A preliminary version of this paper appears in EUROCRYPT 2020. This is the full version.

Security under Message-Derived Keys: Signcryption in iMessage

Mihir Bellare[1] Igors Stepanovs[2]

February 2020

Abstract

At the core of Apple's iMessage is a signcryption scheme that involves symmetric encryption of a message under a key that is derived from the message itself. This motivates us to formalize a primitive we call Encryption under Message-Derived Keys (EMDK). We prove security of the EMDK scheme underlying iMessage. We use this to prove security of the signcryption scheme itself, with respect to definitions of signcryption we give that enhance prior ones to cover issues peculiar to messaging protocols. Our provable-security results are quantitative, and we discuss the practical implications for iMessage.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@eng.ucsd.edu`. URL: `https://cseweb.ucsd.edu/~mihir/`. Supported in part by NSF grant CNS-1717640 and a gift from Microsoft.

[2] Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland. Email: `istepanovs@inf.ethz.ch`. URL: `https://sites.google.com/site/igorsstepanovs/`. Supported in part by grants of first author. Work done while at UCSD.

Contents

1 Introduction

Apple's iMessage app works across iOS (iPhone, iPad) and OS X (MacBook) devices. Laudably, it aims to provide end-to-end security. At its heart is a signcryption scheme.

The current scheme —we refer to the version in iOS 9.3 onwards, revised after the attacks of GGKMR [51] on the iOS 9.0 version— is of interest on two fronts. (1) *Applied*: iMessage encrypts (according to an Internet estimate) 63 quadrillion messages per year. It is important to determine whether or not the scheme provides the security expected by its users. (2) *Theoretical*: The scheme involves (symmetric) encryption of a message under a key that is derived from the message itself, an uncommon and intriguing technique inviting formalization and a foundational treatment.

Contributions in brief. *Signcryption theory*: We extend the prior Signcryption definitions of ADR [4] to capture elements particular to messaging systems, and give general results that simplify the analysis of the candidate schemes. *EMDK*: We introduce, and give definitions (syntax and security) for, Encryption under Message Derived Keys. *iMessage EMDK scheme*: We extract from iMessage an EMDK scheme and prove its security in the random-oracle model. *Composition and iMessage Signcryption*: We give a way to compose EMDK, PKE and signatures to get signcryption, prove it works, and thereby validate the iMessage signcryption scheme for appropriate parameter choices.

Background. By default, the iMessage chatting app encrypts communications between any two iMessage users. The encryption is end-to-end, under keys stored on the devices, meaning Apple itself cannot decrypt. In this way, iMessage joins Signal, WhatsApp and other secure messaging apps as a means to counter mass surveillance, but the cryptography used is quite different, and while the cryptography underlying Signal and WhatsApp, namely ratcheting, has received an extensive theoretical treatment [32, 21, 54, 73, 2, 55, 44], that underlying iMessage has not.

In 2016, Garman, Green, Kaptchuk, Miers and Rushanan (GGKMR) [51] gave chosen-ciphertext attacks on the then current, iOS 9 version, of iMessage that we will denote $\mathsf{iMsg1}$. Its encryption algorithm is shown on the left in Figure 1. In response Apple acknowledged the attack as CVE-2016-1788 [33], and revised the protocol for iOS 9.3. We'll denote this version $\mathsf{iMsg2}$. Its encryption algorithm is shown on the right in Figure 1. It has been stable since iOS 9.3. It was this revision that, for the specific purpose of countering the GGMKR attack, introduced (symmetric) encryption with message-derived keys: message M at line 4 is encrypted under a key K derived, via lines 1–3, from M itself. The question we ask is, does the fix work?

Identifying the goal. To meaningfully answer the above question we must first, of course, identify the formal primitive and security goal being targeted. Neither Apple's iOS Security Guide [5], nor GGKMR [51], explicitly do so. We suggest that it is signcryption. Introduced by Zheng [84], signcryption aims to simultaneously provide privacy of the message (under the receiver's public encryption key) and authenticity (under the sender's secret signing key), and can be seen as the asymmetric analogue of symmetric authenticated encryption. A formalization was given by An, Dodis and Rabin (ADR) [4]. They distinguish between outsider security (the adversary is not one of the users) and the stronger insider security (the adversary could be a sender or receiver).

