN oracle, B_1 queries its own FN oracle with (t,m) and forwards the response to A. Similarly when A makes a query (t,c) to its FN^{-1} oracle, B_1 queries its own FN^{-1} oracle with (t,c) and forwards the response to A. When A queries its KDFN oracle with t, id, adversary B_1 responds with $\text{FN}(t,h)$. When A halts with output b', so

does B_1. We clearly have

$$\Pr[\mathrm{Real}_E^{B_1}(\lambda)] = \Pr[\mathrm{G}_1^A(\lambda)] \ .$$

Since the tweak γ_λ is not used in any queries of A, the point h in G_2 is random and can be viewed as playing the role of k in game Rand_E. Thus we also have

$$\Pr[\mathrm{Rand}_E^{B_1}(\lambda)] = \Pr[\mathrm{G}_2^A(\lambda)] \ .$$

Thus we have Equation (2.1). It remains to upper bound $\Pr[\mathrm{BAD}(\mathrm{G}_1^A)(\lambda)]$. Fix a point $\alpha'_\lambda \in \{0,1\}^{\mu(\lambda)} \setminus \{\alpha_\lambda\}$ and consider games $\mathrm{G}_3, \mathrm{G}_4$ of Fig. 2.3. Game G_3 replaces the $m = k$ test from G_1 with the test $E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda) = h'$. The purpose is to avoid referring explicitly to k, thereby opening the way for a reduction to the assumed prp-cca security of E. This new test, however, will certainly return true if $m = k$ and hence

$$\begin{aligned} \Pr[\mathrm{BAD}(\mathrm{G}_1^A(\lambda))] &\leq \Pr[\mathrm{G}_3^A(\lambda)] \\ &= \Pr[\mathrm{G}_3^A(\lambda)] - \Pr[\mathrm{G}_4^A(\lambda)] + \Pr[\mathrm{G}_4^A(\lambda)] \ . \end{aligned}$$

We stress that in G_4, where we move to the random world by replacing $E(1^\lambda, k, \cdot, \cdot)$ by $\pi(\cdot, \cdot)$, the test still uses E, invoking the blockcipher here explicitly on inputs $m, \gamma_\lambda, \alpha'_\lambda$. (The test does not use π.) We design adversary B_2 so that

$$\Pr[\mathrm{G}_3^A(\lambda)] - \Pr[\mathrm{G}_4^A(\lambda)] \leq \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{E,B_2}(\lambda) \ . \tag{2.2}$$

The simulation is again straightforward now that k is not referred to explicitly in either game. B_2 has oracles $\mathrm{FN}, \mathrm{FN}^{-1}$. It starts by letting $h \leftarrow \mathrm{FN}(\gamma_\lambda, \alpha_\lambda)$ and $h' \leftarrow \mathrm{FN}(\gamma_\lambda, \alpha'_\lambda)$

and initializing boolean variables $\mathsf{bad}_1, \mathsf{bad}_2$ to false. It then runs A. When A makes a query (t, m) to its FN oracle, B_2 does the following:

If $E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda) = h'$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$

Else If $m = h$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$

Return $\mathrm{FN}(t, m)$ to A

Similarly when A makes a query (t, c) to its FN^{-1} oracle, B_2 does the following:

$m \leftarrow \mathrm{FN}^{-1}(t, c)$

If $E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda) = h'$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$

Else If $m = h$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$

Return m to A

When A queries its KDFN oracle with t, id, adversary B responds with $\mathrm{FN}(t, h)$. When A halts with output b', adversary B halts with output 1 if $(\mathsf{bad}_1 \vee \mathsf{bad}_2)$ has value true and 0 otherwise. We have

$$\Pr[\mathrm{Real}_E^{B_2}(\lambda)] = \Pr[\mathrm{G}_3^A(\lambda)] \text{ and } \Pr[\mathrm{Rand}_E^{B_2}(\lambda)] = \Pr[\mathrm{G}_4^A(\lambda)]$$

which implies Equation (2.2). Let B be the adversary that picks c at random from $\{1, 2\}$ and runs B_c. Then at this point we have

$$\mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{F,\Phi,A}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{E,B}(\lambda) + \Pr[\mathrm{G}_4^A(\lambda)]$$

and proceed to upper bound the last term above, considering separately the probability of setting bad_1 and that of setting bad_2. Since γ_λ is not in the tweak space of F, game G_5 of Fig. 2.4 picks π from $\mathsf{TwPm}(\mathbf{T} \setminus \{\gamma_\lambda\}, \{0,1\}^{\mu(\lambda)})$ rather than $\mathsf{TwPm}(\mathbf{T}, \{0,1\}^{\mu(\lambda)})$, and picks h, h' as random, distinct points. The game can move the setting of bad_1 and

$\underline{\text{MAIN}}$ / $\text{G}_5(\lambda)$
$\pi \leftarrow\!\!\$\ \mathsf{TwPm}(\mathbf{T} \setminus \{\gamma_\lambda\}, \{0,1\}^{\mu(\lambda)})$
$h \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)}$; $S \leftarrow \emptyset$
$h' \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)} \setminus \{h\}$
$A^{\text{FN},\text{FN}^{-1},\text{KDFN}}(1^\lambda)$
Return $(h' \in S)$

$\underline{\text{proc FN}(t,m)}$ / G_5
$S \leftarrow S \cup \{E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda)\}$
Return $\pi(t,m)$

$\underline{\text{proc KDFN}(t,\phi)}$ / G_5
Return $\pi(t,h)$

$\underline{\text{proc FN}^{-1}(t,c)}$ / G_5
$m \leftarrow \pi^{-1}(t,c)$
$S \leftarrow S \cup \{E(1^\lambda, m, \gamma_\lambda, \alpha'_\lambda)\}$
Return m

$\underline{\text{MAIN}}$ / G_6, G_7
For all $t \in \mathbf{T} \setminus \{\gamma_\lambda\}$ do
  $y_t \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)}$; $R(t) \leftarrow \{y_t\}$
$h \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)}$; $A^{\text{FN},\text{FN}^{-1},\text{KDFN}}(1^\lambda)$
Return bad

$\underline{\text{proc FN}(t,m)}$ / $\boxed{\text{G}_6}, \text{G}_7$
If $\pi[t,m]$ then return $\pi[t,m]$
$c \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)} \setminus R(t)$
If $m = h$ then bad $\leftarrow$ true; $\boxed{c \leftarrow y_t}$
$D(t) \leftarrow D(t) \cup \{m\}$; $R(t) \leftarrow R(t) \cup \{c\}$
$\pi[t,m] \leftarrow c$; $\pi^{-1}[t,c] \leftarrow m$
Return $\pi[t,m]$

$\underline{\text{proc KDFN}(t,\phi)}$ / G_6, G_7
Return y_t

$\underline{\text{proc FN}^{-1}(t,c)}$ / $\boxed{\text{G}_6}, \text{G}_7$
If $\pi^{-1}[t,c]$ then return $\pi^{-1}[t,c]$
$m \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)} \setminus D(t)$
If $c = y_t$ then bad $\leftarrow$ true; $\boxed{m \leftarrow h}$
Else If $m = h$ then
  bad $\leftarrow$ true
  $\boxed{m \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)} \setminus (D(t) \cup \{h\})}$
$D(t) \leftarrow D(t) \cup \{m\}$; $R(t) \leftarrow R(t) \cup \{c\}$
$\pi[t,m] \leftarrow c$; $\pi^{-1}[t,c] \leftarrow m$
Return m

Figure 2.4. Games G_6 and G_7 used in the proof of Theorem 2.3.1.

the choice of h' to FINALIZE without impacting what is returned to the adversary. At the end of the execution, the set S can have size at most $Q(\lambda)$ so

$$\Pr[\text{G}_4^A(\lambda) \text{ sets } \mathsf{bad}_1] = \Pr[\text{G}_5^A(\lambda)] \leq \frac{Q(\lambda)}{2^{\mu(\lambda)} - 1} .$$

Bounding the probability that bad_2 is set in G_4 is more difficult because information about h does reach the adversary via the KDFN query. Our intent is to move to a game where h is not referred to in replying to adversary oracle queries. We begin with game G_6 of Fig. 2.4 which samples π lazily. The arrays $\pi[\cdot,\cdot]$ and $\pi^{-1}[\cdot,\cdot]$ are assumed initially

MAIN // G_8
For all $t \in \mathbf{T} \setminus \{\gamma_\lambda\}$ do
  $y_t \leftarrow\$ \{0,1\}^{\mu(\lambda)}$; $R(t) \leftarrow \{y_t\}$
$S \leftarrow \emptyset$
$h \leftarrow\$ \{0,1\}^{\mu(\lambda)}$; $A^{\mathrm{FN},\mathrm{FN}^{-1},\mathrm{KDFN}}(1^\lambda)$
Return $(h \in S) \vee \mathsf{bad}$

proc $\mathrm{FN}(t,m)$ // G_8
If $\pi[t,m]$ then return $\pi[t,m]$
$c \leftarrow\$ \{0,1\}^{\mu(\lambda)} \setminus R(t)$; $S \leftarrow S \cup \{m\}$
$D(t) \leftarrow D(t) \cup \{m\}$; $R(t) \leftarrow R(t) \cup \{c\}$
$\pi[t,m] \leftarrow c$; $\pi^{-1}[t,c] \leftarrow m$
Return $\pi[t,m]$

proc $\mathrm{KDFN}(t,\phi)$ // G_8
Return y_t

proc $\mathrm{FN}^{-1}(t,c)$ // G_8
If $\pi^{-1}[t,c]$ then return $\pi^{-1}[t,c]$
$m \leftarrow\$ \{0,1\}^{\mu(\lambda)} \setminus D(t)$
If $c = y_t$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else $S \leftarrow S \cup \{m\}$
$D(t) \leftarrow D(t) \cup \{m\}$
$R(t) \leftarrow R(t) \cup \{c\}$
$\pi[t,m] \leftarrow c$; $\pi^{-1}[t,c] \leftarrow m$
Return m

Figure 2.5. Game G_8 used in the proof of Theorem 2.3.1.

everywhere undefined, and get filled in as the game progresses. A test "If $\pi[t,m]$" returns true if $\pi[t,m]$ is defined, and false otherwise, and similarly for "If $\pi^{-1}[t,c]$". The game begins by picking a random y_t for each t that is intended to stand for $\pi[t,h]$ but not assigned to the latter so as to avoid using h. Instead, whenever $\pi[t,h]$ or $\pi^{-1}[t,y_t]$ are called for, the game sets bad and corrects via the boxed code, which is included in G_6. A variant of the Fundamental Lemma of Game Playing from [25] says that identical until bad games have the same probability of setting bad and hence

$$\Pr[G_4^A(\lambda) \text{ sets } \mathsf{bad}_2] = \Pr[G_6^A(\lambda)] = \Pr[G_7^A(\lambda)] \ .$$

But game G_7, which excludes the boxed code, does not refer to h in replying to adversary oracle queries, and hence

$$\Pr[G_7^A(\lambda)] = \Pr[G_8^A(\lambda)] \ .$$

Since A is legitimate, a $\mathrm{FN}^{-1}(t,c)$ query with $c = y_t$ must occur before it receives y_t from KDFN and hence is made with no knowledge of y_t. Thus the probability that a query

of A sets bad is at most $2^{-\mu(\lambda)}$. We remark that this is the place we use the assumption that A is legitimate, without which bad could be set with probability one. On the other hand the set S has size at most $Q(\lambda)$ at the end of the execution of A with the game. So $\Pr[\mathrm{G}_8^A(\lambda)] \leq 2Q(\lambda) \cdot 2^{-\mu(\lambda)}$. Putting this together with the above concludes the proof. ∎

The IEEE 1619 standard for narrowblock ciphers is based on AES over XEX [101]. **StE** is an alternative narrowblock candidate that is essentially as efficient as the current one but also has provable security for enciphering the key.

2.4 Wideblock cipher

In this section, we provide a simple and general way to enhance a given PRP-CCA wideblock cipher to be able to encrypt messages that might, in any block, contain the key. Our construction simply puts an ECB layer in front of the given cipher and then uses **StE** as the base tweakable blockcipher for both the ECB pass and the application of the wideblock cipher. This works for any given wideblock cipher that is a mode of operation of a blockcipher, meaning it uses the key only to key an underlying blockcipher, as is the case with EME and other standard designs. **StE** is used with one tweak for the ECB pass and then, by fixing a different tweak, yields a blockcipher to instantiate the wideblock mode of operation.

We begin, below, by defining what it means for a wideblock design to be a mode of operation and what it means to assume it is PRP-CCA secure. Then we provide our construction and prove it secure.

Modes

A wideblock mode of operation $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$ is a pair of oracle algorithms. Given oracle access to a permutation g: $\{0,1\}^{\mu(\lambda)} \to \{0,1\}^{\mu(\lambda)}$, algorithm WB takes input a tweak $t \in \mathbf{T}$ and an $\omega(\lambda)$-block message $x \in \{0,1\}^{\omega(\lambda)\mu(\lambda)}$ to return an

$\omega(\lambda)$-block ciphertext denoted $\mathsf{WB}(t,x{:}g)$. Given oracle access to g^{-1}, algorithm WB^{-1} takes input a tweak $t \in \mathbf{T}$ and an $\omega(\lambda)$-block ciphertext $y \in \{0,1\}^{mn}$ to return an $\omega(\lambda)$-block message denoted $\mathsf{WB}^{-1}(t,y{:}g^{-1})$. It is required that $\mathsf{WB}^{-1}(t,\mathsf{WB}(t,x{:}g){:}g^{-1}) = x$ for all choices of the inputs and oracles. Here $\mathbf{T}, \mu, \omega$ are the tweak space, blocklength and number of blocks associated to **WB**.

If N is a blockcipher with κ-bit keys and μ-bit inputs then **WB** associates to it a tweakable blockcipher $W = \mathbf{WB}[N]$ where W has key length κ, tweak space $\mathbf{T}$ and input and output lengths $\{0,1\}^{\omega(\lambda)\mu(\lambda)}$ is defined by $W(k,t,x) = \mathsf{WB}(t,x{:}E(k,\cdot))$ and $W^{-1}(k,t,y) = \mathsf{WB}^{-1}(t,y{:}E^{-1}(k,\cdot))$. EME is an example of a wideblock mode of operation that, in this way, transforms a given blockcipher into a wideblock tweakable cipher. Let $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$ be a wideblock mode of operation. Consider the games of Fig. 2.6 and let

$$\mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{\mathbf{WB},B}(\lambda) = \Pr[\mathrm{RO}^{B}_{\mathbf{WB}}(\lambda)] - \Pr[\mathrm{Rand}^{B}_{\omega(\lambda)\mu(\lambda)}(\lambda)] \ .$$

Game $\mathrm{RO}_{\mathbf{WB}}$ instantiates the oracles of WB and WB^{-1} with a random narrow block permutation and its inverse, respectively, while game Rand_{mn} responds to oracle queries via a random wideblock permutation. Now let

$$\mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{\mathbf{WB},Q,\mu,\omega,\tau}(\lambda) = \max_{B} \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{\mathbf{WB},B}(\lambda)$$

where the maximum is over all adversaries B making at most $Q : \mathbb{N} \to \mathbb{N}$ oracle queries with the tweak argument in each query having length at most $\mu(\lambda)\tau(\lambda)$. Proving security of a mode of operation ubiquitously proceeds by upper bounding this advantage as a function of Q, μ, ω, τ. This is a purely information theoretic setting, and absolute bounds are provided. For example, for EME, it is shown in [79] that this advantage is at most

MAIN / $\mathrm{RO}_{\mathbf{WB}}$
$\pi \leftarrow\!\!\$\ \mathsf{Pm}(\{0,1\}^{\omega(\lambda)})$
$b' \leftarrow\!\!\$\ A^{\mathrm{FN},\mathrm{FN}^{-1}}(1^\lambda)$
Return $(b'=1)$

proc $\mathrm{FN}(t,m)$ / $\mathrm{RO}_{\mathbf{WB}}$
Return $\mathsf{WB}(t,m:\pi)$

proc $\mathrm{FN}^{-1}(t,c)$ / $\mathrm{RO}_{\mathbf{WB}}$
Return $\mathsf{WB}^{-1}(t,c:\pi^{-1})$

MAIN / $\mathrm{Rand}_{\mu(\lambda)\omega(\lambda)}$
$\pi \leftarrow\!\!\$\ \mathsf{TwPm}(\mathbf{T},\{0,1\}^{\mu(\lambda)\omega(\lambda)})$
$b' \leftarrow\!\!\$\ A^{\mathrm{FN},\mathrm{FN}^{-1}}(1^\lambda)$
Return $(b'=1)$

proc $\mathrm{FN}(t,m)$ / $\mathrm{Rand}_{\mu(\lambda)\omega(\lambda)}$
Return $\pi(t,m)$

proc $\mathrm{FN}^{-1}(t,c)$ / $\mathrm{Rand}_{\mu(\lambda)\omega(\lambda)}$
Return $\pi^{-1}(t,c)$

MAIN / J
$\pi \leftarrow\!\!\$\ \mathsf{TwPm}(\mathbf{T},\{0,1\}^{\mu(\lambda)})$; $K \leftarrow\!\!\$\ \{0,1\}^{\kappa(\lambda)}$
$b' \leftarrow\!\!\$\ A^{\mathrm{FN},\mathrm{FN}^{-1},\mathrm{KDF}_{\mathrm{FN}}}(1^\lambda)$; Return $(b'=1)$

proc $\mathrm{FN}(t,m)$ / J
Return $\pi(t,m)$

proc $\mathrm{KDF}_{\mathrm{FN}}(t,\mathrm{id}_m)$ / J
For $i=1,\ldots,m$ do
  If $m[i]=\perp$ then $m[i] \leftarrow k$
Return $\pi(t,m)$

proc $\mathrm{FN}^{-1}(t,c)$ / J
Return $\pi^{-1}(t,c)$

Figure 2.6. On the top are games $\mathrm{RO}_{\mathbf{WB}}, \mathrm{Rand}_{mn}$ to define security of wideblock mode of operation $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$. On the bottom is the final game J for the proof of Theorem 2.4.1.

$7(Q(\lambda)\tau(\lambda)+Q(\lambda)\mu(\lambda)+1)^2/2^{\mu(\lambda)}$. EME requires $\mu(\lambda) \le \omega(\lambda)$, assumed in this bound. PRP-CCA security of $\mathbf{WB}[N]$ follows from the assumption that N is a PRP-CCA secure blockcipher by a standard reduction argument.

Construction

Let ECB denote the oracle algorithm that given oracle access to a permutation $g\colon\ \{0,1\}^{\mu(\lambda)} \to \{0,1\}^{\mu(\lambda)}$, and given a input $m = m[1]\ldots m[\omega(\lambda)]$, returns the $\omega(\lambda)$-bit string $c = \mathsf{ECB}(m:g)$ defined by $c[i] = g(m[i])$ for $1 \le i \le \omega(\lambda)$. Let $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$ be a wideblock mode of operation with tweakspace $\mathbf{T}$, blocklength μ and number of blocks ω. Let F be a blockcipher with block size and key size μ and a tweak space $\mathbf{T}$ with $\mathbf{T}(\lambda) = \{\gamma_\lambda^1, \gamma_\lambda^2\}$ be a tweakable blockcipher with a tweakspace of size two. (Of

course, any tweakable blockcipher with an even larger tweakspace will do. Just drop all but two possible tweaks.) Our **EtE** (ECB-then-encipher) transform associates to **WB** and F a tweakable wideblock cipher $J = \mathbf{EtE}[\mathbf{WB}, \mathsf{F}]$ where J has key length μ and input length $\mu(\lambda)\omega(\lambda)$ and tweak space $\mathbf{T}$ and is defined as follows:

$\underline{J(1^\lambda, k, t, m)}$

01 $x \leftarrow \mathsf{ECB}(m : \mathsf{F}(1^\lambda, k, \gamma_\lambda^1, \cdot))$

02 $y \leftarrow \mathsf{WB}(t, x : \mathsf{F}(1^\lambda, k, \gamma_\lambda^2, \cdot))$

03 Return y

The inverse is defined by

$\underline{J^{-1}(k, t, y)}$

01 $x \leftarrow \mathsf{WB}^{-1}(t, y : \mathsf{F}^{-1}(1^\lambda, k, \gamma_\lambda^2, \cdot))$

02 $m \leftarrow \mathsf{ECB}(x : \mathsf{F}^{-1}(1^\lambda, k, \gamma_\lambda^1, \cdot))$

03 Return m

We take advantage here of the fact that $\mathsf{ECB}(\cdot : g^{-1})$ is the inverse of $\mathsf{ECB}(\cdot : g)$. When instantiated with EME in the role of **WB** this uses three blockcipher invocations per block, but retains the parallelizability of EME. F would be obtained by applying **StE** to some tweakable blockcipher with a tweakspace of size three.

The class id_ω

We formally define the class Φ capturing occurrence of the key in any block of the message. Associate to any $\omega(\lambda)$-vector m over $\{0,1\}^{\mu(\lambda)} \cup \{\perp\}$ the function id_m: $\{0,1\}^{\mu(\lambda)} \to \{0,1\}^{\mu(\lambda)\omega(\lambda)}$ that on input a key $k \in \{0,1\}^{\mu(\lambda)}$ returns the message $m' = m'[1] \ldots m'[\omega(\lambda)]$ defined by $m'[i] = m[i]$ if $m[i] \neq \perp$ and $m'[i] = k$ if $m[i] = \perp$ for all $1 \leq i \leq \omega(\lambda)$. Then id_ω is the class of all id_m as m ranges over all m-vectors over $\{0,1\}^{\mu(\lambda)} \cup \{\perp\}$. This is the class Φ we will consider.

Security

Assume **WB** is PRP-CCA secure and assume F is $\{\mathrm{id}\}$-PRP-CCA secure, meaning can safely encrypt its own key. We claim that J is id_m-PRP-CCA secure, meaning can safely encrypt messages that contain the key in any block.

Theorem 2.4.1 *Let* $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$ *be a wideblock mode of operation with tweak space* $\mathbf{T}$*, blocklength n and number of blocks m. Let* F *be a blockcipher with block size and key size* μ *and a tweak space* $\mathbf{T}$ *with* $\mathbf{T}(\lambda) = \{\gamma_\lambda^1, \gamma_\lambda^2\}$ *be a tweakable blockcipher with a tweakspace of size two. Let* $J = \mathbf{EtE}[\mathbf{WB}, F]$ *be the wideblock tweakable cipher associated to* **WB** *and* F *via the* **EtE** *transform as defined above. Let* $\Phi = \mathrm{id}_\omega$*. Let A be an adversary making at most Q oracle queries with the tweak argument in each query having length at most nt. Then there is an adversary B such that*

$$\mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{J,\Phi,A}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{F,\{\mathrm{id}\},B}(\lambda) + 2\delta(\mathbf{WB}, \lambda) + \frac{2Q^2(\mu(\lambda)^2 + 2)}{2^{\omega(\lambda)+2}},$$

where $\delta(\mathbf{WB}, \lambda) = \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{\mathbf{WB},Q,\mu,\omega,\tau}(\lambda)$*. Moreover, the running time of B is about equal to the running time of A.*

The proof would seem at first to be a quite straightforward simulation in which KDF_N queries of A can be answered via KDF_N queries of B. The subtle issue is that B needs to be legitimate and this means it cannot answer all F_N^{-1} queries of A. Ensuring B is legitimate makes the proof more involved.

MAIN / G_0, G_1
$k \leftarrow\$ \{0,1\}^{\kappa(\lambda)}$; $S \leftarrow \emptyset$
Return $(b' = 1)$

proc $\mathrm{FN}(t,m)$ / G_0, G_1
$x \leftarrow \mathsf{ECB}(m : \mathsf{F}(k, \gamma_\lambda^1, \cdot))$
$y \leftarrow \mathsf{WB}(t, x : \mathsf{F}(k, \gamma_\lambda^2, \cdot))$
Return y

proc $\mathrm{KDFN}(t, \mathrm{id}_m)$ / G_0, G_1
For $i = 1, \ldots, m$ do
  If $m[i] = \perp$ then $m[i] \leftarrow k$
$x \leftarrow \mathsf{ECB}(m : \mathsf{F}(k, \gamma_\lambda^1, \cdot))$
$S \leftarrow S \cup \{ x[i] : 1 \le i \le m \}$
$y \leftarrow \mathsf{WB}(t, x : \mathsf{F}(k, \gamma_\lambda^2, \cdot))$
Return y

proc $\mathrm{FN}^{-1}(t,c)$ / $\boxed{G_0}, G_1$
$x \leftarrow \mathsf{WB}^{-1}(t, c : \mathsf{F}^{-1}(k, \gamma_\lambda^2, \cdot))$
$S' \leftarrow \{ \mathsf{F}[i] : 1 \le i \le m \}$
$m \leftarrow \perp$
If $S \cap S' \neq \emptyset$ then $\mathsf{bad} \leftarrow \mathsf{true}$
  $\boxed{m \leftarrow \mathsf{ECB}(x : \mathsf{F}^{-1}(k, \gamma_\lambda^1, \cdot))}$
Else $m \leftarrow \mathsf{ECB}(x : \mathsf{F}^{-1}(k, \gamma_\lambda^1, \cdot))$
Return m

MAIN / H
$\pi \leftarrow\$ \mathsf{TwPm}(\{\gamma_\lambda^1, \gamma_\lambda^2\}, \{0,1\}^{\mu(\lambda)})$
$k \leftarrow\$ \{0,1\}^n$; $S \leftarrow \emptyset$
Return $(b' = 1)$

proc $\mathrm{FN}(t,m)$ / H
$x \leftarrow \mathsf{ECB}(m : \pi(\gamma_\lambda^1, \cdot))$
$y \leftarrow \mathsf{WB}(t, x : \pi(\gamma_\lambda^2, \cdot))$
Return y

proc $\mathrm{KDFN}(t, \mathrm{id}_m)$ / H
For $i = 1, \ldots, m$ do
  If $m[i] = \perp$ then $m[i] \leftarrow k$
$x \leftarrow \mathsf{ECB}(m : \pi(\gamma_\lambda^1, \cdot))$
$S \leftarrow S \cup \{ x[i] : 1 \le i \le m \}$
$y \leftarrow \mathsf{WB}(t, x : \pi(\gamma_\lambda^2, \cdot))$
Return y

proc $\mathrm{FN}^{-1}(t,c)$ / H
$x \leftarrow \mathsf{WB}^{-1}(t, c : \pi^{-1}(\gamma_\lambda^2, \cdot))$
$S' \leftarrow \{ X[i] : 1 \le i \le m \}$
$m \leftarrow \perp$
If $S \cap S' \neq \emptyset$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else $m \leftarrow \mathsf{ECB}(x : \pi^{-1}(\gamma_\lambda^1, \cdot))$
Return m

Figure 2.7. Games used in the proof of Theorem 2.4.1.

Proof: We begin with the games in Fig. 2.7 and Fig. 2.8. We have

$$\begin{aligned} \Pr[\mathrm{Real}^A_{J,\mathrm{id}_\omega}(\lambda)] &= \Pr[\mathrm{G}_0^A(\lambda)] \\ &= \Pr[\mathrm{G}_0^A(\lambda)] - \Pr[\mathrm{G}_1^A(\lambda)] + \Pr[\mathrm{G}_1^A(\lambda)] \\ &\le \Pr[\mathrm{G}_1^A(\lambda)] + \Pr[\mathrm{BAD}(\mathrm{G}_1^A(\lambda))] \,, \end{aligned} \tag{2.3}$$

the inequality by the Fundamental Lemma of Game Playing [25]. We will design B_1, B_2

MAIN // $\mathrm{I}_0, \mathrm{I}_1$
$\pi_1 \leftarrow\$ \mathsf{TwPm}(\{\gamma_\lambda^1\}, \{0,1\}^{\mu(\lambda)})$
$\pi_2 \leftarrow\$ \mathsf{TwPm}(\mathbf{T}, \{0,1\}^{\mu(\lambda)})$
$k \leftarrow\$ \{0,1\}^n$; $S \leftarrow \emptyset$
Return $(b' = 1)$

proc $\mathrm{FN}(t,m)$ // $\mathrm{I}_0, \mathrm{I}_1$
$x \leftarrow \mathsf{ECB}(m : \pi_1(\gamma_\lambda^1, \cdot))$
$y \leftarrow \pi_2(t,x)$
Return y

proc $\mathrm{KDFN}(t, \mathrm{id}_m)$ // $\mathrm{I}_0, \mathrm{I}_1$
For $i = 1, \ldots, m$ do
  If $m[i] = \perp$ then $m[i] \leftarrow k$
$x \leftarrow \mathsf{ECB}(m : \pi_1(\gamma_\lambda^1, \cdot))$
$S \leftarrow S \cup \{ x[i] : 1 \le i \le m \}$
$y \leftarrow \pi_2(t,x)$
Return y

proc $\mathrm{FN}^{-1}(t,c)$ // $\boxed{\mathrm{I}_0}, \mathrm{I}_1$
$x \leftarrow \pi_2^{-1}(t,c)$
$S' \leftarrow \{ x[i] : 1 \le i \le m \}$
$m \leftarrow \perp$
If $S \cap S' \neq \emptyset$ then $\mathsf{bad} \leftarrow \mathsf{true}$
  $\boxed{m \leftarrow \mathsf{ECB}(x : \pi_1^{-1}(\gamma_\lambda^1, \cdot))}$
Else $m \leftarrow \mathsf{ECB}(x : \pi_1^{-1}(\gamma_\lambda^1, \cdot))$
Return m

Figure 2.8. More games used in the proof of Theorem 2.4.1.

---

so that

$$\Pr[\mathrm{G}_1^A(\lambda)] - \Pr[\mathrm{H}^A(\lambda)] \le \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{\mathsf{F},\{\mathrm{id}\},B_1}(\lambda) \tag{2.4}$$

$$\Pr[\mathrm{BAD}(\mathrm{G}_1^A(\lambda))] - \Pr[\mathrm{BAD}(\mathrm{H}^A(\lambda))] \le \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{\mathsf{F},\{\mathrm{id}\},B_2}(\lambda)\,. \tag{2.5}$$

Adversaries B_1, B_2 are almost the same, differing only in how they take their final decision, and accordingly we unify their descriptions. For $i \in \{1,2\}$, adversary B_i has access to oracles $\mathrm{FN}, \mathrm{KDFN}, \mathrm{FN}^{-1}$ and simulates oracles of the same name for A. It starts by setting $S \leftarrow \emptyset$ and $\mathsf{bad} \leftarrow \mathsf{false}$ and then it runs A, answering its oracle queries as follows. When A queries $\mathrm{FN}(t,m)$, adversary B_i does the following:

For $i = 1, \ldots, m$ do $x[i] \leftarrow \mathrm{FN}(\gamma_\lambda^1, m[i])$
$y \leftarrow \mathsf{WB}(t, x : \mathrm{FN}(\gamma_\lambda^2, \cdot)))$
Return y to A.

The FN calls made here by B_i are to its own FN oracle. When A queries $\mathrm{KDFN}(t, \mathrm{id}_m)$, adversary B_i does the following:

For $i = 1, \ldots, m$ do

If $m[i] = \perp$ then $x[i] \leftarrow \mathrm{KDFN}(\gamma_\lambda^1, \mathrm{id})$

Else $x[i] \leftarrow \mathrm{FN}(\gamma_\lambda^1, m[i])$

$S \leftarrow S \cup \{x[i] : 1 \le i \le m\}$

$y \leftarrow \mathsf{WB}(t, x : \mathrm{FN}(\gamma_\lambda^2, \cdot)))$

Return y to A.

Again the calls made by B_i in the code above are to its own $\mathrm{FN}, \mathrm{KDFN}$ oracles. When A queries $\mathrm{FN}^{-1}(t, c)$, adversary B_i does the following:

$x \leftarrow \mathsf{WB}^{-1}(t, x : \mathrm{FN}^{-1}(\gamma_\lambda^2, \cdot)))$

$S' \leftarrow \{X[i] : 1 \le i \le m\}$

$m \leftarrow \perp$

If $S \cap S' = \emptyset$ then

For $i = 1, \ldots, m$ do $m[i] \leftarrow \mathrm{FN}^{-1}(\gamma_\lambda^1, x[i])$

Else $\mathsf{bad} \leftarrow \mathsf{true}$

Return m to A.

As before, the calls made by B_i here are to its own FN^{-1} oracle. So far there has been no difference between B_1, B_2 but now, when A halts, they compute their outputs differently, B_1 returning the same output as A, but B_2 returning 1 if $\mathsf{bad} = \mathsf{true}$ and 0 otherwise.

We claim that B_i is legitimate, meaning that it never queries FN^{-1} at a point (t, c) which was output by a call to KDFN with tweak t. The FN^{-1} queries issued by B_1 occur at two points, both when processing FN^{-1} queries issued by A. Note that B_i only queries KDFN with tweak γ_λ^1, and since the FN^{-1} queries used in the computation of WB^{-1} are all with the distinct tweak γ_λ^2, these queries will not violate legitimacy. The other queries

made by B_i to FN^{-1}, which *are* under γ_λ^1, are only made if $S \cap S' = \emptyset$, and hence do not violate legitimacy. This explains why bad is set this way in the games. Now we have

$$\begin{aligned}
\Pr[\mathrm{Real}^{B_1}_{F,\{\mathrm{id}\}}(\lambda)] &= \Pr[\mathrm{G}^A_0(\lambda)] \\
\Pr[\mathrm{Rand}^{B_1}_{F,\{\mathrm{id}\}}(\lambda)] &= \Pr[\mathrm{H}^A(\lambda)] \\
\Pr[\mathrm{Real}^{B_2}_{F,\{\mathrm{id}\}}(\lambda)] &= \Pr[\mathrm{BAD}(\mathrm{G}^A_0(\lambda))] \\
\Pr[\mathrm{Rand}^{B_2}_{F,\{\mathrm{id}\}}(\lambda)] &= \Pr[\mathrm{BAD}(\mathrm{H}^A(\lambda))]
\end{aligned}$$

which yields Equations (2.4) and (2.5).

Let B pick $i \in \{1,2\}$ at random and run B_i, and let $\delta(F,\lambda) = \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{F,\{\mathrm{id}\},B}(\lambda)$. Then from Equations (2.3), (2.4), (2.5) we have

$$\Pr[\mathrm{G}^A_0(\lambda)] \le 2\delta(F,\delta) + \Pr[\mathrm{H}^A] + \Pr[\mathrm{BAD}(\mathrm{H}^A)]\ . \tag{2.6}$$

We will design B_3, B_4 so that

$$\begin{aligned}
\Pr[\mathrm{H}^A(\lambda)] - \Pr[\mathrm{I}^A_1(\lambda)] &\le \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{\mathbf{WB},B_3}(\lambda) && (2.7)\\
\Pr[\mathrm{BAD}(\mathrm{H}^A(\lambda))] - \Pr[\mathrm{BAD}(\mathrm{I}^{A(\lambda)}_1)] &\le \mathsf{Adv}^{\mathsf{prp\text{-}cca}}_{\mathbf{WB},B_4}(\lambda)\ . && (2.8)
\end{aligned}$$

Adversaries B_3, B_4 are almost the same, differing only in how they take their final decision, and accordingly we unify their descriptions. For $i \in \{3,4\}$, adversary B_i has access to oracles $\mathrm{FN}, \mathrm{FN}^{-1}$ and provides oracles $\mathrm{FN}, \mathrm{KDFN}, \mathrm{FN}^{-1}$ to A. It starts by selecting k at random from $\{0,1\}^{\mu(\lambda)}$. It will then pick π at random from $\mathsf{TwPm}(\{\gamma_\lambda^1\}, \{0,1\}^{\mu(\lambda)})$. This is a conceptual simplification. Adversary B_i can't really pick π in advance like this and remain efficient. Instead it will build π on the fly via lazy sampling. It sets $S \leftarrow \emptyset$ and $\mathsf{bad} \leftarrow \mathsf{false}$ and then runs A. When A queries $\mathrm{FN}(t,m)$, adversary B_i does the following:

$x \leftarrow \mathsf{ECB}(m : \pi(\gamma_\lambda^1, \cdot))$

$y \leftarrow \mathrm{FN}(t, x)$

Return y to A.

When A queries $\mathrm{KDFN}(t, \mathrm{id}_m)$, adversary B_i does the following:

For $i = 1, \ldots, m$ do

If $m[i] = \perp$ then $m[i] \leftarrow k$

$x \leftarrow \mathsf{ECB}(m : \pi(\gamma_\lambda^1, \cdot))$

$S \leftarrow S \cup \{x[i] : 1 \leq i \leq m\}$

$y \leftarrow \mathrm{FN}(t, x)$

Return y to A.

When A queries $\mathrm{FN}^{-1}(t, c)$, adversary B_i does the following:

$x \leftarrow \mathrm{FN}^{-1}(t, c)$

$S' \leftarrow \{X[i] : 1 \leq i \leq m\}$

$m \leftarrow \perp$

If $S \cap S' = \emptyset$ then $x \leftarrow \mathsf{ECB}(m : \pi(\gamma_\lambda^1, \cdot))$

Else $\mathsf{bad} \leftarrow \mathsf{true}$

Return m to A.