Identifying the iMessage goal as signcryption gives some perspective on, and understanding of, the schemes and history. The iMessage schemes can be seen as using some form of ADR's Encrypt-then-Sign ($\mathcal{E}t\mathcal{S}$) method. The $\mathsf{iMsg1}$ scheme turns out to be a simple scheme from ADR [4]. It may be outsider-secure, but ADR give an attack that shows it is not insider secure. (The adversary queries the sender encryption oracle to get a ciphertext $((C_1, C_2), S)$, substitutes S with a signature S' of $H = \mathsf{SHA1}(C_1 \| C_2)$ under its own signing key, which it can do as an insider, and then queries this modified ciphertext to the recipient decryption oracle to get back the message underlying the

```
iMsg1.Enc($pk_r, sk_s, M$)
1. $K \leftarrow\$ \{0,1\}^{128}$
2. $C_1 \leftarrow \mathsf{AES\text{-}CTR.Enc}(K, M)$
3. $C_2 \leftarrow \mathsf{RSA\text{-}OAEP.Enc}(pk_r, K)$
4. $H \leftarrow \mathsf{SHA1}(C_1 \| C_2)$
5. $S \leftarrow \mathsf{EC\text{-}DSA.Sign}(sk_s, H)$
6. Return $((C_1, C_2), S)$
```

```
iMsg2.Enc($pk_r, sk_s, M$)
1. $L \leftarrow\$ \{0,1\}^{88}$
2. $h \leftarrow \mathsf{HMAC}(L, pk_s \| pk_r \| M)[1..40]$
3. $K \leftarrow L \| h$
4. $C_1 \leftarrow \mathsf{AES\text{-}CTR.Enc}(K, M)$
5. $C_2 \leftarrow \mathsf{RSA\text{-}OAEP.Enc}(pk_r, K)$
6. $H \leftarrow \mathsf{SHA1}(C_1 \| C_2)$
7. $S \leftarrow \mathsf{EC\text{-}DSA.Sign}(sk_s, H)$
8. Return $((C_1, C_2), S)$
```

Figure 1: Encryption in iMsg1 (left) and iMsg2 (right). Here pk_r is the recipient's public RSA encryption key, sk_s is the sender's ECDSA secret signing key and pk_s is the sender's ECDSA public verification key. Our analysis and proofs consider general schemes of which the above emerge as instantiations corresponding to particular choices of primitives and parameters.

original ciphertext.) The GGKMR [51] attack on iMsg1 is a clever improvement and real-world rendition of the ADR attack. That Apple acknowledged the GGKMR attack, and modified the scheme to protect against it, indicates that they want insider security, not just outsider security, for their modified iMsg2 scheme. So the question becomes whether this goal is achieved.

Signcryption theory extended. We could answer the above question relative to ADR's (existing) definitions of insider-secure signcryption, but we do more, affirming the iMsg2 signcryption scheme under stronger definitions that capture elements particular to messaging systems, making our results of more applied value.

When you send an iMessage communication to Alice, it is encrypted to *all* her devices (her iPhone, MacBook, iPad, ...), so that she can chat seamlessly across them. To capture this, we *enhance signcryption syntax*, making the encryption algorithm multi-recipient. (It takes not one, but a list of receiver public encryption keys.) We also allow associated data as in symmetric authenticated encryption [75].

We give, like in prior work [4], a privacy definition (priv) and an authenticity definition (auth); but, unlike prior work, we also give a strong, unified definition (sec) that implies auth+priv. We show that (under certain conditions) sec is implied by auth+priv, mirroring analogous results for symmetric authenticated encryption [24, 17]. Proving that a scheme satisfies sec (the definition more intuitively capturing the practical setting) now reduces to the simpler tasks of separately showing it satisfies auth and priv. These definitions and results are for both insider and outsider security, and parameterized by choices of *relaxing relations* that allow us to easily capture variants reflecting issues like plaintext or ciphertext integrity [16], gCCA2 [4] and RCCA [28].

EMDK definitions. Recall that a scheme for conventional symmetric encryption specifies a key-generation algorithm that is run once, a prioi, to return a key k; the encryption algorithm then takes k and message m to return a ciphertext. In our definition of a scheme for (symmetric) Encryption under Message-Derived Keys (EMDK), there is no dedicated key-generation algorithm. Encryption algorithm EMDK.Enc takes only a message m, returning both a key k and a ciphertext c, so that k may depend on m. Decryption algorithm EMDK.Dec takes k —in the overlying signcryption scheme, this is communicated to the receiver via asymmetric encryption— and c to return either m or $\perp$.

We impose two security requirements on an EMDK scheme. (1) The first, called ae, adapts the authenticated encryption requirement of symmetric encryption [75]. (Our game formalizing ae is in

Figure 8.) (2) The second, called rob, is a form of robustness or wrong-key detection [1, 27, 46, 47]. (Our game formalizing rob is also in Figure 8.) Of course one may define many other and alternative security goals for EMDK, so why these? We have focused on these simply because they suffice for our results.

EMDK is different from both (Symmetric) Encryption of Key-Dependent Messages (EKDM) [23, 26] and (Symmetric) Encryption secure against Related-Key Attack (ERKA) [15]. To begin with, these definitions apply to *syntactically different objects*. Namely, both EKDM and ERKA are security metrics for the standard symmetric encryption syntax where the encryption algorithm takes a key and message as input and returns a ciphertext, while in EMDK the encryption algorithm takes only a message and itself produces a key along with the ciphertext. (Note that the latter is also different from the syntax of a Key-Encapsulation mechanism, where encryption does produce a key and ciphertext, but takes no input message.) These syntactic differences make comparison moot, but one can still discuss intuitively how the security requirements relate. In the security games for EKDM there is an honestly and randomly chosen target key k, and challenge messages to be encrypted may depend on k, but in our security games for EMDK, the key is not chosen honestly and could depend on the message being encrypted. In ERKA also, like EKDM but unlike EMDK, a target key k is chosen honestly and at random. One can now have the game apply the encryption algorithm under a key k' derived from k, but this does not capture the encryption algorithm not taking a key as input but itself producing it as a function of the message, as in EKDM.

Deconstructing iMessage. Equipped with the above, we show how to cast the iMsg2 signcryption scheme as the result of a general transform (that we specify and call IMSG-SC) on a particular EMDK scheme (that we specify) and some standard auxiliary primitives (that we also specify). In Section 5, we prove that IMSG-SC works, reducing insider security (priv, auth, sec) of the signcryption scheme to the security of the constituents, leaving us with what is the main technical task, namely showing security of the EMDK scheme.

In more detail, IMSG-SC takes a scheme EMDK for encryption under message-derived keys, a public-key encryption scheme PKE and a digital signature scheme DS to return a signcryption scheme $\mathsf{SC} = \mathsf{IMSG\text{-}SC}[\mathsf{EMDK}, \mathsf{PKE}, \mathsf{DS}]$. (In the body of the paper, this is done in two steps, with a multi-recipient public-key encryption scheme [13] as an intermediate point, but for simplicity we elide this here.) Both iMessage signcryption schemes (i.e. iMsg1 and iMsg2) can be seen as results of this transform. The two make the same choices of PKE and DS, namely RSA-OAEP and EC-DSA respectively, differing only in their choice of EMDK, which for iMsg1 is a trivial scheme that we call the basic scheme, and for iMsg2 a more interesting scheme that we denote $\mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$ and discuss below. Our Section 5 result is that signcryption scheme $\mathsf{SC} = \mathsf{IMSG\text{-}SC}[\mathsf{EMDK}, \mathsf{PKE}, \mathsf{DS}]$ provides insider security (priv, auth, sec) assuming ae- and rob-security of EMDK and under standard assumptions on PKE and DS.

EMDK results. In Figure 10 we specify an EMDK scheme $\mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$ constructed from a given function family F and a given, ordinary one-time (assumed deterministic) symmetric encryption scheme SE. Setting F to HMAC and SE to AES-CTR recovers the EMDK scheme underlying iMsg2 signcryption. This EMDK scheme captures the heart of iMsg2 signcryption, namely lines 1–4 of the right side of Figure 1.