So far there has been no difference between B_3, B_4 but now, when A halts, they compute their outputs differently, B_3 returning the same output as A, but B_4 returning 1 if $\mathsf{bad} =$

true and 0 otherwise. We have

$$\begin{aligned}
\Pr[\mathrm{RO}_{\mathbf{WB}}^{B_3}(\lambda)] &= \Pr[\mathrm{H}^A(\lambda)] \\
\Pr[\mathrm{Rand}_{mn}^{B_3}(\lambda)] &= \Pr[\mathrm{I}_1^A(\lambda)] \\
\Pr[\mathrm{RO}_{\mathbf{WB}}^{B_3}(\lambda)] &= \Pr[\mathrm{BAD}(\mathrm{H}^A(\lambda))] \\
\Pr[\mathrm{Rand}_{\omega(\lambda)\mu(\lambda)}^{B_3}(\lambda)] &= \Pr[\mathrm{BAD}(\mathrm{I}_1^A(\lambda))]
\end{aligned}$$

which yields Equations (2.7) and (2.8). Recall that $\delta(\mathbf{WB}) = \mathsf{Adv}_{\mathbf{WB}}^{\mathsf{prp\text{-}cca}}(Q,\mu,\omega,\tau)$. Then from Equations (2.6), (2.7), (2.8) we have

$$\Pr[\mathrm{G}_0^A(\lambda)] - \Pr[\mathrm{I}_1^A(\lambda)] \le 2\delta(F,\lambda) + 2\delta(\mathbf{WB},\lambda) + \Pr[\mathrm{BAD}(\mathrm{I}_1^A(\lambda))]\,. \tag{2.9}$$

Consider game J of Fig. 2.6. Composing a random permutation with any independently chosen permutation yields a random permutation, regardless of the distribution of the second permutation, so

$$\begin{aligned}
\mathsf{Adv}_{J,\mathrm{id}_\omega,A}^{\mathsf{prp\text{-}cca}}(\lambda) &= \Pr[\mathrm{Real}_{J,\mathrm{id}_\omega,A}^{\lambda}] - \Pr[\mathrm{Rand}_{J,\mathrm{id}_\omega}^{A}(\lambda)] \\
&= \Pr[\mathrm{G}_0^A(\lambda)] - \Pr[\mathrm{J}^A(\lambda)] \\
&= \Pr[\mathrm{G}_0^A(\lambda)] - \Pr[\mathrm{I}_0^A(\lambda)] \\
&= (\Pr[\mathrm{G}_0^A(\lambda)] - \Pr[\mathrm{I}_1^A(\lambda)]) + (\Pr[\mathrm{I}_1^A(\lambda)] - \Pr[\mathrm{I}_0^A(\lambda)]) \\
&\le (\Pr[\mathrm{G}_0^A(\lambda)] - \Pr[\mathrm{I}_1^A(\lambda)]) + \Pr[\mathrm{BAD}(\mathrm{I}_1^A(\lambda))]\,.
\end{aligned}$$

Putting this together with Equation (2.9) we have

$$\mathsf{Adv}_{J,\mathrm{id}_\omega,A}^{\mathsf{prp\text{-}cca}}(\lambda) \le 2\delta(F,\lambda) + 2\delta(\mathbf{WB},\lambda) + 2\cdot\Pr[\mathrm{BAD}(\mathrm{I}_1^A(\lambda))]\,. \tag{2.10}$$

To complete the proof we bound the last term above. The assumption that A is legitimate (this is where we use it) implies that if it makes query $\mathrm{FN}^{-1}(t,c)$ then it made no previous $\mathrm{KDFN}(t,\cdot)$ query that returned c. We can wlog also assume that no previous $\mathrm{FN}(t,\cdot)$ query returned c. Taking a hit of $Q(\lambda)^2 \cdot 2^{-\omega(\lambda)\mu(\lambda)-1}$ in the bound, we can view $\pi(t,\cdot)$ as a random function rather than a random permutation, so the response to a new $\mathrm{FN}^{-1}(t,c)$ query is a new, random string, meaning each block is uniformly distributed in $\{0,1\}^{\mu(\lambda)}$. If A made $q_1(\lambda)$ queries to KDFN then the set S has size at most $q_1(\lambda)\omega(\lambda)$ and each FN^{-1} inverse query has chance at most $q_1\omega(\lambda)^2 2^{-\mu(\lambda)}$ of setting bad. If it made $q_2(\lambda)$ queries to FN^{-1}, the overall chance of setting bad is at most $q_1(\lambda)q_2(\lambda)\omega(\lambda)^2 2^{-\mu(\lambda)}$. But $q_1(\lambda)+q_2(\lambda) \le Q(\lambda)$ so $q_1(\lambda)q_2(\lambda) \le Q(\lambda)^2/4$ and thus

$$\Pr[\mathrm{BAD}(\mathrm{I}_1^A(\lambda))] \le \frac{Q(\lambda)^2}{2^{\omega(\lambda)\mu(\lambda)+1}} + \frac{Q(\lambda)^2\omega(\lambda)^2}{2^{\mu(\lambda)+2}} \le \frac{Q(\lambda)^2(\omega(\lambda)^2+2)}{2^{\mu(\lambda)+2}} .$$

Combining this with Equation (2.10) completes the proof.∎

2.5 Implementation

We describe a fast AES-based instantiation of **EtE**. As a starting point, we need an AES-based tweakable blockcipher. LRW [87] and XEX [79] are possible choices. Both of them require two calls to the base AES blockcipher, but with XEX, one of the calls can be can be made just once and the value cached as the cipher is used to encipher many message blocks. We thus choose XEX. Applying **StE** to XEX yields a tweakable blockcipher F. Now we need to pick a wideblock cipher to play the role of **WB**. We pick EME which uses about two blockcipher calls per message block together with overhead for offset computations. Let J be the wideblock tweakable cipher produced by applying **EtE** to **WB** and F. We optimized J and compared its speed to that of EME.

XEX

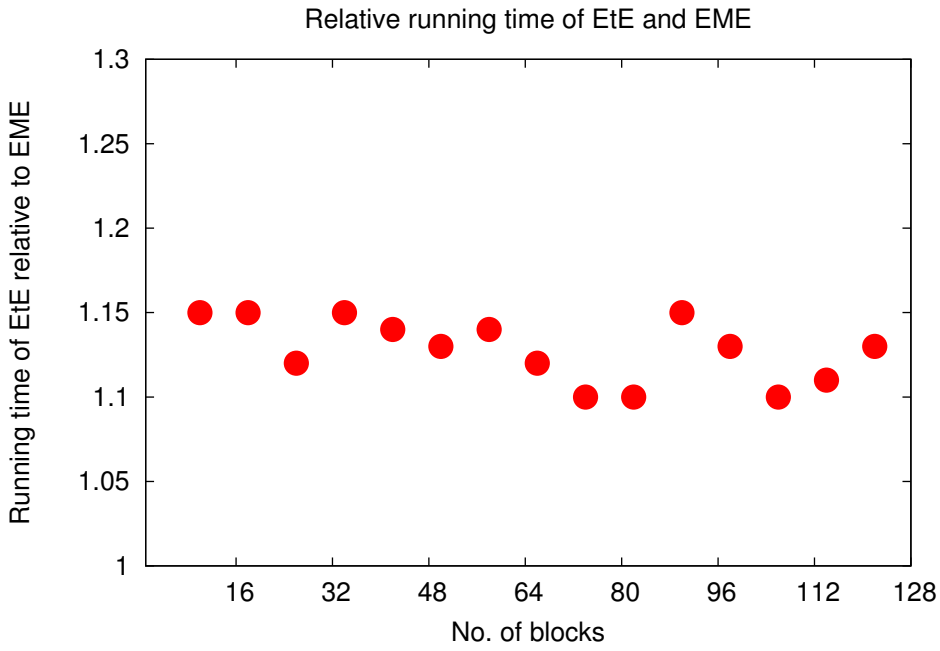


Figure 2.9. Running time of J on a scale where the running time of EME has been normalized to 1.

Fix a security parameter $\lambda = n$. XEX over a blockcipher E: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ has tweakspace $\{0,1\}^n \times \mathbb{I}$, where $\mathbb{I}$ is the set of integers $[1,\ldots,2^n-2]$. Arithmetic operations in XEX are done in the field GF(2^n), with elements of GF(2^n) viewed interchangeably as n-bit strings, integers in $[0,2^n-1]$ and as formal polynomials of degree $n-1$ with binary coefficients. Addition is bitwise XOR and multiplication of two points is performed by multiplying them as formal polynomials modulo the irreducible polynomial $p_{128}(x) = x^{127}+x^7+x^2+x+1$. XEX is defined by $\tilde{E}_K^{(N,i)}(M) = E_K(M \oplus \Delta) \oplus \Delta$ where $\Delta = 2^i\mathsf{N}$ and $\mathsf{N} = E_K(N)$. For our purposes, we require a much smaller tweak space of two elements. We pick $t_0 = (0^n,1)$ and $t_1 = (10^{n-1},1)$. Now, Δ can assume only two values (Δ_0,Δ_1) which can be precomputed and cached between successive calls, avoiding running E for the second time to calculate N. The XOR operations still remain, but they are parallelizable, using vector instructions [57].

Experiments

We compare the speeds of **EtE** and EME on a processor that has the AES-NI support for executing AES in hardware [73]. Our results show that encrypting with **EtE** is slower than EME by a modest 15% or less. We ran the tests described below on a Intel Xeon W3690 processor at 3.47 GHz with support for AES-NI running Linux version 2.6 with code compiled using `gcc -O3 -ftree-vectorize -msse2 -ffast-math` to enable vector instructions and perform other general optimizations. The base code was from Brian Gladman's EME2 implementation [64], which we modified to run as EME and use the hardware AES instructions.

Acknowledgements

This chpater is largely a reproduction of the material as it appears in Ciphers That Securely Encipher Their Own Keys, by Mihir Bellare, David Cash, and Sriram Keelveedhi from Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, 2011. I wish to thank David Cash for being an excellent co-author on this work and for being a knowledgeable yet friendly presence during my first couple of years.

Chapter 3

Encryption of key-dependent messages

We provide a comprehensive treatment of security for key-dependent data for authenticated encryption, the central practical goal of symmetric cryptography. 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.

3.0.1 Background

As, we discussed in Section 2, in the past, encyrption schemes were not been designed to work in settings when the data being encrypted is related to the encyrption key, following the view that key-dependent data is not a concern in practical systems. However, security of key-dependent data has been a growing concern of late. The first indications of prevalance of key-dependent data came from applications such as anonymous credentials [43] and connections of cryptography to formal methods [29], the latter being a more direct and widespread concern for secure systems.

Wider scenarios where key-dependent data is a reality started appearing following these cases. Disk encryption is also an area where encryption or enciphering of key-dependent messages can occur. For instance, the key used by disk-encryption tools such

as BitLocker, could end up on disk due to various events, such as forced shutdowns and the machine going into standby. As a result, the key under which a filesystem is encrypted may itself be stored in a file on the same system.

Password based encryption is another area where key-dependent data is a particularly acute issue. Passwords are by nature derived from users personal contexts, and could have strong relations with data being encyrpted by the users. A user could set her password to the name of her dog, and then encrypt emails under the password. These emails could of course have her dog's name appearing in the text. Moreover, many of us store our passwords on our systems and systems store password hashes.

The main takeaway from these examples is that security in the presence of key-dependent data is essential for robust and misuse-resistant cryptography, because one cannot expect applications to ensure or certify that their data is *not* key-dependent. Thus, it is important to design practical cryptosystems which remain secure when encrypting key-dependent data. We should start by designing security goals designed to enforce the sort of key-dependencies that arise in real systems, and follow this by designing practical and efficient schemes achieving these targets.

3.0.2 Related work

The issue of key-dependent messages was pointed out as early as Goldwasser and Micali [67], and asymmetric encryption of decryption keys was treated by Camenisch and Lysyanskaya [43], but a full treatment of key-dependent message (KDM) encryption awaited BRS [29], 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 [36, 82, 5, 38, 42, 37, 9, 47, 27, 4, 90]. On the more practical side, Backes, Dürmuth and Unruh [6] show that RSA-OAEP [23, 59] is

KDM-secure in the RO model. Backes, Pfitzmann and Scedrov [7] 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 [1] and Green and Hohenberger [71] 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, Belenkiy, Bellare, and Cash [1] also connect KDM secure encryption to cryptographic agility. Haitner and Holenstein [74] study the difficulty of proving KDM security by blackbox reduction to standard primitives.

Halevi and Krawczyk [77] consider blockciphers under key-dependent inputs. Muñiz and Steinwandt [94] study KDM secure signatures. González, in an unpublished thesis [68], studies KDM secure MACs.

Motivated by attacks on SSH, Paterson and Watson [96] consider notions of security in the standard key independent 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 [96] to allow key-dependent data is an interesting direction for future work.

Black, Rogaway and Shrimpton (BRS) [29] introduced the problem of encryption of key-dependent messages, and proposed a new notion, KDM security, to extend IND-CPA to allow key-dependent messages (KDMs). A new notion was necessary as the standard goals for symmetric encyrption, IND-CPA and IND-CCA, that our schemes are proven to meet, do not guarantee security when the message being encrypted depends on the key. In this notion, the game provides the adversary with an encryption oracle that takes input a function ϕ, called a message-deriving function, that the game applies to the target key k to get a message m. Then, the adversary is returned either an encryption of m under k or the encryption of $0^{|m|}$. To win, the adversary must distinguish between the

two scenarios. In reality, both BRS and our work actually considers a multi-key setting, where messages can depend on several keys. We restrict attention to the single-key setting to simplify the current discussion. BRS present a simple random-oracle (RO) model solution that achieves KDM security.

Post-BRS work has aimed mainly at building schemes secure against as large as possible a class of message deriving functions without random oracles [36, 82, 5, 38, 42, 37, 9, 47, 27, 4, 90]. The resulting schemes suffer from one or more of the following: they are in the asymmetric setting while data encryption in practice is largely in the symmetric setting; 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) [7] define KDM-security for a basic form of authenticated encryption and show that Encrypt-then-MAC [21] 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 [29]. We now extend their treatment of AE in several directions.

Authenticated encryption

In several applications, along with privacy, authenticity is also needed, meaning plain, IND-CPA secure encryption does not suffice. The primitive needed is authenticated encryption (AE), which provides both privacy and integrity. This is evidenced by numerous standards and high usage: CCM [109, 108] is in IEEE 802.11, IEEE 802.15.4, IPSEC ESP and IKEv2; GCM [92] is standardized by NIST as SP 800-38D; EAX [26] is in ANSI C12.22 and ISO/IEC 19772; OCB 2.0 [101, 103] is in ISO 19772.

Authenticated encryption schemes follow the same syntax as regular symmetric encryption schemes. The same correctness conditions apply. Symmetric encryption schemes take as input a nonce, also called an IV. In the classical [67, 15] framework,

the nonce gets chosen at random by the party performing the encryption; we call this random-nonce security, denoted by `r`. Subsequent schemes sought to achieve more robust security by targeting universal-nonce security [100, 102, 104] denoted by `u` 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.

KDM security for authenticated encryption

We look at KDM security for authenticated encryption, specifically towards analyzing the security of the aforementioned standards. Moreover, besides key, nonce and message, modern AE schemes, including the above standards, take input a header, or associated data [100]. 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.

We 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. Existing works have treated the `ki` variants, and in works such as Backes, Pfitzmann and Scedrov [7] the special case of $(\mathtt{nt},\mathtt{mt},\mathtt{ht}) = (\mathtt{r},\mathtt{kd},\mathtt{ki})$ in which the header is absent is treated. Here, we provide treatment for all variants.

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 ϕ_m, ϕ_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 ϕ_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 ϕ_m, ϕ_h are drawn yields the different

Table 3.1. Key-depdendent security for authenticated encryption. The definitions and meaning of Yes and No entries are explained in text.

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

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 [29, 7], we allow any functions in this role. These functions are even allowed to call the RO, which complicates the proofs, but is essential to capture real applications.

Underlying the above definition is a new one of the standard AE goal that simplifies that of [104] 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 [21]. When data is key-independent, these and prior formulations [21, 83] are equivalent, but the difference is important with key-dependent data.

We establish security in the eight variants as follows. Consider the table in Fig. 3.1. 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. We first 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 Fig. 3.1. 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 different and stronger security requirement.

An attack aiming to show that no stateful scheme is KDM-CPA secure was described in [29] 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 [29], 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, and this holds 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 Fig. 3.1. 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.

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 [29] 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 (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. RHtE, given key ℓ and randomness R, derives subkey $k = H(\mathtt{r}\|\ell)$ 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 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, 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 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 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 E be the (deterministic) encryption function of the base scheme. Let $\phi_1, \ldots, \phi_{q_e}$ be the message-deriving functions in A's encryption queries. We begin with a natural hybrid in which the key $k = H(r_g \| \ell)$ 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 $r_g \| \ell$ 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 ϕ_i is given ℓ as input so *we cannot avoid* it querying $r_g \| \ell$ 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 ϕ_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 3.3.1 is weaker than we would like. It is an interesting open problem to either prove a better bound for RHtE or provide an alternative scheme with such a bound.

Extensions

In filesystem encryption, as with most applications, security is likely to stem from the user's password p. The system stores a hash $\overline{\mathsf{pw}} = H(\mathsf{p})$ of it to authenticate the user and an AE scheme must then encrypt or decrypt using p. 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 p that must be stored on the system and thus will be encrypted with disk encryption. To address this, we show that 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.

Implementation

We implemented RHtE for base schemes CCM, EAX and GCM, with SHA256 instantiating the RO. The results, provided in Section 3.4, show for example that with CCM the slowdown is 11% for 5KB messages and only 1% for 50KB messages.

3.1 Definitions

We defined the syntax of authenticated encryption in Section 1. Here, 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

$\underline{\text{MAIN}(1^\lambda)\ \ /\!/\ \text{KIAE}^A_{\mathsf{SE},\mathtt{nt}}(\lambda)}$
$K \leftarrow\!\!\$\ \mathsf{K}(1^\lambda)\ ;\ S \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ A^{\text{ENC},\text{DEC}}(1^\lambda)$
Return $(b' = b)$

$\underline{\text{proc ENC}(n,d,m)\ \ /\!/\ \text{KIAE}^A_{\mathsf{SE},\mathtt{nt}}(\lambda)}$
If $(\mathtt{nt} = \mathtt{r})$ then $n \leftarrow\!\!\$\ \{0,1\}^{\rho(\lambda)}$
If $(b = 1)$ then $c \leftarrow \mathsf{E}(1^\lambda, k, n, d, m)$
Else
  $\gamma \leftarrow \mathsf{cl}(\lambda, |m|, |d|)$; $c \leftarrow\!\!\$\ \{0,1\}^\gamma$
$S \leftarrow S \cup \{(n,d,c)\}$
Return (n,c)

$\underline{\text{proc DEC}(n,d,c)\ \ /\!/\ \text{KIAE}^A_{\mathsf{SE},\mathtt{nt}}(\lambda)}$
If $(n,d,c) \in S$ then return $\perp$
If $(b = 1)$ then $m \leftarrow \mathsf{D}(1^\lambda, k, d, n, c)$
Else $m \leftarrow \perp$
If $m = \perp$ then $v \leftarrow 0$ else $v \leftarrow 1$
Return v

$\underline{\text{MAIN}(1^\lambda)\ \ /\!/\ \text{AE}^A_{\mathsf{SE},\mathtt{nt}}(\lambda)}$
For $j = 1, \ldots, \nu(\lambda)$ do
  $k_j \leftarrow\!\!\$\ \mathsf{K}(1^\lambda)$; $S_j \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$; $b' \leftarrow\!\!\$\ A^{\text{ENC},\text{DEC}}(1^\lambda)$
Return $(b' = b)$

$\underline{\text{proc ENC}(j,n,\phi_h,\phi_m)\ \ /\!/\ \text{AE}^A_{\mathsf{SE},\mathtt{nt}}(\lambda)}$
$m \leftarrow \phi_m(k_1, \ldots, k_{\nu(\lambda)})$
$d \leftarrow \phi_h(k_1, \ldots, k_{\nu(\lambda)})$
If $(\mathtt{nt} = \mathtt{r})$ then $n \leftarrow\!\!\$\ \{0,1\}^{\rho(\lambda)}$
If $(b = 1)$ then $c \leftarrow \mathsf{E}(1^\lambda, k_j, n, d, m)$
Else
  $\gamma \leftarrow \mathsf{cl}(\lambda, \mathsf{ol}(\phi_m), \mathsf{ol}(\phi_h))$
  $c \leftarrow\!\!\$\ \{0,1\}^\gamma$
$S_j \leftarrow S_j \cup \{(n,d,c)\}$; Return (n,c)

$\underline{\text{proc DEC}(j,n,\phi_h,c)\ \ /\!/\ \text{AE}^A_{\mathsf{SE},\mathtt{nt}}(\lambda)}$
$d \leftarrow \phi_h(k_1, \ldots, k_{\nu(\lambda)})$
If $(n,d,c) \in S_j$ then return $\perp$
If $(b = 1)$ then
  $m \leftarrow \mathsf{D}(k_j, n, d, c)$ else $m \leftarrow \perp$
If $m = \perp$ then return 0 else return 1

Figure 3.1. Game $\text{KIAE}_{\mathsf{SE},\mathtt{nt}}$ (left) defining AE-security of encryption scheme $\mathsf{SE} = (\mathsf{K},\mathsf{E},\mathsf{D})$, where $\mathtt{nt} \in \{\mathtt{u},\mathtt{r}\}$ indicates universal or random nonce, and (right) game $\text{AE}_{\mathsf{SE},\nu(\lambda),\mathtt{nt}}$ defining KDI AE-security of SE.

messages and headers.

AE security

We now define standard (neither message nor header is key-dependent) AE security for $\mathsf{SE} = (\mathsf{K},\mathsf{E},\mathsf{D})$. Consider game $\text{KIAE}^A_{\mathsf{SE},\mathtt{nt}}$ shown on the left side of Fig. 3.1. Define the advantage of adversary A via

$$\text{Adv}^{\mathtt{ae\text{-}nt}}_{\mathsf{SE},A}(\lambda) = 2\Pr[\text{KIAE}^A_{\mathsf{SE},\mathtt{nt}}(\lambda) \Rightarrow \mathsf{true}] - 1.$$

When $\mathtt{nt} = \mathtt{u}$ the definition captures what we call universal-nonce security. (It is simply called nonce-based security in [102, 100, 104].) 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, d, m and n', d', 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 [67, 15]. 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 [21, 83, 24]. Our version is a blend of the single-game formulation of [104] 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 [15] privacy, namely that ciphertexts of different messages are indistinguishable from each other. (A subtle point is that the length-respecting property assumed of 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 [104], 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 [21, 100].

KDI security of AE

We now extend the above along the lines of [29, 7] 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 $\mathrm{AE}_{\mathsf{SE},\mathtt{nt}}$ shown on the right side of

Fig. 3.1. Define the advantage of adversary A via

$$\mathsf{Adv}^{\mathtt{ae\text{-}nt}}_{\mathsf{SE},A}(\lambda) = 2\Pr[\mathrm{AE}^{A}_{\mathsf{SE},\mathtt{nt}}(\lambda) \Rightarrow \mathsf{true}] - 1.$$

The argument $\nu(\lambda)$ to INITIALIZE is the number of keys; arguments ϕ_m, ϕ_h (message and header deriving functions, respectively) in the ENC, DEC queries must be functions in $\mathsf{F}(\kappa(\lambda), \nu(\lambda))$; $\mathsf{ol}(\phi)$ is the output length of $\phi \in \mathsf{F}(\kappa(\lambda), \nu(\lambda))$; and cl is the ciphertext length of 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 $j \in [1..\nu(\lambda)]$ we have $n \neq n'$ for any two ENC queries j, n, ϕ_m, ϕ_h and j, n', ϕ'_m, ϕ'_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 $\Phi_m(\lambda), \Phi_h(\lambda) \subseteq \mathsf{F}(\kappa(\lambda), \nu(\lambda))$ let $\mathbf{A}[\Phi_m(\lambda), \Phi_h(\lambda)]$ be the class of all adversaries A for which ϕ_m in A's ENC queries is in $\Phi_m(\lambda)$ and ϕ_h in its ENC, DEC queries is in $\Phi_h(\lambda)$. Let $f(a,b)$ and $c(a,b)$ denote $\mathsf{F}(a(\lambda), b(\lambda))$ and $\mathsf{C}(a(\lambda), b(\lambda))$ respectively. Let $\mathbf{A}[\mathtt{mt}, \mathtt{ht}] = \mathbf{A}[\Phi_m(\lambda), \Phi_h(\lambda)]$ where the values of $(\Phi_m(\lambda), \Phi_h(\lambda))$ corresponding to $(\mathtt{mt}, \mathtt{ht}) = (\mathtt{kd}, \mathtt{kd}), (\mathtt{kd}, \mathtt{ki}), (\mathtt{ki}, \mathtt{kd}), (\mathtt{ki}, \mathtt{ki})$ are, respectively, $(f(\kappa, \nu), f(\kappa, \nu))$, $(f(\kappa, \nu), c(\kappa, \nu))$, $(c(\kappa, \nu), f(\kappa, \nu))$, and $(c(\kappa, \nu), c(\kappa, \nu))$. We say that $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ is $(\mathtt{nt}, \mathtt{mt}, \mathtt{ht})$-AE secure if $\mathsf{Adv}^{\mathtt{ae\text{-}nt}}_{\mathsf{SE}}(A)$ is negligible for all efficient $A \in \mathbf{A}[\mathtt{mt}, \mathtt{ht}]$.

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 ϕ_m, ϕ_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 K in our syntax $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ does not have to output random $\kappa(\lambda)$-bit strings but could induce an arbitrary distribution, allowing us to capture passwords. The metric of interest in this case is the min-entropy $\mathbf{H}_\infty[\mathsf{K}] : \mathbb{N} \to \mathbb{R}$ defined via $\mathbf{H}_\infty[\mathsf{K}](\lambda) - \log_2(\mathbf{GP}(\mathsf{K}(1^\lambda)))$, where the guessing probability $\mathbf{GP}(\mathsf{K})$ is defined as the maximum, over all $\kappa(\lambda)$-bit strings k, of the probability that $k' = k$ when $k' \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$. 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.

3.2 Impossibility Results

We now rule out universal-nonce security for key-dependent messages as well as security for key-dependent headers, by first presenting 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. As we explore below, the attack requires only that the nonce is predictable, and works 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. Morevoer, the attack does not use the decryption oracle, so rules out even KDM universal-nonce CPA secure encryption. An important consequence is that the universal-nonce security proven for the standardized schemes for key-independent messages fails to extend to key-dependent ones.

We then present an attack showing 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 rules out security of the standardized schemes even with random nonces in a setting where headers may be key-dependent.

All the standardized schemes achieve universal-nonce security for $\mathtt{ki}$-messages. This is convenient as applications 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 3.2.1 Let $\mathsf{SE} = (\mathsf{K},\mathsf{E},\mathsf{D})$ be an encryption scheme. Then there is an efficient adversary $A \in \mathbf{A}[\mathtt{kd},\mathtt{ki}]$ such that for all $\lambda \in \mathbb{N}$, it holds that

$$\mathsf{Adv}^{\mathsf{ae\text{-}u}}_{\mathsf{SE},A}(\lambda) \geq \frac{1}{4}.$$

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 3.2.1, and finally show how to apply an underlying lemma to provide the first analysis of an attack in BRS [29].

Background and overview

BRS [29, 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 $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$. Let $\kappa : \mathbb{N} \to \mathbb{N}$ be the keylength of the scheme. We will use messages of length $\mu : \mathbb{N} \to \mathbb{N}$. Let $\mathsf{cl} : \mathbb{N} \to \mathbb{N}$ denote the length of the resulting ciphertexts. Let $H_{\mathrm{ip}}(v, c) = v[1]c[1] + \cdots + v[\mathsf{cl}(\lambda)]c[\mathsf{cl}(\lambda)] \bmod 2$ denote the inner product modulo two of $\mathsf{cl}(\lambda)$-bit strings v, c. Let $\phi_{v,i}$ denote the message-deriving function that on input a key k returns the first $\mu(\lambda)$-bit message m such that $H_{\mathrm{ip}}(v, \mathsf{E}(1^\lambda, k, i, \varepsilon, m)) = k[i]$, or $0^{\mu(\lambda)}$ if there is no such message. (Here we use i as the nonce and ε 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 $H_{\mathrm{ip}}(v, c)$, repeating for $i = 1, \ldots, \kappa(\lambda)$ 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 $O(1/(\kappa(\lambda))$. (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 $H_{\mathrm{ip}}(v, \mathsf{E}(1^\lambda, k, i, \varepsilon, m)) = k[i]$ 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. One approach to modify the attack so that $\phi_{v_1,\ldots,v_{\ell(\lambda)},i}$ now depends on a sequence $v_1, \ldots, v_{\ell(\lambda)}$ of strings, chosen independently at random by the adversary. On input the key k, the function computes the smallest j such that $H_{\mathrm{ip}}(v_j, \mathsf{E}(1^\lambda, k, i, \varepsilon, m_j)) = k[i]$, where $m_1, m_2, \ldots, m_{\ell(\lambda)}$ 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 $k[i] = H_{\mathrm{ip}}(v_j, c)$ from the ciphertext $c = \mathsf{E}(1^\lambda, k, i, \varepsilon, m_j)$, 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 $H\colon \{0,1\}^{\sigma(\lambda)} \times \{0,1\}^{\mathsf{cl}(\lambda)} \to \{0,1\}$ of pairwise independent functions. The message-deriving function $\phi_{s,i}$, on input k, will now search for m such that $H(s, \mathsf{E}(1^\lambda, k, i, \varepsilon, m)) = k[i]$. 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 $i = 1, \ldots, \kappa(\lambda)$ to get k. We will prove that $O(\kappa(\lambda))$ trials suffice for the search to have failure probability at most $O(1/\kappa(\lambda))$ 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 $H\colon \{0,1\}^{\mathsf{cl}(\lambda)+1} \times \{0,1\}^{\mathsf{cl}(\lambda)} \to \{0,1\}$ defined by

$$H(s, c) = H_{\mathrm{ip}}(s[1] \ldots s[\mathsf{cl}(\lambda)], c) + s[\mathsf{cl}(\lambda) + 1] \bmod 2$$

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, $0^{\mathsf{cl}(\lambda)}$ 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. We now proceed to the details.

Lemma 3.2.2 *Let* $H\colon \{0,1\}^{\alpha} \times \{0,1\}^{\beta} \to \{0,1\}$ *be a family of pairwise independent hash functions. Let* $c_1,\ldots,c_l \in \{0,1\}^{\beta}$ *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* $\{0,1\}^{\alpha}$.

Proof:[Lemma 3.2.2] For each $j \in \{1,\ldots,l\}$ define $x_j\colon \{0,1\}^{\alpha} \to \{0,1\}$ 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 $\{0,1\}^{\alpha}$. Let $x = x_1 + \cdots + x_l$ and let $\mu = \mathbf{E}[x]$. By Chebyshev's inequality, the probability above is

$$\Pr[x=0] \leq \Pr[|x-\mu| \geq \mu] \leq \frac{\mathbf{Var}[x]}{\mu^2}.$$

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 $\mathbf{E}\left[x_j\right] = 1/2$ 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 3.2.1.

Proof:[Proposition 3.2.1] Let $\kappa(\lambda)$ be the key length, $\rho(\lambda)$ the nonce length and $\mathsf{cl}(\lambda)$ the ciphertext length of SE. Let $\ell : \mathbb{N} \to \mathbb{N}$ be the function defined by $\ell(\lambda) = 4 \cdot \kappa(\lambda)$. Let $\mathsf{NumToStr}(j)$ denote a representation of integer $j \in \{0,\ldots,\ell(\lambda)\}$ as a string of length

exactly $v = \lceil \log_2(\ell(\lambda)+1) \rceil$ bits. Let H: $\{0,1\}^{\sigma(\lambda)} \times \{0,1\}^{\mathsf{cl}(v,0)} \to \{0,1\}$ denote a family of pairwise independent hash functions with $\sigma(\lambda)$-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 $\mathrm{AE}_{\mathsf{SE},\mathrm{u}}$ 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 $\rho(\lambda)$-bit strings:

Adversary $B(1^\lambda)$	Function $\phi_{\mathbf{m},s,i}(k)$
For $j = 1,\ldots,\ell(\lambda)$ do $\mathbf{m}[j] \leftarrow \mathsf{NumToStr}(j)$	$m \leftarrow \mathsf{NumToStr}(0)$
$s \leftarrow\!\!\$\ \{0,1\}^{\sigma(\lambda)}$	For $j = 1,\ldots,l$ do
For $i = 1,\ldots,\kappa(\lambda)$ do	$\quad c_j \leftarrow \mathsf{E}(K,i,\varepsilon,\mathbf{m}[j])$
$\quad (i,c) \leftarrow\!\!\$\ \mathrm{ENC}(1,i,\phi_\varepsilon,\phi_{\mathbf{m},s,i})$	$\quad$ If $H(s,c_j) = k[i]$ then
$\quad L[i] \leftarrow H(s,c)$	$\quad\quad m \leftarrow \mathbf{m}[j]$
Return L	Return m

Above $\mathbf{m}$ is a $\ell(\lambda)$-vector over $\{0,1\}^v$ and ϕ_ε is the constant function that returns the empty string on every input. In its first step, B initializes the game to play with $v(\lambda) = 1$, meaning a single target key. Function $\phi_{\mathbf{m},s,i}(k)$ 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 3.2.2 says that for each i, adversary B fails to recover $k[i]$ with probability at most $1/4\kappa(\lambda)$. 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 3.2.2 we not only show that the inner-product function works but that it is worse only by a factor of two:

Lemma 3.2.3 *Let* H_{ip}: $\{0,1\}^{\mathsf{cl}(\lambda)} \times \{0,1\}^{\mathsf{cl}(\lambda)} \to \{0,1\}$ *be the function defined by* $H_{\text{ip}}(v,c) = v[1]c[1] + \cdots + v[\mathsf{cl}(\lambda)]c[\mathsf{cl}(\lambda)] \bmod 2$. *Let* $c_1, \ldots, c_l \in \{0,1\}^{\mathsf{cl}(\lambda)}$ *be distinct and let* $t \in \{0,1\}$. *Then*

$$\Pr\left[\forall j : H_{\text{ip}}(v,c_j) \neq t\right] \leq \frac{2}{l} \tag{3.1}$$

where the probability is over a random choice of V from $\{0,1\}^{\mathsf{cl}(\lambda)}$.

Proof:[Lemma 3.2.3] Define H: $\{0,1\}^{\mathsf{cl}(\lambda)+1} \times \{0,1\}^{\mathsf{cl}(\lambda)} \to \{0,1\}$ by

$$H(s,c) = H_{\text{ip}}(s[1]\ldots s[\mathsf{cl}(\lambda)],c) + s[\mathsf{cl}(\lambda)+1] \bmod 2\ .$$

This family of functions is pairwise independent. Let G be the set of all $s \in \{0,1\}^{\mathsf{cl}(\lambda)+1}$ 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 $s[\mathsf{cl}(\lambda)+1] = b$. Let $\varepsilon = 1/\ell(\lambda)$. Lemma 3.2.2 says that $|G| \geq (1-\varepsilon)2^{\mathsf{cl}(\lambda)+1}$. But $G = G_0 \cup G_1$ and G_0, G_1 are disjoint so

$$|G_0| = |G| - |G_1| \ \geq\ |G| - 2^{\mathsf{cl}(\lambda)} \ \geq\ (1-\varepsilon)2^{\mathsf{cl}(\lambda)+1} - 2^{\mathsf{cl}(\lambda)} \ =\ (1-2\varepsilon)2^{\mathsf{cl}(\lambda)}\ .$$

We note that the probability on the left of Equation (3.1) equals $1 - |G_0|/2^{\mathsf{cl}(\lambda)}$. This concludes the proof. ▮

With this in hand, one can substitute H by H_{ip} in the proof of Lemma 3.2.1. 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.

3.2.1 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 3.2.4 Let $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be an encryption scheme. Then for any $\mathtt{nt} \in \{\mathtt{u}, \mathtt{r}\}$ there is an efficient adversary $A \in \mathbf{A}[\mathtt{ki}, \mathtt{kd}]$ such that

$$\mathsf{Adv}^{\mathtt{ae\text{-}nt}}_{\mathsf{SE},A}(1^\lambda) \geq 1/2.$$

Proof:[Proposition 3.2.4] Let $\kappa : \mathbb{N} \to \mathbb{N}$ be the keylength of 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 $\rho(\lambda)$-bit strings:

Adversary $B(1^\lambda)$	
For $i = 1, \ldots, \kappa(\lambda)$ do	Function $\mathsf{bit}_i(k)$
$\quad (n_i, c_i) \leftarrow\!\!\$\ \textsc{Enc}(1, i, \mathsf{bit}_i, \phi_0)$; $v_i \leftarrow \textsc{Dec}(1, n_i, \phi_0, c_i)$	Return $k[i]$
$\quad$ If $v_i = \perp$ then $k'[i] \leftarrow 0$ else $k'[i] \leftarrow 1$	
Return k'	

Here ϕ_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 $\textsc{Enc}$ query is $k[i]$. The header computed and used by the game in response to the i-th $\textsc{Dec}$ query is 0. Thus, $\textsc{Dec}$ 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. ∎

Remarks

The message-deriving functions used by the adversary in the proof of Proposition 3.2.1 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.

3.3 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.

The transform

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

random, h is the header and m is the message:

Algorithm $\overline{\mathsf{E}}(1^\lambda, \ell, r, h, m)$	Algorithm $\overline{\mathsf{D}}(1^\lambda, \ell, r, h, c)$
$k \leftarrow H(r \,\|\, \ell)$; $c \leftarrow \mathsf{E}(1^\lambda, k, h, m)$	$k \leftarrow H(r \,\|\, \ell)$; $m \leftarrow \mathsf{D}(1^\lambda, k, h, c)$
Return c	Return m

The base scheme $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ 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 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 $\mathbf{L}(\lambda)$ returning $\omega(\lambda)$-bit strings under some arbitrary distribution, and design $\overline{\mathsf{SE}}$ to have $\mathbf{L}(\lambda)$ as its key-generation algorithm.

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 3.4) show that for the first the slowdown is 11% for 5KB messaegs and only 1% for 50KB messages.

The BRS scheme [29] 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.

3.3.1 Security of 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 3.3.1 *Let* $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be a base symmetric encryption scheme as above. Let* $\mathbf{L}(\lambda)$ *be a key-generation algorithm with keylength* ω *and let* $\rho : \mathbb{N} \to \mathbb{N}$ *be a function. Let* $\overline{\mathsf{SE}} = \mathsf{RHtE}[\mathsf{SE}, \mathbf{L}(\lambda), r]$ *be the RO model symmetric encryption scheme associated to* $\mathsf{SE}, \mathbf{L}(\lambda), r$ *as above. Let* $A \in \mathbf{A}[\mathtt{kd}, \mathtt{ki}]$ *be an adversary making* $\mathsf{q_e}(\lambda)$ ENC *queries,* $\mathsf{q_d}(\lambda)$ DEC *queries and* $\mathsf{q_h}(\lambda)$ HASH *queries, and let* $\nu(\lambda) \leq 2^{\mathbf{H}_\infty[\mathbf{L}(\lambda)]-1}$ *be the number of keys, meaning the argument of A's* INITIALIZE *query. Then there is an adversary D such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ae\text{-}r}}_{\overline{\mathsf{SE}}}(A) \leq\ & (24\mathsf{q_e}(\lambda)^2 + 2\mathsf{q_d}(\lambda)) \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{SE}}(D) \\ & + \frac{8\nu(\lambda)\mathsf{q_e}(\lambda)\mathsf{q_h}(\lambda) + 2\nu(\lambda)(\nu(\lambda)-1)\mathsf{q_e}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}(\lambda)]}} \\ & + \frac{2\mathsf{q_e}(\lambda)(\mathsf{q_h}(\lambda) + 2\mathsf{q_e}(\lambda)\nu(\lambda))}{2^r}\ . \end{aligned} \tag{3.2}$$

Adversary D makes only one ENC *query and has the same number of* DEC *queries and*

MAIN // $\mathrm{F}_{\alpha,\beta}(\lambda)$
For $j = 1,\ldots,\nu(\lambda)$ do
  $\ell_j \leftarrow\$ \mathbf{L}(\lambda)$; $S_j \leftarrow \emptyset$
Return $(b' = 1)$

proc $\textsc{Dec}(j,r,h,c)$ // $\mathrm{F}_{1,1},\mathrm{F}_{0,1}$
If $(r,h,c) \in s_j$ then return $\perp$
$m \leftarrow \overline{\mathsf{D}}(1^\lambda,\ell_j,r,h,c)$
If $m = \perp$ then $v \leftarrow 0$ else $v \leftarrow 1$
Return v

proc $\textsc{Dec}(j,r,h,c)$ // $\mathrm{F}_{1,0},\mathrm{F}_{0,0}$
If $(r,h,c) \in s_j$ then return $\perp$
Return 0

proc $\textsc{Enc}(j,n,h,\phi)$ // $\mathrm{F}_{1,1},\mathrm{F}_{1,0}$
$r \leftarrow\$ \{0,1\}^{\rho(\lambda)}$
$c \leftarrow\$ \overline{\mathsf{E}}(1^\lambda,\ell_j,r,h,\phi(\ell_1,\ldots,\ell_{\nu(\lambda)}))$
$s_j \leftarrow s_j \cup \{(r,h,c)\}$
Return (r,c)

proc $\textsc{Enc}(j,n,h,\phi)$ // $\mathrm{F}_{0,1},\mathrm{F}_{0,0}$
$\gamma \leftarrow \mathsf{cl}(\lambda,\mathsf{ol}(\phi),|h|)$; $c \leftarrow\$ \{0,1\}^\gamma$
$r \leftarrow\$ \{0,1\}^{\rho(\lambda)}$; $s_j \leftarrow s_j \cup \{(r,h,c)\}$
Return (r,c)

Figure 3.2. Games $\mathrm{F}_{\alpha,\beta}$ for $\alpha,\beta \in \{0,1\}$. Next to each procedure we write the games in which it occurs.

the same time complexity as A. ∎

We have omitted the nt superscript in the advantage of D because SE is nonceless. That only one-time security is required of SE is reflected in the fact that D makes only one ENC query. We remark that the bound in Theorem 3.3.1 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.

Proof overview

As we noted in Section 4.1 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 $\nu(\lambda) = 1$, meaning there is a single target key, denoted ℓ. Also assume A makes no DEC queries. Denote by $\phi_1,\ldots,\phi_{\mathsf{q_e}(\lambda)}$ 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 ϕ_i is answered by encrypting message $\phi_i(\ell)$ 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 $r_1, \ldots, r_{\mathsf{q_e}(\lambda)}$ denote the random nonces chosen by the game. The reduction B cannot answer hash oracle query $r_g \| \ell$ 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 ϕ_i from querying $r_g \| \ell$ 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 ϕ_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 $r_g \| \ell$ 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.

3.3.2 Proof of Theorem 3.3.1

Consider executing A with the games of Fig. 3.2. For simplicity we have not displayed procedure HASH, which implements the RO and is called by $\overline{\mathsf{E}}, \overline{\mathsf{D}}$. We need to bound

$$\begin{aligned} &\Pr[\mathrm{F}^A_{1,1}(\lambda)] - \Pr[\mathrm{F}^A_{0,0}(\lambda)] \\ &\qquad = \Big(\Pr[\mathrm{F}^A_{1,1}(\lambda)] - \Pr[\mathrm{F}^A_{0,1}(\lambda)]\Big) + \Big(\Pr[\mathrm{F}^A_{0,1}(\lambda)] - \Pr[\mathrm{F}^A_{0,0}(\lambda)]\Big) \end{aligned} \tag{3.3}$$

$\underline{\text{MAIN}(1^\lambda)}$ // $\text{G}(\lambda),\text{H}(\lambda)$
000 For $j = 1,\ldots,\nu(\lambda)$ do
001 $\quad \ell_j \leftarrow\$ \mathbf{L}(\lambda)(1^\lambda)$; $S_j \leftarrow \emptyset$
002 $\quad$ For $i = 1,\ldots,\mathsf{q_e}(\lambda)$ do
003 $\quad\quad r_{i,j} \leftarrow\$ \{0,1\}^{\rho(\lambda)}$
004 $i \leftarrow 0$; $b \leftarrow\$ \{0,1\}$; $k \leftarrow\$ \{0,1\}^{\kappa(\lambda)}$
005 $g \leftarrow\$ \{1,\ldots,\mathsf{q_e}(\lambda)\}$
006 $b' \leftarrow\$ A^{\text{ENC},\text{IHASH},\text{DEC}}(\lambda)$
007 Return $(b' = 1)$

$\underline{\text{proc ENC}(j,n,h,\phi)}$ // $\boxed{\text{G}(\lambda)}$, $\text{H}(\lambda)$
101 $i \leftarrow i+1$; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi),|h|)$
102 If $(i \le g)$ then
103 $\quad m \leftarrow \text{EVAL}^{\text{IHASH}}(\phi,\ell_1,\ldots,\ell_\nu(\lambda))$
104 If $i < g$ then
105 $\quad c_i \leftarrow \mathsf{E}(1^\lambda,\text{IHASH}(r_{i,j}\,\|\,\ell_j),h,m)$
106 If $(i = g$ and $b = 1)$ then
107 $\quad$ If $H[r_{i,j}\,\|\,\ell_j]$ then
108 $\quad\quad \mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{k \leftarrow H[r_{i,j}\,\|\,\ell_j]}$
109 $\quad s \leftarrow j$; $c_i \leftarrow \mathsf{E}(1^\lambda,k,h,m)$
110 If $(i = g$ and $b = 0)$ then
111 $\quad s \leftarrow j$; $c_i \leftarrow\$ \{0,1\}^c$
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 $(r_{i,j},c_i)$

$\underline{\text{proc IHASH}(r\,\|\,\ell)}$ // $\boxed{\text{G}(\lambda)}$, $\text{H}(\lambda)$
200 If not $H[r\,\|\,\ell]$ then
201 $\quad H[r\,\|\,\ell] \leftarrow\$ \{0,1\}^{\kappa(\lambda)}$
202 $\quad$ If $(r\,\|\,\ell = r_{g,s}\,\|\,\ell_s)$ then
203 $\quad\quad \mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{H[r\,\|\,\ell] \leftarrow k}$
204 Return $H[r\,\|\,\ell]$

$\underline{\text{proc HASH}(r\,\|\,\ell)}$ // $\boxed{\text{G}}$, H
300 If not $H[r\,\|\,\ell]$ then
301 $\quad H[r\,\|\,\ell] \leftarrow\$ \{0,1\}^{\kappa(\lambda)}$
302 $\quad$ If $(r\,\|\,\ell = r_{g,s}\,\|\,\ell_s)$ then
303 $\quad\quad \mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{H[r\,\|\,\ell] \leftarrow k}$
304 Return $H[r\,\|\,\ell]$

$\underline{\text{proc DEC}(j,r,h,c)}$ // $\text{G}(\lambda),\text{H}(\lambda)$
400 If $(r,h,c) \in S_j$ then return $\perp$
401 If $(r\,\|\,\ell_j = r_{g,s}\,\|\,\ell_s)$ then
402 $\quad m \leftarrow \mathsf{D}(1^\lambda,k,h,c)$
403 Else
404 $\quad k'\text{IHASH}(r\,\|\,\ell_j)$
405 $\quad m \leftarrow \mathsf{D}(1^\lambda,k',h,c)$
406 If $m = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
407 Return V

Figure 3.3. Games G, H for the proof of Theorem 3.3.1. A box around a game next to a procedure means the boxed code of that procedure is included in the game.

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 Fig. 3.3. 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 SE for the g-th

$\underline{\text{MAIN}(1^\lambda)}$ / All
$k \xleftarrow{\$} \{0,1\}^{\kappa(\lambda)}$; $S \leftarrow \emptyset$
$b' \xleftarrow{\$} A^{\text{ENC},\text{DEC},\text{HASH}}(\lambda)$; Return $(b' = 1)$
$\underline{\text{proc DEC}(r,h,c)}$ / $\text{E}_{1,1}(\lambda), \text{E}_{0,1}(\lambda)$
If $(r,h,c) \in S$ then return $\perp$
$m \leftarrow \text{D}(1^\lambda, k, r, h, c)$
If $m = \perp$ then $v \leftarrow 0$ else $v \leftarrow 1$
Return v
$\underline{\text{proc DEC}(r,h,c)}$ / $\text{E}_{1,0}(\lambda), \text{E}_{0,0}(\lambda)$
If $(r,h,c) \in S$ then return $\perp$
Return 0

$\underline{\text{proc ENC}(n,h,m)}$ / $\text{E}_{1,1}(\lambda), \text{E}_{1,0}(\lambda)$
$r \xleftarrow{\$} \{0,1\}^{\rho(\lambda)}$
$c \leftarrow \text{E}(1^\lambda, k, r, h, m)$
$S \leftarrow S_j \cup \{(h,c)\}$; Return (r,c)
$\underline{\text{proc ENC}(n,h,m)}$ / $\text{E}_{0,1}(\lambda), \text{E}_{0,0}(\lambda)$
$c \leftarrow \text{cl}(|m|,|h|)$; $r \xleftarrow{\$} \{0,1\}^{\rho(\lambda)}$
$c \xleftarrow{\$} \{0,1\}^{\text{cl}(\lambda)}$; $S \leftarrow S \cup \{(h,c)\}$
Return (r,c)

Figure 3.4. Games $\text{E}_{\alpha,\beta}$ for $\alpha, \beta \in \{0,1\}$.

encryption with k playing the role of the key underlying the game of the adversary attacking 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 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 ϕ, 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 3.3.3.

Lemma 3.3.2 *There exists an adversary B_1 so that*

$$\frac{1}{\text{q}_\text{e}(\lambda)}(\Pr[\text{F}^A_{1,1}(\lambda)] - \Pr[\text{F}^A_{0,1}(\lambda)]) \\ \leq \Pr[\text{E}^{B_1}_{1,1}(\lambda)] - \Pr[\text{E}^{B_1}_{0,1}(\lambda)] + 2\Pr[\text{BAD}(\text{H}^A(\lambda))].$$

The games $\text{E}_{\alpha,\beta}(\lambda)$ referred to here are in Fig. 3.4. We must now bound the probability that $\text{H}^A(\lambda)$ sets bad. The proof of the following lemma appears in Section 3.3.4.

Lemma 3.3.3 *The probability of* $\mathsf{BAD}(\mathrm{H}^A(\lambda))$ *is at most*

$$\Pr[\mathsf{BAD}(\mathrm{H}^A(\lambda), 303)] + \frac{\mathsf{q_h}(\lambda) + 2\mathsf{q_e}(\lambda)\nu(\lambda)}{2^{\rho(\lambda)}} \ . \tag{3.4}$$

Here $\mathsf{BAD}(\mathrm{H}^A(\lambda), x)$ 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 ϕ 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 $\Pr[\mathsf{BAD}(\mathrm{H}^A(\lambda), 303)]$ that $\mathrm{H}^A(\lambda)$ 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 $r_{g,s} \,\|\, \ell_s$. We might think, accordingly, that $\Pr[\mathsf{BAD}(\mathrm{H}^A(\lambda), 303)]$ is unconditionally bounded by $\mathsf{q_h}(\lambda)/2^{\mathbf{H}_\infty[\mathbf{L}]}$, but this turns out to be wrong. In fact, the argument is necessarily computational, relying on the assumed security of SE. To see this, suppose $\mathsf{E}(1^\lambda, k, m) = m$ for all m (no header). Let A's first ENC query be $1, \phi$ where $\phi(\ell_1, \ldots, \ell_n) = \ell_1$, and suppose $g > 1$. The ENC procedure returns $(r_{1,1}, \mathsf{E}(1^\lambda, \mathrm{IHASH}(r_{1,1} \,\|\, \ell_1), \ell_1)) = (r_{1,1}, \ell_1)$, to A and thus A has ℓ_1. It uses $j = 1$ in all subsequent ENC queries as well (it does not matter what is the corresponding ϕ) 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 $r_{g,1} \,\|\, \ell_1$ since it knows ℓ_1 and thus sets bad whenever $g > 1$, meaning with (high) probability $1 - 1/\mathsf{q_e}(\lambda)$.

Of course, if SE is secure it will not be that $\mathsf{E}(1^\lambda, k, m) = m$ for all m. But what this means is that bounding $\Pr[\mathsf{BAD}(\mathrm{H}^A(\lambda), 303)]$ *must rely on the assumed security of* SE. Towards obtaining this bound, consider game $\mathrm{I}_{g,h}$ of Fig. 3.5, which is parameterized by g, h. It provides random responses to the first h ENC queries, then provides real

$\underline{\text{MAIN}(1^\lambda)}$ // $\mathrm{I}_{g,h}(\lambda)$
000 For $j = 1,\ldots,\nu(\lambda)$ do
001 $\quad \ell_j \leftarrow\$ \mathbf{L}(\lambda)$; $S_j \leftarrow \emptyset$
002 $\quad$ For $i = 1,\ldots,\mathsf{q_e}(\lambda)$ do
003 $\quad\quad r_{i,j} \leftarrow\$ \{0,1\}^{\rho(\lambda)}$
004 $i \leftarrow 0$; $b \leftarrow\$ \{0,1\}$; $k \leftarrow\$ \{0,1\}^{\kappa(\lambda)}$
005 $b' \leftarrow\$ A^{\text{ENC},\text{DEC},\text{IHASH}}(\lambda)$
006 Return $(b' = 1)$

$\underline{\text{proc ENC}(j,n,h,\phi)}$ // $\mathrm{I}_{g,h}(\lambda)$
100 $i \leftarrow i+1$; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi),|h|)$
101 If $(i \le g)$ then
102 $\quad m \leftarrow \text{EVAL}^{\text{IHASH}}(\phi,\ell_1,\ldots,\ell_{\nu(\lambda)})$
103 If $(i \le h)$ then $c_i \leftarrow\$ \{0,1\}^c$
104 If $(h < i < g)$ then
105 $\quad c_i \leftarrow \mathsf{E}(\text{IHASH}(r_{i,j} \,\|\, \ell_j), h, m)$
106 If $(i = g$ and $b = 1)$ then
107 $\quad s \leftarrow j$; $c_i \leftarrow \mathsf{E}(k,h,m)$
108 If $(i = g$ and $b = 0)$ then
109 $\quad s \leftarrow j$; $c_i \leftarrow\$ \{0,1\}^c$
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 $(r_{i,j},c_i)$

$\underline{\text{proc IHASH}(r \,\|\, \ell)}$ // $\mathrm{I}_{g,h}(\lambda)$
200 If not $H[r \,\|\, \ell]$ then
201 $\quad H[r \,\|\, \ell] \leftarrow\$ \{0,1\}^{\kappa(\lambda)}$
202 Return $H[r \,\|\, \ell]$

$\underline{\text{proc HASH}(r \,\|\, \ell)}$ // $\mathrm{I}_{g,h}$
300 If not $H[r \,\|\, \ell]$ then
301 $\quad H[r \,\|\, \ell] \leftarrow\$ \{0,1\}^{\kappa(\lambda)}$
302 $\quad$ If $(r \,\|\, \ell = r_{g,s} \,\|\, \ell_s)$ then
303 $\quad\quad \mathsf{bad} \leftarrow \mathsf{true}$
304 Return $H[r \,\|\, \ell]$

$\underline{\text{proc DEC}(j,r,h,c)}$ // $\mathrm{I}_{g,h}(\lambda)$
400 If $(r,h,c) \in S_j$ then return $\perp$
401 If $(r \,\|\, \ell_j = r_{g,s} \,\|\, \ell_s)$ then
402 $\quad m \leftarrow \mathsf{D}(k,h,c)$
403 Else $m \leftarrow \mathsf{D}(\text{IHASH}(r \,\|\, \ell_j), h, c)$
404 If $m = \perp$ then $V \leftarrow 0$ else $V \leftarrow 1$
405 Return V

Figure 3.5. Games $\mathrm{I}_{g,h}$ for $0 \le h \le g-1 \le \mathsf{q_e}(\lambda)-1$.

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

$$\Pr[\mathsf{BAD}(\mathrm{H}^A(\lambda),303)] \;=\; \tfrac{1}{\mathsf{q_e}(\lambda)} \textstyle\sum_{g=1}^{\mathsf{q_e}(\lambda)} \Pr[\mathrm{I}^A_{g,0}(\lambda)] \;. \tag{3.5}$$

Towards bounding this we begin by considering game $\mathrm{I}_{g,g-1}$. 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 $H[\cdot \,\|\, \ell_s]$ we do succeed in unconditionally bounding the probability that bad is set in game $\mathrm{I}_{g,g-1}$ 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 3.3.5.

Lemma 3.3.4 *There is an adversary B_2 so that*

$$\frac{1}{\mathsf{q_e}(\lambda)} \sum_{g=1}^{\mathsf{q_e}(\lambda)} \Pr\left[\mathrm{I}^A_{g,g-1}(\lambda)\right] \leq \frac{1}{2}\Pr[\mathrm{E}^{B_2}_{1,1}(\lambda)] - \frac{1}{2}\Pr[\mathrm{E}^{B_2}_{0,1}(\lambda)] + \frac{4\nu(\lambda)\mathsf{q_h}(\lambda) + \nu(\lambda)(\nu(\lambda)-1)}{2^{\mathbf{H}_\infty[\mathbf{L}]+1}}.$$

The next lemma bridges the gap from $\mathrm{I}_{g,g-1}$ to $\mathrm{I}_{g,0}$. An unusual aspect of the proof of the following, which appears in Section 3.3.6, is that adversary B_3 will need to assign different weights to the different hybrids.

Lemma 3.3.5 *There exists an adversary B_3 so that*

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

From Lemma 3.3.5 and Lemma 3.3.4 we have the following, which bounds the first term of Equation (3.3).

Lemma 3.3.6 *There is an adversary D_1 such that*

$$\begin{aligned}\Pr[\mathrm{F}^A_{1,1}(\lambda)] - \Pr[\mathrm{F}^A_{0,1}(\lambda)] \leq\ & 24\mathsf{q_e}(\lambda)^2 \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{SE}}(D_1) \\ & + \frac{4\nu(\lambda)\mathsf{q_e}(\lambda)\mathsf{q_h}(\lambda) + \nu(\lambda)(\nu(\lambda)-1)\mathsf{q_e}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}} \\ & + \frac{2\mathsf{q_e}(\lambda)(\mathsf{q_h}(\lambda) + 2\mathsf{q_e}(\lambda)\nu(\lambda))}{2^{\rho(\lambda)}}. \qquad (3.6)\end{aligned}$$

$\underline{\text{MAIN}(1^\lambda)}$ // $\mathrm{J}(\lambda), \mathrm{L}(\lambda)$
000 For $j = 1, \ldots, \nu(\lambda)$ do
001 $\quad \ell_j \leftarrow\!\!\$\ \mathbf{L}(\lambda)(1^\lambda)$; $S_j \leftarrow \emptyset$
002 $\quad$ For $i = 1, \ldots, \mathsf{q_e}(\lambda)$ do
003 $\quad\quad r_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\rho(\lambda)}$
004 $g \leftarrow\!\!\$\ \{1, \ldots, \mathsf{q_d}(\lambda)\}$
005 For $d = 1, \ldots, \mathsf{q_d}(\lambda)$ do
005 $\quad k[d] \leftarrow\!\!\$\ \{0,1\}^{\kappa(\lambda)}$
007 $i, d \leftarrow 0$; $b \leftarrow\!\!\$\ \{0,1\}$

$\underline{\text{proc ENC}(j, n, h, \phi)}$ // $\mathrm{J}(\lambda), \mathrm{L}(\lambda)$
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 $(r_{i,j}, c_i)$

$\underline{\text{proc HASH}(r \,\|\, \ell)}$ // $\boxed{\mathrm{J}(\lambda)}, \mathrm{L}(\lambda)$
200 If not $H[r \,\|\, \ell]$ then
201 $\quad H[r \,\|\, \ell] \leftarrow\!\!\$\ \{0,1\}^{\kappa(\lambda)}$
202 $\quad$ If $\text{Ind}[r \,\|\, \ell]$ then
203 $\quad\quad \mathsf{bad} \leftarrow \mathsf{true}$
204 $\quad\quad \boxed{H[r \,\|\, \ell] \leftarrow k[\text{Ind}[r \,\|\, \ell]]}$
205 Return $H[r \,\|\, \ell]$

$\underline{\text{proc DEC}(j, r, h, c))}$ // $\mathrm{J}(\lambda)$
300 If $(r, h, c) \in S_j$ then return $\perp$
301 If not $\text{Ind}[r \,\|\, \ell_j]$ then
302 $\quad d \leftarrow d + 1$; $\text{Ind}[r \,\|\, \ell_j] \leftarrow d$
303 $e \leftarrow \text{Ind}[r \,\|\, \ell_j]$
304 If $H[r \,\|\, \ell_j]$ then
305 $\quad \mathsf{bad} \leftarrow \mathsf{true}$
306 $\quad \boxed{k[e] \leftarrow H[r \,\|\, \ell_j]}$
307 If $(e < g)$ then
308 $\quad m \leftarrow \mathsf{D}(1^\lambda, k[e], h, c)$
309 If $((e = g)$ and $(b = 1))$ then
310 $\quad m \leftarrow \mathsf{D}(1^\lambda, k[e], h, c)$
311 If $((e = g)$ and $(b = 0))$ then
312 $\quad$ Return 0
313 If $(e > g)$ then return 0
314 If $m = \perp$ then $v \leftarrow 0$ else $v \leftarrow 1$
315 Return v

Figure 3.6. Games J, L to bound second term in Equation (3.3).

The proof is in Section 3.3.7. We proceed to bound the second term of Equation (3.3). Game J of Fig. 3.6 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 3.3.8.

Lemma 3.3.7 *There exists an adversary B_4 so that*

$$\frac{1}{\mathsf{q_d}(\lambda)}\left(\Pr[\mathrm{F}^A_{0,1}(\lambda)]-\Pr[\mathrm{F}^A_{0,0}(\lambda)]\right) \le \Pr[\mathrm{E}^{B_4}_{\mathsf{SE},0,1}(\lambda)]-\Pr[\mathrm{E}^{B_4}_{\mathsf{SE},0,0}(\lambda)] + 2\Pr[\mathsf{BAD}(\mathrm{L}^A(\lambda))].$$

We must now bound the probability that L^A sets bad. Assume $\ell_1,\ldots,\ell_{\nu(\lambda)}$ are distinct, which happens except with probability $\nu(\lambda)(\nu(\lambda)-1)/2^{\mathbf{H}_\infty[\mathbf{L}]+1}$. 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 bad done in FINALIZE. At this point $\ell_1,\ldots,\ell_{\nu(\lambda)}$ are not referred to by the oracles and may also be chosen in FINALIZE. The flag bad is set only when a query to HASH involves ℓ_j for some j hence is set with probability at most $\nu(\lambda)\mathsf{q_h}(\lambda)/2^{\mathbf{H}_\infty[\mathbf{L}]}$. We conclude that

$$\begin{aligned}\Pr[\mathsf{BAD}(\mathrm{L}^A(\lambda))] &\le \frac{\nu(\lambda)(\nu(\lambda)-1)}{2^{\mathbf{H}_\infty[\mathbf{L}]+1}} + \sum_{j=1}^{\nu}(\lambda)\frac{\mathsf{q_h}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}-(j-1)} \\ &\le \frac{\nu(\lambda)(\nu(\lambda)-1)}{2^{\mathbf{H}_\infty[\mathbf{L}]+1}} + \frac{2\nu(\lambda)\mathsf{q_h}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}} \\ &= \frac{4\nu(\lambda)\mathsf{q_h}(\lambda)+\nu(\lambda)(\nu(\lambda)-1)}{2^{\mathbf{H}_\infty[\mathbf{L}]+1}}\,. \end{aligned} \tag{3.7}$$

The second inequality above used the assumption, made in the theorem statement, that $\nu(\lambda) \le 2^{\mathbf{H}_\infty[\mathbf{L}]-1}$. From Lemma 3.3.7 and Equation (3.7) we have

$$\begin{aligned}\Pr[\mathrm{F}^A_{0,1}(\lambda)]-\Pr[\mathrm{F}^A_{0,0}(\lambda)] \le{}& \mathsf{q_d}(\lambda)(\Pr[\mathrm{E}^{B_4(\lambda)}_{\mathsf{SE},0,1}]-\Pr[\mathrm{E}^{B_4(\lambda)}_{\mathsf{SE},0,0}]) \\ &+\frac{4\nu(\lambda)\mathsf{q_d}(\lambda)\mathsf{q_h}(\lambda)+\nu(\lambda)(\nu(\lambda)-1)\mathsf{q_d}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}}\,.\end{aligned}$$

Apply Lemma 3.3.8 (Section 3.3.9) to B_4. This yields adversary D_2 such that

$$\Pr[\mathrm{F}^A_{0,1}(\lambda)] - \Pr[\mathrm{F}^A_{0,0}(\lambda)] \leq 2\mathsf{q_d}(\lambda) \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{SE}}(D_2) + \frac{4\nu(\lambda)\mathsf{q_d}(\lambda)\mathsf{q_h}(\lambda) + \nu(\lambda)(\nu(\lambda)-1)\mathsf{q_d}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}} \,. \tag{3.8}$$

Finally let adversary D run D_1 with probability $24\mathsf{q_e}(\lambda)^2/(24\mathsf{q_e}(\lambda)^2 + 2\mathsf{q_d}(\lambda))$ and D_2 otherwise. From equations 3.3, 3.6 and 3.8 we have Equation (3.2), concluding the proof of Theorem 3.3.1.

3.3.3 Proof of Lemma 3.3.2

The proof refers to the games of Fig. 3.2 and Fig. 3.4. We have

$$\begin{aligned} \frac{1}{\mathsf{q_e}(\lambda)}\Big(\Pr[\mathrm{F}^A_{1,1}(\lambda)] - \Pr[\mathrm{F}^A_{0,1}(\lambda)]\Big) &= 2\Pr[\mathrm{G}^A(\lambda)] - 1 && (3.9)\\ &= 2(\Pr[\mathrm{H}^A(\lambda)] + \Pr[\mathrm{G}^A(\lambda)] - \Pr[\mathrm{H}^A(\lambda)]) - 1 \\ &\leq 2\Pr[\mathrm{H}^A(\lambda)] - 1 + 2\Pr[\mathsf{BAD}(\mathrm{H}^A(\lambda))] \,. && (3.10) \end{aligned}$$

Equation (3.10) follows from the Fundamental Lemma of Game Playing [25] because games G, H are identical until bad, meaning differ only in code following the setting of bad to true. To justify Equation (3.9) 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 $j = 1, \ldots, \nu(\lambda)$ do
  $\ell_j \leftarrow\!\!\$\, \mathbf{L}(1^\lambda)$; $S_j \leftarrow \emptyset$
  For $i = 1, \ldots, \mathsf{q_e}(\lambda)$ do $r_{i,j} \leftarrow\!\!\$\, \{0,1\}^{\rho(\lambda)}$
$i \leftarrow 0$; $g \leftarrow\!\!\$\, \{1, \ldots, \mathsf{q_e}(\lambda)\}$

Now B_1 runs A. It replies to ENC query j, d, ϕ of A via

$i \leftarrow i+1$; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi, |d|))$

If $(i \leq g)$ then

$m \leftarrow \text{EVAL}^{\text{IHASH}}(\phi, \ell_1, \ldots, \ell_{v(\lambda)})$

If $i < g$ then $c_i \leftarrow \mathsf{E}(1^\lambda, \text{IHASH}(r_{i,j} \| \ell_j), d, m)$

If $(i = g)$ then $s \leftarrow j$; $c_i \leftarrow\!\!\$\, \text{ENC}(d, m)$

If $i > g$ then $c_i \leftarrow\!\!\$\, \{0,1\}^{\mathsf{cl}(\lambda)}$

$S_j \leftarrow S_j \cup \{(r_{i,j}, d, c_i)\}$

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 $r \| \ell$ via

If not $H[r \| \ell]$ then $H[r \| \ell] \leftarrow\!\!\$\, \{0,1\}^{\kappa(\lambda)}$

Return $H[r \| \ell]$

It replies to DEC query j, r, d, c via

If $(r, d, c) \in S_j$ then return $\perp$

If $(r \| \ell_j = r_{g,s} \| \ell_s)$ then $m \leftarrow \text{DEC}(d, c)$

Else $m \leftarrow \mathsf{D}(1^\lambda, \text{IHASH}(r \| \ell_j), d, c)$

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 $\text{AE}_{\mathsf{SE},1,1}, \text{AE}_{\mathsf{SE},0,1}$ for B_1. Thus

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

3.3.4 Proof of Lemma 3.3.3

No information about $r_{g,s}$ is provided to A until the g-th ENC query is answered, so the probability that bad is set at line 106 is at most $\mathsf{q_h}(\lambda)/2^{\rho(\lambda)}$. Once $r_{g,s}$ is released, however, A *could* in fact specify a ϕ whose evaluation would result in hash query $r_{g,s} \| \kappa$. This difficulty is circumvented by the If statement on line 102, which performs the evaluation of ϕ 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 bad. Thus IHASH sets bad only with the probability that some $r_{i,j}$ equals $r_{g,s}$ which is at most $\mathsf{q_e}(\lambda)\nu(\lambda)/2^{\rho(\lambda)+1}$.

3.3.5 Proof of Lemma 3.3.4

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

$$\Pr[\mathrm{I}^A_{g,g-1}(\lambda) \,|\, b=0] \;\leq\; \frac{2\nu(\lambda)\mathsf{q_h}(\lambda)+\nu(\lambda)(\nu(\lambda)-1)}{2^{\mathbf{H}_\infty[\mathbf{L}]+1}}\,. \tag{3.11}$$

Towards justifying this the first observation is that in $\mathrm{I}_{g,g-1}$ 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 $\mathsf{q_h}(\lambda)/2^l$. This is true if there are no DEC queries. To obtain a bound in the presence of the latter we claim

$$\Pr[\mathrm{I}^A_{g,g-1}(\lambda) \,|\, b=0 \text{ and } \ell_1,\ldots,\ell_{\nu(\lambda)} \text{ are distinct}] \;\leq\; \Pr[\mathrm{K}^A_g(\lambda)] \;\leq\; \frac{2\nu(\lambda)\mathsf{q_h}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}}$$

where game K_g is in Fig. 3.7. Briefly, when $\ell_1,\ldots,\ell_{\nu(\lambda)}$ 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 bad to FINALIZE.

$\underline{\text{MAIN}(1^\lambda)}$ // $\text{K}_g(\lambda)$
000 For $j = 1,\ldots,\nu(\lambda)$ do
001 $\quad S_j \leftarrow \emptyset$
002 $\quad$ For $i = 1,\ldots,\mathsf{q_e}(\lambda)$ do
003 $\quad\quad r_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\rho(\lambda)}$
004 $i \leftarrow 0$; $k \leftarrow\!\!\$\ \{0,1\}^{\kappa(\lambda)}$; $t \leftarrow \emptyset$
005 For $j = 1,\ldots,\nu(\lambda)$ do $\ell_j \leftarrow 0^{\ell(\lambda)}$
006 While $|\{\ell_1,\ldots\ell_\nu(\lambda)\}| < \nu(\lambda)$
007 $\quad$ For $j = 1,\ldots,\nu(\lambda)$ do
008 $\quad \ell_j \leftarrow\!\!\$\ \mathbf{L}(1^\lambda)$
009 If $(T \cap \{\ell_1,\ldots,\ell_{\nu(\lambda)}\} \neq \emptyset)$ then
010 $\quad \mathsf{bad} \leftarrow \mathsf{true}$
011 Return bad

$\underline{\text{proc ENC}(j,n,d,\phi)}$ // $\text{K}_g(\lambda)$
100 $i \leftarrow i+1$; $\gamma \leftarrow \mathsf{cl}(\mathsf{ol}(\phi),|d|)$
101 $c_i \leftarrow\!\!\$\ \{0,1\}^\gamma$; $S_j \leftarrow S_j \cup \{(r_{i,j},d,c_i)\}$
102 If $(i = g)$ then $s \leftarrow j$
103 Return $(r_{i,j},c_i)$

$\underline{\text{proc IHASH}(r \,\|\, j)}$ // $\text{K}_g(\lambda)$
200 If not $H'[r \,\|\, j]$ then
201 $\quad H'[r \,\|\, j] \leftarrow\!\!\$\ \{0,1\}^{\kappa(\lambda)}$
202 Return $H'[r \,\|\, j]$

$\underline{\text{proc HASH}(r \,\|\, \ell)}$ // K_g
300 If not $H[r \,\|\, \ell]$ then
301 $\quad H[r \,\|\, \ell] \leftarrow\!\!\$\ \{0,1\}^{\kappa(\lambda)}$
302 $\quad t \leftarrow t \cup \{\ell\}$
303 Return $H[r \,\|\, \ell]$

$\underline{\text{proc DEC}(j,r,d,c)}$ // $\text{K}_g(\lambda)$
400 If $(r,d,c) \in S_j$ then return $\perp$
401 If $(r \,\|\, j = r_{g,s} \,\|\, s)$ then
402 $\quad m \leftarrow \mathsf{D}(1^\lambda,k,d,c)$
403 Else $m \leftarrow \mathsf{D}(\text{IHASH}(r \,\|\, j),d,c)$
404 If $m = \perp$ then $v \leftarrow 0$ else $v \leftarrow 1$
405 Return v

Figure 3.7. Game K_g for proof of Lemma 3.3.4 where $1 \le g \le \mathsf{q_e}(\lambda)$.

The probability that bad is set is then at most

$$\frac{\mathsf{q_h}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}-1} + \cdots + \frac{\mathsf{q_h}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}-(\nu(\lambda)-1)} \le \frac{2\nu(\lambda)\mathsf{q_h}(\lambda)}{2^{\mathbf{H}_\infty[\mathbf{L}]}},$$

the last because we assumed $\nu(\lambda) \le 2^{\mathbf{H}_\infty[\mathbf{L}]-1}$. Since $\ell_1,\ldots,\ell_{\nu(\lambda)}$ are distinct except with probability less than $\nu(\lambda)(\nu(\lambda)-1)/2^{\mathbf{H}_\infty[\mathbf{L}]}$ we have Equation (3.11).

Below we will construct B_2 so that

$$\Pr[\text{E}_{1,1}^{B_2(1^\lambda)}] - \Pr[\text{E}_{0,1}^{B_2}(1^\lambda)] \ge \tag{3.12}$$

$$\frac{1}{\mathsf{q_e}(\lambda)} \sum_{g=1}^{\mathsf{q_e}(\lambda)} \Big(\Pr[\text{I}_{g,g-1}^A(\lambda) \,|\, b=1] - \Pr[\text{I}_{g,g-1}^A(\lambda) \,|\, b=0] \Big) . \tag{3.13}$$

Assuming this we have

$$\begin{aligned}
&\frac{1}{\mathsf{q_e}(\lambda)} \sum_{g=1}^{\mathsf{q_e}(\lambda)} \Pr\left[\mathrm{I}^{A}_{g,g-1}(\lambda)\right] \\
&= \frac{1}{2\mathsf{q_e}(\lambda)} \sum_{g=1}^{\mathsf{q_e}(\lambda)} \Big(\Pr[\mathrm{I}^{A}_{g,g-1}(\lambda) \,|\, b=1] + \Pr[\mathrm{I}^{A}_{g,g-1}(\lambda) \,|\, b=0]\Big) \\
&= \frac{1}{2\mathsf{q_e}(\lambda)} \sum_{g=1}^{\mathsf{q_e}(\lambda)} \Big(\Pr[\mathrm{I}^{A}_{g,g-1}(\lambda) \,|\, b=1] - \Pr[\mathrm{I}^{A}_{g,g-1}(\lambda) \,|\, b=0] + 2\Pr[\mathrm{I}^{A}_{g,g-1}(\lambda) \,|\, b=0]\Big) \\
&\leq \frac{1}{2}\Pr[\mathrm{E}^{B_2}_{1,1}(1^\lambda)] - \frac{1}{2}\Pr[\mathrm{E}^{B_2}_{0,1}(1^\lambda)] + \frac{1}{\mathsf{q_e}(\lambda)} \sum_{g=1}^{\mathsf{q_e}(\lambda)} \frac{4\nu(\lambda)\mathsf{q_h}(\lambda) + \nu(\lambda)(\nu(\lambda)-1)}{2^{\mathbf{H}_\infty[\mathbf{L}]+1}} \\
&= \frac{1}{2}\Pr[\mathrm{E}^{B_2}_{1,1}(1^\lambda)] - \frac{1}{2}\Pr[\mathrm{E}^{B_2}_{0,1}(1^\lambda)] + \frac{4\nu(\lambda)\mathsf{q_h}(\lambda) + \nu(\lambda)(\nu(\lambda)-1)}{2^{\mathbf{H}_\infty[\mathbf{L}]+1}}
\end{aligned}$$

which proves the lemma.

We now construct B_2 so that Equation (3.13) is true. Adversary B_2 begins with initializations

For $j = 1, \ldots, \nu(\lambda)$ do
  $\ell_j \leftarrow\!\!\$\ \mathbf{L}(1^\lambda)$; $S_j \leftarrow \emptyset$
  For $i = 1, \ldots, \mathsf{q_e}(\lambda)$ do $r_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\rho(\lambda)}$
$i \leftarrow 0$; $g \leftarrow\!\!\$\ \{1, \ldots, \mathsf{q_e}(\lambda)\}$

Now B_2 runs A. It replies to ENC query of A via

$i \leftarrow i+1$; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |d|)$
If $(i \leq g)$ then
  $m \leftarrow \mathrm{EVAL}^{\mathrm{IHASH}}(\phi, \ell_1, \ldots, \ell_{\nu(\lambda)})$
If $(i \leq g-1)$ then $c_i \leftarrow\!\!\$\ \{0,1\}^{\gamma(\lambda)}$
If $(i = g)$ then $s \leftarrow j$; $c_i \leftarrow\!\!\$\ \mathrm{ENC}(d, M)$
If $i > g$ then $c_i \leftarrow\!\!\$\ \{0,1\}^{\gamma(\lambda)}$

$S_j \leftarrow S_j \cup \{(r_{i,j}, d, c_i)\}$

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 Fig. 3.5. It replies to DEC query j, R, d, c via

If $(r, d, c) \in S_j$ then return $\perp$

If $(r \,\|\, \ell_j = r_{g,s} \,\|\, \ell_s)$ then

  $v \leftarrow \text{DEC}(d, c)$

  Return v

Else $m \leftarrow \mathsf{D}(1^\lambda, \text{IHASH}(r \,\|\, \ell_j), d, c)$

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 bad was set and 0 otherwise.

3.3.6 Proof of Lemma 3.3.5

Adversary B_3 will perform a hybrid based on the games of Fig. 3.5 but with the twist that different hybrids are differently weighted. It begins with initializations

For $j = 1, \ldots, \nu(\lambda)$ do

  $\ell_j \xleftarrow{\$} \mathbf{L}(1^\lambda)$; $S_j \leftarrow \emptyset$

  For $i = 1, \ldots, \mathsf{q_e}(\lambda)$ do $r_{i,j} \xleftarrow{\$} \{0,1\}^{\rho(\lambda)}$

$i \leftarrow 0$; $b \xleftarrow{\$} \{0,1\}$; $g \xleftarrow{\$} \{1, \ldots, \mathsf{q_e}(\lambda)\}$; $m \xleftarrow{\$} \{1, \ldots, \mathsf{q_e}(\lambda)\}$

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 $m \geq g$ then B_3 outputs 0 and halts. Else —this happens when $1 \leq m \leq g-1$ hence with probability $(g-1)/\mathsf{q_e}(\lambda)$— it runs A. It replies to ENC query (j,d,ϕ) of A via

$i \leftarrow i+1$; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi),|d|)$

If $(i \leq g)$ then

  $m \leftarrow \text{EVAL}^{\text{IHASH}}(\phi,\ell_1,\ldots,\ell_{v(\lambda)})$

If $(i \leq h-1)$ then $c_i \leftarrow\!\!\$\ \{0,1\}^{\gamma(\lambda)}$

If $(i = h)$ then $c_i \leftarrow\!\!\$\ \text{ENC}(d,m)$

If $(h < i < g)$ then $c_i \leftarrow\!\!\$\ \mathsf{E}(1^\lambda, \text{IHASH}(r_{i,j} \,\|\, \ell_j), d, m)$

If $(i = g$ and $b = 1)$ then $s \leftarrow j$; $c_i \leftarrow\!\!\$\ \mathsf{E}(1^\lambda, k, d, m)$

If $(i = g$ and $b = 0)$ then $s \leftarrow j$; $c_i \leftarrow\!\!\$\ \{0,1\}^{\gamma(\lambda)}$

If $i > g$ then $c_i \leftarrow\!\!\$\ \{0,1\}^{\gamma(\lambda)}$

$S_j \leftarrow S_j \cup \{(r_{i,j}, d, c_i)\}$

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 Fig. 3.5. It replies to DEC query j,R,d,C via

If $(r,d,c) \in S_j$ then return $\perp$

If $(r \,\|\, \ell_j = r_{g,s} \,\|\, \ell_s)$ then

  $v \leftarrow \text{DEC}(d,c)$

  Return v

Else $c \leftarrow \mathsf{D}(1^\lambda, \text{IHASH}(r \,\|\, \ell_j), d, c)$

If $c = \perp$ 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

$$\begin{aligned}\Pr[\mathrm{E}^{B_3}_{1,1}(\lambda)] &= \frac{1}{\mathsf{q_e}(\lambda)} \sum_{g=2}^{\mathsf{q_e}(\lambda)} \frac{1}{g-1} \frac{g-1}{\mathsf{q_e}(\lambda)} \sum_{h=1}^{g-1} \Pr\left[\mathrm{I}^{A}_{g,h-1}(\lambda)\right] \\ &= \frac{1}{\mathsf{q_e}(\lambda)^2} \sum_{g=2}^{\mathsf{q_e}(\lambda)} \sum_{h=1}^{g-1} \Pr\left[\mathrm{I}^{A}_{g,h-1}(\lambda)\right]\end{aligned}$$

and

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

Subtracting we get

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

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.