The security analysis of $\mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$ is somewhat complex. We prove ae-security of this EMDK scheme assuming F is a random oracle and SE has the following properties: one-time IND-CPA privacy, a property we define called uniqueness, and partial key recovery security. The latter strengthens key recovery security to say that, not only is it hard to recover the key, but it is hard to recover even a prefix, of a certain prescribed length, of this key. We prove rob-security of the

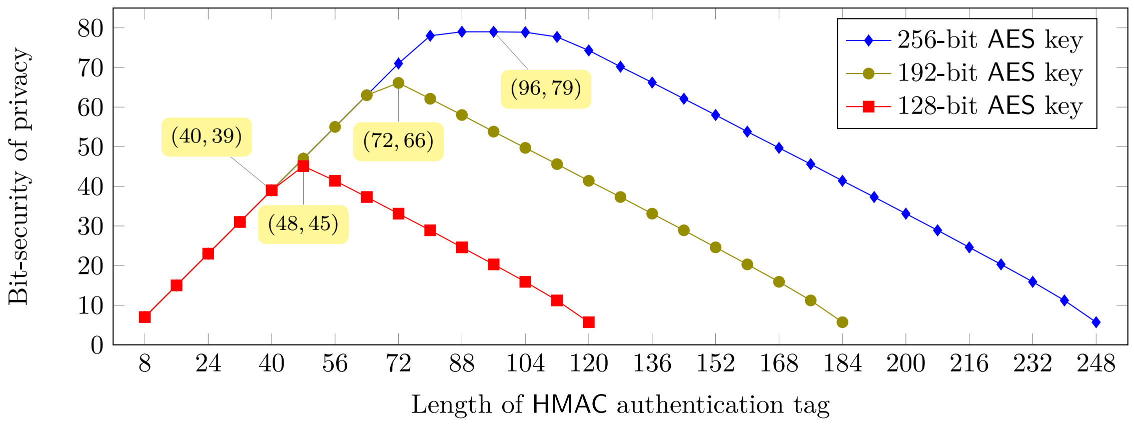

Figure 2: Lower bounds for the bit-security of privacy achieved by iMessage, depending on the key size of AES-CTR and the length of the authentication tag returned by HMAC. iMessage 10 uses 128-bit AES key and 40-bit long HMAC authentication tag, and hence guarantees at least 39 bits of security for privacy. (Any choice of parameters guarantees 71 bits of security for authenticity.)

EMDK scheme assuming F is a random oracle and SE satisfies uniqueness and weak robustness. The properties assumed of SE appear to be true for the AES-CTR used in iMessage, and could be shown in idealized models.

PRACTICAL IMPLICATIONS FOR IMESSAGE. What we have proved is that iMsg2 signcryption is secure in principle, in the sense that the underlying template is sound. (That is, the signcryption scheme given by our IMSG-SC transform is secure assuming the underlying primitives are secure.) For the practical implications, we must consider the quantitative security guaranteed by our theorems based on the particular choices of parameters and primitives made in iMsg2 signcryption scheme. Here, things seem a bit borderline, because iMsg2 signcryption has made some specific parameter choices that seem dangerous. Considering again the right side of Figure 1, the 128-bit AES key K at line 3 has only 88 bits of entropy —all the entropy is from the choice of L at line 1— which is not only considered small in practice but also is less than for iMsg1. (On the left side of the Figure we see that line 1 selects an AES key K with the full 128 bits of entropy.) Also the tag h produced at line 2 of the right-hand-side of the Figure is only 40 bits, shorter than recommended lengths for authentication tags. To estimate the impact of these choices, we give concrete attacks on the scheme. They show that the bounds in our theorems are tight, but do not contradict our provable-security results.

Numerical estimates based on our provable-security results say that iMessage 10 guarantees at least 39 bits of security for privacy, and 71 bits of security for authenticity, if HMAC and AES are modeled as ideal primitives. Fig. 2 shows the guaranteed bit-security of privacy for different choices of AES key length and HMAC tag length. For the small parameter choices made in iMsg2 signcryption, the attacks do approach feasibility in terms of computational effort, but we wouldn't claim they are practical, for two reasons. First, they only violate the very stringent security goals that are the target of our proofs. Second, following the GGKMR [51] attacks, Apple has implemented decryption-oracle throttling that will also curtail our attacks.