3.3.7 Proof of Lemma 3.3.6

We have

$$\begin{aligned}
&\frac{1}{\mathsf{q_e}(\lambda)}\sum_{g=1}^{\mathsf{q_e}(\lambda)}\Pr\left[\mathrm{I}^A_{g,0}(\lambda)\right]\\
&\leq\ \mathsf{q_e}(\lambda)\cdot\left(\Pr[\mathrm{E}^{B_3}_{1,1}(\lambda)]-\Pr[\mathrm{E}^{B_3}_{0,1}(\lambda)]\right)+\frac{1}{\mathsf{q_e}(\lambda)}\sum_{g=1}^{\mathsf{q_e}(\lambda)}\Pr\left[\mathrm{I}^A_{g,g-1}(\lambda)\right]\\
&\leq\ \mathsf{q_e}(\lambda)\cdot\left(\Pr[\mathrm{E}^{B_3}_{1,1}(\lambda)]-\Pr[\mathrm{E}^{B_3}_{0,1}(\lambda)]\right)+\frac{1}{2}\left(\Pr[\mathrm{E}^{B_2}_{1,1}(\lambda)]-\Pr[\mathrm{E}^{B_2}_{0,1}(\lambda)]\right)\\
&\quad+\frac{4\nu(\lambda)\mathsf{q_h}(\lambda)+\nu(\lambda)(\nu(\lambda)-1)}{2^{\mathbf{H}_\infty[\mathbf{L}]+1}}\,. \qquad (3.14)
\end{aligned}$$

From Lemma 3.3.2, Equation (3.4), Equation (3.5) and Equation (3.14) we have

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

Let adversary B'_1 pick j at random in $\{1,2,3\}$ and run B_j. Now apply Lemma 3.3.8 (Appendix 3.3.9) to B'_1.

3.3.8 Proof of Lemma 3.3.7

We have

$$\begin{aligned}
\frac{1}{\mathsf{q_d}(\lambda)}\left(\Pr[\mathrm{F}^A_{0,1}(\lambda)]-\Pr[\mathrm{F}^A_{0,0}(\lambda)]\right) &= 2\Pr[\mathrm{J}^A]-1\\
&= 2(\Pr[\mathrm{L}^A(\lambda)]+\Pr[\mathrm{J}^A(\lambda)]-\Pr[\mathrm{L}^A(\lambda)])-1\\
&\leq 2\Pr[\mathrm{L}^A(\lambda)]-1+2\Pr[\mathsf{BAD}(\mathrm{L}^A(\lambda))]\,. \qquad (3.15)
\end{aligned}$$

Equation (3.15) follows from the Fundamental Lemma of Game Playing [25] because games J, L are identical until bad, meaning differ only in code following the setting of bad to true. To justify Equation (3.15) 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 $j = 1, \ldots, \nu(\lambda)$ do
  $\ell_j \leftarrow\!\!\$\ \mathbf{L}(1^\lambda)$; $S_j \leftarrow \emptyset$
  For $i = 1, \ldots, \mathsf{q_e}(\lambda)$ do $r_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\rho(\lambda)}$
$g \leftarrow\!\!\$\ \{1, \ldots, \mathsf{q_d}(\lambda)\}$
For $d = 1, \ldots, \mathsf{q_d}(\lambda)$ do If $(d \neq g)$ then $k[d] \leftarrow\!\!\$\ \mathsf{K}(1^\lambda)$
$i, d \leftarrow 0$

Now B_4 runs A. It replies to ENC query j, d, ϕ of A via

$i \leftarrow i+1$; $c \leftarrow \mathsf{cl}(\mathsf{ol}(\phi), |d|)$; $c_i \leftarrow\!\!\$\ \{0,1\}^{\gamma(\lambda)}$; $S_j \leftarrow S_j \cup \{(r_{i,j}, d, c_i)\}$
Return $(r_{i,j}, c_i)$

B_4 replies to HASH query $r \,\|\, \ell$ via

If not $H[r \,\|\, \ell]$ then $H[r \,\|\, \ell] \leftarrow\!\!\$\ \{0,1\}^{\kappa(\lambda)}$
Return $H[r \,\|\, \ell]$

It replies to DEC query j, r, d, c via

If $(r, d, c) \in S_j$ then return $\perp$
If not $\mathrm{Ind}[r \,\|\, \ell_j]$ then $d \leftarrow d+1$; $\mathrm{Ind}[r \,\|\, \ell_j] \leftarrow d$

$e \leftarrow \text{Ind}[r \,\|\, \ell_j]$

If $(e < g)$ then $m \leftarrow \mathsf{D}(1^\lambda, k[e], d, c)$

If $(e = g)$ then $v \leftarrow \text{DEC}(d, c)$; 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 $\text{E}_{0,1}, \text{E}_{0,0}$ for B_4. Thus

$$2\Pr[\text{L}^A(\lambda)] - 1 \quad = \quad \Pr[\text{E}_{0,1}^{B_4}(1^\lambda)] - \Pr[\text{E}_{0,0}^{B_4}(\lambda)] \; .$$

3.3.9 Splitting lemma

Lemma 3.3.8 *Let* $\mathsf{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be a symmetric encryption scheme. Let* B_1, B_2 *be adversaries. Then there are adversaries* D_1, D_2 *such that*

$$\Pr[\text{E}_{1,1}^{B_1}(1^\lambda)] - \Pr[\text{E}_{0,1}^{B_1}(1^\lambda)] \quad \leq \quad 2 \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{SE}}(D_1) \tag{3.16}$$

$$\Pr[\text{E}_{0,1}^{B_2}(1^\lambda)] - \Pr[\text{E}_{0,0}^{B_2}(1^\lambda)] \quad \leq \quad 2 \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{SE}}(D_2) \; . \tag{3.17}$$

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

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

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

Table 3.2. Table showing relative slowdown of RHtE with SHA256. The entries are explained in text.

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

Let D_1 pick $d \in \{0,1\}$ at random and run $D_{1,d}$ and Equation (3.16) 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 (3.17) is similar and is omitted. ∎

3.4 Implementation results

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++ [52], a standard cryptography library. The measurements in the table of Fig. 3.2 show 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.

These 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 Fig. 3.2, 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.

Acknowledgements

This chapter is largely a reproduction of the material as it appears in Authenticated and Misuse-resistant Encryption of Key-dependent Data, by Mihir Bellare and Sriram Keelveedhi from Proceedings of the 31st Annual Conference on Advances in Cryptology, CRYPTO 2011.

Chapter 4

Message-Locked Encryption

4.1 Overview

To save space, commercial cloud storage services such as Google Drive [69], Dropbox [56] and bitcasa [28] perform file-level deduplication across all their users. Say a user Alice stores a file M and Bob requests to store the same file M. Observing that M is already stored, the server, instead of storing a second copy of M, simply updates metadata associated to M to indicate that Bob and Alice both stored M. In this way, no file is stored more than once, moving storage costs for a file stored by u users from $\mathcal{O}(u \cdot |M|)$ to $\mathcal{O}(u + |M|)$ where the big-O notation hides implementation-dependent constants.

However, as users we may want our files to be encrypted. We may not want the storage provider to see our data. Even if we did trust the provider, we may legitimately worry about errant employees or the risk of server compromise by an external adversary. When users themselves are corporations outsourcing their data storage, policy or government regulation may mandate encryption.

Message-Locked Encryption (MLE) is an intriguing new primitive in which the key used for encryption and decryption is itself derived from the message. Instances of this primitive have been seeing widespread deployment and application for the purpose of secure deduplication [28, 112, 49, 2, 51, 10, 50, 91, 106, 98, 3, 65, 58]. We provide a

theoretical treatment towards investigating the security of these applications, by providing definitions of privacy and integrity peculiar to this domain. Subsequently, we utilize the theoretical framework to explore practical and theoretical improvements. In the practical side, we analyze existing schemes and new variants, breaking some and justifying others with proofs in the random-oracle-model (ROM) [22]. In the theoretical side we address the challenging question of finding a standard-model MLE scheme, making connections with deterministic public-key encryption [12], correlated-input-secure hash functions [70] and locally-computable extractors [8, 88, 107] to provide schemes exhibiting different trade-offs between assumptions made and the message distributions for which security is proven. From our treatment MLE emerges as a primitive that combines practical impact with theoretical depth and challenges, making it well worthy of further study and a place in the cryptographic pantheon. Below we provide a high level overview of the rest of this chapter, before proceeding with the details.

Commercial storage services need deduplication to keep costs low, and users want their data to be encrypted. Conventional encryption, however, makes deduplication impossible. Say Alice stores not her file M but its encryption c_A under her password p_A. Bob would store c_B, the encryption of M under his password p_B. Two issues arise: (1) how the server is to detect that the data underlying the two ciphertexts is the same, and (2) even if it can so detect, what can it store short of (c_A, c_B) that allows both parties, based on their separate respective passwords, to recover the data from what is stored. Standard IND-CPA encryption means even (1) is not possible. We might use some kind of searchable encryption [105, 35, 12] but it is still not clear how to solve (2). Just storing Alice's ciphertext, for example, does not work because Bob cannot later decrypt it to recover the file, and visa versa.

Douceur et. al. (DABST) [55] proposed a clever solution called convergent encryption (CE). Alice derives a key $k = \mathsf{H}(M)$ from her message M and then encrypts

the message as $c = E(k,M) = E(\mathsf{H}(M),M)$, where H is a cryptographic hash function and E is a block cipher. (They assume the message is one block long.) The ciphertext is given to the server and the user retains k. Since encryption is deterministic, if Bob starts from the same message he would produce the same key and ciphertext. The server can now perform deduplication on the ciphertext c, checking, when it receives c, whether or not it is already stored, and, if the latter, as before, not re-storing but instead updating meta-data to indicate an additional owner. Both Alice and Bob can decrypt c since both have the same key k.

These ideas have been attractive enough to see significant usage, with CE or variants deployed in [112, 28, 49, 91, 106, 98, 3, 65, 58]. It is not however clear what precisely is the underlying security goal and whether deployed schemes achieve it.

We introduce Message-Locked Encryption (MLE) —so named because the message is locked, as it were, under itself— with the goal of providing an encryption primitive that provably enables secure deduplication. As depicted in Fig. 4.2, the key generation algorithm of an MLE scheme K maps a message m to a key k. The encryption algorithm E takes input the key k and a message m and produces a ciphertext c. The decryption algorithm D allows recovery of m from c given the key k. The tagging algorithm T maps the ciphertext c to a tag t used by the server to detect duplicates. (Tag correctness requires that tags corresponding to messages m_1, m_2 are likely to be the same iff m_1, m_2 are the same.) All algorithms may depend on a parameter p but the latter is public and common to all parties including the adversary, and thus is not a key.

Any MLE scheme enables deduplication of ciphertexts. CE is captured by our syntax as the MLE scheme that lets $k = \mathsf{H}(m)$, $c = \mathsf{E}(k,m)$ and tag $t = H(c)$.

MLE is trivially achieved by letting the key k equal the message m. Set $c = t = \varepsilon$ to the empty string and have decryption simply return the key. This degenerate solution is however useless for deduplication since the client stores as k the entire file and no

storage savings result. We rule it out by requiring that keys be shorter than messages, ideally keys are of a fixed, short length.

Privacy

No MLE scheme can achieve semantic-security-style privacy in the spirit of [67, 15]. Indeed, if the target message m is drawn from a space S of size s then an adversary, given an encryption c of m, can recover m in $\mathcal{O}(s)$ trials. For each candidate $m' \in S$ test whether $\mathsf{D}(\mathsf{K}(m'), c) = m'$ and if so return m'. As with deterministic public-key encryption [12], we therefore ask for the best possible privacy, namely semantic security when messages are unpredictable (have high min-entropy). Adapting definitions from [12, 16, 13, 39] we formalize a PRV-CDA notion where encryptions of two unpredictable messages should be indistinguishable. ("cda" stands for "chosen-distribution attack" [13].) We also formalize a stronger PRV$-CDA notion where the encryption of an unpredictable message must be indistinguishable from a random string of the same length (cf. [103]).

These basic notions are for non-adaptive adversaries. The corresponding adaptive versions are PRV-CDA-A and PRV$-CDA-A. We show that PRV-CDA does not imply PRV-CDA-A but, interestingly, that PRV$-CDA does imply PRV$-CDA-A. See the right hand side of Fig. 4.1 for a comprehensive relations summary. In the figure, an arrow X $\rightarrow$ Y means we can construct primitive Y from primitive X. Dark arrows are our results while light arrows indicate trivial or known implications. Thus PRV$-CDA emerges as the preferred target for designs because non-adaptive security is easier to prove yet adaptive security is implied.

Tag consistency

Suppose client Alice has a message m_A and client Bob has a different message m_B. Alice is malicious and uploads not an honest encryption of m_A but a maliciously-

generated ciphertext c_A such that, when Bob tries to upload c_B, the server sees a tag match $\mathrm{T}(c_A) = \mathrm{T}(c_B)$. This does not contradict the correctness requirement that tags are usually equal iff the messages are equal because that holds for honestly-generated ciphertexts. The server thus keeps only c_A, deleting c_B. Yet later, when Bob downloads to get c_A, the decryption is m_A, not m_B, meaning the integrity of his data has been compromised.

This is a serious concern, and not mere speculation, for such "duplicate-faking" attacks have been found on some CE variants [106]. We define tag consistency to rule out these types of integrity violations. Notion TC asks that it be hard to create (m,c) such that $\mathrm{T}(c) = \mathrm{T}(\mathsf{E}(\mathsf{K}(m),m))$ but $\mathsf{D}(\mathsf{K}(m),c)$ is a string different from m. In words, an adversary cannot make an honest client recover an incorrect message, meaning one different from the one it uploaded. Notion STC ("S" for "strong") asks that it additionally be hard to create (m,c) such that $\mathrm{T}(c) = \mathrm{T}(\mathsf{E}(\mathsf{K}(m),m))$ but $\mathsf{D}(\mathsf{K}(M),c) = \perp$, meaning an adversary cannot erase an honest client's message. STC is strictly stronger than TC; we define both because, as we will see, some schemes meet only the weaker, but still meaningful, TC version.

Practical Contributions

The definitional framework outlined above puts us in a position to rigorously assess —a decade after its inception in [55]— the security of convergent encryption (CE). The task is complicated by the presence and deployment of numerous variants of the basic CE idea. We address this by formulating two MLE schemes, that we call CE and $\mathsf{HCE1}$, that represent two major variants of CE and between them capture the prominent existing schemes. They each make use of a RO hash function H and a deterministic symmetric encryption scheme SE. CE with SE set to a blockcipher, for example, is the scheme of [55] and $\mathsf{HCE1}$ with SE as a blockcipher in counter mode with fixed IV is used within the Tahoe FileSystem (TahoeFS) [112].

CE sets $k = \mathsf{H}(m)$, $c = \mathsf{SE}(k,m)$ and tag $t = \mathsf{H}(c)$, while $\mathsf{HCE1}$ sets $k = \mathsf{H}(m)$,

$c = \mathsf{SE}(k,m)||\mathsf{H}(k)$ and $t = \mathsf{H}(k)$. The rationale for HCE1 is to offer better performance for the server who can simply read the tag as the second part of the ciphertext rather than needing to compute it by hashing the possibly long ciphertext. But we observe that HCE1 is vulnerable to duplicate faking attacks, meaning it does not even achieve TC security. We discuss the implications for the security of TahoeFS in Section 4.3.

We ask whether performance gains of the type offered by HCE1 over CE can be obtained without loss in consistency, and offer as answers two new schemes, HCE and RCE. The former is as efficient as HCE1. RCE however is even more efficient, needing just one concerted pass over the data to generate the key, encrypt the message and produce the tag. On the other hand, HCE needs two passes, one pass to generate the key and a second for encryption, while CE needs a third pass for producing the tag. RCE achieves this via a novel use of randomization (all previous schemes were deterministic). Roughly (see Fig. 4.4), encryption picks a fresh random key L and then computes $\mathsf{SE}(L,m)$ and $k = \mathsf{H}(m)$ in the same pass, finally placing an encryption of L under k, together with an appropriate tag, in the ciphertext. We have implemented all three schemes and the results (cf. Appendix 4.7) show that RCE does indeed outperform the other two.

Fig. 4.1 (table, first four rows) summarizes the findings of our security analysis of the four schemes. Here, for each MLE scheme that we construct, we indicate whether it is in the RO or standard model; whether it is deterministic or randomized; and which security properties it is proven to possess. The assumptions for XtCIH, XtDPKE and XtESPKE are, respectively, a CI-H function, a D-PKE scheme and an ES-PKE scheme, while the others assume only a symmetric encryption scheme. Under standard assumptions on the deterministic symmetric encryption scheme SE (one-time real-or-random ciphertext, or ROR, security as well as key-recovery security) and with H a RO, we show that all four MLE schemes meet our strong privacy notion PRV$-CDA. The consistency findings are more involved. As mentioned, HCE1 provides no tag consistency. The good news

is that CE, HCE and RCE all achieve TC security, so that an adversary cannot make a client recover a file different from the one she uploaded. But only CE offers STC security, implying that the reduction in server cost offered by HCE1, HCE and RCE comes at a price, namely loss of STC-security. The conclusion is that designers will need to trade performance for strong tag consistency. Whether this is fundamental or if better schemes exist is an interesting open question.

Theoretical Contributions

Is MLE possible in the standard-model? This emerges as the natural and most basic theoretical question in this domain. Another question is, how does MLE relate to other (existing) primitives? MLE has in common with Deterministic Public-Key Encryption (D-PKE) [12] and Correlated-input-secure Hash Functions (CI-H) [70] a goal of privacy on unpredictable but possibly related inputs, so it is in particular natural to ask about the relation of MLE to these primitives. The two questions are related, for showing that a primitive X implies MLE yields a construction of an MLE scheme based on X. In exploring these questions it is instructive to distinguish between D-MLE (where encryption is deterministic) and R-MLE (where encryption may be randomized). The connections we now discuss are summarized by the picture on the right side of Fig. 4.1:

- D-PKE $\Rightarrow$ D-MLE: We show how to construct an MLE scheme from any D-PKE scheme that is PRIV-secure in the sense of [12]. The first idea that may come to mind is to make public a public key ek for the D-PKE scheme DE and MLE-encrypt m as $\mathsf{DE}(ek, m)$. But this does not make sense because dk is needed to decrypt and the latter is not derived from m. Our XtDPKE ("extract-then-D-PKE") solution, specified and proven in Section 4.4, is quite different and does not exploit the decryptability of DE at all. We apply a strong randomness extractor to m to get the MLE key k and then encrypt m bit-by-bit, the encryption of the i-th bit $m[i]$ being

Scheme	Model	D/R	Privacy		Integrity	
			PRV-CDA	prv$-cda	TC	STC
CE	RO	D	✓	✓	✓	✓
HCE1	RO	D	✓	✓	✗	✗
HCE	RO	D	✓	✓	✓	✗
RCE	RO	R	✓	✓	✓	✗
XtCIH	STD	D	✓	✓	✓	✓
XtDPKE	STD	D	✓	✗	✓	✓
XtESPKE	STD	R	✓	✗	✓	✓
SXE	STD	D	✓	✓	✓	✓

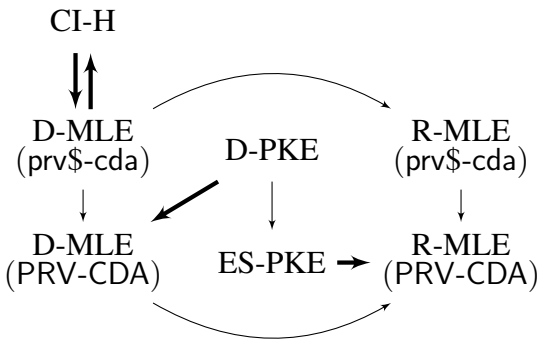


Figure 4.1. Security properties of constructed MLE schemes (**top**) and relations between security notions (**bottom**).

$c[i] = \mathsf{DE}(ek, k\|i\|m[i])$. Decryption, given k, is done by re-encrypting, for each i, both possible values of the i-th message bit and seeing which ciphertext matches $c[i]$. We assume a trusted generation of ek in which nobody retains dk. XtDPKE has PRV-CDA privacy and provides STC (strong) tag consistency.

- CI-H $\Leftrightarrow$ D-MLE: Our XtCIH ("extract-then-CI-Hash") scheme derives a D-MLE scheme from any CI-H hash function [70] by using the latter in place of the D-PKE scheme in the above. XtCIH is PRV$-CDA private while retaining STC consistency. Conversely, any PRV$-CDA D-MLE scheme can be used to construct a CI-H hash function, making the primitives equivalent.

We believe these results are interesting as connections between prominent primitives.

However, they do not, right now, yield MLE schemes under standard assumptions because providing the required D-PKE schemes or CI-H functions under such assumptions is still open and deemed challenging. Indeed, Wichs [110] shows that secure D-PKE schemes or CI-H functions may not be obtained via blackbox reductions from any assumption that may be modeled as a game between an adversary and a challenger. We note that his result applies to D-MLE as well but, as far as we can tell, not to R-MLE. One potential route to MLE with standard assumptions may thus be to exploit randomization but we are unaware of how to do this beyond noting that XtDPKE extends to a R-MLE scheme XtESPKE based on any ES-PKE (Efficiently Searchable PKE) scheme [12], a weaker primitive than D-PKE.

In the D-PKE domain, progress was made by restricting attention to special message distributions. In particular D-PKE under standard assumptions have been achieved for independent messages or block sources [16, 32, 39, 60]. CI-H functions have been built for messages given by polynomials evaluated at the same random point [70]. It is thus natural to ask whether we can obtain MLE under standard assumptions for special message distributions. One might think that this follows from our D-PKE $\Rightarrow$ D-MLE and CI-H $\Rightarrow$ D-MLE constructions and the known results on D-PKE and CI-H, but this is not the case because our constructions do not preserve the message distribution.

The final contribution we mention here is MLE schemes under standard assumptions for certain classes of message distributions. Our SXE (Sample-extract-encrypt) MLE scheme is inspired by locally-computable extractors [8, 88, 107] and the sample-then-extract paradigm [107, 95]. The idea is to put a random subset of the message bits through an extractor to get a key used to encrypt the rest of the bits, and the only assumption made is a standard, ROR-secure symmetric encryption scheme.

Related Work

There are folklore suggestions along the lines of CE predating [55]. See [97].

We have introduced an indistinguishability-from-random notion (PRV$-CDA) for MLE and showed that it implied its adaptive counterpart. This is of broader interest for the parent settings of deterministic and hedged encryption. Here achieving adaptive security has been challenging [13]. We suggest that progress can be made by defining and then targeting indistinguishability-from-random style definitions.

Mironov, Pandey, Reingold and Segev [93] suggest deduplication as a potential application of their incremental deterministic public-key encryption scheme. But this will only work with a single client. It won't allow deduplication across clients, since they would all have to share the secret key.

Recent work showed that client-side deduplication gives rise to side-channel attacks because users are told if another user already uploaded a file [80]. MLE is compatible with either client- or server-side deduplication (the latter prevents such side-channels). We note that one of our new schemes, RCE, gives rise to such a side-channel (see Section 4.3). MLE targets a different class of threats than proofs of ownership [76], which were proposed for deduplication systems in order to mitigate abuse of services for surreptitious content distribution. In independent and concurrent work, Xu, Chang and Zhou [113] consider leakage resilience in the deduplication setting. They provide a randomized construction similar to RCE.

4.2 Message-Locked Encryption

We now look at the syntax and correctness requirements of message-locked encryption.

SYNTAX AND CORRECTNESS. An MLE scheme $\mathsf{MLE} = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$ is a five-tuple of PT algorithms, the last two deterministic — see Fig. 4.2. The parameter generation algorithm is not shown. On input 1^λ the parameter generation algorithm P returns a public parameter p. On input p and a message m, the key-generation algorithm

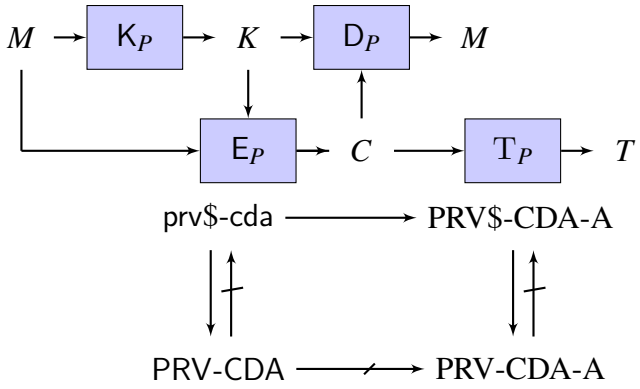


Figure 4.2. Left: Depiction of syntax of MLE scheme $\mathsf{MLE} = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$. Right: Relations between notions of privacy for MLE schemes.

K returns a message-derived key $k \leftarrow\!\!\$\, \mathsf{K}(1^\lambda, p, m)$. On inputs p, k, m the encryption algorithm E returns a ciphertext $c \leftarrow\!\!\$\, \mathsf{E}(p, k, m)$. On inputs p, k and a ciphertext c, the decryption algorithm D returns $\mathsf{D}(p, k, c) \in \{0,1\}^* \cup \{\perp\}$. On inputs p, c the tag generation algorithm returns a tag $T \leftarrow \mathrm{T}(p, c)$. Associated to the scheme is a *message space* M_{MLE} that associates to any $\lambda \in \mathbb{N}$ a set $M_{\mathsf{MLE}}(\lambda) \subseteq \{0,1\}^*$. We require that there is a function Cl such that, for all $\lambda \in \mathbb{N}$, all $p \in [P(1^\lambda)]$ and all $m \in \{0,1\}^*$, any output of $\mathsf{E}(p, \mathsf{K}(1^\lambda, p, m), m)$ has length $\mathsf{Cl}(p, \lambda, |m|)$, meaning the length of a ciphertext depends on nothing about the message other than its length. The *decryption correctness* condition requires that $\mathsf{D}(p, k, c) = m$ for all $\lambda \in \mathbb{N}$, all $p \in [P(1^\lambda)]$, all $m \in M_{\mathsf{MLE}}(\lambda)$, all $k \in [\mathsf{K}(1^\lambda, p, m)]$ and all $c \in [\mathsf{E}(p, k, m)]$. The *tag correctness* condition requires that there is a negligible function δ: $\mathbb{N} \to [0,1]$, called the false negative rate, such that $\Pr[\mathrm{T}(p, c) \neq \mathrm{T}(p, c')] \leq \delta(\lambda)$ for all $\lambda \in \mathbb{N}$, all $p \in [P(1^\lambda)]$ and all $m \in M_{\mathsf{MLE}}(\lambda)$, where the probability is over $c \leftarrow\!\!\$\, \mathsf{E}(p, \mathsf{K}(1^\lambda, p, m), m)$ and $c' \leftarrow\!\!\$\, \mathsf{E}(p, \mathsf{K}(1^\lambda, p, m), m)$. We say that MLE is deterministic if K and E are deterministic. We observe that if MLE is deterministic then it has perfect tag correctness, meaning a false negative rate of 0.

In the application to secure deduplication, the server publishes p and maintains a database that we view as a table Da, initially everywhere $\perp$. In the UPLOAD protocol, the client, having p, m, computes $k \leftarrow\!\!\$\ \mathsf{K}(1^\lambda, p, m)$ and $c \leftarrow\!\!\$\ \mathsf{E}(p, k, m)$. The client stores k securely. (It may do so locally or store k encrypted under its password on the server, but the implementation is not relevant here.) It sends c to the server. The latter computes $T \leftarrow \mathrm{T}(p, c)$. If $\mathsf{Da}[T] = \perp$ then it lets $\mathsf{Da}[T] \leftarrow c$. The server provides the client with a filename or pointer that we may, for simplicity, just view as the tag T. In the DOWNLOAD protocol, the client sends the server a tag T and the server returns $\mathsf{Da}[T]$. If Alice uploads M and Bob later does the same, tag correctness means that their tags will most likely be equal and the server will store a single ciphertext on their behalf. Downloads will return to both this common ciphertext c, and decryption correctness guarantees that both can decrypt c under their respective (although possibly different) keys to recover m.

A trivial construction of an MLE scheme $\mathsf{MLE} = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$ may be obtained by setting the key to the message. In more detail, let $P(1^\lambda) = \varepsilon$; let $\mathsf{K}_\varepsilon(m) = m$; let $\mathsf{E}_\varepsilon(m, m) = \mathrm{T}_\varepsilon(c) = \varepsilon$; let $\mathsf{D}_\varepsilon(m, c) = m$. This will meet the decryption and tag correctness conditions besides meeting the security requirements (privacy and tag consistency) we will formalize below. However, this scheme is of no use for deduplication because the client stores the entire file as the key and no storage savings are gleaned. To avoid this kind of degenerate scheme, we insist that an MLE scheme have keys that are shorter than the message. Formally, there must be a constants $c, d < 1$ such that the function that on input $\lambda \in \mathbb{N}$ returns $\max_{p,m} \Pr[|\mathsf{K}(1^\lambda, p, m)| > d \cdot |m|^c]$ is negligible where the probability is over the choices of K and the maximum is over all $p \in [P(1^\lambda)]$ and all $m \in M_{\mathsf{MLE}}(\lambda)$. Particular schemes we construct or analyze, however, do much better, with the key-length for most of them depending only on the security parameter.

Our formulation of search via tag comparison enables fast search: the server can

use the tag to index directly into a table or perform a logarithmic-time binary search as in [12]. These requirements could be relaxed to define MLE variants where search was allowed linear time (cf. [35]) or search ability was not even provided. MLE does not appear easy to achieve even in the last case.

Privacy

As we noted in Section 4.1, no MLE scheme can provide privacy for predictable messages, meaning ones drawn from a space of polynomial size, in particular ruling out classical semantic security. We now formalize two notions of privacy for unpredictable messages. A *source* is a PT algorithm S that on input 1^λ returns $(\mathbf{m}_0, \ldots, \mathbf{m}_{n-1}, z)$ where $\mathbf{m}_0, \ldots, \mathbf{m}_{n-1}$ are vectors over $\{0,1\}^*$ and $z \in \{0,1\}^*$. Here $n \geq 1$ is a constant called the arity of the source. We will only consider $n \in \{1,2\}$. We require that all the vectors have the same length $\mu(\lambda)$ for some function m called the number of messages of the source. We require that there is a function ℓ, called the message length of the source, such that the string $\mathbf{m}_j[i]$ has length $\ell(\lambda, i)$ for all $i \in [\mu(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$. We require that $\mathbf{m}_j[i_1] \neq \mathbf{m}_j[i_2]$ for all distinct $i_1, i_2 \in [\mu(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$, meaning the entries of each vector are distinct. We refer to z as the auxiliary information. The guessing probability g_S of source S is defined as the function which on input $\lambda \in \mathbb{N}$ returns $\max_{i,j} \mathbf{GP}(\mathbf{m}_j[i] \,|\, z)$ where the probability is over $(\mathbf{m}_0, \ldots, \mathbf{m}_{n-1}, z) \xleftarrow{\$} S(1^\lambda)$ and the maximum is over all $i \in [\mu(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$. We say that S is *unpredictable* if $g_S(\cdot)$ is negligible. Meaning, messages are unpredictable given the auxiliary information. We do *not* require that the components $\mathbf{m}_j[1], \ldots, \mathbf{m}_j[\mu(\lambda)]$ of a vector are independent, just that each, individually, is unpredictable. We refer to $-\log(g_S(\cdot))$ as the min-entropy of the source. We say that S is MLE-valid if $\mathbf{m}_j[i] \in M_{\mathsf{MLE}}(\lambda)$ for all $\lambda \in \mathbb{N}$, all $(\mathbf{m}_0, \ldots, \mathbf{m}_{n-1}, z) \in [S(1^\lambda)]$, all $i \in [\mu(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$.

In the games of Fig. 4.3, "CDA" stands for "Chosen-Distribution Attack," referring to the distribution on messages imposed by the MLE-valid source S, which in

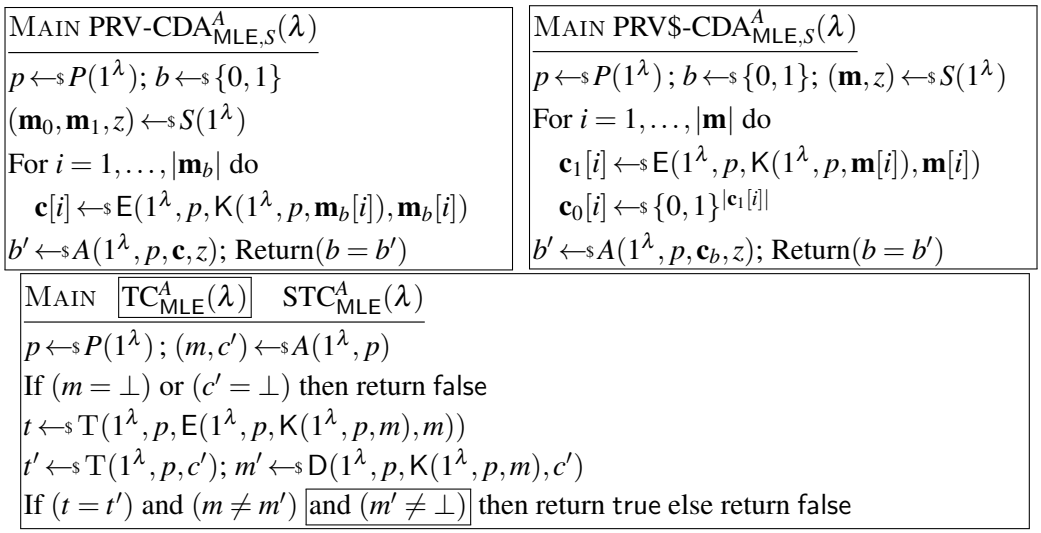


Figure 4.3. Games defining PRV-CDA, PRV$-CDA privacy and TC, STC tag consistency security of MLE scheme $\mathsf{MLE} = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$.

game PRV-CDA has arity 2 and in game PRV$-CDA has arity 1. If A is an adversary we let $\mathsf{Adv}^{\mathsf{prv\text{-}cda}}_{\mathsf{MLE},S,A}(\lambda) = 2 \cdot \Pr[\text{PRV-CDA}^A_{\mathsf{MLE},S}(\lambda)] - 1$ and $\mathsf{Adv}^{\mathsf{prv\$\text{-}cda}}_{\mathsf{MLE},S,A}(\lambda) = 2 \cdot \Pr[\text{PRV\$-CDA}^A_{\mathsf{MLE},S}(\lambda)] - 1$. We say that MLE is PRV-CDA (resp. PRV$-CDA) secure over a class $\overline{S}$ of PT, MLE-valid sources if $\mathsf{Adv}^{\mathsf{prv\text{-}cda}}_{\mathsf{MLE},S,A}(\cdot)$ (resp. $\mathsf{Adv}^{\mathsf{prv\$\text{-}cda}}_{\mathsf{MLE},S,A}(\cdot)$) is negligible for all PT A and all $S \in \overline{S}$. We say that MLE is PRV-CDA (resp. PRV$-CDA) secure if it is PRV-CDA (resp. PRV$-CDA) secure over the class of all PT, unpredictable MLE-valid sources. PRV-CDA asks for indistinguishability of encryptions of two unpredictable messages and is based on formalizations of deterministic [12, 16, 39] and hedged [13] PKE. PRV$-CDA is a new variant, asking for the stronger property that encryptions of unpredictable messages are indistinguishable from random strings, an adaption to this setting of the corresponding notion for symmetric encryption from [103].

The source is not given the parameter p as input, meaning privacy is only assured for messages that do not depend on the parameter. This is analogous to the restriction that messages do not depend on the public key in D-PKE [12], and without this restriction,

privacy is not possible. However, the adversary A does get the parameter.

The notions here are non-adaptive in the sense that the distribution of the next message does not depend on the previous ciphertext. In Appendix 4.5 we give corresponding adaptive definitions PRV-CDA-A and PRV$-CDA-A, and prove the relations summarized in Fig. 4.2. Here, an arrow from A to B means that any A-secure MLE scheme is also B-secure. A barred arrow means there is an A-secure MLE scheme that is not B-secure. The one we highlight is that the non-adaptive PRV$-CDA implies its adaptive counterpart. This is not true for PRV-CDA and makes PRV$-CDA preferable to achieve.

Tag consistency

Consider the games of Fig. 4.3 and let A be an adversary. Game $\mathrm{TC}_{\mathsf{MLE}}$ includes the boxed statement, while $\mathrm{STC}_{\mathsf{MLE}}$ does not. We let $\mathsf{Adv}^{\mathsf{TC}}_{\mathsf{MLE},A}(\lambda) = \Pr[\mathrm{TC}^{A}_{\mathsf{MLE}}(\lambda)]$ and $\mathsf{Adv}^{\mathsf{STC}}_{\mathsf{MLE},A}(\lambda) = \Pr[\mathrm{STC}^{A}_{\mathsf{MLE}}(\lambda)]$. We say that MLE is TC (resp. STC) secure if $\mathsf{Adv}^{\mathsf{TC}}_{\mathsf{MLE},A}(\cdot)$ (resp. $\mathsf{Adv}^{\mathsf{STC}}_{\mathsf{MLE},A}(\cdot)$) is negligible.

Tag consistency (TC) aims to provide security against duplicate faking attacks in which a legitimate message is undetectably replaced by a fake one. In such an attack we imagine the adversary A creating and uploading c'. Later, an honest client, holding m (the formalism allows A to pick m) computes $k \leftarrow\!\!\$\, \mathsf{K}(1^{\lambda}, p, m)$ and uploads $c \leftarrow\!\!\$\, \mathsf{E}(1^{\lambda}, p, k, m)$. The server finds that the tags of c and c' are equal and thus continues to store only c'. Later, the honest client downloads c' and decrypts under k. It expects to recover m, but in a successful duplicate-faking attack it recovers instead some message $m' \neq m$. The integrity of its data has thus been violated. TC security protects against this. Note that TC explicitly excludes an attack in which $m' = \perp$. Thus TC secure schemes may still admit duplicate faking attacks that lead to erasures: a client can detect corruption but no longer be able to recover their message. STC (strong tag consistency) aims to additionally provide security against such erasure attacks. In terms of implications, STC

implies TC but TC does not imply STC.

Duplicate faking attacks are not just a theoretical concern. They were first discussed in [106], yet currently deployed schemes are still vulnerable, as we'll see in the next section. Discussions with practitioners suggest that security against them is viewed as an important requirement in practice.

Given any TC secure scheme, we can prevent all of the attacks above by having a client, upon being informed that her ciphertext is already stored, download it immediately and check that decryption yields her message. If not, she complains. This however is not optimal, being expensive and leading to deduplication side-channels (cf. [80]).

If an MLE scheme is deterministic, letting the tag equal the ciphertext will result in a scheme that is STC secure. (In a strong sense, for the advantage of even a computationally unbounded adversary is 0 in either case. We omit the simple proof.) This provides a relatively easy way to ensure resistance to duplicate faking attacks, but the price paid is that the tag is as long as the ciphertext. CR-hashing the ciphertext (still for a D-MLE scheme) preserves STC, but for efficiency other, less effective options have been employed in practice, as we will see.

In lifting the privacy definitions to the ROM, we do not give the source access to H. This is to simplify our proofs. Our methods and proofs can be extended to handle sources with access to H under an extension of the definition of unpredictability to this setting in which, following [13, 99], the unpredictability of the source is independent of the coins underlying H.

4.3 Practical Contributions

We propose two new practical MLE schemes, and compare these with two in-use MLE schemes. Fig. 4.4 provides pseudocode for all four schemes. All schemes inherit their message space from SE (typically $\{0,1\}^*$), use as parameter generation

K_H, and share a common key generation algorithm. All schemes generate keys as $k \leftarrow \mathsf{H}(1^\lambda, p, m)$. Schemes HCE1 and HCE additionally use the same encryption and tag generation algorithms.

Ingredients

The schemes are built from a one-time symmetric encryption scheme and a hash function family $\mathsf{H} = (\mathsf{K}_\mathsf{H}, \mathsf{H})$. The former is a tuple of algorithms $\mathsf{SE} = (\mathsf{SK}, \mathsf{SE}, \mathsf{SD})$: key generation SK, on input 1^λ, outputs a key k of length $\kappa(\lambda)$; deterministic encryption SE maps a key K and plaintext m to a ciphertext c; and deterministic decryption SD maps a key k and ciphertext c to a message m. We require that $\Pr[\mathsf{SD}(k, \mathsf{SE}(k,m)) = m] = 1$ for all $\lambda \in \mathbb{N}$, all $k \in [\mathsf{SK}(1^\lambda)]$, and all $m \in \{0,1\}^*$. We assume that there exists a function cl_SE such that for all $\lambda \in \mathbb{N}$ and all $m \in \{0,1\}^*$ any output of $\mathsf{SE}(k,m)$ has length $\mathrm{cl}_\mathsf{SE}(\lambda, |m|)$. For simplicity we assume that H and SE are compatible: $\mathsf{H}(k_h, m)$ outputs a message of length $\kappa(\lambda)(\lambda)$ for any $k_h \in [\mathsf{K}_\mathsf{H}(1^\lambda)]$.

We require schemes that provide both key recovery security and one-time real-or-random security [103]. Let game KR_SE, on input 1^λ, run the adversary A. The latter can query at most once to an encryption oracle ENC a message m to which the game replies with an encryption of m under a freshly chosen key k. Adversary A outputs a bit string k' and wins if $k' = k$. Advantage is defined as $\mathsf{Adv}^{\mathrm{kr}}_{\mathsf{SE},A}(\lambda) = \Pr[\mathrm{KR}^A_\mathsf{SE}(1^\lambda)]$ and we say that SE is KR-secure if $\mathsf{Adv}^{\mathrm{kr}}_{\mathsf{SE},A}(\cdot)$ is a negligible function for any PT A.

Let game ROR_SE on input 1^λ, first choose a random bit b, and then run the adversary A. Adversary A can make multiple queries to an encryption oracle ENC, each query a plaintext $m \in \{0,1\}^*$. If $b = 1$, then ENC will first choose a random key $k \leftarrow\!\!\$\, \mathsf{SK}(1^\lambda)$ and return $c \leftarrow \mathsf{SE}(k,m)$. Note that each encryption query chooses a fresh key. If $b = 0$, then ENC will just return a random bit string of length $\mathrm{cl}_\mathsf{SE}(\lambda, |m|)$. To win, the adversary should guess b. Advantage is defined as $\mathsf{Adv}^{\mathrm{ror}}_{\mathsf{SE},A}(\lambda) = 2 \cdot \Pr[\mathrm{ROR}^A_\mathsf{SE}(1^\lambda)] - 1$ and we say that SE is ROR-secure if $\mathsf{Adv}^{\mathrm{ror}}_{\mathsf{SE},A}(\cdot)$ is a negligible function for any PT A.

The four schemes

The first scheme, that we simply call convergent encryption (CE), generalizes the original scheme of DABST [55]. CE encrypts by hashing the message to generate a symmetric key k, which is then used to encrypt the message using SE. Tags are computed by hashing the entire ciphertext. One could alternatively use the ciphertext itself as the tag, but this is typically not practical.

The second scheme, HCE1 (Hash-and-CE 1), is a popular variant of the CE scheme used in a number of systems [112, 51, 58, 50]. Compared to CE, HCE1 computes tags during encryption by hashing the per-message key and including the result in the ciphertext. Tag generation just extracts this embedded tag. This offloads work from the server to the client and reduces the number of passes needed to encrypt and generate a tag from three to two.

HCE1 is vulnerable to attacks that break TC security, as first discussed in [106]. The attack is straightforward: adversary A chooses two messages $m \neq m'$, computes $c \leftarrow \mathsf{SE}(1^\lambda, \mathsf{H}(1^\lambda, p, m), m')$ and $t \leftarrow \mathsf{H}(1^\lambda, p, \mathsf{H}(1^\lambda, p, m))$, and finally outputs $(m, c \,\|\, t)$. This means an adversary, given knowledge of a user's to-be-stored message, can undetectably replace it with any arbitrary message of the adversary's choosing. In TahoeFS's use of HCE1, the client additionally stores a message authentication code (MAC) computed over the message, and checks this MAC during decryption. This means that the TC attack against HCE1 would be detected. TahoeFS is, however, still vulnerable to erasure attacks. We have reported this to the developers, and are discussing possible fixes with them.

We suggest a new scheme, HCE, that modifies HCE1 to directly include a mechanism, that we call guarded decryption, that helps it to achieve TC security. in the ciphertext by recomputing the tag using the just-decrypted message. The decryption routine now additionally checks the tag embedded If the check fails, then $\perp$ is returned. As we argue below, this provably ensures TC security (but not STC).

For performance in practice, the important operation is deriving the ciphertext and tag from the message. This involves generating the key, followed by encryption and tag generation. CE requires three full passes to perform encryption and tag generation, while HCE1 and HCE require two. This is fundamental: deterministic MLE schemes that output bits of ciphertext before processing most of the message will not achieve PRV-CDA security.

The fourth scheme, Randomized Convergent Encryption (RCE), takes advantage of randomization to give a version of HCE that can generate the key, encrypt the message, and produce the tag, all together, in a single pass. RCE accomplishes this by first picking a random symmetric encryption key ℓ and then encrypting the message with ℓ, and deriving the MLE key k in a single pass. Finally it encrypts ℓ using k as a one-time pad, and derives the tag from k. Like HCE it uses guarded decryption. It is easy to verify decryption correctness. Tag correctness follows as the tags are all deterministic.

Privacy

We prove the prv$-cda security of the four schemes when modeling H as a RO. In Appendix 4.6 we give a concrete-security statement together with proof that covers all four schemes. The following theorem ends up an easy corollary.

Theorem 4.3.1 *Let* H *be a RO and let* $\mathsf{SE} = (\mathsf{SK}, \mathsf{SE}, \mathsf{SD})$ *be a one-time symmetric encryption scheme with key length* $\kappa(\lambda)(\cdot)$. *Then if* SE *is both* KR*-secure and* ROR*-secure, the scheme* $\mathsf{XXX}[\mathsf{SE}, \mathrm{H}]$ *for* $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE1}, \mathsf{HCE}, \mathsf{RCE}\}$ *is* prv$-cda*-secure.*

TAG CONSISTENCY. We turn to security in the sense of tag consistency. As discussed in Section 5.2.1, any deterministic scheme is STC-secure when tags are CR-hashes of the ciphertext. So too with CE. For HCE and RCE, a straightforward reduction establishes the following theorem. We omit the details.

Scheme	Encrypt $\mathsf{E}(1^\lambda, p, k, m)$	Tag $\mathrm{T}(1^\lambda, p, c)$	Decrypt $\mathsf{D}(1^\lambda, p, k, c)$
CE[SE, H] Convergent Encryption	$c \leftarrow \mathsf{SE}(k, m)$ Return c	$\mathsf{H}(1^\lambda, p, c)$	$\mathsf{SD}(k, c)$
HCE1[SE, H] Hash and CE w/o tag check	$T \leftarrow \mathsf{H}(1^\lambda, p, k)$ $c \leftarrow \mathsf{SE}(k, m)$ Return $c \,\|\, T$	$c_1 \,\|\, t \leftarrow c$ Return t	$c_1 \,\|\, t \leftarrow c$ Return $\mathsf{SD}(1^\lambda, k, c)$
HCE[SE, H] Hash and CE w/ tag check		$c_1 \,\|\, T \leftarrow c$ $m \leftarrow \mathsf{SD}(1^\lambda, k, c)$ $T' \leftarrow \mathsf{H}(1^\lambda, p, \mathsf{H}(1^\lambda, p, m))$ If $T' \neq T$ then $\perp$ Return$m$	
RCE[SE, H] Randomized Convergent Encryption	$L \leftarrow\!\!\$\, \{0,1\}^{\kappa(\lambda)}$ $T \leftarrow \mathsf{H}(1^\lambda, p, k)$ $c_1 \leftarrow \mathsf{SE}(L, m)$ $c_2 \leftarrow L \oplus k$ Return$c_1 \,\|\, c_2 \,\|\, T$	$c_1 \,\|\, c_2 \,\|\, t \leftarrow c$ Return$t$	Parse c as c_1, c_2, T $L \leftarrow c_2 \oplus k$ $m \leftarrow \mathsf{SD}(L, c_1)$ $T' \leftarrow \mathsf{H}(1^\lambda, p, \mathsf{H}(1^\lambda, p, m))$ If $T' \neq T$ then Return$\perp$ Ret m

Figure 4.4. MLE schemes built using symmetric encryption scheme SE and hash function family H.

Theorem 4.3.2 *Let* $\mathsf{SE} = (\mathsf{SK}, \mathsf{SE}, \mathsf{SD})$ *be a one-time symmetric encryption scheme and let* $\mathsf{H} = (\mathsf{K_H}, \mathsf{H})$ *be a hash function family. If* H *is* CR*-secure then* HCE[SE, H] *and* RCE[SE, H] *are* TC*-secure.*

HCE and RCE are not STC-secure, by the same attack as used against the TC security of HCE1. (The tag check makes it so that decryption outputs $m' = \perp$.) One could in theory achieve STC security using non-interactive zero-knowledge proofs [30], but this would obviate the speedups offered by the schemes compared to CE. We conclude that finding fast, STC-secure schemes with $\mathcal{O}(1)$ tag generation is an interesting open problem, surfaced by our definitions and results above.

Discussion

The above schemes use a hash function family H. In practice, we might use SHA-256 or SHA-3, and key them appropriately by choosing a uniform bit string to

prepend to messages. In Appendix 4.7 we explore various instantiations of the MLE schemes, including ones that implement the hash function by way of a block cipher. This specifically yields MLE schemes entirely built from AES, which provides efficiency benefits due to widespread hardware support for AES (i.e., AES-NI). We also report on performance there.

4.4 Constructions without ROs

We present a paradigm for constructing MLE schemes that we call Extract-Hash-Check. In particular this paradigm yields standard model constructions of MLE-schemes from D-PKE schemes and CI-H hash functions. We then present an MLE scheme based on a weaker assumption, namely any ROR symmetric encryption scheme, for particular classes of sources, based on a method we call Sample-Extract-Encrypt. It also uses an extractor.

4.4.1 Extract-Hash-Check

Overview

It is natural to aim to build MLE from a D-PKE scheme or a CI-H function because the latter primitives already provide privacy on unpredictable messages. However, in attempting to build MLE from these primitives, several problems arise. One is that neither of the base primitives derives the decryption key from the message. Indeed, in both, keys must be generated upfront and independently of the data. A related problem is that it is not clear how an MLE scheme might decrypt. CI-H functions are not required to be efficiently invertible. D-PKE does provide decryption, but it requires the secret key, and it is not clear how this can yield message-based decryption.

Our solution will in fact not use the decryptability of the D-PKE scheme, but rather view the latter as providing a CI-H function keyed by the public key. We apply an

extractor (its seed S will be in the parameters of the MLE scheme) to the message m to get the MLE key k. Given S, m, this operation is deterministic. The scheme encrypts the message bit by bit, creating from $m = m[1] \dots m[|m|]$ the ciphertext $c = c[1] \dots c[|m|]$ in which $c[i]$ is a hash of $k \| \langle i \rangle \| m[i]$. (The key for the hash function is also in the parameters.) To decrypt $c[i]$ given k, hash both $k \| \langle i \rangle \| 1$ and $k \| \langle i \rangle \| 0$ and see which equals $c[i]$. (This is the "check" part.) The proof of privacy relies on the fact that each input to each application of the hash function will have a negligible guessing probability even given the parameters. The reduction will take an MLE source and build a source for the hash function that itself computes k and produces the inputs to the hash function. We now proceed to the details.

Ingredients

The first tool we need is a family of hash functions $\mathsf{H} = (\mathsf{K}_{\mathsf{H}}, \mathsf{H})$. We define privacy in the same manner as for MLE. Namely, game PRV-CDA$_{\mathsf{H}}$ is the same as PRV-CDA$_{\mathsf{MLE}}$ of Fig. 4.3 except that it uses K_{H} and H instead of P and E. Likewise for PRV\$-CDA$_{\mathsf{H}}$. If A is an adversary we let $\mathsf{Adv}^{\text{prv-cda}}_{\mathsf{H},S,A}(\lambda) = 2 \cdot \Pr[\text{PRV-CDA}^{A}_{\mathsf{H},S}(\lambda)] - 1$ and $\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},S,A}(\lambda) = 2 \cdot \Pr[\text{PRV\$-CDA}^{A}_{\mathsf{H},S}(\lambda)] - 1$. We say that H is PRV-CDA (resp. PRV\$-CDA) secure if $\mathsf{Adv}^{\text{prv-cda}}_{\mathsf{H},S,A}(\cdot)$ (resp. $\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},S,A}(\cdot)$) is negligible for all PT S, A such that S is unpredictable. The PRV-CDA formulation follows [12, 16, 70] while the PRV\$-CDA formulation follows [70].

The second tool we need is a family of extractors. This is a family $\text{Ext} = \{\text{Ext}_\lambda\}_{\lambda \in \mathbb{N}}$ where $\text{Ext}_\lambda \colon \{0,1\}^{s(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{k(\lambda)}$ for each $\lambda \in \mathbb{N}$. We refer to s, ℓ, k as the seed, input and output lengths respectively. We require that the map $1^\lambda, S, X \mapsto \text{Ext}_\lambda(S, X)$ be PT computable. For all $\lambda \in \mathbb{N}$ and all random variables $(X, Z), S, k$ we require that $\mathsf{SD}((S, \text{Ext}_\lambda(S, X), Z); (S, k, Z)) \leq \sqrt{2^{k(\lambda)} \cdot \mathbf{GP}(X \,|\, Z)}$ under the following conditions: $(X, Z), S, k$ are independent, S is uniformly distributed over $\{0,1\}^{s(\lambda)}$, and k is uniformly distributed over $\{0,1\}^{k(\lambda)}$. A construction with this guar-

$\underline{P(1^\lambda)}$
$S \xleftarrow{\$} \{0,1\}^{s(\lambda)}$
$k_h \xleftarrow{\$} \mathsf{K}_{\mathsf{H}}(1^\lambda)$
Return $S \| k_h$

$\underline{\mathsf{K}(1^\lambda, S \| k_h, m)}$
$k \leftarrow \mathrm{Ext}_\lambda(S, m)$
Return k

$\underline{\mathsf{E}(1^\lambda, S \| k_h, k, m)}$
For $i = 1$ to $|m|$ do
  $c[i] \leftarrow \mathsf{H}(1^\lambda, k_h, k \| \langle i \rangle \| m[i])$
Return c

$\underline{\mathrm{T}(1^\lambda, S \| k_h, c)}$
Return c

$\underline{\mathsf{D}(1^\lambda, S \| k_h, k, c)}$
For $i = 1$ to $|c|$ do
  If $c[i] = \mathsf{H}(1^\lambda, k_h, k \| \langle i \rangle \| 1)$ then
    $m[i] \leftarrow 1$
  Else
    If $c[i] = \mathsf{H}(1^\lambda, k_h, k \| \langle i \rangle \| 0)$ then
      $m[i] \leftarrow 0$
    Else Return $\perp$
Return m

Figure 4.5. MLE scheme $\mathrm{HC}[\mathsf{H}, \mathrm{Ext}]$ associated to hash family $\mathsf{H} = (\mathsf{K}_{\mathsf{H}}, \mathsf{H})$ and extractor family $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda \in \mathbb{N}}$.

antee may be obtained via the (average-case version of the) Leftover Hash Lemma (LHL) [81, 53].

Extract-Hash-Check construction.

Let $\mathsf{H} = (\mathsf{K}_{\mathsf{H}}, \mathsf{H})$ be a family of hash functions. Let $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of extractors with seed length s, input length ℓ and output length k. Our construction associates to them the MLE scheme $\mathrm{HC}[\mathsf{H}, \mathrm{Ext}] = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$ whose constituent algorithms are defined in Fig. 4.5. The message space of this scheme is defined by $M(\lambda) = \{0,1\}^{\ell(\lambda)}$ for all $\lambda \in \mathbb{N}$. We let $\langle i \rangle$ denote the encoding of $i \in \mathbb{N}$ as a λ-bit string. (We are assuming $\ell(\lambda) \leq 2^\lambda$ for all $\lambda \in \mathbb{N}$.) The ciphertext c is a $\ell(\lambda)$-vector over $\{0,1\}^*$.

This MLE scheme is deterministic, hence provides perfect tag correctness. It satisfies decryption correctness as long as H is injective. (A weaker, computational decryption correctness condition is met if H is not injective but is collision-resistant.) As a consequence of being deterministic and using ciphertexts for tags, it also has unconditional consistency, namely perfect STC security. (We are, for simplicity, using the ciphertext as the tag. For greater efficiency one could CR-hash it. STC would still

hold, but now computationally.) The main task is to prove privacy, which is done by the following, whose proof is in Appendix 4.8. Here, when $\mathsf{MLE} = \mathrm{HC}[\mathsf{H}, \mathrm{Ext}]$, we denote by $\overline{S}_{\mathsf{MLE}}$ the class of all PT, MLE-valid sources S such that $2^{k(\cdot)} \cdot g_S(\cdot)$ is negligible, where k is the output length of Ext.

Theorem 4.4.1 *Let* $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a family of extractors. Let* $\mathsf{H} = (\mathsf{K}_\mathsf{H}, \mathsf{H})$ *be a family of hash functions. Let* $\mathsf{MLE} = \mathrm{HC}[\mathsf{H}, \mathrm{Ext}]$ *be the MLE scheme associated to them via our Extract-Hash-Check construction. Then (1) If* H *is* PRV-CDA*-secure then* MLE *is* PRV-CDA*-secure over* $\overline{S}_{\mathsf{MLE}}$ *, and (2) If* H *is* PRV$-CDA*-secure then* MLE *is* PRV$-CDA*-secure over* $\overline{S}_{\mathsf{MLE}}$ *.*

We can directly instantiate H by a CI-H function as defined in [70]. Given a D-PKE scheme $\mathsf{DPKE} = (\mathsf{DK}, \mathsf{DE}, \mathsf{DD})$, we obtain a hash family $\mathsf{H} = (\mathsf{K}_\mathsf{H}, \mathsf{H})$ as follows. Let $\mathsf{K}_\mathsf{H}(1^\lambda)$ run $\mathsf{DK}(1^\lambda)$ to get (ek, dk) and return ek as the hash key. Let $\mathsf{H}(1^\lambda, ek, m) = \mathsf{DE}(ek, m)$. (We note that we must assume a trusted setup which executes K_H as shown and discards dk, for if the server knows dk it can break the scheme. The parameters must be generated by a third party or via a secure computation protocol so that the server does not learn dk.) If DPKE meets the PRIV-security condition of [12, 16] appropriately extended to handle auxiliary inputs as above, then H will be PRV-CDA-secure. Note that this hash family is injective. Finally we note that the construction can be adapted to turn an efficiently-searchable PKE scheme [12] into an MLE scheme, but since the former may be randomized, the latter may be as well. These constructions account for the schemes called XtCIH, XtDPKE and XtESPKE in the table of Fig. 4.1 and the corresponding implication arrows in the picture.

D-MLE implies CI-H

Finally we justify the claim of Fig. 4.1 that deterministic MLE implies CI-H. (Combined with the above, this makes the primitives equivalent.) Given a deterministic

MLE scheme $\mathsf{MLE} = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$ we define the family of hash functions $\mathsf{H} = (P, \mathsf{H})$ as follows. Algorithm H, given key P and message m, lets $k \leftarrow \mathsf{K}_P(m)$ and $C \leftarrow \mathsf{E}_P(k, m)$, and returns C. (Both K and E are deterministic by assumption.) It is easy to see that if MLE is PRV$-CDA-secure then so is H.

4.5 Relations Between MLE Privacy Notions

We first define the adaptive versions of PRV-CDA and PRV$-CDA, and then go on to show the relations between PRV-CDA, PRV$-CDA and their adaptive versions, as described in Fig. 4.2. We first generalize sources to take inputs.

A *source with input* is a PT algorithm S that on input 1^λ and a string d returns $(\mathbf{m}_0, \ldots, \mathbf{m}_{n-1}, z)$ where $\mathbf{m}_0, \ldots, \mathbf{m}_{n-1}$ are vectors over $\{0,1\}^*$ and $z \in \{0,1\}^*$. As before $n \geq 1$ is called the arity of the source. We require that

- all the vectors have the same length $\mu(\lambda)$, for some function m called the number of messages;
- there is a function ℓ, called the message length of the source, such that the string $\mathbf{m}_j[i]$ has length $\ell(\lambda, i)$ for all $i \in [\mu(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$; and
- $\mathbf{m}_j[i_1] \neq \mathbf{m}_j[i_2]$ for all distinct $i_1, i_2 \in [\mu(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$, meaning the entries of each vector are distinct.

These requirements are as before for sources (without inputs). As before, we refer to z as the auxiliary information.

The guessing probability g_S of source with input S is defined as the function which on input $\lambda \in \mathbb{N}$ returns $\max_{i,j,d} \mathbf{GP}(\mathbf{m}_j[i] \,|\, z)$ where the probability is over $(\mathbf{m}_0, \ldots, \mathbf{m}_{n-1}, z) \leftarrow\!\!\$\, S(1^\lambda, d)$ and the maximum is over all $i \in [\mu(\lambda)]$, all $j \in \{0, \ldots, n-1\}$ and all $d \in \{0,1\}^*$. (The domain for the latter being infinite, the max here is interpreted as a sup and it is an implicit assumption that for a source-with-input to be valid,

this sup must exist.) As compared to a source (without input) the important point here is that we maximize over d as well, so only the coins underlying the execution of S can contribute to the guessing probability. We refer to $-\log(g_S(\cdot))$ as the min-entropy of S. We say that S is *unpredictable* if $g_S(\cdot)$ is negligible. We say that S is MLE-valid if $\mathbf{m}_j[i] \in M_{\mathsf{MLE}}(\lambda)$ for all $\lambda \in \mathbb{N}$, all $(\mathbf{m}_0, \ldots, \mathbf{m}_{n-1}, z) \in [S(1^\lambda, d)]$, all $i \in [\mu(\lambda)]$, all $j \in \{0, \ldots, n-1\}$ and all $d \in \{0,1\}^*$.

Adaptive privacy notions

Let $\mathsf{MLE} = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$ be an MLE scheme and S be MLE-valid. The adaptive chosen distribution attack games PRV-CDA-$\mathrm{A}_{\mathsf{MLE},S}$ and PRV\$-CDA-$\mathrm{A}_{\mathsf{MLE},S}$ for MLE are detailed in Fig. 4.6. For the former S will have arity one and for the latter, arity two. In these games, the adversary A can make multiple, adaptive queries to its ENC oracle, specifying each time an input d for S. It can then ask for the parameter with a reveal query, after which no further ENC queries are allowed. We define advantage as $\mathsf{Adv}^{\text{prv-cda-a}}_{\mathsf{MLE},S,A}(\lambda) = 2 \cdot \Pr[\text{PRV-CDA-}\mathrm{A}^A_{\mathsf{MLE},S}(\lambda)] - 1$ and $\mathsf{Adv}^{\text{prv\$-cda-a}}_{\mathsf{MLE},S,A}(\lambda) = 2 \cdot \Pr[\text{PRV\$-CDA-}\mathrm{A}^A_{\mathsf{MLE},S}(\lambda)] - 1$. We say that MLE is PRV-CDA-A (resp. PRV\$-CDA-A) secure over a class $\overline{S}$ of PT, MLE-valid sources if $\mathsf{Adv}^{\mathsf{prv\text{-}cdaa}}_{\mathsf{MLE},S,A}(\cdot)$ (resp. $\mathsf{Adv}^{\mathsf{prv\$\text{-}cdaa}}_{\mathsf{MLE},S,A}(\cdot)$) is negligible for all PT A and all $S \in \overline{S}$. We say that MLE is PRV-CDA-A (resp. PRV\$-CDA-A) secure if it is PRV-CDA-A (resp. PRV\$-CDA-A) secure over the class of all PT, unpredictable MLE-valid sources.

Relations between the notions

We have introduced four notions: PRV-CDA, PRV\$-CDA, PRV-CDA-A and PRV\$-CDA-A. Fig. 4.2 states the relations between the four notions. The two trivial implications are that security in the adaptive sense implies security in the corresponding non-adaptive sense: PRV-CDA-A $\Rightarrow$ PRV-CDA and PRV\$-CDA-A $\Rightarrow$ PRV\$-CDA. (The double-arrow notation meaning security in the left sense implies security in the right.)

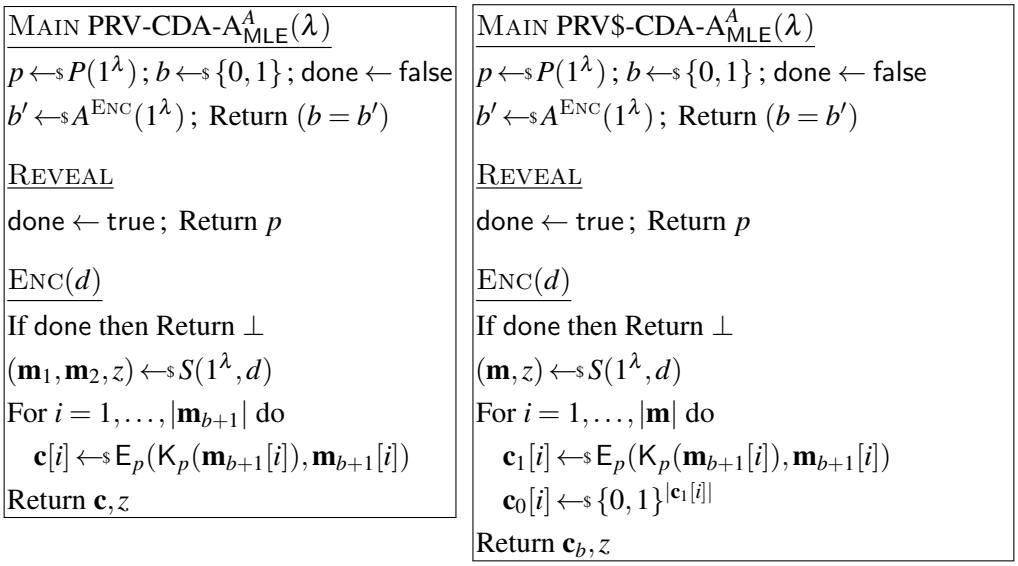


Figure 4.6. The PRV-CDA-A and PRV$-CDA-A games.

We now show through a series of propositions the other implications and separations. Most are simple; we start with the most interesting and useful, that PRV$-CDA $\Rightarrow$ PRV$-CDA-A. In the following, let $\mathsf{MLE} = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$ be an MLE scheme.

Proposition 4.5.1 If MLE is PRV$-CDA secure, then it is PRV$-CDA-A secure.

Proof: We prove the above proposition by showing that for every adversary A and source S, there exists another adversary B and source S' such that for any $\lambda \in \mathbb{N}$

$$\mathsf{Adv}^{\text{prv\$-cda-a}}_{\mathsf{MLE},S,A}(\lambda) \leq q_e(\lambda) \cdot \mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{MLE},S',B}(\lambda) \ ,$$

where function q_e is a bound on the number of encryption queries made by A. Moreover, S' has the same guessing probability as S, and $\mathbf{t_{S'}} = \mathscr{O}(\mathbf{t_S})$, and $\mathbf{t_B} = \mathscr{O}(\mathbf{t_A})$.

We use a hybrid argument. Consider game H with behavior intermediate between PRV$-CDA-A with $b = 0$ and with $b = 1$. In H, a random $g \leftarrow\$ [q_e(\lambda)]$ and $b \leftarrow\$ \{0,1\}$ are chosen. Up to the g-th encryption query, the adversary gets random bits as replies.

The g-th query gets real ciphertexts or random bits depending on b being 1 or 0. Further queries get real ciphertexts as replies. It follows that

$$\Pr[H^A(\lambda)] \geq \frac{1}{q_e(\lambda)} \mathsf{Adv}^{\mathsf{prv\$\text{-}cdaa}}_{\mathsf{MLE},S,A}(\lambda).$$

Consider adversary B which simulates A on game H as follows. Let S' be the algorithm that chooses random coins R and $g \in q_e(\lambda)$ and simulates A up to the g-th encryption query by using bits from R to supply for the randomness required by A. When A makes a query to ENC, then B runs S with fresh random coins from R to get $\mathbf{m}, z$ and replies to A with random bits (from R) for ciphertexts, along with z. When A makes its g-th encryption query d_g, then S' evaluates $S(d)$ on fresh random coins — not from R — and outputs the resulting vector of plaintexts $\mathbf{m}$ and side information z, R, g.

Now, consider the PRV$-CDA game with S' as the source. Importantly, the side information output by S', which includes the random bits R, does not reduce the conditional entropy of the output $\mathbf{m}$ of $S(d_g)$ as the coins used to run $S(d_g)$ to get $\mathbf{m}$ were picked at random, not using R. The PRV$-CDA game then invokes adversary $B(p, \mathbf{c}, z, R, g)$, where $\mathbf{c}$ is either an encryption of $\mathbf{m}$ or a vector of random bits, depending on the choice of the bit b in the CDR game. This bit b actually corresponds to the bit in the H game as well. Adversary B then continues to run A with the same random coins R. A continues to make ENC queries which are now handled by B which replies to them since it knows the parameter p. When A makes a reveal query and B releases the parameter. Finally A finishes execution and outputs a bit b which is echoed by B. We have

$$\Pr[H^A(\lambda)] = \Pr[\text{PRV\$-CDA}^B_{\mathsf{MLE},S'}(\lambda)].$$

Moreover, the sum of the running times of S' and B are both proportional to the running time of A, which includes the running time of S on the inputs provided by A. This proves

the proposition. ▌

Security in the PRV$-CDA sense also implies security in the PRV-CDA sense, with sources of comparable entropy.

Proposition 4.5.2 If MLE is prv$-cda secure, then it is PRV-CDA secure.

Proof: We can prove this proposition by showing that for every adversary A, for every source S, there exists another adversary B, and another source S' with $\mathbf{GP}_S(\cdot) = \mathbf{GP}_{S'}(\cdot)$ such that

$$\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{MLE},S',B}(\lambda) \geq \frac{1}{2}\mathsf{Adv}^{\text{PRV-CDA}}_{\mathsf{MLE},S,A}(\lambda),$$

for all $\lambda \in \mathbb{N}$ and $\mathbf{t_B} = \mathscr{O}(\mathbf{t_A})$, and $\mathbf{t_{S'}} = \mathscr{O}(\mathbf{t_S})$, from which the proposition follows immediately. Towards that, consider source S' which picks a random bit b, runs S to get $\mathbf{m}_0, \mathbf{m}_1, z$, and returns $\mathbf{m}_b, (z, b)$. Clearly, S' has the same guessing probability as S and the running time of S' is proportional to S, depending on implementation constants. Adversary B runs A with z to get b', and returns 1 if $b = b'$ and 0 otherwise. The relation between the advantages follows. ▌

A similar argument can be used to show the adaptive equivalent, PRV$-CDA-A $\Rightarrow$ PRV-CDA-A. The following proposition shows that the converse is not true: PRV-CDA $\not\Rightarrow$ PRV$-CDA.

Proposition 4.5.3 There exists an MLE scheme which is PRV-CDA secure but not PRV$-CDA secure.

Proof: Assume there exists an PRV-CDA secure $\mathsf{MLE} = (P, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathrm{T})$. Then, on top of MLE, we build another scheme $\mathsf{MLE}' = (P, \mathsf{K}, \mathsf{E}', \mathsf{D}', \mathrm{T})$ such that (1) for every adversary A and source S there exists another adversary B running in comparable time with $\mathsf{Adv}^{\text{prv-cda}}_{\mathsf{MLE},S,B}(\lambda) = \mathsf{Adv}^{\text{prv-cda}}_{\mathsf{MLE}',S,A}(\lambda)$, for all $\lambda \in \mathbb{N}$; and (2) there exists an efficient

adversary C such that $\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{MLE}',S,C}(\lambda) \geq \frac{1}{2}$ for all $\lambda \in \mathbb{N}$, for all sources S. Let MLE' be such that E' runs E on its inputs and appends a 0-bit to the resulting ciphertext, and

D' chops off the trailing bit of its ciphertext input and returns the result of D on the result. Given adversary A, adversary B simply appends a 0-bit to its input ciphertexts and runs A. Adversary C outputs 1 if the last bit of the first component of its input ciphertext is 0 and it outputs 0 otherwise. ▌

The adaptive analogue PRV-CDA-A $\not\Rightarrow$ PRV\$-CDA-A follows from a similar proposition and proof as above. The following proposition shows that PRV-CDA $\not\Rightarrow$ PRV-CDA-A as one-time security does not imply adaptive security in the PRV-CDA-A sense. The proof follows from results for deterministic encryption [12] and hedged encryption [13] which rule out public-key dependent security.

Proposition 4.5.4 There exists a scheme MLE' which is PRV-CDA secure but not PRV-CDA-A secure.

Proof: We prove the proposition by taking an PRV-CDA-secure MLE scheme MLE, and modifying it so that it is still PRV-CDA secure, but no longer PRV-CDA-A secure. Specifically, we show that for every adversary A and source S there exists another adversary B running in comparable time with

$$\mathsf{Adv}^{\text{PRV-CDA}}_{\mathsf{MLE},S,B}(\lambda) = \mathsf{Adv}^{\text{PRV-CDA}}_{\mathsf{MLE}',S,A}(\lambda),$$

for all $\lambda \in \mathbb{N}$ and there exists an efficient adversary C and source S' with $\mathbf{GP}_{S'}(\cdot) \leq 2\mathbf{GP}_S(\cdot)$ such that $\mathsf{Adv}^{\text{prv-cdaa}}_{\mathsf{MLE}',S',C}(\lambda) \geq 1/2$, for all $\lambda \in \mathbb{N}$. Consider the MLE scheme MLE' such that E' runs E on its inputs and appends the parameter p to the ciphertext returned by E. D' chops off the trailing parameter p of its ciphertext input and returns the result of D on the result. Given adversary A, adversary B, when invoked with parameter

p, ciphertext vector $\mathbf{c}$ and side-information z, simply appends the parameter to each element of $\mathbf{c}$ and calls 0-bit to its input ciphertexts and runs A, echoing its output. In the PRV-CDA-A game, adversary C receives the parameter after the first query to ENC, and it can provide this parameter to S', which can now generate parameter-dependent messages. Now, we can show that S' can fix one or more bits of the tag, and thus communicate a single bit to the adversary. We omit the details, noting that this scenario of insecurity when the source has the parameter is similar to others arising in deterministic encryption [12] and hedged encryption [13] which rule out public-key dependent security in these settings. ▌

4.5.1 Sample-Extract-Encrypt

Overview

We now give a construction of an MLE scheme that relies only on a standard and weak assumption, namely a one-time symmetric encryption scheme as defined in Section 4.3, which can be built from any one-way function. The tradeoff is that the scheme only works for a limited class of sources.

Stepping back, if we are to consider special sources, the obvious starting point is uniform and independent messages. Achieving MLE here is easy because we can use part of the message as the key to encrypt the other part. The next obvious target is block sources, where each message is assumed to have negligible guessing probability given the previous ones. D-PKE for such sources was achieved in [32]. We might hope, via the above HC construction, to thus automatically obtain MLE for the same sources, but HC does not preserve the block source restriction because the inputs to the hash function for different bits of the same message are highly correlated.

Our Sample-Extract-Encrypt (SXE) construction builds an MLE scheme for certain classes of block sources where a random subset of the bits of each message

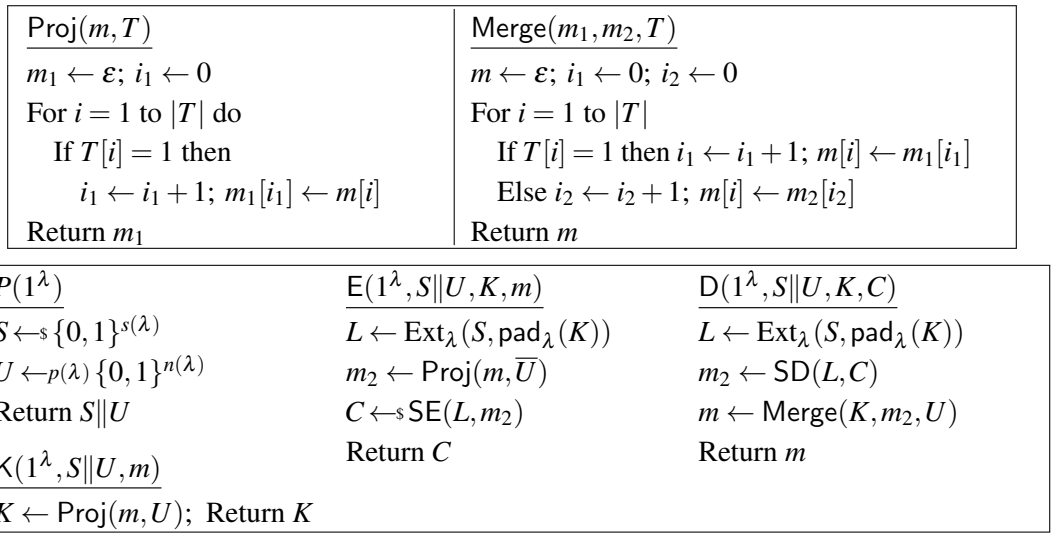


Figure 4.7. Top: The Proj and Merge algorithms. Bottom: MLE scheme $\mathsf{SXE}[\mathsf{SE}, \mathrm{Ext}, p]$ associated to symmetric encryption scheme SE, extractor family Ext and $p\colon \mathbb{N} \to [0,1]$.

remains unpredictable even given the rest of the bits and previous messages. For example, if a message has some subset of uniform bits embedded within it. The scheme then uses a random subset of the message bits as a key, applies an extractor, and then symmetrically encrypts the rest of the message.