Still, ideally, a practical scheme would implement cryptography that meets even our stringent security goals without recourse to extraneous measures like throttling. We suggest that parameter and primitive choices in iMessage signcryption be revisited, for if they are chosen properly, our results do guarantee that the scheme provides strong security properties.

Discussion. When a new primitive (like EMDK) is defined, the first question of a theoretical cryptographer is often, does it exist, meaning, can it be built, and under what assumptions? At least in the random-oracle model [18] in which our results are shown, it is quite easy to build, under standard assumptions, an EMDK scheme that provides the ae+rob-security we define, and we show such a scheme in Figure 9. The issue of interest for us is less existence (to build some secure EMDK scheme) and more the security of the *particular* IMSG-EMDK[F, SE] scheme underlying iMsg2 signcryption. The motivation is mainly applied, stemming from this scheme running in security software (iMessage) that is used by millions.

But, one may then ask, WHY did Apple use their (strange) EMDK scheme instead of one like that in Figure 9, which is simpler and provable under weaker assumptions? We do not know. In that vein, one may even ask, why did Apple use EMDK at all? The literature gives Signcryption schemes that are efficient and based on standard assumptions. Why did they not just take one of them? Again, we do not know for sure, but we can speculate. The EMDK-based template that we capture in our IMSG-SC transform provides *backwards decryption compatibility*; an iMsg1 implementation can decrypt an iMsg2 ciphertext. (Of course, security guarantees revert to those of the iMsg1 scheme under such usage, but this could be offset by operational gains.) Moving to an entirely new signcryption scheme would *not* provide this backwards compatibility. But we stress again that this is mere speculation; we did not find any Apple documents giving reasons for their choices.

Related work. We have discussed some related work above. However, signcryption is a big research area with a lot of work. We overview this in Appendix A.

2 Preliminaries

In Appendix D we provide the following standard definitions. We state syntax, correctness and security definitions for function families, symmetric encryption, digital signatures, public-key encryption, and multi-recipient public-key encryption. We define the random oracle model, the ideal cipher model, and provide the birthday attack bounds. In this section we introduce the basic notation and conventions we use throughout the paper.

Basic notation and conventions. Let $\mathbb{N} = \{1, 2, \ldots\}$ be the set of positive integers. For $i \in \mathbb{N}$ we let $[i]$ denote the set $\{1, \ldots, i\}$. If X is a finite set, we let $x \leftarrow\!\!\$\ X$ denote picking an element of X uniformly at random and assigning it to x. Let ε denote the empty string. By $x \,\|\, y$ we denote the concatenation of strings x and y. If $x \in \{0,1\}^*$ is a string then $|x|$ denotes its length, $x[i]$ denotes its i-th bit, and $x[i..j] = x[i] \ldots x[j]$ for $1 \le i \le j \le |x|$. If mem is a table, we use $\mathsf{mem}[i]$ to denote the element of the table that is indexed by i. We use a special symbol $\perp$ to denote an empty table position; we also return it as an error code indicating an invalid input to an algorithm or an oracle, including invalid decryption. We assume that adversaries never pass $\perp$ as input to their oracles.

Uniquely decodable encoding. We write $\langle a, b, \ldots \rangle$ to denote a string that is a uniquely decodable encoding of $a, b, \ldots$, where each of the encoded elements can have an arbitrary type (e.g. string or set). For any $n \in \mathbb{N}$ let $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ be two sequences of elements such that for each $i \in [n]$ the following holds: either $x_i = y_i$, or both x_i and y_i are strings of the same length. Then we require that $|\langle x_1, \ldots, x_n \rangle| = |\langle y_1, \ldots, y_n \rangle|$, and that $\langle x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n \rangle \oplus \langle x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n \rangle = \langle x_1, \ldots, x_{i-1}, (x_i \oplus y_i), x_{i+1}, \ldots, x_n \rangle$ for all $i \in [n]$.