Preliminaries

First, some notation. If $T \in \{0,1\}^*$, we let $\mathsf{hw}(T)$ denote the Hamming weight of T and $\overline{T}$ the bitwise complement of T. For $T \in \{0,1\}^{|m|}$, function $\mathsf{Proj}(m,T)$ of Fig. 4.7 returns the $\mathsf{hw}(T)$-length string formed by selecting from m the coordinates i in which $T[i] = 1$. If $q \in [0,1]$ we let $U \leftarrow_q \{0,1\}^n$ mean that we let $U[i] = 1$ with probability q and 0 with probability $1-q$, independently for each $i \in [n]$. Note that the expected Hamming weight of U is then qn. Looking ahead, we will be interested in sources for which one can "gather" sufficient min-entropy by randomly taking a subset of message bits. The fraction q controls the bias with which we select any particular bit.

In this section, all sources have arity 1. Let S be a source with number of messages m and message length ℓ such that $\ell(\lambda, i) = n(\lambda)$ for all $\lambda, i \in \mathbb{N}$, meaning all messages

$\mathbf{m}[1],\ldots,\mathbf{m}[m(\lambda)] \in [S(1^\lambda)]$ have the same length $n(\lambda)$. For p: $\mathbb{N} \to [0,1]$ and $\lambda \in \mathbb{N}$ we let

$$g_{S,p}(\lambda) = \max_i \mathbf{GP}(\mathsf{Proj}(\mathbf{m}[i],U) \,|\, (\mathbf{m}[1],\ldots,\mathbf{m}[i-1],\mathsf{Proj}(\mathbf{m}[i],\overline{U}),Z))$$

where the probability is over $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$ and $(\mathbf{m},Z) \leftarrow S(1^\lambda)$ and the maximum is over all $i \in [m(\lambda)]$. In other words, picking U at random from our $p(\lambda)$-biased distribution, we are measuring the probability of guessing the projection of the i-th message onto U, given the other bits of the message as well as the previous messages. We say that S is p-unpredictable if $g_{S,p}(\cdot)$ is negligible. Note that a blocksource is a source that is 1-unpredictable. Considering smaller values of p thus relaxes the usual blocksource requirement.

Some natural sources are, or can be shown to be, p-unpredictable for small p. One obvious example is uniform messages. Short of that is any source that is sufficiently dense, meaning that a large enough fraction of the message bits have high min-entropy conditioned on all other bits. A concrete example would be sources that embed uniform bits in arbitrary locations within a message. For such a source, one can use a Chernoff bound to show that it is a p-unpredictable source for a reasonable value of p related to the density of the message.

The Sample-Extract-Encrypt construction

Let $\mathsf{SE} = (\mathsf{SK},\mathsf{SE},\mathsf{SD})$ be a (deterministic) one-time symmetric encryption scheme with key length $\kappa(\lambda)(\cdot)$. Let $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of extractors with seed length s, output length k and input length ℓ. (We've assumed that the output length of Ext is equal to the key length of SE.) We let $n(\cdot) = \ell(\cdot) - 1$. Let p: $\mathbb{N} \to [0,1]$. Our construction associates to them the MLE scheme $\mathsf{SXE}[\mathsf{SE},\mathrm{Ext},p] = (P,\mathsf{K},\mathsf{E},\mathsf{D},\mathrm{T})$ whose constituent algorithms are defined in Fig. 4.7. The omitted tag algorithm $\mathrm{T}(1^\lambda,S\|U,C)$

simply returns C. Here we let $\mathsf{pad}_\lambda(K) = K\|1\|0^{n(\lambda)-|K|}$ so that outputs of pad_λ are valid inputs for $\mathrm{Ext}_\lambda(S,\cdot)$. The message space of this scheme is defined by $M(\lambda) = \{0,1\}^{n(\lambda)}$ for all $\lambda \in \mathbb{N}$.

This MLE scheme is deterministic, hence provides perfect tag correctness. It satisfies decryption correctness due to the corrections of SE. Note that the scheme is not strictly non-trivial, since it could be that $\mathsf{hw}(U) = n(\lambda)$. However, it is non-trivial in expectation whenever $p(\lambda) < 1$, since $\mathbf{E}[\mathsf{hw}(U)] = p(\lambda)\cdot n(\lambda)$. As a consequence of being deterministic, and using ciphertext as the tag, it also has perfect STC security. (We are, for simplicity, using the ciphertext as the tag. For greater efficiency one could CR-hash it. STC would still hold, but now computationally.) The following theorem establishes privacy. Here, when $\mathsf{MLE} = \mathsf{SXE}[\mathsf{SE},\mathrm{Ext},p]$, we denote by $\overline{S}_{\mathsf{MLE}}$ the class of all PT, MLE-valid sources S such that $2^{k(\cdot)}\cdot g_{S,p}(\cdot)$ is negligible, where k is the output length of Ext. The proof is in Appendix 4.9.

Theorem 4.5.5 *Let* $\mathsf{SE} = (\mathsf{SK},\mathsf{SE},\mathsf{SD})$ *be a one-time symmetric encryption scheme providing* ROR *security. Let* $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda\in\mathbb{N}}$ *be a family of extractors Let* p: $\mathbb{N}\to[0,1]$. *Let* $\mathsf{MLE} = \mathsf{SXE}[\mathsf{SE},\mathrm{Ext},p]$ *be the MLE scheme associated to them via our Sample-Extract-Encrypt construction. Then* MLE *is* PRV$-CDA*-secure over* $\overline{S}_{\mathsf{MLE},p}$.

The key in this scheme has expected length $p(\cdot)\cdot n(\cdot)$. If we increase p, we get security for a larger class of sources at the cost of a larger key length, so the construction can be seen as trading key size for security.

4.6 Proof of prv$-cda for CE, HCE1, HCE, RCE

We state a concrete security version of Theorem 4.3.1; the latter is a corollary of the concrete version.

Theorem 4.6.1 *Let* SE *be a symmetric encryption scheme with key length* $\kappa(\cdot)$ *and let* H *be a random oracle. Let* $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE1}, \mathsf{HCE}, \mathsf{RCE}\}$ *be constructed using* SE *and* H. *For any source S with min-entropy* $\mu(\cdot)$ *and number of messages* $m(\cdot)$ *and any adversary A making* $q(\cdot)$ *queries to* H, *there exists adversaries* B_1, B_2 *such that for all* $\lambda \in \mathbb{N}$

$$\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{XXX},S,A}(\lambda) \;\leq\; qm \cdot \mathsf{Adv}^{\text{kr}}_{\mathsf{SE},B_1}(\lambda) + 2 \cdot \mathsf{Adv}^{\text{ror}}_{\mathsf{SE},B_2}(\lambda) + \frac{4m^2}{2^{\kappa}} + \frac{qm}{2^{\mu}}$$

where $q = q(\lambda)$, $\mu = \mu(\lambda)$, $m = m(\lambda)$, *and* $\kappa = \kappa(\lambda)$. *The running time of adversary* B_1 *and* B_2 *are each at most* $\mathbf{t_A} + \mathbf{t_S} + cq(\lambda)$, *where c is a small implementation-dependent constant.* B_2 *makes at most m queries to its oracle.* □

Proof: We prove the theorem for the case of RCE; the case of HCE proceeds similarly. The prv$-cda security of HCE implies immediately both the security of HCE1 and the security of CE, the latter by way of a straightforward reduction. In this setting, where S does not have access to H, the parameter is unnecessary, and so we omit it in the rest of the proof.

Let $\text{PRV\$-CDA1}_{\mathsf{RCE},S}$ be the $\text{PRV-CDA}_{\mathsf{RCE},S}$ game with $b = 1$ and let $\text{PRV\$-CDA0}_{\mathsf{RCE},S}$ be the $\text{PRV\$-CDA}_{\mathsf{RCE},S}$ game with $b = 0$. A standard argument yields

$$\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{RCE},S,A}(\lambda) = \Pr\left[\text{PRV\$-CDA1}^{A}_{\mathsf{RCE},S}(\lambda)\right] - \Pr\left[\text{PRV\$-CDA0}^{A}_{\mathsf{RCE},S}(\lambda)\right].$$

We write $m(\lambda)$, $\kappa(\lambda)$, $q(\lambda)$, and $\mu(\lambda)$ in short as m, κ, q, and μ for the rest of the proof. The first game G_1 (Fig. 4.8) is identical to $\text{PRV\$-CDA1}_{\mathsf{RCE},S}$. Note that while the loop in main does not check if $\mathtt{H}[\mathbf{m}[i]]$ was already defined by a previous iteration, but this does not change the implementation of a random oracle H because all sources ensure that $\mathbf{m}[i] \neq \mathbf{m}[j]$ for $i \neq j$. The game sets a flag bad, however, should a value $\mathbf{k}[i]$ collide with

a value $\mathbf{k}[j]$ or $\mathbf{m}[j]$ for $j < i$. Game G_2 removes the boxed statement after bad. The two games are identical-until-bad, and so from the fundamental lemma of game-playing [25],

$$\Pr\left[\mathrm{G}_1^A\right] \leq \Pr\left[\mathrm{G}_2^A\right] + \Pr\left[\mathrm{G}_2^A \text{ sets bad}\right] .$$

Because $\mathbf{k}[i]$ are chosen uniformly and independently of $\mathbf{m}$, we can bound the probability of bad being set via a union bound, giving that $\Pr[\mathrm{G}_2^A \text{ sets bad}] \leq 4m^2/2^\kappa$.

Game G_3 defers updating of the table H with regards to the points $\mathbf{k}[i]$ and $\mathbf{T}[i]$ until they are needed due to a query to H. This change is invisible to the adversary, and we have $\Pr[\mathrm{G}_2^A] = \Pr[\mathrm{G}_3^A]$. Game G_3 sets a flag bad or bad$'$ should such an H query occur. Game G_4 removes the boxed statements of G_3, which occur after bad or bad$'$ is set. Games G_3 and G_4 are identical-until-bad or bad$'$ and so

$$\Pr\left[\mathrm{G}_3^A\right] \leq \Pr\left[\mathrm{G}_4^A\right] + \Pr\left[\mathrm{G}_4 \text{ sets bad}\right] + \Pr\left[\mathrm{G}_4 \text{ sets bad}'\right] .$$

In game G_5, we change the way in which $\mathbf{k}[i]$ is selected. Now, $\mathbf{c}_2[i]$ is sampled uniformly, and $\mathbf{k}[i]$ is set to be $\mathbf{c}_2[i] \oplus \mathbf{l}[i]$. This, in particular, makes the ciphertexts in $\mathbf{c}$ independent of the symmetric keys in $\mathbf{l}$. We have that $\Pr[\mathrm{G}_4^A] = \Pr[\mathrm{G}_5^A]$ as well as $\Pr[\mathrm{G}_4^A \text{ sets bad}] = \Pr[\mathrm{G}_5^A \text{ sets bad}]$ and $\Pr[\mathrm{G}_4^A \text{ sets bad}'] = \Pr[\mathrm{G}_5^A \text{ sets bad}']$.

In game G_6 we defer the computation of $\mathbf{k}[i]$ values and the setting of bad or bad$'$ until after A finishes execution. We have that $\Pr[\mathrm{G}_5^A] = \Pr[\mathrm{G}_6^A]$ as well as $\Pr[\mathrm{G}_5^A \text{ sets bad}] = \Pr[\mathrm{G}_6^A \text{ sets bad}]$ and $\Pr[\mathrm{G}_5^A \text{ sets bad}'] = \Pr[\mathrm{G}_6^A \text{ sets bad}']$.

We now bound in game G_6 the probability that bad$'$ is set, which corresponds to the adversary A querying one of the $\mathbf{k}[i]$ values. The adversary doing so reveals the symmetric key $\mathbf{l}[i]$ associated to that query, by way of $\mathbf{c}_2[i] \oplus \mathbf{k}[i]$. Let B_1 be $\mathrm{KR}_{\mathsf{SE}}$ adversary that works as shown in Fig. 4.8. It guesses a hash query j^* and an encryption

MAIN $\boxed{\mathrm{G}_1}$, G_2

$(\mathbf{m}, z) \leftarrow\!\!\$\, S(1^\lambda)$
For $i = 1, \ldots, m$
  $\mathbf{l}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}_1[i] \leftarrow \mathsf{SE}(\mathbf{l}[i], \mathbf{m}[i])$
  $\mathbf{k}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathtt{H}[\mathbf{m}[i]] \leftarrow \mathbf{k}[i]$
  $\mathbf{c}_2[i] \leftarrow \mathbf{l}[i] \oplus \mathbf{k}[i]$
  $\mathbf{T}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  If $\mathtt{H}[\mathbf{k}[i]] \neq \bot$ then
    bad $\leftarrow$ true
    $\boxed{\mathbf{T}[i] \leftarrow \mathtt{H}[\mathbf{k}[i]]}$
  $\mathtt{H}[\mathbf{k}[i]] \leftarrow \mathbf{T}[i]$
  $\mathbf{c}[i] \leftarrow \mathbf{c}_1[i] \,\|\, \mathbf{c}_2[i] \,\|\, \mathbf{T}[i]$
$b' \leftarrow\!\!\$\, A^{\mathrm{H}}(\mathbf{c}, z)$; Return b'

PROC. $\mathrm{H}(X)$ // G_1, G_2

If $\mathtt{H}[X] \neq \bot$ then Return $\mathtt{H}[X]$
$Y \leftarrow\!\!\$\, \{0,1\}^k$
$\mathtt{H}[X] \leftarrow Y$
Return Y

MAIN $\boxed{\mathrm{G}_3}$, G_4

$(\mathbf{m}, z) \leftarrow\!\!\$\, S(1^\lambda)$
For $i = 1, \ldots, m$
  $\mathbf{l}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}_1[i] \leftarrow \mathsf{SE}(\mathbf{l}[i], \mathbf{m}[i])$
  $\mathbf{k}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}_2[i] \leftarrow \mathbf{l}[i] \oplus \mathbf{k}[i]$
  $\mathbf{T}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}[i] \leftarrow \mathbf{c}_1[i] \,\|\, \mathbf{c}_2[i] \,\|\, \mathbf{T}[i]$
$b' \leftarrow\!\!\$\, A^{\mathrm{H}}(\mathbf{c}, z)$; Return b'

PROC. $\mathrm{H}(X)$

For $i = 1, \ldots, m$
  If $X = \mathbf{m}[i]$ then
    bad $\leftarrow$ true
    $\boxed{\text{Return } \mathbf{k}[i]}$
  If $X = \mathbf{k}[i]$ then
    bad$' \leftarrow$ true
    $\boxed{\text{Return } \mathbf{T}[i]}$
If $\mathtt{H}[X] \neq \bot$ then Return $\mathtt{H}[X]$
$Y \leftarrow\!\!\$\, \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$
Return Y

MAIN G_5

$(\mathbf{m}, z) \leftarrow\!\!\$\, S(1^\lambda)$
For $i = 1, \ldots, m$
  $\mathbf{l}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}_1[i] \leftarrow \mathsf{SE}(\mathbf{l}[i], \mathbf{m}[i])$
  $\mathbf{c}_2[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{k}[i] \leftarrow \mathbf{c}_2[i] \oplus \mathbf{l}[i]$
  $\mathbf{T}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}[i] \leftarrow \mathbf{c}_1[i] \,\|\, \mathbf{c}_2[i] \,\|\, \mathbf{T}[i]$
$b' \leftarrow\!\!\$\, A^{\mathrm{H}}(\mathbf{c}, z)$; Return b'

PROC. $\mathrm{H}(X)$

For $i = 1, \ldots, m$
  If $X = \mathbf{m}[i]$ then
    bad $\leftarrow$ true
  If $X = \mathbf{k}[i]$ then
    bad$' \leftarrow$ true
If $\mathtt{H}[X] \neq \bot$ then
  Return $\mathtt{H}[X]$
$Y \leftarrow\!\!\$\, \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$
Return Y

MAIN G_6

$(\mathbf{m}, z) \leftarrow\!\!\$\, S(1^\lambda)$
For $i = 1, \ldots, m$
  $\mathbf{l}[i] \leftarrow\!\!\$\, \{0,1\}^k$; $\mathbf{c}_2[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}_1[i] \leftarrow \mathsf{SE}(\mathbf{l}[i], \mathbf{m}[i])$; $\mathbf{T}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}[i] \leftarrow \mathbf{c}_1[i] \,\|\, \mathbf{c}_2[i] \,\|\, \mathbf{T}[i]$
$b' \leftarrow\!\!\$\, A^{\mathrm{H}}(\mathbf{c}, z)$
For $i = 1, \ldots, m$
  $\mathbf{k}[i] \leftarrow \mathbf{c}_2[i] \oplus \mathbf{l}[i]$
  If $\mathbf{m}[i] \in \mathscr{X}$ then bad $\leftarrow$ true
  If $\mathbf{k}[i] \in \mathscr{X}$ then bad$' \leftarrow$ true
Ret b'

PROC. $\mathrm{H}(X)$

$\mathscr{X} \leftarrow \mathscr{X} \cup \{X\}$
If $\mathtt{H}[X] \neq \bot$ then Return $\mathtt{H}[X]$
$Y \leftarrow\!\!\$\, \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Return Y

ADVERSARY B_1^{ENC}:

$(\mathbf{m}, z) \leftarrow\!\!\$\, S(1^\lambda)$; $i^* \leftarrow\!\!\$\, [1..m]$; $j^* \leftarrow\!\!\$\, [1..q]$
For $i = 1, \ldots, m$
  If $i = i^*$ then $\mathbf{c}_1[i] \leftarrow \mathrm{ENC}(\mathbf{m}[i])$
  Else
    $\mathbf{k}[i] \leftarrow\!\!\$\, \{0,1\}^k$; $\mathbf{c}_1[i] \leftarrow \mathsf{SE}(\mathbf{k}[i], \mathbf{m}[i])$
  $\mathbf{c}_2[i] \leftarrow\!\!\$\, \{0,1\}^k$; $\mathbf{T}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}[i] \leftarrow \mathbf{c}_1[i] \,\|\, \mathbf{c}_2[i] \,\|\, \mathbf{T}[i]$
$b' \leftarrow\!\!\$\, A^{\mathrm{H}}(\mathbf{c}, z)$
If $|\mathbf{X}[j^*]| = \kappa$ then ret $\mathbf{c}_2[i^*] \oplus \mathbf{X}[j^*]$
Ret $\bot$

PROC. $\mathrm{H}(X)$

$j \leftarrow j + 1$; $\mathbf{X}[j] \leftarrow X$
If $\mathtt{H}[X] \neq \bot$ then Return $\mathtt{H}[X]$
$Y \leftarrow\!\!\$\, \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Return Y

Figure 4.8. Games used in the proof of Theorem 4.3.1.

MAIN G_7 , $\boxed{\mathrm{G}_8}$
$(\mathbf{m}, z) \leftarrow\!\!\$\, S(1^\lambda)$
For $i = 1, \ldots, m$
  $\mathbf{l}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}_1[i] \leftarrow \mathsf{SE}(\mathbf{l}[i], \mathbf{m}[i])$
  $\boxed{\mathbf{c}_1[i] \leftarrow\!\!\$\, \{0,1\}^{\mathsf{clSE}(\lambda, \omega(\lambda, i))}}$
  $\mathbf{c}_2[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{T}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}[i] \leftarrow \mathbf{c}_1[i] \,\|\, \mathbf{c}_2[i] \,\|\, \mathbf{T}[i]$
$b' \leftarrow\!\!\$\, A^{\mathrm{H}}(P, \mathbf{c}, z)$
For $i = 1, \ldots, m$
  If $\mathbf{m}[i] \in \mathscr{X}$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Ret b'
PROC. $\mathrm{H}(X)$
$\mathscr{X} \leftarrow \mathscr{X} \cup \{X\}$
If $\mathtt{H}[X] \neq \perp$ then Return $\mathtt{H}[X]$
$Y \leftarrow\!\!\$\, \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Return Y

ADVERSARY $B^{\mathrm{ENC}}(1^\lambda)$:
$(\mathbf{m}, z) \leftarrow\!\!\$\, S(1^\lambda)$
For $i = 1, \ldots, m$
  $\mathbf{c}_1[i] \leftarrow \mathrm{ENC}(\mathbf{m}[i])$
  $\mathbf{c}_2[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{T}[i] \leftarrow\!\!\$\, \{0,1\}^k$
  $\mathbf{c}[i] \leftarrow \mathbf{c}_1[i] \,\|\, \mathbf{c}_2[i] \,\|\, \mathbf{T}[i]$
$b' \leftarrow\!\!\$\, A^{\mathrm{H}}(\mathbf{c}, z)$
$\mathsf{bad} \leftarrow 0$
For $i = 1, \ldots, m$
  If $\mathbf{m}[i] \in \mathscr{X}$ then $\mathsf{bad} \leftarrow 1$
$c \leftarrow\!\!\$\, \{0,1\}$
If $c = 0$ then Ret b'
If $c = 1$ then Ret bad
PROC. $\mathrm{H}(X)$
$\mathscr{X} \leftarrow \mathscr{X} \cup \{X\}$
If $\mathtt{H}[X] \neq \perp$ then Return $\mathtt{H}[X]$
$Y \leftarrow\!\!\$\, \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Return Y

Figure 4.9. Games used in the proof of Theorem 4.3.1.

---

index i^*. After A finishes executing, it outputs $L = \mathbf{c}_2[i^*] \oplus \mathbf{X}[j^*]$ should $|X^{j^*}| = \kappa$. A hybrid argument gives that

$$\Pr\left[\mathrm{G}_6^A \text{ sets } \mathsf{bad}'\right] \leq qm \cdot \mathsf{Adv}^{\mathrm{kr}}_{\mathsf{SE}, B_1}(\lambda) \,.$$

Game G_7 (Fig. 4.9) is the same as G_6 except that the setting of bad' is dropped, and so $\Pr[\mathrm{G}_6^A] = \Pr[\mathrm{G}_7^A]$ as well as $\Pr[\mathrm{G}_6^A \text{ sets } \mathsf{bad}] = \Pr[\mathrm{G}_7^A \text{ sets } \mathsf{bad}]$ Game G_8 adds adds the boxed statement, replacing the symmetric encryption ciphertext with random bits.

We now upper bound the transition from G_7 to G_8 by reducing to an $\mathrm{ROR}_{\mathsf{SE}}$ adversary B_2. The adversary is shown in Fig. 4.9. It executes exactly G_7^A except that it uses its own ENC oracle to generate $\mathbf{c}_1[i]$ ciphertexts and it computes its output differently,

randomly choosing whether to return b' or bad. (Here bad is set to 0 or 1 as opposed to true or false so B_2 can output the value of the flag as its return value.) We have that

$$\mathsf{Adv}^{\mathrm{ror}}_{\mathsf{SE},B_2}(\lambda) = \Pr\left[\mathrm{ROR}^{B_2}_{\mathsf{SE}}\right] = \frac{1}{2}\left(\Pr[\mathrm{ROR}^{B_2}_{\mathsf{SE}} \,|\, c=1] + \Pr[\mathrm{ROR}^{B_2}_{\mathsf{SE}} \,|\, c=2]\right) . \quad (4.1)$$

We investigate each conditional probability in the sum in turn.

Let $\mathrm{ROR1}_{\mathsf{SE}}$ (resp. $\mathrm{ROR0}_{\mathsf{SE}}$) be the $\mathrm{ROR}_{\mathsf{SE}}$ game but with the challenge bit b set to 1 (resp. 0). Then

$$\begin{aligned}\Pr[\mathrm{ROR}^{B_2}_{\mathsf{SE}} \,|\, c=0] &= \Pr[\mathrm{ROR1}^{B_2}_{\mathsf{SE}} \Rightarrow 1 \,|\, c=0] - \Pr[\mathrm{ROR0}^{B_2}_{\mathsf{SE}} \Rightarrow 1 \,|\, c=0] \\ &= \Pr\left[\mathrm{G}^A_7\right] - \Pr\left[\mathrm{G}^A_8\right]\end{aligned}$$

Likewise we have that

$$\begin{aligned}\Pr[\mathrm{ROR}^{B_2}_{\mathsf{SE}} \,|\, c=1] &= \Pr[\mathrm{ROR1}^{B_2}_{\mathsf{SE}} \Rightarrow 1 \,|\, c=1] - \Pr[\mathrm{ROR0}^{B_2}_{\mathsf{SE}} \Rightarrow 1 \,|\, c=1] \\ &= \Pr\left[\mathrm{G}^A_7 \text{ sets bad}\right] - \Pr\left[\mathrm{G}^A_8 \text{ sets bad}\right]\end{aligned}$$

By rearranging and substituting into Equation (4.1) we get that

$$\Pr\left[\mathrm{G}^A_7\right] + \Pr\left[\mathrm{G}^A_7 \text{ sets bad}\right] = \Pr\left[\mathrm{G}^A_8\right] + \Pr\left[\mathrm{G}^A_8 \text{ sets bad}\right] + 2 \cdot \mathsf{Adv}^{\mathrm{ror}}_{\mathsf{SE},B_2}(\lambda) .$$

In game G_8, the choice of $\mathbf{m}$ is independent of the generation of $\mathbf{c}$. We can therefore apply the min-entropy of S in order to bound the probability that any hash query by A equals $P \,\|\, \mathbf{m}[i]$ for some i. Specifically, a union bound gives that

$$\Pr\left[\mathrm{G}^A_8 \text{ sets bad}\right] \leq \frac{qm}{2^{\mu}} .$$

Moreover, G_8 implements for A exactly the oracles of PRV$-CDA0$_{\mathsf{RCE}}$. ∎

4.7 Instantiations of CE, HCE, and RCE

We define variants of CE, HCE1, HCE, RCE that rely solely on a block cipher, such as AES. This has significant efficiency benefits given the widespread hardware support for AES. A block cipher is a map E: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ for which $E(k,\cdot) = E_k(\cdot)$ is a permutation with inverse $D(k,\cdot) = D_k(\cdot)$. Both E and D must be efficiently computable. We have for simplicity assumed that the key size and block size are equal. This is true of AES128, and one can extend the constructions below to other cases.

In Fig. 4.10 we define the hash function $MD[E]$ with output length n bits and the one-time symmetric encryption algorithm CTR$[E]$. The Merkle-Damgard transform $MD[E]: \{0,1\}^* \to \{0,1\}^n$ over (k,n)-block cipher E in Davies-Meyer mode for messages of length $< 2^k$, and CTR$[E]$, the counter mode of operation of E. If $\ell \in \mathbb{N}$, then $\langle \ell \rangle_n$ denotes an n-bit encoding of ℓ. The former is the Merkle-Damgärd with strengthening transform applied to the Davies-Meyer compression function using E. The latter is standard CTR mode with a fixed IV. We define the MLE scheme RCE$[E] = (P, \mathsf{K}, \mathsf{E}, \mathrm{T}, \mathsf{D})$ via the algorithms below. A diagram is shown in Fig. 4.11. The scheme CE$[E]$ is derived from RCE$[E]$ by: (1) setting $L = 0^n$, using c_2 as the symmetric key for CTR$[E]$, and not including c_2 in the ciphertext; and (2) having tag generation hash the ciphertext by $E(MD[E](m), p)$. Scheme HCE$[E]$ is the same except that encryption uses as key for CTR$[E]$ the value c_2 and c_2 is not included in the ciphertext. The prv$-cda security of these variants can be shown in the ideal cipher model [84] using a proof that uses the preimage-awareness [54] of $MD[E]$ together with an adaptation of the proof techniques used in the analysis of deterministic encryption schemes given in [99]. We omit it for the sake of brevity. The STC and TC security of the schemes are analogous to the results in

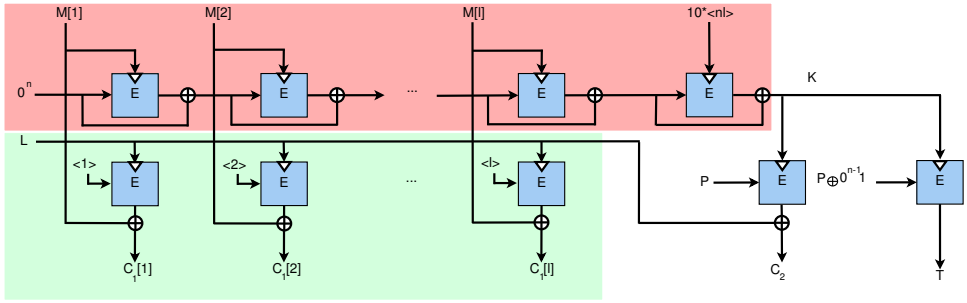
$\underline{MD[E](m)}$
$\mathbf{y}[0] \leftarrow 0^n$; $\mathbf{b} \leftarrow \mathsf{split}(\mathsf{pad}(m,k),k)$
$\ell \leftarrow |\mathbf{b}|$
For $i = 1,\ldots,\ell$ do
  $\mathbf{y}[i] \leftarrow E(\mathbf{b}[i], \mathbf{y}[i-1]) \oplus \mathbf{y}[i-1]$
Return $\mathbf{y}[\ell]$

$\underline{\mathsf{split}(m,k)}$
For $i = 1,\ldots,\lfloor |m|/k \rfloor$ do $\mathbf{b}[i] \leftarrow m[ik,(i+1)k]$
$\mathbf{b}[\lfloor |m|/k \rfloor + 1] \leftarrow m[\lfloor |m|/k \rfloor k, |m|]$; Return $\mathbf{b}$

$\underline{\mathsf{CTR}[E](k,m)}$
$\mathbf{b} \leftarrow \mathsf{split}(m,n)$; $\ell \leftarrow |\mathbf{b}|$
For $i = 1,\ldots,\ell-1$ do
  $\mathbf{c}[i] \leftarrow E(k, \langle i \rangle_n) \oplus \mathbf{b}[i]$
$X \leftarrow E(k, \langle \ell \rangle_n)$
$\mathbf{c}[\ell] \leftarrow \mathbf{b}[\ell] \oplus X[0, |\mathbf{b}[\ell]|]$
Return $\mathbf{c}$

$\underline{\mathsf{pad}(m,k)}$
$\ell \leftarrow k(\lfloor |m|/k \rfloor + 1) - k$;
Return $m \,\|\, 1 \,\|\, 0^\ell \,\|\, \langle |m| \rangle_k$

Figure 4.10. The Merkle-Damgard transform over a block cipher in Davies-Meyer mode.



$\underline{P(1^\lambda)}$
$p \xleftarrow{\$} \{0,1\}^n$; Return p

$\underline{\mathsf{K}_p(m)}$
$k \leftarrow MD[E](m)$; Return k

$\underline{\mathsf{E}_p(k,m)}$
$L \xleftarrow{\$} \{0,1\}^n$
$c_1 \leftarrow \mathsf{CTR}[E](L,m)$
$c_2 \leftarrow L \oplus E(k,p)$
$T \leftarrow E(k, p \oplus 0^{n-1}1)$
Return (c_1,c_2,T)

$\underline{\mathsf{D}_p(k,(c_1,c_2,T))}$
$L \leftarrow c_2 \oplus E(k,p)$
$m \leftarrow \mathsf{CTR}[E](L,c_1)$
$k' \leftarrow MD[E](m)$
If $E(k', p \oplus 0^{n-1}1) = T$ then
  Return m
Else Return $\perp$

Figure 4.11. Key generation and encryption, together in a single pass, for a message of length ℓn using $\mathsf{RCE}[E]$ for a block cipher E.

Section 4.3.

Speed comparisons.

We implemented the schemes

- $\mathsf{XXX}[\mathsf{CTR}[\mathsf{AES128}],\mathsf{H}]$ for $\mathsf{XXX} \in \{\mathsf{CE},\mathsf{HCE},\mathsf{RCE}\}$ and $\mathsf{H} \in \{\mathsf{SHA256},\mathsf{SHA512}\}$

- $\mathsf{XXX}[E]$ for $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE}, \mathsf{RCE}\}$ and $E \in \{\mathsf{AES128}, \mathsf{AES256}\}$

using OpenSSL version 1.0.0, with AES-NI [72] turned on for both AES128 and AES256. We measured the encryption performance when processing 4KB inputs. (Larger sizes have performance that degrades as expected due to inputs not fitting into the CPU cache.) The tests were performed with a warm cache (prior to the timings, the inputs were accessed several times) on an x86-64 Intel Core i7-970 processor clocking at 3.2 GHz; the test machine had 12GB of memory. The compiler, kernel and operating system on the machine were gcc-4.6, Linux kernel 3.0.0-13 and Ubuntu 11.04 respectively. The programs were compiled with the `-O3 -march=native` flags to produce optimized code and `-msse4` flag to enable support for Streaming SIMD instructions. Processor frequency scaling was turned off while running the experiments. The system was otherwise idle during the tests, and we used the `rdtsc` instruction of the x86 instruction set to measure time. Fig. 4.1 lists the measured performance, in cycles per byte, of the various MLE schemes. In terms of absolute performance, the fastest among the schemes, $\mathsf{RCE}[\mathsf{AES256}]$ can encrypt a 4KB input and generate the tag in just over 8 microseconds. On the other hand, the CE instantiation with AES256 spends about 15 microseconds to generate the ciphertext and tag.

4.8 Proof of Theorem 4.4.1

As per the theorem statement, let $\mathsf{H} = (\mathsf{K}_\mathsf{H}, \mathsf{H})$ be a family of hash functions and let $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of extractors. Let $\ell(\cdot)$ be the input length and $\kappa(\cdot)$ the output length of Ext. Below we let $\mathrm{HC} = \mathrm{HC}[\mathsf{H}, \mathrm{Ext}]$. We concentrate on the second part of the theorem, for prv$-cda. The second part of the proof (showing that PRV-CDA security follows from PRV-CDA of the underlying hash family) proceeds in a similar fashion.. Sources below are always therefore of arity one.

The following lemma relates the PRV$-CDA advantage of an adversary A against

Table 4.1. Performance of CE instantiations in cycles per byte.

CE variant	Operation	Choice of H			
		SHA256	SHA512	AES128	AES256
CE	K	21.1	12.1	7.5	**5.3**
HCE		21.1	12.1	7.5	**5.3**
RCE		21.1	12.1	7.5	**5.3**
CE	E	1.2	1.2	1.2	**1.2**
HCE		1.4	1.3	1.3	1.3
RCE		1.4	1.3	1.3	1.3
CE	T	21.1	12.1	7.5	**5.3**
HCE		–	–	–	–
RCE		–	–	–	–
CE	K + E + T	43.4	25.4	16.3	11.8
HCE		22.5	13.6	8.9	6.6
RCE		22.3	13.3	8.7	**6.5**
CE	D	1.2	1.2	1.2	**1.2**
HCE		22.5	13.6	8.9	6.6
RCE		22.3	13.3	8.7	6.5

$\mathsf{HC}[\mathrm{Ext}_\lambda, \mathsf{H}]$ to the PRV$-CDA advantage of a H adversary B.

Lemma 4.8.1 *Let S be an* HC*-valid source with message number $m(\cdot)$ and message lengths $\ell(\cdot)$. Let A be an adversary. Then there exists a* H*-valid source S' and adversary B such that for all $\lambda \in \mathbb{N}$*

$$\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{HC},S,A}(\lambda) \leq \mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},S',B}(\lambda)\ .$$

Moreover, S' is such that $g_{S'}(\lambda) \leq 2^{-k(\lambda)} + \sqrt{2^{k(\lambda)} \cdot g_S(\lambda)}$; its message number is $m'(\lambda) = m(\lambda) \cdot \ell(\lambda)$; and the length of each message it outputs is $\kappa(\lambda)(\lambda) + 1 + \lceil \log_2(\ell(\lambda)) \rceil$. The running time of adversary B is $\mathbf{t_B} = \mathscr{O}(\mathbf{t_A} + m(\cdot)\ell(\cdot)t_{\mathsf{H}}(\cdot))$, and the running time of S' is $\mathbf{t_{S'}} = \mathscr{O}(\mathbf{t_S} + m(\cdot)t_{\mathrm{Ext}_\lambda(\cdot)})$ where t_{H} and t_{Ext_λ} are bounds on running times of of H *and* Ext_λ. $\square$

MAIN $G_1(\lambda)$
$S \leftarrow\$ \{0,1\}^{s(\lambda)}$; $k_h \leftarrow\$ \mathsf{K}_{\mathsf{H}}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$; $(\mathbf{m}, z) \leftarrow\$ S(1^\lambda)$
For $i = 1, \ldots, \lvert\mathbf{m}\rvert$ do
$m \leftarrow \mathbf{m}[i]$; $k \leftarrow \mathrm{Ext}_\lambda(S, m)$
$\ell \leftarrow \lvert m\rvert$; $m_1 \| \ldots \| m_\ell \leftarrow m$
For $j = 1$ to ℓ do
$c_j^1 \leftarrow \mathsf{H}(k_h, k \| j \| m_j)$; $c_j^0 \leftarrow\$ \{0,1\}^{\lvert c_j^1\rvert}$
$\mathbf{c}[i] \leftarrow (c_1^b, \ldots, c_\ell^b)$
$b' \leftarrow\$ A(S, k_h, \mathbf{c}, z)$
Return $(b = b')$

$S'(1^\lambda)$
$S \leftarrow\$ \{0,1\}^{s(\lambda)}$; $(\mathbf{m}, z) \leftarrow\$ S(1^\lambda)$
$i' \leftarrow 0$
For $i = 1, \ldots, \lvert\mathbf{m}\rvert$ do
$m \leftarrow \mathbf{m}[i]$; $k \leftarrow \mathrm{Ext}_\lambda(S, m)$
$\ell \leftarrow \lvert m\rvert$; $m_1 \| \ldots \| m_\ell \leftarrow m$
For $j = 1$ to ℓ do
$\mathbf{m}'[i'] \leftarrow k \| j \| m_j$
$i' \leftarrow i' + 1$
Return $\mathbf{m}', (z, S)$

Figure 4.12. Game G_1 and source S' for Theorem 4.4.1.

Proof: Consider game G_1, source S' and adversary B of Fig. 4.12. Game G_1 is the PRV$-CDA$_{\mathrm{HC}}$ game and so it follows that

$$\mathsf{Adv}^{\text{prv\$-cda}}_{\mathrm{HC},S,A}(\lambda) = 2 \cdot \Pr\left[G_1^A(\lambda)\right] - 1.$$

Adversary B, on inputs $k_h, \mathbf{c}, (z', S)$, plays the PRV$-CDA$_{\mathsf{H},S'}$ game by running $A(S, k_h, \mathbf{c}, z')$ and outputing the output of A. The source S' first picks a random extractor key S, then runs S to get $\mathbf{m}$ and then runs each component of $\mathbf{m}$ through Ext_λ with S to get the MLE keys $\mathbf{k}$. In the next step, it splits each $\mathbf{m}$ component into individual bits and prepends them with the appropriate MLE key and index to get $m(\lambda)\ell(\lambda)$ messages of the form $\mathbf{k}[i] \| \langle j \rangle \| \mathbf{m}[i][j]$ for $j \in [\ell(\lambda)]$ for $i \in [m(\lambda)]$. Then, S' outputs these messages along with auxiliary information z of S and S. As B plays the PRV-CDA game, depending on the bit b in the game, either the hashes corresponding to $\mathbf{m}$, or random strings are provided to B, and this corresponds to getting HC ciphertexts for $\mathbf{m}$, or random bits. Thus, B simulates G_1 for A with the bit of its PRV$-CDA game playing the role of the bit in G_1. Finally, when A finishes and outputs a bit, B also exits, echoing that bit. We

have $\Pr[G_1^A(\lambda)] = \Pr[\text{PRV\$-CDA}^B_{\mathsf{H},S'}(\lambda)]$ and hence

$$\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},S',B}(\lambda) = 2 \cdot \Pr\left[G_1^A\right] - 1 = \mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{HC},S,A}(\lambda).$$

Now, we relate S' and S, by observing that S is picked at random, it follows from the properties of Ext_λ that for a randomly picked $k \leftarrow\!\!\$\ \{0,1\}^{k(\lambda)}$

$$\begin{aligned}\mathbf{GP}(\text{Ext}_\lambda(S,\mathbf{m}[i]) \mid\mid j \mid\mid \mathbf{m}[i][j] | S, z) &\leq \mathbf{GP}(k|S,z) + \mathsf{SD}((S,\text{Ext}_\lambda(S,\mathbf{m}[i]),z);(S,k,z)) \\ &\leq \frac{1}{2^{k(\lambda)}} + \sqrt{2^{k(\lambda)}\mathbf{GP}(\mathbf{m}[i]|z)} \leq \frac{1}{2^{k(\lambda)}} + \sqrt{2^{k(\lambda)}\mathbf{GP}_S(\lambda)}\end{aligned}$$

for all $i \in [\ell(\lambda)]$, for all $i \in [m(\lambda)]$. This completes the proof of the lemma. ▮

To finish the proof of the theorem in the prv\$-cda case, we note that $g_{S'}(\cdot)$ is negligible when $2^{k(\cdot)} \cdot g_S(\cdot)$ is negligible. In turn, $\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},S',B}(\cdot)$ is negligible for all PT B and hence $\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{HC},S,A}(\cdot)$ is negligible for all PT A.

4.9 Proof of Theorem 4.5.5

Proof: Game G_1 of Fig. 4.13 is a hybrid between the PRV\$-CDA with $b = 0$ and $b = 1$, with the code of SXE and S. We have

$$\Pr[G_1^A(\lambda)] \geq \frac{1}{m(\lambda)}\mathsf{Adv}^{\text{prv\$-cda}}_{\mathsf{SXE}[\text{Ext},\mathsf{SE},p],S,A}(\lambda).$$

Game G_2 is like G_1, but makes a modification in how the ciphertext corresponding to the switching value g is treated. Specifically, it picks a random $g \leftarrow\!\!\$\ [m(\lambda)]$, creates the first $g-1$ ciphertexts and later $m(\lambda)-g$ ciphertexts as in G_1, but the g-th ciphertext is set to an encryption of the g-th plaintext under a random key, if $b = 0$, as opposed to a random string as in G_1.

The difference between G_1 and G_2 can be bounded by a simple single key ROR

MAIN $G_1(\lambda)$	MAIN G_2
$S \leftarrow\$ \{0,1\}^{\kappa}$; $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$	$S \leftarrow\$ \{0,1\}^{\kappa}$; $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$
$g \leftarrow\$ m(\lambda)$; $b \leftarrow\$ \{0,1\}$; $\mathbf{m}, z \leftarrow\$ S(1^{\lambda})$	$g \leftarrow\$ m(\lambda)$; $b \leftarrow\$ \{0,1\}$; $\mathbf{m}, z \leftarrow\$ S(1^{\lambda})$
For $i = 1, \ldots, \lvert\mathbf{m}\rvert$ do	For $i = 1, \ldots, \lvert\mathbf{m}\rvert$ do
$m_1 \leftarrow \mathsf{Proj}(\mathbf{m}[i], U)$	$m_1 \leftarrow \mathsf{Proj}(\mathbf{m}[i], U)$
$m_2 \leftarrow \mathsf{Proj}(\mathbf{m}[i], [n(\lambda)] \setminus U)$	$m_2 \leftarrow \mathsf{Proj}(\mathbf{m}[i], [n(\lambda)] \setminus U)$
$k \leftarrow \mathrm{Ext}_{\lambda}(S, m_1)$	$k \leftarrow \mathrm{Ext}_{\lambda}(S, m_1)$
If $i < g + b$ then $\mathbf{c}[i] \leftarrow \mathsf{SE}(1^{\lambda}, k, m_2)$	If $i < g + b$ then $\mathbf{c}[i] \leftarrow \mathsf{SE}(1^{\lambda}, k, m_2)$
If $i = g$ and $b = 1$ then	If $i = g$ and $b = 1$ then $\mathbf{c}[i] \leftarrow \mathsf{SE}(1^{\lambda}, k, m_2)$
$\mathbf{c}[i] \leftarrow \mathsf{SE}(1^{\lambda}, k, m_2)$	If $i = g$ and $b = 0$ then
If $i = g$ and $b = 0$ then	$k' \leftarrow\$ \{0,1\}^{k_s}$; $\mathbf{c}[i] \leftarrow \mathsf{SE}(1^{\lambda}, k', m_2)$
$\mathbf{c}[i] \leftarrow\$ \{0,1\}^{\lvert\mathsf{SE}(1^{\lambda}, k, m_2)\rvert}$	If $i > g$ then $\mathbf{c}[i] \leftarrow\$ \{0,1\}^{\lvert\mathsf{SE}(1^{\lambda}, k, m_2)\rvert}$
If $i > g$ then	$b' \leftarrow\$ A(S \lvert\rvert U, \mathbf{c}, z)$; Return$(b = b')$
$c' \leftarrow \mathsf{SE}(1^{\lambda}, k, m_2)$; $\mathbf{c}[i] \leftarrow\$ \{0,1\}^{c'}$	
$b' \leftarrow\$ A(S \lvert\rvert U, \mathbf{c}, z)$; Return$(b = b')$	

Figure 4.13. Games for Theorem 4.5.5.

adversary B which runs the hybrid, and handles its switching ciphertext as follows. It flips a random bit b, and if $b = 1$, it runs the extractor on the g-th input to get the key and encrypts the g-th message under this key. If $b = 0$, then it forwards the g-th message to its ENC oracle. Depending on whether the bit in the ROR game is 0 or 1, adversary A is simulated either in G_1 or G_2. Moreover, B never makes more than one query to its ENC oracle. We have, $(|\Pr[G_1^A(\lambda)] - \Pr[G_2^A(\lambda)]|)/2 = \mathsf{Adv}^{\mathsf{1\text{-}ror}}_{\mathsf{SE},B}(\lambda)$ for all $\lambda \in \mathbb{N}$.

In game G_2, the difference between the settings of $b = 0$ and $b = 1$ is in how the key for g-th plaintext is derived. If $b = 0$, the extractor output is used, and if $b = 1$ then a random key is chosen. Importantly, this is the last plaintext for which the game replies

with a real encryption; subsequent components of $\mathbf{c}$ are simply random strings. We have

$$\begin{aligned}\Pr[G_2^A(\lambda)] \leq \mathsf{SD}((\mathsf{SE}(1^\lambda, k, \mathsf{Proj}(m_g, [n(\lambda)] \setminus U)), m_1, \ldots, m_{g-1}, z, S),\\ (\mathsf{SE}(1^\lambda, \mathrm{Ext}_\lambda(S, \mathsf{Proj}(m_g, U)),\\ \mathsf{Proj}(m_g, [n(\lambda)] \setminus U)), m_1, \ldots, m_{g-1}, z, S))\\ \leq \mathsf{SD}(k, \mathsf{Proj}(m_g, [n(\lambda)] \setminus U), m_1, \ldots, m_{g-1}, z, S),\\ (\mathrm{Ext}_\lambda(S, \mathsf{Proj}(m_g, U))), \mathsf{Proj}(m_g, [n(\lambda)] \setminus U), m_1, \ldots, m_{g-1}, z, S)).\end{aligned}$$

In the above probabilities, $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$. The source S is such that

$$g_{S,p}(\lambda) = \max_i \mathbf{GP}(\mathsf{Proj}(\mathbf{m}[i], U) \mid (\mathbf{m}[1], \ldots, \mathbf{m}[i-1], \mathsf{Proj}(\mathbf{m}[i], [n(\lambda)] \setminus U), z))$$

which in turn is equal to $2^{-k(\lambda)}\nu(\lambda)$ where ν is some negligible function. Applying that Ext is an extractor, the above statistical distance is bounded by $\sqrt{2^{k(\lambda)} g_{S,p}(\lambda)} = \sqrt{\nu(\lambda)}$. Putting these together, we have

$$\mathsf{Adv}^{\mathsf{prv\$\text{-}cda}}_{\mathsf{SXE}[\mathrm{Ext},\mathsf{SE},p],S,A}(\lambda) \leq m(\lambda)(\sqrt{\nu(\lambda)} + \mathsf{Adv}^{\mathsf{1\text{-}ror}}_{\mathsf{SE},B}(\lambda)).$$

Since both the quantities on the right are negligible and A and S are PT algorithms, the prv\$-cda advantage of A is negligible and $\mathsf{SXE}[\mathrm{Ext}, \mathsf{SE}, p]$ is prv\$-cda secure for these types of sources.

Acknowledgements

This chapter is a reproduction of the material as it appears in Message-locked encryption and secure deduplication, by Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart from Advances in Cryptology, EUROCRYPT 2013. I wish to thank Thomas

Ristenpart for several fruitful collaborations, and for guiding me towards improving my technical skill, writing style, and productivity.

Chapter 5

Universal Computational Extractors

We now look at a new notion of security for (keyed) hash functions called UCE (Universal Computational Extractor). UCE-security is the first well-defined, standard-model security attribute of a hash function shown to permit the latter to securely instantiate ROs across a fairly broad spectrum of schemes and goals.

The random-oracle paradigm of BR93 [22] has two steps: (1) Design your scheme, and prove it secure, in the ROM, where the scheme algorithms and adversary have access to a RO denoted RO (2) Instantiate the RO to get the standard model scheme that is actually implemented and used. We will consider instantiation via a family of functions H, which means that the instantiated scheme is obtained by replacing RO calls of the ROM-scheme algorithms by evaluations of the deterministic function $\mathsf{H}(k,\cdot)$ specified by a key $k \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$, where λ is the security parameter. The key k is put in the public key of the instantiated scheme if the latter is public key, else enters in some scheme-dependent way. The suggestion of BR93 was that if H "behaved like a RO," the instantiated scheme would be secure in the standard model. They suggested to obtain such instantiations, heuristically, via cryptographic hash functions. The fundamental subsequent concern has been the lack of a proof of security for the instantiated scheme.

The lack of a proof of security for the instantiated scheme is a consequence of an even more fundamental lack, namely that of a *definition*, of what it means for a family of

functions to "behave like a RO," that could function as an assumption on which to base the proof. The PRF definition [66], which has worked so well in the symmetric setting, is inadequate here because PRF-security relies on the adversary not knowing the key. And collision-resistance (CR) is far from sufficient in any non-trivial usage of a RO.

Canetti [44] was the first to articulate the need for defintions for function families to model properties of random oracles, and seek a standard-model primitive sufficient to capture some usages of a RO. Notions such as Perfectly One-Way Probabilistic Hash Functions (POWHFs) [44, 48, 45] and non-malleable hash functions [31] have however proven of limited applicability [33]. Another direction has been to try to instantiate the RO in particular schemes like OAEP [23], again with limited success [34, 33] or under strong assumptions on RSA [85]. These works [44, 48] aimed for security notions that they could achieve under standard assumptions. Expectedly, applicability was limited. UCEs start from the perspective of maximizing applicability while being seen as an assumption rather than something to achieve under other assumptions.

UCE

At a high level, our definition considers a *source* S which is an algorithm executed with access to an oracle HASH, the latter being $\mathsf{H}(k,\cdot)$ for key $k \leftarrow\!\!\$\, \mathsf{K}(1^\lambda)$ if the challenge bit b is 1, and a RO otherwise. If security now asks that S not figure out b, then, if we deny it hk, we would be back to PRFs, and if we give it hk, security would be unachievable. So we don't ask S to figure out b. Instead, it must pass to an accomplice adversary D, called the distinguisher, some information L called the leakage. The distinguisher *is given the key* hk and must figure out b. For a class $\mathbf{S}$ of sources, let us say that $\mathsf{H} \in \mathsf{UCE}[\mathbf{S}]$, or is $\mathsf{UCE}[\mathbf{S}]$-secure, if, for all $S \in \mathbf{S}$ and all PT D, the advantage of S, D in figuring out b, in the game sketched above, is negligible.

Clearly, $\mathsf{UCE}[\mathbf{S}]$-security is not achievable if $\mathbf{S}$ is the class of *all* PT sources. For example, the source could include in L a point x and the result $y = \text{HASH}(x)$ of its oracle

on x, and D, having hk, can test whether or not $y = \mathsf{H}(k,x)$. We seek, accordingly, classes $\mathbf{S}$ small enough that the assumption of $\mathsf{UCE}[\mathbf{S}]$-security (that is, that this set is non-empty) is plausible, yet large enough that the same assumption is useful for applications. The classes are obtained by restricting the source. Here, we consider a restriction we term unpredictability. Unpredictability of S requires that it be infeasible for a predictor adversary P, given the leakage produced by the source in the *random* ($b = 0$) game, to find any of the inputs queried by S to its oracle. Note that unpredictability is a property of the source, not of the family of functions H, the latter not figuring in the definition at all. We let $\mathbf{S}^{\mathrm{sup}}$ be the classes of PT sources unpredictable to unbounded adversaries, and this leads to UCE class $\mathsf{UCE}[\mathbf{S}^{\mathrm{sup}}]$. In the basic definitions, only a single hashing key is involved, and we also define mUCE, a multi-key analogue.

The full version of the UCE paper [18] provides a comprehensive description of the ROM landscape, the UCE notion, and shows how a variety of applications can be moved to the standard model via UCE. The paper also discusses a variety of UCE classes to support these applications, including restrictions based on notions of unpredictability, reset-security, splitting and run-time limitations. Although the framework we develop here can be used to model several classes of sources and hence several classes of UCE security, our applications, namely, KDM secure symmetric encryption schemes and MLE schemes, will need only the $\mathsf{UCE}[\mathbf{S}^{\mathrm{sup}}]$ UCE class of [18]. Some classes of UCE sources have been affected by a family of attacks due to Brzuska, Farshim, and Mittelbach [41]. These classes of attacks rely on indistinguishability obfuscation (IO) [61], and have been instantiated by the authors based on the candidate IO schemes due to [61]. However, these attacks do not apply to the class $\mathsf{UCE}[\mathbf{S}^{\mathrm{sup}}]$ which we consider here. This follows from the fact that in $\mathsf{UCE}[\mathbf{S}^{\mathrm{sup}}]$, the predictors are allowed to run for unbounded time, and the attacks rely on the unpredictability of a source which leaks an obfuscated circuit containing a query point. But, there is some evidence that statistically-secure iO for

general circuits is not possible. Garg, Gentry, Halevi, Raykova, Sahai, Waters [62] give a construction of witness encryption from iO, and we observe that if the latter is statistically secure, so is the former. But Garg, Gentry, Sahai, and Waters [63] show that statistically-secure witness encryption implies a collapse of the Polynomial-Time Hierarchy. Since IO secure against unbounded adversaries does not likely exist, such a source cannot be unpredictable in this setting.

5.1 Definitions

We now extend the syntax of hash functions from Section 1 to allow variable length outputs. A family of functions $\mathsf{H} = (\mathsf{K}, \mathsf{H})$ specifies the following. On input the security parameter 1^λ, key generation algorithm K returns a key $k \in \{0,1\}^{\kappa(\lambda)}$, where $\kappa\colon\ \mathbb{N} \to \mathbb{N}$ is the keylength function associated to H. The deterministic, PT evaluation algorithm H takes 1^λ, a key $k \in [\mathsf{K}(1^\lambda)]$, an input $x \in \{0,1\}^*$ with $|x| \in \Omega(\lambda)$, and a unary encoding 1^ℓ of an output length $\ell \in \rho(\lambda)$ to return an output $\mathsf{H}(1^\lambda, k, x, 1^\ell) \in \{0,1\}^\ell$. Here Ω is the input-length function associated to H, so that $\Omega(\lambda) \subseteq \mathbb{N}$ is the (non-empty) set of allowed input lengths, and similarly ρ is the output-length function associated to H, so that $\rho(\lambda) \subseteq \mathbb{N}$ is the (non-empty) set of allowed output lengths.

5.1.1 UCE security

Let H be a family of functions. Let S be an adversary called the *source* and D an adversary called the *distinguisher*. We associate to them and H the game $\mathrm{UCE}^{S,D}_{\mathsf{H}}(\lambda)$ of Fig. 5.2. The source has access to an oracle HASH and we require that any query $x, 1^\ell$ made to this oracle satisfy $|x| \in \Omega(\lambda)$ and $\ell \in \rho(\lambda)$. When the challenge bit b is 1 (the "real" case) the oracle responds via H under a key k that is chosen by the game and *not* given to the source. When $b = 0$ (the "random" case) it responds as a RO. The source communicates to its accomplice distinguisher a string $L \in \{0,1\}^*$ we call the *leakage*.

$\underline{\text{MAIN UCE}^{S,D}_{\mathsf{H}}(\lambda)}$	$\underline{\text{MAIN Pred}^{P}_{S}(\lambda)}$	$\underline{\text{MAIN SPred}^{P'}_{S}(\lambda)}$
$b \leftarrow\!\!\$\ \{0,1\};\ k \leftarrow\!\!\$\ \mathsf{K}(1^\lambda)$	$\mathsf{done} \leftarrow \mathsf{false};\ Q \leftarrow \emptyset$	$Q \leftarrow \emptyset$
$L \leftarrow\!\!\$\ S^{\text{HASH}}(1^\lambda)$	$L \leftarrow\!\!\$\ S^{\text{HASH}}(1^\lambda);\quad \mathsf{done} \leftarrow \mathsf{true}$	$L \leftarrow\!\!\$\ S^{\text{HASH}}(1^\lambda)$
$b' \leftarrow\!\!\$\ D(1^\lambda, k, L)$	$Q' \leftarrow\!\!\$\ P^{\text{HASH}}(1^\lambda, L)$	$x \leftarrow\!\!\$\ P'(1^\lambda, L)$
Return $(b' = b)$	Return $(Q \cap Q' \neq \emptyset)$	Return $(x \in Q)$
$\underline{\text{HASH}(x, 1^\ell)}$	$\underline{\text{HASH}(x, 1^\ell)}$	$\underline{\text{HASH}(x, 1^\ell)}$
If $T[x,\ell] = \perp$ then	If $\mathsf{done} = \mathsf{false}$ then $Q \leftarrow Q \cup \{x\}$	$Q \leftarrow Q \cup \{x\}$
If $b = 1$ then $T[x,\ell] \leftarrow \mathsf{H}(1^\lambda, k, x, 1^\ell)$	If $T[x,\ell] = \perp$ then	If $T[x,\ell] = \perp$ then
Else $T[x,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$	$T[x,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$	$T[x,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $T[x,\ell]$	Return $T[x,\ell]$	Return $T[x,\ell]$

Figure 5.1. Games UCE, Pred used to define UCE security of family of functions H, and game SPred defining the simplified but equivalent form of unpredictability. Here S is the source, D is the distinguisher, P is the predictor and P' is the simple predictor.

The distinguisher *does* get the key k as input and must now return its guess $b' \in \{0,1\}$ for b. The game returns true iff $b' = b$, and the uce-advantage of (S,D) is defined for $\lambda \in \mathbb{N}$ via

$$\mathsf{Adv}^{\mathsf{uce}}_{\mathsf{H},S,D}(\lambda) = 2\Pr[\text{UCE}^{S,D}_{\mathsf{H}}(\lambda)] - 1\ . \tag{5.1}$$

One's first thought may now be to say that H is UCE-secure if $\mathsf{Adv}^{\mathsf{uce}}_{\mathsf{H},S,D}(\cdot)$ is negligible for all PT S and all PT D. But an obvious attack shows that no H can meet this definition. Indeed, S can pick some x and ℓ, let $h \leftarrow \text{HASH}(x, 1^\ell)$ and return leakage $L = (x, h, 1^\ell)$ to D. The latter, knowing k, can return 1 if $h = \mathsf{H}(1^\lambda, k, x, 1^\ell)$ and 0 otherwise.

To obtain useful and potentially achievable definitions of UCE-security for H, we will restrict the adversaries. In [19], we consider many ways to do this, so that UCE will be not a single definition but rather a framework in which many definitions are possible. However, for our purposes here, we restrict attention to the class of unpredictable sources. Let **S** be a class of sources and **D** a class of distinguishers. Then we let $\mathsf{UCE}[\mathbf{S},\mathbf{D}]$ be

the set of all H such that $\mathsf{Adv}^{\mathsf{uce}}_{\mathsf{H},S,D}(\cdot)$ is negligible for all $(S,D) \in \mathbf{S} \times \mathbf{D}$. To say that H is $\mathsf{UCE}[\mathbf{S},\mathbf{D}]$-secure simply means that $\mathsf{H} \in \mathsf{UCE}[\mathbf{S},\mathbf{D}]$. We let $\mathbf{D}^{\mathsf{poly}}$ the class of all PT distinguishers. We will almost always restrict attention to the latter and it is thus convenient to let $\mathsf{UCE}[\mathbf{S}] = \mathsf{UCE}[\mathbf{S},\mathbf{D}^{\mathsf{poly}}]$.

Unpredictable sources

Let S be a source. Consider game $\mathrm{Pred}^P_S(\lambda)$ of Fig. 5.2 associated to S and an adversary P called a *predictor*. Given the leakage, the latter outputs a set Q'. It wins if this set contains any HASH-query of the source. For $\lambda \in \mathbb{N}$ we let

$$\mathsf{Adv}^{\mathsf{pred}}_{S,P}(\lambda) = \Pr[\mathrm{Pred}^P_S(\lambda)] \ .$$

We consider the class of unpredictable sources where the predictor P is not necessarily a PT algorithm, but there exist polynomials q, q' such that for all $\lambda \in \mathbb{N}$, predictor P makes at most $q(\lambda)$ oracle queries and outputs a set Q' of size at most $q'(\lambda)$ in game $\mathrm{Pred}^P_S(\lambda)$. We say S is *unpredictable* if $\mathsf{Adv}^{\mathsf{pred}}_{S,P}(\cdot)$ is negligible for all such predictors P. We let $\mathbf{S}^{\mathrm{sup}}$ the class of all PT, unpredictable sources. This gives the UCE class $\mathsf{UCE}[\mathbf{S}^{\mathrm{sup}}]$.

We stress that in the prediction game, the HASH oracle of the source is a RO like in the random game, and the predictor gets the same oracle. The family H is not involved in the definition of unpredictability: the latter is a property of the source. For implementation, we suggest an instantiation based on HMAC [14, 11].

Simple unpredictability

Applications of unpredictability-based UCE assumptions involve proving the unpredictability of sources, and this task is simplified by using a simpler formulation of unpredictability, called simple unpredictability, that is equivalent to the original. The formalization considers game $\mathrm{SPred}^{P'}_S(\lambda)$ of Fig. 5.2 associated to source S and an adversary P' called a *simple predictor*. There are two simplifications: the simple predictor

MAIN $\mathrm{UCE}_{\mathsf{H}}^{S,D}(\lambda)$	MAIN $\mathrm{Pred}_{S}^{P}(\lambda)$	MAIN $\mathrm{SPred}_{S}^{P'}(\lambda)$
$b \leftarrow\$ \{0,1\}$; $k \leftarrow\$ \mathsf{K}(1^\lambda)$	done $\leftarrow$ false; $Q \leftarrow \emptyset$	$Q \leftarrow \emptyset$
$L \leftarrow\$ S^{\mathrm{HASH}}(1^\lambda)$	$L \leftarrow\$ S^{\mathrm{HASH}}(1^\lambda)$; done $\leftarrow$ true	$L \leftarrow\$ S^{\mathrm{HASH}}(1^\lambda)$
$b' \leftarrow\$ D(1^\lambda, k, L)$	$Q' \leftarrow\$ P^{\mathrm{HASH}}(1^\lambda, L)$	$x \leftarrow\$ P'(1^\lambda, L)$
Return $(b' = b)$	Return $(Q \cap Q' \neq \emptyset)$	Return $(x \in Q)$
HASH$(x, 1^\ell)$	HASH$(x, 1^\ell)$	HASH$(x, 1^\ell)$
If $T[x,\ell] = \perp$ then	If done = false then $Q \leftarrow Q \cup \{x\}$	$Q \leftarrow Q \cup \{x\}$
If $b = 1$ then	If $T[x,\ell] = \perp$ then	If $T[x,\ell] = \perp$ then
$T[x,\ell] \leftarrow \mathsf{H}(1^\lambda, k, x, 1^\ell)$	$T[x,\ell] \leftarrow\$ \{0,1\}^\ell$	$T[x,\ell] \leftarrow\$ \{0,1\}^\ell$
Else $T[x,\ell] \leftarrow\$ \{0,1\}^\ell$	Return $T[x,\ell]$	Return $T[x,\ell]$
Return $T[x,\ell]$		

Figure 5.2. Games UCE, Pred used to define UCE security of family of functions H, and game SPred defining the simplified but equivalent form of unpredictability. Here S is the source, D is the distinguisher, P is the predictor and P' is the simple predictor.

---

does not have access to the RO HASH, and its output is a single string x rather than a set of strings. It wins if x is a HASH-query of the source. For $\lambda \in \mathbb{N}$ we let

$$\mathsf{Adv}^{\mathsf{spred}}_{S,P'}(\lambda) = \Pr[\mathrm{SPred}^{P'}_{S}(\lambda)] \ .$$

We say that source S is *simple unpredictable* if $\mathsf{Adv}^{\mathsf{spred}}_{S,P'}(\cdot)$ is negligible for all simple predictors P'. The following lemma says that simple unpredictability is equivalent to unpredictability.

Lemma 5.1.1 Let S be a source. Then S is unpredictable if and only if it is simple unpredictable.

Proof:[Lemma 5.1.1] Suppose P' is a simple predictor. Let $P^{\mathrm{HASH}}(1^\lambda, L)$ be the algorithm which runs $x \leftarrow\$ P'(1^\lambda, L)$ and returns $\{x\}$. Then $\mathsf{Adv}^{\mathsf{spred}}_{S,P'}(\cdot) \leq \mathsf{Adv}^{\mathsf{pred}}_{S,P}(\cdot)$. This shows that if S is unpredictable then it is also simple unpredictable. Turning to the converse, let P be a predictor. We may assume wlog that S and P never repeat HASH

queries. We may also assume wlog that the output Q' of P contains every x for which there exists ℓ such that HASH-query $(x, 1^\ell)$ was made by P. (This is wlog because we can modify P to include all such x in Q'.) Let q be a polynomial that bounds the number of elements in the output Q' of P. Game $\mathrm{G}_1^{S,P}(\lambda)$ below includes the boxed code while game $\mathrm{G}_2^{S,P}(\lambda)$ does not:

$\underline{P'(1^\lambda, L)}$	$\underline{\textsc{Main}\ \boxed{\mathrm{G}_1^{S,P}(\lambda)}, \mathrm{G}_2^{S,P}(\lambda)}$
$Q' \leftarrow\!\!\$\ P^{\textsc{HashSim}}(1^\lambda, L)$	$Q \leftarrow \emptyset$; $L \leftarrow\!\!\$\ S^{\textsc{Hash}_1}(1^\lambda)$; $Q' \leftarrow\!\!\$\ P^{\textsc{Hash}_2}(1^\lambda, L)$
$x \leftarrow\!\!\$\ Q'$	Return $(Q \cap Q' \neq \emptyset)$
Return x	$\underline{\textsc{Hash}_1(x, 1^\ell)}$
$\underline{\textsc{HashSim}(x, 1^\ell)}$	$Q \leftarrow Q \cup \{x\}$; $T[x,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$;
$y \leftarrow\!\!\$\ \{0,1\}^\ell$; Return y	Return $T[x,\ell]$
	$\underline{\textsc{Hash}_2(x, 1^\ell)}$
	If $x \in Q$ then bad $\leftarrow$ true; $\boxed{\text{Return } T[x,\ell]}$
	$y \leftarrow\!\!\$\ \{0,1\}^\ell$; Return y

Game $\mathrm{G}_1^{S,P}(\lambda)$ is identical to $\mathrm{Pred}_S^P(\lambda)$, except that it separates the HASH procedures used by S and P, while maintaining consistency. Setting bad has no effect on the outcome of the game. Games $\mathrm{G}_1^{S,P}(\lambda)$ and $\mathrm{G}_2^{S,P}(\lambda)$ are identical-until-bad. From the fundamental lemma of game-playing [25],

$$\mathsf{Adv}^{\mathsf{pred}}_{S,P}(\cdot) = \Pr[\mathrm{G}_1^{S,P}(\cdot)] \leq \Pr[\mathrm{G}_2^{S,P}(\cdot)] + \Pr[\mathrm{G}_2^{S,P}(\cdot) \text{ sets bad}].$$

From the assumption that Q' contains every x such that P queried some $(x, 1^\ell)$ to HASH, if game G_2 sets bad then P will surely win. Hence $\Pr[\mathrm{G}_2^{S,P}(\cdot) \text{ sets bad}] \leq \Pr[\mathrm{G}_2^{S,P}(\cdot)]$.

$\underline{\text{MAIN mUCE}^{S,D}_{\mathsf{H}}(\lambda)}$	$\underline{\text{MAIN mPred}^{P}_{S}(\lambda)}$	$\underline{\text{MAIN mSPred}^{P'}_{S}(\lambda)}$
$(1^n,t) \leftarrow\!\!\$\ S(1^\lambda,\varepsilon)$	$(1^n,t) \leftarrow\!\!\$\ S(1^\lambda,\varepsilon)$	$(1^n,t) \leftarrow\!\!\$\ S(1^\lambda,\varepsilon)$
For $i = 1$ to n do $\mathbf{hk}[i] \leftarrow\!\!\$\ \mathsf{K}(1^\lambda)$	done $\leftarrow$ false; $Q \leftarrow \emptyset$	$Q \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$; $L \leftarrow\!\!\$\ S^{\text{HASH}}(1^n,t)$	$L \leftarrow\!\!\$\ S^{\text{HASH}}(1^n,t)$; done $\leftarrow$ true	$L \leftarrow\!\!\$\ S^{\text{HASH}}(1^n,t)$
$b' \leftarrow\!\!\$\ D(1^\lambda,\mathbf{hk},L)$	$Q' \leftarrow\!\!\$\ P^{\text{HASH}}(1^\lambda,1^n,L)$	$x \leftarrow\!\!\$\ P'(1^\lambda,1^n,L)$
Return $(b' = b)$	Return $(Q \cap Q' \neq \emptyset)$	Return $(x \in Q)$
$\underline{\text{HASH}(x,1^\ell,i)}$	$\underline{\text{HASH}(x,1^\ell,i)}$	$\underline{\text{HASH}(x,1^\ell,i)}$
If $T[x,\ell,i] = \perp$ then	If done $=$ false then	$Q \leftarrow Q \cup \{x\}$
If $b = 1$ then	$Q \leftarrow Q \cup \{x\}$	If $T[x,\ell,i] = \perp$ then
$T[x,\ell,i] \leftarrow \mathsf{H}(1^\lambda,\mathbf{hk}[i],x,1^\ell)$	If $T[x,\ell,i] = \perp$ then	$T[x,\ell,i] \leftarrow\!\!\$\ \{0,1\}^\ell$
Else $T[x,\ell,i] \leftarrow\!\!\$\ \{0,1\}^\ell$	$T[x,\ell,i] \leftarrow\!\!\$\ \{0,1\}^\ell$	Return $T[x,\ell,i]$
Return $T[x,\ell,i]$	Return $T[x,\ell,i]$	

Figure 5.3. Games mUCE, mPred, and mSPred used to define mUCE security of family of functions H, and game mSPred defining the simplified but equivalent form of unpredictability. Here S is the multi-source, D is the distinguisher, P is the predictor and P' is the simple predictor.

Now, consider the simple predictor P' as above. Then

$$\mathsf{Adv}^{\mathsf{spred}}_{S,P'}(\cdot) = \frac{1}{q}\Pr[\mathrm{G}^{S,P}_2(\cdot)] \geq \frac{1}{2q}\mathsf{Adv}^{\mathsf{pred}}_{S,P}(\cdot)\ .$$

This concludes the proof. ∎

5.1.2 mUCE security

In UCE, there is a single target key hk. Some of our applications will depend on an extension involving multiple keys. Here we define this mUCE extension of UCE.

Framework

Let H be a family of functions. Consider game $\text{mUCE}^{S,D}_{\mathsf{H}}(\lambda)$ of Fig. 5.3 involving a multi-source S and distinguisher D. Adversary S now begins by returning a unary-encoded integer $n \geq 1$ indicating the number of instances, together with state information

t. The game creates n, independent keys. The oracle HASH given to S now allows it to query any instance $i \in [1,n]$ of its choice. As before S returns leakage L based on which the distinguisher D, now given the entire vector **hk** of keys, returns its guess bit b'. The mUCE-advantage of (S,D) is defined for $\lambda \in \mathbb{N}$ by

$$\mathsf{Adv}^{\mathsf{m\text{-}uce}}_{\mathsf{H},S,D}(\lambda) = 2\Pr[\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\lambda)] - 1\ . \tag{5.2}$$

Let $\mathbf{S}$ be a class of multi-sources and $\mathbf{D}$ a class of distinguishers. Then we let $\mathsf{mUCE}[\mathbf{S},\mathbf{D}]$ be the set of all H such that $\mathsf{Adv}^{\mathsf{m\text{-}uce}}_{\mathsf{H},S,D}(\cdot)$ is negligible for all $(S,D) \in \mathbf{S} \times \mathbf{D}$. We let $\mathsf{mUCE}[\mathbf{S}] = \mathsf{mUCE}[\mathbf{S},\mathbf{D}^{\mathsf{poly}}]$.

Classes

The single-key class $\mathsf{UCE}[\mathbf{S}^{\mathrm{sup}}]$ has a natural multi-key analogue. For $\lambda \in \mathbb{N}$ we let $\mathsf{Adv}^{\mathsf{m\text{-}pred}}_{S,P}(\lambda) = \Pr[\mathrm{mPred}^{P}_{S}(\lambda)]$ where game $\mathrm{mPred}^{P}_{S}(\lambda)$ is in Fig. 5.3. Here P can run in unbounded time, but there must exist polynomials q,q' such that for all $\lambda \in \mathbb{N}$, predictor P makes at most $q(\lambda,n)$ oracle queries and outputs a set Q' of size at most $q'(\lambda,n)$ in game $\mathrm{Pred}^{P}_{S}(\lambda)$, where the number of keys n is defined via the output of S in the first line of the game. We say S is unpredictable if $\mathsf{Adv}^{\mathsf{m\text{-}pred}}_{S,P}(\cdot)$ is negligible for all predictors P. associated class of assumptions is $\mathsf{mUCE}[\mathbf{S}^{\mathrm{sup\text{-}m}}]$.

As with UCE, unpredictability is equivalent to simple unpredictability. In detail, consider game $\mathrm{mSPred}^{P'}_{S}(\lambda)$ of Fig. 5.3 and let $\mathsf{Adv}^{\mathsf{m\text{-}spred}}_{S,P'}(\lambda) = \Pr[\mathrm{mSPred}^{P'}_{S}(\lambda)]$. We say that multi-source S is simple unpredictable if $\mathsf{Adv}^{\mathsf{m\text{-}spred}}_{S,P'}(\cdot)$ is negligible for all simple predictors P'. The following analogue of Lemma 5.1.1 shows equivalence of simple unpredictability and unpredictability for multi-sources. The proof of Lemma 5.1.2 is similar to the proof of Lemma 5.1.1 and is omitted.

Lemma 5.1.2 Let S be a multi-source. Then S is unpredictable if and only if it is simple unpredictable.

5.2 Security for key-dependent messages

In Section 5.2 we defined security for key-dependent messages (KDM) and provided a construction that is KDM secure in the random oracle. We now show that KDM security is possible in the standard model, using UCE. Black, Rogaway, and Shrimpton (BRS) [29] formalized security in the presence of key-dependent messages (KDM) and described a simple and efficient KDM-secure symmetric encryption scheme in the ROM. We now instantiate the RO in the BRS scheme with a mUCE family and obtain an efficient KDM-secure symmetric encryption scheme in the standard model. There are several other standard-model KDM-secure encryption schemes [36, 5, 9, 90, 4] but they are significantly more complex and less efficient than our instantiated BRS scheme.

Definitions

Let SE be a symmetric encryption (SE) scheme as defined in Section 5.2.1. We define non-adaptive KDM security for symmetric encryption via game $\mathrm{KDM}^{A}_{\mathsf{SE}}(\lambda)$ of Fig. 5.4, an adversary $A = (A_1, A_2)$ is a pair of algorithms. Algorithm A_1, when invoked with $(1^{\lambda}, \varepsilon)$, returns $(1^{n}, t)$ where n is the number of keys it is requesting be created, and t is state information. Then when invoked with $(1^{\lambda}, (t, \mathbf{k}))$ where $\mathbf{k} \in (\{0,1\}^{\mathsf{SE.kl}(\lambda)})^{n}$ is a vector of keys, it outputs a triple of vectors $\mathbf{s}, \mathbf{m}_0, \mathbf{m}_1$ satisfying the following: (1) $|\mathbf{s}| = |\mathbf{m}_0| = |\mathbf{m}_1|$, and (2) $\mathbf{s}[i] \in [1, n]$ and $\mathbf{m}_0[i], \mathbf{m}_1[i] \in \mathsf{SE.il}(\lambda)$ for all $i \in [1, |\mathbf{s}|]$. We say that SE is KDM-secure if $\mathrm{Adv}^{\mathsf{kdm}}_{\mathsf{SE},A}(\cdot)$ is negligible for every PT KDM adversary A, where $\mathrm{Adv}^{\mathsf{kdm}}_{\mathsf{SE},A}(\lambda) = 2\Pr[\mathrm{KDM}^{A}_{\mathsf{SE}}(\lambda)] - 1$. Our definitions capture non-adaptive security, but this includes the cases that have been most prominent in past work, namely key cycles and cliques [36, 5, 1, 40].