Algorithms and adversaries. Algorithms may be randomized unless otherwise indicated. Running time is worst case. If A is an algorithm, we let $y \leftarrow A(x_1, \ldots ; r)$ denote running A with random coins r on inputs $x_1, \ldots$ and assigning the output to y. We let $y \leftarrow\!\!\$\; A(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \ldots; r)$. We let $[A(x_1, \ldots)]$ denote the set of all possible outputs of A when invoked with inputs $x_1, \ldots$. The instruction $\mathbf{abort}(x_1, \ldots)$ is used to immediately halt the algorithm with output $(x_1, \ldots)$. Adversaries are algorithms.

<u>Security games and reductions.</u> We use the code based game playing framework of [20]. (See Fig. 5 for an example.) We let $\Pr[\mathrm{G}]$ denote the probability that game G returns true. In the security reductions, we omit specifying the running times of the constructed adversaries when they are roughly the same as the running time of the initial adversary.

<u>Implicit initialization values.</u> In algorithms and games, uninitialized integers are assumed to be initialized to 0, Booleans to false, strings to the empty string, sets to the empty set, and tables are initially empty.

<u>Bit-security of cryptographic primitives.</u> Let prim be any cryptographic primitive, and let sec be any security notion defined for this primitive. We say that prim has n bits of security with respect to sec (or n bits of sec-security) if for every adversary $\mathcal{A}$ that has advantage $\epsilon_{\mathcal{A}}$ and runtime $T_{\mathcal{A}}$ against sec-security of prim it is true that $\epsilon_{\mathcal{A}}/T_{\mathcal{A}} < 2^{-n}$. In other words, if there exists an adversary $\mathcal{A}$ with advantage $\epsilon_{\mathcal{A}}$ and runtime $T_{\mathcal{A}}$ against sec-security of prim, then prim has at most $-\log_2(\epsilon_{\mathcal{A}}/T_{\mathcal{A}})$ bits of security with respect to sec. This is the folklore definition of bit-security for cryptographic primitives. Micciancio and Walter [68] recently proposed an alternative definition for bit-security.

<u>Bit-security lower bounds.</u> Let $\mathcal{BS}(\mathsf{prim}, \mathsf{sec})$ denote the bit-security of cryptographic primitive prim with respect to security notion sec. Consider any security reduction showing $\mathsf{Adv}^{\mathsf{sec}}_{\mathsf{prim}}(\mathcal{A}) \leq \sum_i \mathsf{Adv}^{\mathsf{sec_i}}_{\mathsf{prim}_i}(\mathcal{B}_i^{\mathcal{A}})$ by constructing for any adversary $\mathcal{A}$ and for each i a new adversary $\mathcal{B}_i^{\mathcal{A}}$ with runtime roughly $T_{\mathcal{A}}$. Then we can lower bound the bit-security of prim with respect to sec as

$$\mathcal{BS}(\mathsf{prim}, \mathsf{sec}) = \min_{\forall \mathcal{A}} -\log_2\left(\frac{\epsilon_{\mathcal{A}}}{T_{\mathcal{A}}}\right) \geq \min_{\forall \mathcal{A}} -\log_2\left(\frac{\sum_i \mathsf{Adv}^{\mathsf{sec_i}}_{\mathsf{prim}_i}(\mathcal{B}_i^{\mathcal{A}})}{T_{\mathcal{A}}}\right) \geq -\log_2\left(\sum_i 2^{-\mathcal{BS}(\mathsf{prim}_i, \mathsf{sec}_i)}\right).$$

3 Signcryption

In this section we define syntax, correctness and security notions for multi-recipient signcryption schemes. We assume that upon generating any signcryption key pair (pk, sk), it gets associated to some identity id. This captures a system where users can indepenently generate their cryptographic keys prior to registering them with a public-key infrastructure. We require that all identities are distinct values in $\{0,1\}^*$. Depending on the system, each identity id serves as a label that uniquely identifies a device or a user. Note that pk cannot be used in place of the identity, because different devices can happen to use the same public keys (either due to generating the same key pairs by chance, or due to maliciously claiming someone's else public key). We emphasize that our syntax is not meant to capture identity-based signcryption, where a public key would have to depend on the identity. In Appendix A we provide an extensive summary of prior work on signcryption.