Results

BRS [29] showed that encrypting a message m under key k by picking a random

$\underline{\text{MAIN KDM}^{A}_{\mathsf{SE}}(\lambda)}$	$\underline{\mathsf{SE.Enc}(1^\lambda, k, m)}$
$(1^n, t) \leftarrow\!\$\, A_1(1^\lambda, \varepsilon)$	$k \leftarrow\!\$\, \mathsf{K}(1^\lambda)$
For $i = 1$ to n do $\mathbf{k}[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{SE.kl}(\lambda)}$	$h \leftarrow \mathsf{H}(1^\lambda, k, k, 1^{\rho(\lambda)})$
$(\mathbf{s}, \mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\$\, A_1(1^\lambda, (t, \mathbf{k}))$	$c \leftarrow (k, h \oplus m)$; Return c
$b \leftarrow\!\$\, \{0,1\}$	$\underline{\mathsf{SE.Dec}(1^\lambda, k, (k, z))}$
For $i = 1$ to $\lvert\mathbf{m}_b\rvert$ do	$h \leftarrow \mathsf{H}(1^\lambda, k, k, 1^{\lvert z\rvert})$
$\quad \mathbf{c}[i] \leftarrow\!\$\, \mathsf{SE.Enc}(1^\lambda, \mathbf{k}[\mathbf{s}[i]], \mathbf{m}_b[i])$	$m \leftarrow\!\$\, h \oplus z$; Return m
$b' \leftarrow\!\$\, A_2(1^\lambda, t, \mathbf{c})$; Return $(b = b')$	

Figure 5.4. Left: The KDM game. Middle: The RKA game. Right: The SE scheme $\mathsf{SE} = \mathsf{HtX}[\mathsf{H}]$.

---

r and returning $(r, \mathsf{RO}(r \,\|\, k) \oplus m)$ is KDM secure when RO is a random oracle. The natural first attempt to instantiate via a family H would be to add $k \leftarrow\!\$\, \mathsf{K}(1^\lambda)$ to the encryption key and then replace RO with $\mathsf{H}(1^\lambda, k, \cdot, 1^{\rho(\lambda)})$, but this fails because in the KDM setting the messages are chosen by A_1 as a function of the encryption key(s), and UCE-security will not apply if the messages depend on k. Instead, we leave the key unchanged relative to the BRS scheme and view the random value r of the BRS scheme as a key for H, so that a fresh key k is chosen for each encryption. Given H with $\lambda \in \Omega(\lambda)$ for all $\lambda \in \mathbb{N}$, our instantiated transform produces the SE scheme $\mathsf{SE} = \mathsf{HtX}[\mathsf{H}]$ whose encryption and decryption algorithms are described in Fig. 5.4. (Here "HtX" stands for "Hash-then-XOR.") Its key length is defined by $\mathsf{SE.kl}(\lambda) = \lambda$ for all $\lambda \in \mathbb{N}$ and its input length is $\mathsf{SE.il} = \rho$. The following theorem says that $\mathsf{HtX}[\mathsf{H}]$ is KDM secure if H is $\mathsf{mUCE}[\mathbf{S}^{\text{sup-m}}]$-secure.

Theorem 5.2.1 If H is $\mathsf{mUCE}[\mathbf{S}^{\text{sup-m}}]$-secure, then $\mathsf{HtX}[\mathsf{H}]$ is non-adaptive KDM secure.

Proof:[Theorem 5.2.1] Let $\mathsf{SE} = \mathsf{HtX}[\mathsf{H}]$. Let $A = (A_1, A_2)$ be a PT KDM adversary. Assume that A_1 outputs messages of length $\rho(\lambda)$. We will construct a PT unpredictable

multi-source S and a PT distinguisher D such that

$$\mathsf{Adv}^{\mathsf{kdm}}_{\mathsf{SE},A}(\cdot) \le 2 \cdot \mathsf{Adv}^{\mathsf{m\text{-}uce}}_{\mathsf{H},S,D}(\cdot) \ . \tag{5.3}$$

The theorem then follows from the assumption that $\mathsf{H} \in \mathsf{mUCE}[\mathbf{S}^{\text{sup-m}}]$. Let q and $\overline{n}$ be polynomials such that, in game $\mathrm{KDM}^A_{\mathsf{SE}}(\lambda)$, we have $|\mathbf{m}_0| \le q(\lambda)$ and $n \le \overline{n}(\lambda)$ for all $\lambda \in \mathbb{N}$. The constructions of S and D are shown below:

$\underline{S^{\mathrm{HASH}}(1^\lambda, t)}$	$\underline{D(1^\lambda, \mathbf{hk}, L)}$
$(1^n, t') \leftarrow t$; $d \leftarrow\$ \{0,1\}$	$(\mathbf{c}', t', d) \leftarrow L$
If $t = \varepsilon$ then	For $i = 1$ to $\lvert\mathbf{c}'\rvert$ do
$(1^n, t') \leftarrow\$ A_1(1^\lambda, \varepsilon)$; Return $(1^{q(\lambda)}, (1^n, t'))$	$\mathbf{c}[i] \leftarrow (\mathbf{hk}[i], \mathbf{c}'[i])$
Else	$d' \leftarrow\$ A_2(1^\lambda, t', \mathbf{c})$
For $i = 1$ to n do $\mathbf{k}[i] \leftarrow\$ \{0,1\}^\lambda$	If $(d = d')$ then
$(\mathbf{s}, \mathbf{m}_0, \mathbf{m}_1) \leftarrow\$ A_1(1^\lambda, (t', \mathbf{k}))$	$b' \leftarrow 1$
For $i = 1$ to $\lvert\mathbf{m}_d\rvert$ do	else
$\mathbf{c}'[i] \leftarrow \mathrm{HASH}(\mathbf{k}[\mathbf{s}[i]], 1^{\rho(\lambda)}, i) \oplus \mathbf{m}_d[i]$	$b' \leftarrow 0$
$L \leftarrow (\mathbf{c}', t', d)$; Return L	Return b'

Let b denote the challenge bit in game $\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\cdot)$. Then

$$\Pr[\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\cdot) \,|\, b = 1] = \Pr[\mathrm{KDM}^A_{\mathsf{SE}}(\cdot)]$$
$$\Pr[\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\cdot) \,|\, b = 0] = \frac{1}{2} \ .$$

Summing yields Equation (5.3). It remains to show that S is unpredictable. It suffices to show that S is simple unpredictable by Lemma 5.1.2. Consider an arbitrary simple predictor P'. Given $|\mathbf{k}|$ and the leakage $(\mathbf{c}', t', d)$, the components of $\mathbf{k}$ are still uniformly and independently distributed. Hence $\mathsf{Adv}^{\mathsf{m\text{-}spred}}_{S,P'}(\lambda) \le \overline{n}(\lambda)/2^\lambda$ for every $\lambda \in \mathbb{N}$. ∎

5.2.1 Message-locked encryption

In Chapter 4, we introduced the notion of message-locked encryption, and provided constructions in the standard model and random oracle models. However, the standard model constructions either achived limited security, such as the SXE construction, or relied on the existence of other primitives (such as deterministic PKE) which are themselves not known to exist in the standard model. Here we sort this issue by providing a standard model instantiation of the Convergent Encryption construction which we initially showed to be secure in the random oracle model. Specifically, we now show that a particular variant of convergent encryption can be instantiated with a UCE secure hash function family. Let $\mathsf{H} = (\mathsf{K}, \mathsf{H})$ be a hash function family. Then, we associate an MLE scheme $\mathsf{CE}[\mathsf{H}]$ with H as follows. Parameter generation $\mathsf{CE.Pg}$ simply runs K to get k_h which it returns as the public parameters.

$\underline{\mathsf{CE.Kg}(1^\lambda, k_h, m)}$	$\underline{\mathsf{CE.Enc}(1^\lambda, k_h, k, m)}$	$\underline{\mathsf{CE.Dec}(1^\lambda, k_h, k, c)}$
$k \leftarrow \mathsf{H}(1^\lambda, k_h, m, 1^\lambda)$	$c \leftarrow m \oplus \mathsf{H}(1^\lambda, k_h, k, 1^{\lvert m \rvert})$	$m \leftarrow c \oplus \mathsf{H}(1^\lambda, k_h, k, 1^{\lvert c \rvert})$
Return k	Return c	Return m

Here $\mathsf{CE.Tag}(1^\lambda, k_h, c)$ simply returns c. Correctness and tag consistency are easy to verify. The following theorem shows that $\mathsf{CE}[\mathsf{H}]$ is PRV$-CDA-secure.

Theorem 5.2.2 If H is $\mathsf{UCE}[\mathbf{S}^{\mathrm{sup}}]$-secure, then $\mathsf{CE}[\mathsf{H}]$ is PRV$-CDA-secure.

Proof:[Theorem 5.2.2] Let A be a PT high min-entropy PRV$-CDA adversary. Let v, ℓ be functions associated to A as per the definitions. Let a, c be the challenge bits of games $\mathrm{UCE}^{S,D}_{\mathsf{H}}(\cdot)$, and $\text{PRV\$-CDA}^{A}_{\mathsf{CE}[\mathsf{H}]}(\cdot)$ respectively. Consider the source and distinguisher S, D described below.

$\underline{S^{\text{HASH}}(1^\lambda)}$	$\underline{D(1^\lambda, L)}$
$\mathbf{m} \leftarrow\!\!\$\, A_1(1^\lambda)$	$a' \leftarrow A_2(1^\lambda, hk, L)$
For $i = 1$ to $\lvert\mathbf{m}\rvert$ do	Return a'
$\quad \mathbf{k}[i] \leftarrow \text{HASH}(\mathbf{m}[i], 1^\lambda)$	
$\quad \mathbf{c}[i] \leftarrow \mathbf{m}[i] \oplus \text{HASH}(\mathbf{k}[i], 1^{\lvert\mathbf{m}[i]\rvert})$	
$L \leftarrow \mathbf{c}$	
Return L	

It can be seen that

$$\Pr[\text{PRV\$-CDA}^A_{\mathsf{CE}[\mathsf{H}]}(\cdot) \,|\, c = 1] = \Pr[\text{UCE}^{S,D}_{\mathsf{H}}(\cdot) \,|\, a = 1]\,,$$
$$\Pr[\text{PRV\$-CDA}^A_{\mathsf{CE}[\mathsf{H}]}(\cdot) \,|\, c = 0] = \Pr[\text{UCE}^{S,D}_{\mathsf{H}}(\cdot) \,|\, a = 0]\,,$$

leading to $\mathsf{Adv}^{\mathsf{prv\$\text{-}cda}}_{\mathsf{CE}[\mathsf{H}],A}(\cdot) = \mathsf{Adv}^{\mathsf{uce}}_{\mathsf{H},S,D}(\cdot)$. It remains to show that S is statistically unpredictable and by Lemma 5.1.1 it suffices to show that S is simple statistically unpredictable. If P' is a simple predictor, it follows that $\mathsf{Adv}^{\mathsf{spred}}_{S,P'}(\cdot) \le v \cdot (\text{Guess}_A(\cdot) + 2^{-\lambda})$. Here v is the function associated to A as per the definitions. The simple statistical unpredictability of S then follows from the assumption that A has high min-entropy. ∎

5.3 Constructions of UCE families

In this section, we describe constructions of UCE-secure function families. We provide a ROM construction and prove that it is $\mathsf{mUCE}[\mathbf{S}^{\text{sup}}]$-secure. We go on to explore practical instantiations of UCE-secure functions. There is value of validating UCE in the ROM, even though it seem at first as not helping the goal of relying on random oracles.

Currently, RO-based design *directly* proves schemes secure in the ROM. UCE instead enables a layered approach where base primitives with standard-model security definitions are validated in the ROM. End goals are then reached from the base

$\underline{\text{MAIN mUCE}^{S,D}_{\mathsf{H}}(\lambda)}$	$\underline{\text{MAIN mPred}^{P}_{S}(\lambda)}$
$(1^n, t) \leftarrow\!\!\$\ S^{\mathsf{RO}}(1^\lambda, \varepsilon)$	$\mathsf{done} \leftarrow \mathsf{false}$; $Q \leftarrow \emptyset$
For $i = 1$ to n do $\mathbf{hk}[i] \leftarrow\!\!\$\ \mathsf{K}(1^\lambda)$	$L \leftarrow\!\!\$\ S^{\text{HASH},\mathsf{RO}}(1^\lambda)$; $\mathsf{done} \leftarrow \mathsf{true}$
$b \leftarrow\!\!\$\ \{0,1\}$; $L \leftarrow\!\!\$\ S^{\mathsf{RO},\text{HASH}}(1^n, t)$	$Q' \leftarrow\!\!\$\ P^{\text{HASH},\mathsf{RO}}(1^\lambda, L)$
$b' \leftarrow\!\!\$\ D^{\mathsf{RO}}(1^\lambda, \mathbf{hk}, L)$	Return $(Q \cap Q' \neq \emptyset)$
Return $(b' = b)$	
$\underline{\text{HASH}(x, 1^\ell, i)}$	$\underline{\text{HASH}(x, 1^\ell, i)}$
If $T[x,\ell,i] = \perp$ then	If $\mathsf{done} = \mathsf{false}$ then $Q \leftarrow Q \cup \{x\}$
If $b = 1$ then $T[x,\ell,i] \leftarrow \mathsf{H}^{\mathsf{RO}}(1^\lambda, \mathbf{hk}[i], x, 1^\ell)$	If $T[x,\ell,i] = \perp$ then
Else $T[x,\ell,i] \leftarrow\!\!\$\ \{0,1\}^\ell$	$T[x,\ell,i] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $T[x,\ell,i]$	Return $T[x,\ell,i]$
$\underline{\mathsf{RO}(v, 1^\ell)}$	
If $H[v,\ell] = \perp$ then $H[v,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$	
Return $H[v,\ell]$	

Figure 5.5. Left: Game mUCE defines multi-key UCE security in the ROM. Right: Game mPred defines multi-key unpredictability in the ROM.

primitives purely in the standard model, the ROM being entirely dispensed with in the second step. In implementations, we would instantiate families assumed UCE-secure via appropriately-keyed cryptographic hash functions whenever these appear to meet the particular UCE notion being used. UCE lets us be precise about the properties sought from the hash functions, and enables cryptanalytic validation, which, even if difficult, is at least meaningful.

5.3.1 Achieving UCE in the ROM

Definitions

The first step is to extend the syntax. In a ROM family of functions H, the algorithm H has oracle access to RO. The rest is as before. We now define $\mathsf{mUCE}[\mathbf{S}^{\text{sup-m}}]$ security of H in the ROM. The multi-source S now has access to RO in addition to HASH, and the distinguisher gets access to RO as well. We continue to define m-uce advantage

via Equation (5.2), with game $\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\lambda)$ now being that of Fig. **??**. The adversary now gets oracle access to RO in addition to HASH. We say that S is unpredictable if $\mathsf{Adv}^{\mathsf{m\text{-}pred}}_{S,P}(\cdot)$ is negligible for any P, where $\mathsf{Adv}^{\mathsf{m\text{-}pred}}_{S,P}(\lambda) = \Pr[\mathrm{Pred}^{P}_{S}(\lambda)]$ and game $\mathsf{Adv}^{\mathsf{m\text{-}pred}}_{S,P}(\lambda)$ is in Fig. 5.5. We say that H is $\mathsf{mUCE}[\mathbf{S}^{\text{sup-m}}]$-secure in the ROM if $\mathsf{Adv}^{\mathsf{m\text{-}uce}}_{\mathsf{H},S,D}(\cdot)$ is negligible for every PT unpredictable multi-source S and every PT D. Let $\mathsf{mUCE}^{\mathsf{ro}}[\mathbf{S}^{\text{sup-m}}]$ denote the set of all ROM function families H that are $\mathsf{mUCE}[\mathbf{S}^{\text{sup-m}}]$-secure in the ROM.

Results

We now describe a $\mathsf{mUCE}[\mathbf{S}^{\text{sup-m}}]$-secure ROM family of functions. The construction H is as follows. Let $\mathsf{K}(1^\lambda)$ return $hk \leftarrow\!\!\$\, \{0,1\}^\lambda$ for every $\lambda \in \mathbb{N}$. Let $\Omega = \mathbb{N}$ and $\rho = \mathbb{N}$. Let $\mathsf{H}^{\mathsf{RO}}(1^\lambda, k, m, 1^\ell)$ return $\mathsf{RO}(k \| m, 1^\ell)$ for every $k \in \{0,1\}^\lambda$, every $m \in \{0,1\}^*$, every $\ell \in \mathbb{N}$ and every $\lambda \in \mathbb{N}$. The following says that H is $\mathsf{mUCE}[\mathbf{S}^{\text{sup-m}}]$-secure in the ROM.

Theorem 5.3.1 Let H be the ROM function family defined above. Then H is $\mathsf{mUCE}^{\mathsf{ro}}[\mathbf{S}^{\text{sup-m}}]$ secure.

Proof: Let S be a PT unpredictable multi-source and let D be a PT distinguisher. Let $\nu(\lambda), q$ be polynomials such that $n \le \nu(\lambda)$ and S, D between them make at most $q(\lambda)$ RO-queries in game $\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\lambda)$, for all $\lambda \in \mathbb{N}$. Assume $\nu(\lambda) < 2^\lambda$ for all $\lambda \in \mathbb{N}$. Wlog, assume that S doesn't repeat a query to HASH or RO, and D does not repeat a query to RO. We'll construct a predictor P such that for all $\lambda \in \mathbb{N}$ we have

$$\mathsf{Adv}^{\mathsf{m\text{-}uce}}_{S,D,\mathsf{H}}(\lambda) \le \mathsf{Adv}^{\mathsf{m\text{-}pred}}_{S,P}(\lambda) + \frac{2\nu(\lambda)\cdot q(\lambda) + \nu(\lambda)^2}{2^\lambda} \quad . \tag{5.4}$$

The theorem follows from the assumption that S is unpredictable. Consider games G_1–G_6 in Fig. 5.6 and Fig. 5.7. We let RO_1 be the interface of S to access to the random oracle, and RO_2 be that of D. Let d be the challenge bit of game $\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\cdot)$. Then

$\underline{\text{MAIN } \mathrm{G}_1^{S,D}(\lambda),\ \boxed{\mathrm{G}_2^{S,D}(\lambda)}}$
$(1^n,t) \leftarrow\!\!\$\ S(1^\lambda,\varepsilon);\ M \leftarrow \emptyset$
For $i = 1$ to n do
  $\mathbf{k}[i] \leftarrow\!\!\$\ \{0,1\}^\lambda$
  If $\mathbf{k}[i] \in M$ then
    bad $\leftarrow$ true
    $\boxed{\mathbf{k}[i] \leftarrow\!\!\$\ \{0,1\}^\lambda \setminus M}$
  $M \leftarrow M \cup \{\mathbf{k}[i]\}$
$L \leftarrow\!\!\$\ S^{\text{HASH},\mathsf{RO}_1}(1^n,t)$
$b \leftarrow\!\!\$\ D^{\mathsf{RO}_2}(1^\lambda,\mathbf{k},L)$
Return $(b = 1)$

$\underline{\text{HASH}(x,1^\ell,i)}$
$v \leftarrow \mathbf{k}[i] \,\|\, x$
If $H[v,\ell] = \perp$ then
  $H[v,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $H[v,\ell]$

$\underline{\mathsf{RO}_1(v,1^\ell)}$
If $H[v,\ell] = \perp$ then $H[v,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $H[v,\ell]$

$\underline{\mathsf{RO}_2(v,1^\ell)}$
If $H[v,\ell] = \perp$ then $H[v,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $H[v,\ell]$

$\underline{\text{MAIN } \boxed{\mathrm{G}_3^{S,D}(\lambda)},\ \mathrm{G}_4^{S,D}(\lambda)}$
$(1^n,t) \leftarrow\!\!\$\ S(1^\lambda,\varepsilon);\ M \leftarrow \emptyset$
For $i = 1$ to n do
  $\mathbf{k}[i] \leftarrow\!\!\$\ \{0,1\}^\lambda$
  If $\mathbf{k}[i] \in M$ then
    bad $\leftarrow$ true; $\boxed{\mathbf{k}[i] \leftarrow\!\!\$\ \{0,1\}^\lambda \setminus M}$
  $M \leftarrow M \cup \{\mathbf{k}[i]\}$
$L \leftarrow\!\!\$\ S^{\text{HASH},\mathsf{RO}_1}(1^n,t);\ b \leftarrow\!\!\$\ D^{\mathsf{RO}_2}(1^\lambda,\mathbf{k},L)$
Return $(b = 1)$

$\underline{\text{HASH}(x,1^\ell,i)}$
If $T[x,\ell,i] = \perp$ then $T[x,\ell,i] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $T[x,\ell,i]$

$\underline{\mathsf{RO}_1(v,1^\ell)}$
$x \leftarrow v[\lambda+1,|v|];\ K \leftarrow v[1,\lambda]$
For $i = 1$ to n do
  If $K = \mathbf{k}[i]$ then
    coll $\leftarrow$ true
    If $T[x,\ell,i] = \perp$ then $T[x,\ell,i] \leftarrow\!\!\$\ \{0,1\}^\ell$
    Return $T[x,\ell,i]$
If $H[v,\ell] = \perp$ then $H[v,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $H[v,\ell]$

$\underline{\mathsf{RO}_2(v,1^\ell)}$
$x \leftarrow v[\lambda+1,|v|];\ K \leftarrow v[1,\lambda]$
For $i = 1$ to n do
  If $K = \mathbf{k}[i]$ then
    If $T[x,\ell,i] = \perp$ then $T[x,\ell,i] \leftarrow\!\!\$\ \{0,1\}^\ell$
    Return $T[x,\ell,i]$
If $H[v,\ell] = \perp$ then $H[v,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $H[v,\ell]$

Figure 5.6. Games G_1–G_4 for the proof of Theorem 5.3.1. Games $\mathrm{G}_2, \mathrm{G}_3$ include the corresponding boxed statement, while the other games do not.

$\Pr[\mathrm{G}_1^{S,D}(\lambda)] = \Pr[\mathrm{mUCE}_{\mathsf{H}}^{S,D}(\lambda) \mid d = 1]$ and $\Pr[\mathrm{G}_6^{S,D}(\lambda)] = 1 - \Pr[\mathrm{mUCE}_{\mathsf{H}}^{S,D}(\lambda) \mid d = 0]$.

We explain the game chain up to the terminal game. In game $\mathrm{G}_2^{S,D}(\lambda)$, we sample the

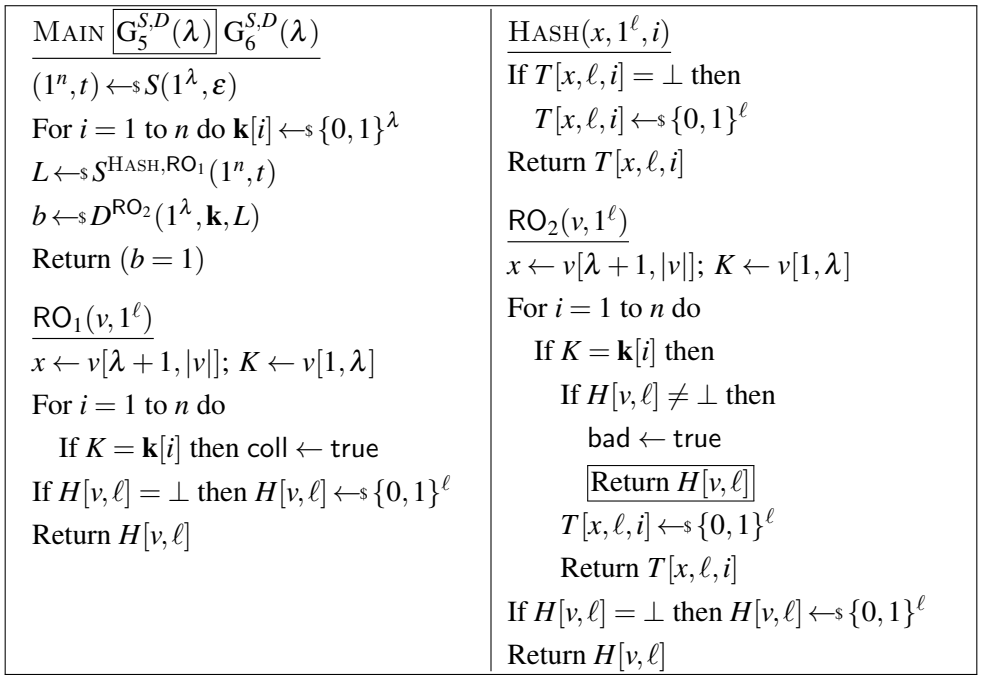


Figure 5.7. Games G_5 and G_6 for the proof of Theorem 5.3.1. Game G_5 includes the corresponding boxed statement, while the other game does not.

---

hash keys so that they are distinct. The two games $G_1^{S,D}(\lambda)$ and $G_2^{S,D}(\lambda)$ are identical-until-bad. Then, for all $\lambda \in \mathbb{N}$, we have

$$\Pr[G_1^{S,D}(\lambda)] - \Pr[G_2^{S,D}(\lambda)] \leq \Pr[G_2^{S,D}(\lambda) \text{ sets bad}] \leq \frac{\nu(\lambda)^2}{2^{\lambda+1}} \ .$$

In game $G_3^{S,D}(\lambda)$, for each string v, if there is $i \leq n$ such that $v[1,\lambda] = \mathbf{k}[i]$, instead of reading/writing to $H[v,\ell]$, we'll use $T\left[v[\lambda+1,|v|],\ell,i\right]$. Since the keys are distinct, for every $\lambda \in \mathbb{N}$,

$$\Pr[G_2^{S,D}(\lambda)] = \Pr[G_3^{S,D}(\lambda)] \ .$$

In game $G_4^{S,D}(\lambda)$, the keys now are sampled independently. The two games $G_3^{S,D}(\lambda)$ and $G_4^{S,D}(\lambda)$ are identical-until-bad. Then for every $\lambda \in \mathbb{N}$,

$$\Pr[G_3^{S,D}(\lambda)] - \Pr[G_4^{S,D}(\lambda)] \leq \Pr[G_4^{S,D}(\lambda) \text{ sets bad}] \leq \frac{\nu(\lambda)^2}{2^{\lambda+1}} \ .$$

In game $\mathrm{G}_5^{S,D}(\lambda)$, replies from RO_2 are no longer consistent with HASH replies. The two games $\mathrm{G}_4^{S,D}(\lambda)$ and $\mathrm{G}_5^{S,D}(\lambda)$ are identical-until-coll. Then for every $\lambda \in \mathbb{N}$,

$$\Pr[\mathrm{G}_4^{S,D}(\lambda)] - \Pr[\mathrm{G}_5^{S,D}(\lambda)] \leq \Pr[\mathrm{G}_5^{S,D}(\lambda) \text{ sets coll}] \leq \frac{v(\lambda)\cdot q(\lambda)}{2^\lambda},$$

where the last inequality is due to the fact that the keys now are uniformly random, and independent of whatever S receives.

Games $\mathrm{G}_5^{S,D}(\lambda)$ and $\mathrm{G}_6^{S,D}(\lambda)$ are identical-until-bad. The flag bad is set only if S or D queries $(\mathbf{k}[i] \,\|\, x, 1^\ell)$ to RO_1, for some $i \leq n$. Then, it is easy to see that there exists a predictor P which captures D setting bad by running D with access to an oracle, noting the queries, and returning as its guess set the set of queries made by D such that

$$\Pr[\mathrm{G}_5^{S,D}(\lambda)] - \Pr[\mathrm{G}_6^{S,D}(\lambda)] \leq \Pr[\mathrm{G}_6^{S,D}(\lambda) \text{ sets bad}] \leq \frac{v(\lambda)\cdot q(\lambda)}{2^\lambda} + \mathsf{Adv}^{\mathsf{m\text{-}pred}}_{S,P}(\lambda) \ .$$

Hence, for every $\lambda \in \mathbb{N}$,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{m\text{-}uce}}_{S,D,\mathsf{H}}(\lambda) &= \Pr[\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\lambda) \,|\, d=1] + \Pr[\mathrm{mUCE}^{S,D}_{\mathsf{H}}(\lambda) \,|\, d=0]] - 1 \\
&= \Pr[\mathrm{G}_1^{S,D}(\lambda)] - \Pr[\mathrm{G}_6^{S,D}(\lambda)] \\
&\leq \sum_{i=1}^{5} \Pr[\mathrm{G}_i^{S,D}(\lambda)] - \Pr[\mathrm{G}_{i+1}^{S,D}(\lambda)] \\
&\leq \mathsf{Adv}^{\mathsf{m\text{-}pred}}_{S,R}(\lambda) + \frac{2v(\lambda)\cdot q(\lambda) + v(\lambda)^2}{2^\lambda}
\end{aligned}$$

yielding Equation (5.4). ∎

A practical UCE construction

We consider a practical, heuristic instantiation for a family of functions assumed UCE secure. The UCE framework and security definitions are asymptotic, while real-world instantiations are going to be based on non-asymptotic hash functions, so we make

no formal claims about security. We ignore the security parameter and consider FOL families, so that we view K as taking no inputs and we view H as taking only a key and an input. One could consider instantiations based on cryptographic hash functions such as SHA256, but UCE security requires a keyed function, and SHA256 is not keyed. This suggests that we use the HMAC construction of [14, 86]. This is indeed our leading suggestion for a practical way to instantiate families assumed UCE secure.

Acknowledgements

This chapter contains material from Instantiating random oracles via UCEs, by Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi from Advances in Cryptology, CRYPTO 2013. I wish to thank Viet Tung Hoang for being an excellent co-author in this work.

Bibliography

[1] T. Acar, M. Belenkiy, M. Bellare, and D. Cash. Cryptographic agility and its relation to circular encryption. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 403–422. Springer, May 2010.

[2] A. Adya, W. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. *ACM SIGOPS Operating Systems Review*, 36(SI):1–14, 2002.

[3] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In *Proc. of USENIX LISA*, 2010.

[4] B. Applebaum. Key-dependent message security: Generic amplification and completeness. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 527–546. Springer, May 2011.

[5] 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, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, Aug. 2009.

[6] M. Backes, M. Dürmuth, and D. Unruh. OAEP is secure under key-dependent messages. In J. Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 506–523. Springer, Dec. 2008.

[7] 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.

[8] Z. Bar-Yossef, O. Reingold, R. Shaltiel, and L. Trevisan. Streaming computation of combinatorial objects. In *Computational Complexity, 2002. Proceedings. 17th IEEE Annual Conference on*, pages 133–142. IEEE, 2002.

[9] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 423–444. Springer, May 2010.

[10] C. Batten, K. Barr, A. Saraf, and S. Trepetin. pStore: A secure peer-to-peer backup system. *Unpublished report, MIT Laboratory for Computer Science*, 2001.

[11] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In C. Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619. Springer, Aug. 2006.

[12] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552. Springer, Aug. 2007.

[13] 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, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249. Springer, Dec. 2009.

[14] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, Aug. 1996.

[15] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, Oct. 1997.

[16] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In D. Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 360–378. Springer, Aug. 2008.

[17] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: The case of hashing and signing. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 216–233. Springer, Aug. 1994.

[18] M. Bellare, V. T. Hoang, and S. Keelveedhi. Instantiating random oracles via uces. Cryptology ePrint Archive, Report 2013/424, 2013. http://eprint.iacr.org/.

[19] M. Bellare, V. T. Hoang, and S. Keelveedhi. Instantiating random oracles via uces. In *Advances in Cryptology–CRYPTO 2013*, pages 398–415. Springer, 2013.

[20] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, May 2003.

[21] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, Dec. 2000.

[22] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, Nov. 1993.

[23] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, May 1994.

[24] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer, Dec. 2000.

[25] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006.

[26] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In B. K. Roy and W. Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, Feb. 2004.

[27] N. Bitansky and R. Canetti. On strong simulation and composable point obfuscation. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 520–537. Springer, Aug. 2010.

[28] Bitcasa. Inifinite storage. http://blog.bitcasa.com/tag/patented-de-duplication/.

[29] 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: 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75. Springer, Aug. 2003.

[30] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

[31] A. Boldyreva, D. Cash, M. Fischlin, and B. Warinschi. Foundations of non-malleable hash and one-way functions. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 524–541. Springer, Dec. 2009.

[32] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In D. Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 335–359. Springer, Aug. 2008.

[33] A. Boldyreva and M. Fischlin. Analysis of random oracle instantiation scenarios for OAEP and other practical schemes. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 412–429. Springer, Aug. 2005.

[34] A. Boldyreva and M. Fischlin. On the security of OAEP. In X. Lai and K. Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 210–225. Springer, Dec. 2006.

[35] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, May 2004.

[36] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision diffie-hellman. In D. Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer, Aug. 2008.

[37] Z. Brakerski and S. Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Aug. 2010.

[38] Z. Brakerski, S. Goldwasser, and Y. T. Kalai. Black-box circular-secure encryption beyond affine functions. In Y. Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 201–218. Springer, Mar. 2011.

[39] Z. Brakerski and G. Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 543–560. Springer, Aug. 2011.

[40] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In P. Rogaway, editor, *Advances in*

Cryptology – CRYPTO 2011, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, Aug. 2011.

[41] C. Brzuska, P. Farshim, and A. Mittelbach. Indistinguishability obfuscation and uces: The case of computationally unpredictable sources. Cryptology ePrint Archive, Report 2014/099, 2014. http://eprint.iacr.org/.

[42] 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, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 351–368. Springer, Apr. 2009.

[43] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, May 2001.

[44] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In B. S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer, Aug. 1997.

[45] R. Canetti and R. R. Dakdouk. Extractable perfectly one-way functions. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 449–460. Springer, July 2008.

[46] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, May 1998.

[47] R. Canetti, Y. T. Kalai, M. Varia, and D. Wichs. On symmetric encryption and point obfuscation. In D. Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 52–71. Springer, Feb. 2010.

[48] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 131–140. ACM Press, May 1998.

[49] Ciphertite. Ciphertite data backup. https://www.cyphertite.com/faq.php.

[50] J. Cooley, C. Taylor, and A. Peacock. ABS: the apportioned backup system. *MIT Laboratory for Computer Science*, 2004.

[51] L. P. Cox, C. D. Murray, and B. D. Noble. Pastiche: making backup cheap and easy. *SIGOPS Oper. Syst. Rev.*, 36:285–298, Dec. 2002.

[52] W. Dai. Crypto++ library. http://www.cryptopp.com.

[53] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[54] Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for practical applications. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer, Apr. 2009.

[55] J. Douceur, A. Adya, W. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pages 617–624. IEEE, 2002.

[56] Dropbox. Deduplication in Dropbox. https://dropbox.com.

[57] N. Firasta, M. Buxton, P. Jinbo, K. Nasri, and S. Kuo. Intel AVX: New frontiers in performance improvements and energy efficiency. *Intel Software Network*, 2009.

[58] Flud. The Flud backup system. http://flud.org/wiki/Architecture.

[59] 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.

[60] B. Fuller, A. O'Neill, and L. Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In R. Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 582–599. Springer, Mar. 2012.

[61] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49. IEEE, 2013.

[62] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *Proc. of FOCS (to appear)*, 2013.

[63] S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 467–476. ACM, 2013.

[64] B. Gladman. AES implementation. http://gladman.plushost.co.uk.

[65] GNUnet. A framework for secure peer-to-peer networking. https://gnunet.org/.

[66] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33:792–807, 1986.

[67] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[68] M. González. *Cryptography in the Presence of Key Dependent Messages*. PhD thesis, Florida Atlantic University, 2009.

[69] Google. Google Drive. http://drive.google.com/.

[70] V. Goyal, A. O'Neill, and V. Rao. Correlated-input secure hash functions. In Y. Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 182–200. Springer, Mar. 2011.

[71] 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/.

[72] S. Gueron. Advanced encryption standard (AES) instructions set. *Intel Corporation*, 25, 2008.

[73] S. Gueron. Advanced encryption standard (AES) instructions set. Intel, http://softwarecommunity.intel.com/articles/eng/3788.htm, 2010.

[74] I. Haitner and T. Holenstein. On the (im)possibility of key dependent encryption. In O. Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 202–219. Springer, Mar. 2009.

[75] S. Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In A. Canteaut and K. Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004: 5th International Conference in Cryptology in India*, volume 3348 of *Lecture Notes in Computer Science*, pages 315–327. Springer, Dec. 2004.

[76] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM CCS 11: 18th Conference on Computer and Communications Security*, pages 491–500. ACM Press, Oct. 2011.

[77] 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: 14th Conference on Computer and Communications Security*, pages 466–475. ACM Press, Oct. 2007.

[78] S. Halevi and P. Rogaway. A tweakable enciphering mode. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer, Aug. 2003.

[79] S. Halevi and P. Rogaway. A parallelizable enciphering mode. In T. Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, Feb. 2004.

[80] D. Harnik, B. Pinkas, and A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. *Security & Privacy, IEEE*, 8(6):40–47, 2010.

[81] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[82] D. Hofheinz and D. Unruh. Towards key-dependent message security in the standard model. In N. P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 108–126. Springer, Apr. 2008.

[83] J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In B. Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer, Apr. 2000.

[84] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *Journal of Cryptology*, 14(1):17–35, 2001.

[85] E. Kiltz, A. O'Neill, and A. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 295–313. Springer, Aug. 2010.

[86] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. IETF Internet Request for Comments 2104, Feb. 1997.

[87] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, Aug. 2002.

[88] C.-J. Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 257–271. Springer, Aug. 2002.

[89] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2), 1988.

[90] T. Malkin, I. Teranishi, and M. Yung. Efficient circuit-size independent public key encryption with KDM security. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 507–526. Springer, May 2011.

[91] L. Marques and C. Costa. Secure deduplication on mobile devices. In *Proceedings of the 2011 Workshop on Open Source and Design of Communication*, pages 19–26. ACM, 2011.

[92] 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, *Progress in Cryptology - INDOCRYPT 2004: 5th International Conference in Cryptology in India*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, Dec. 2004.

[93] I. Mironov, O. Pandey, O. Reingold, and G. Segev. Incremental deterministic public-key encryption. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 628–644. Springer, Apr. 2012.

[94] 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.

[95] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[96] K. G. Paterson and G. J. Watson. Plaintext-dependent decryption: A formal security treatment of SSH-CTR. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 345–361. Springer, May 2010.

[97] J. Pettitt. Content based hashing. http://cypherpunks.venona.com.

[98] A. Rahumed, H. Chen, Y. Tang, P. Lee, and J. Lui. A secure cloud backup system with assured deletion and version control. In *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*, pages 160–167. IEEE, 2011.

[99] T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, May 2011.

[100] P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *ACM CCS 02: 9th Conference on Computer and Communications Security*, pages 98–107. ACM Press, Nov. 2002.

[101] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In P. J. Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, Dec. 2004.

[102] P. Rogaway. Nonce-based symmetric encryption. In B. K. Roy and W. Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359. Springer, Feb. 2004.

[103] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 196–205. ACM Press, Nov. 2001.

[104] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, May / June 2006.

[105] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society Press, May 2000.

[106] M. Storer, K. Greenan, D. Long, and E. Miller. Secure data deduplication. In *Proceedings of the 4th ACM international workshop on Storage security and survivability*, pages 1–10. ACM, 2008.

[107] S. P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 61–77. Springer, Aug. 2003.

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

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

[110] D. Wichs. Barriers in cryptography with weak, correlated and leaky sources. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 111–126, New York, NY, USA, 2013. ACM.

[111] Z. Wilcox-O'Hearn. Convergent encryption reconsidered, 2011. http://www.mail-archive.com/cryptography@metzdowd.com/msg08949.html.

[112] Z. Wilcox-O'Hearn and B. Warner. Tahoe: The least-authority filesystem. In *Proceedings of the 4th ACM international workshop on Storage security and survivability*, pages 21–26. ACM, 2008.

[113] J. Xu, E.-C. Chang, and J. Zhou. Leakage-resilient client-side deduplication of encrypted data in cloud storage. Cryptology ePrint Archive, Report 2011/538, 2011. http://eprint.iacr.org/.