We focus on authenticity and privacy of signcryption in the *insider* setting, meaning that the adversary is allowed to adaptively compromise secret keys of any identities as long as that does not enable the adversary to trivially win the security games. Our definitions can also capture the *outsider* setting by considering limited classes of adversaries. We define our security notions with respect to *relaxing relations*. This allows us to capture a number of weaker security notions in a fine-grained way, by choosing an appropriate relaxing relation in each case. In Appendix C we define a *combined* security notion for signcryption that simultaneously encompasses authenticity

```
$\pi \leftarrow\$ \mathsf{SC.Setup}$
$(pk, sk) \leftarrow\$ \mathsf{SC.Kg}(\pi)$
$\mathcal{C} \leftarrow\$ \mathsf{SC.SigEnc}(\pi, id_s, pk_s, sk_s, \mathcal{R}, m, ad)$
$m \leftarrow \mathsf{SC.VerDec}(\pi, id_s, pk_s, id_r, pk_r, sk_r, c, ad)$
```

Figure 3: Syntax of the constituent algorithms of signcryption scheme SC.

```
$\underline{\mathsf{R_m.Vf}(z_0, z_1)}$
$(x_0, y_0) \leftarrow z_0$ ; $(x_1, y_1) \leftarrow z_1$
Return $x_0 = x_1$

$\underline{\mathsf{R_{id}.Vf}(z_0, z_1)}$
Return $z_0 = z_1$
```

Figure 4: Relaxing relations $\mathsf{R_m}$ and $\mathsf{R_{id}}$.

and privacy, and prove that it is equivalent to the separate notions under certain conditions.

Multi-recipient signcryption schemes. A multi-recipient signcryption scheme SC specifies algorithms SC.Setup, SC.Kg, SC.SigEnc, SC.VerDec, where SC.VerDec is deterministic. Associated to SC is an identity space SC.ID. The setup algorithm SC.Setup returns public parameters π. The key generation algorithm SC.Kg takes π to return a key pair (pk, sk), where pk is a public key and sk is a secret key. The signcryption algorithm SC.SigEnc takes π, sender's identity $id_s \in \mathsf{SC.ID}$, sender's public key pk_s, sender's secret key sk_s, a set $\mathcal{R}$ of pairs (id_r, pk_r) containing recipient identities and public keys, a plaintext $m \in \{0,1\}^*$, and associated data $ad \in \{0,1\}^*$ to return a set $\mathcal{C}$ of pairs (id_r, c), each denoting that signcryption ciphertext c should be sent to the recipient with identity id_r. The unsigncryption algorithm SC.VerDec takes π, sender's identity id_s, sender's public key pk_s, recipient's identity id_r, recipient's public key pk_r, recipient's secret key sk_r, signcryption ciphertext c, and associated data ad to return $m \in \{0,1\}^* \cup \{\perp\}$, where $\perp$ indicates a failure to recover plaintext. The syntax used for the constituent algorithms of SC is summarized in Fig. 3.

Correctness of signcryption. The correctness of a signcryption scheme SC requires that for all $\pi \in [\mathsf{SC.Setup}]$, all $n \in \mathbb{N}$, all $(pk_0, sk_0), \ldots, (pk_n, sk_n) \in [\mathsf{SC.Kg}(\pi)]$ all $id_0 \in \mathsf{SC.ID}$, all *distinct* $id_1, \ldots, id_n \in \mathsf{SC.ID}$, all $m \in \{0,1\}^*$, and all $ad \in \{0,1\}^*$ the following conditions hold. Let $\mathcal{R} = \{(id_i, pk_i)\}_{1 \le i \le n}$.We require that for all $\mathcal{C} \in [\mathsf{SC.SigEnc}(\pi, id_0, pk_0, sk_0, \mathcal{R}, m, ad)]$: (i) $|\mathcal{C}| = |\mathcal{R}|$; (ii) for each $i \in \{1, \ldots, n\}$ there exists a unique $c \in \{0,1\}^*$ such that $(id_i, c) \in \mathcal{C}$; (iii) for each $i \in \{1, \ldots, n\}$ and each c such that $(id_i, c) \in \mathcal{C}$ we have $m = \mathsf{SC.VerDec}(\pi, id_0, pk_0, id_i, pk_i, sk_i, c, ad)$.

Relaxing relations. A relaxing relation $\mathsf{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ is a set containing pairs of arbitrary strings. Associated to a relaxing relation R is a membership verification algorithm R.Vf that takes inputs $z_0, z_1 \in \{0,1\}^*$ to return a decision in $\{\mathsf{true}, \mathsf{false}\}$ such that $\forall z_0, z_1 \in \{0,1\}^*: \mathsf{R.Vf}(z_0, z_1) = \mathsf{true}$ iff $(z_0, z_1) \in \mathsf{R}$. We will normally define relaxing relations by specifying their membership verification algorithms. Two relaxing relations that will be used throughout the paper are defined in Fig. 4.

We define our security notions for signcryption with respect to relaxing relations. Relaxing relations are used to restrict the queries that an adversary is allowed to make to its unsigncryption oracle. The choice of different relaxing relations can be used to capture a variety of different security notions for signcryption in a fine-grained way. We will use relaxing relations $\mathsf{R_{id}}$ and $\mathsf{R_m}$ to capture strong vs. standard authenticity (or unforgeability) of signcryption, and IND-CCA vs. RCCA [28, 53] style indistinguishability of signcryption. In Section 5.3 we will aso define unforgeability of digital signatures with respect to relaxing relations, allowing to capture standard

```
Games $\mathrm{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}}$
$\pi \leftarrow\!\!\$\ \mathsf{SC.Setup}$ ; $\mathcal{F}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\textsc{SigEnc},\textsc{VerDec}}(\pi)$ ; Return $\mathsf{win}$

$\underline{\textsc{NewH}(id)}$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(pk, sk) \leftarrow\!\!\$\ \mathsf{SC.Kg}(\pi)$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $pk$

$\underline{\textsc{NewC}(id, pk, sk)}$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $\mathsf{true}$

$\underline{\textsc{Exp}(id)}$
If not $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$

$\underline{\textsc{SigEnc}(id_s, \mathcal{I}, m, ad)}$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) then return $\bot$
$\mathcal{R} \leftarrow \emptyset$ ; For each $id \in \mathcal{I}$ do $\mathcal{R} \leftarrow \mathcal{R} \cup \{(id, \mathsf{pk}[id])\}$
$\mathcal{C} \leftarrow\!\!\$\ \mathsf{SC.SigEnc}(\pi, id_s, \mathsf{pk}[id_s], \mathsf{sk}[id_s], \mathcal{R}, m, ad)$
For each $(id_r, c) \in \mathcal{C}$ do $Q \leftarrow Q \cup \{((id_s, id_r, m, ad), c)\}$
Return $\mathcal{C}$

$\underline{\textsc{VerDec}(id_s, id_r, c, ad)}$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return $\bot$
$m \leftarrow \mathsf{SC.VerDec}(\pi, id_s, \mathsf{pk}[id_s], id_r, \mathsf{pk}[id_r], \mathsf{sk}[id_r], c, ad)$ ; If $m = \bot$ then return $\bot$
$z_0 \leftarrow ((id_s, id_r, m, ad), c)$ ; If $\exists z_1 \in Q$: $\mathsf{R.Vf}(z_0, z_1)$ then return $m$
$\mathsf{cheated} \leftarrow \mathsf{exp}[id_s]$ ; If not $\mathsf{cheated}$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $m$
```

Figure 5: Game defining authenticity of signcryption scheme SC with respect to relaxing relation R.

and strong unforgeability notions in a unified way.

<u>AUTHENTICITY OF SIGNCRYPTION.</u> Consider game $\mathrm{G}^{\mathsf{auth}}$ of Fig. 5 associated to a signcryption scheme SC, a relaxing relation R and an adversary $\mathcal{F}$. The advantage of adversary $\mathcal{F}$ in breaking the AUTH-security of SC with respect to R is defined as $\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}}(\mathcal{F}) = \Pr[\mathrm{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}}]$. Adversary $\mathcal{F}$ has access to oracles NEWH, NEWC, EXP, SIGENC, and VERDEC. The oracles can be called in any order. Oracle NEWH generates a key pair for a new <u>h</u>onest identity *id*. Oracle NEWC associates a key pair (pk, sk) of adversary's choice to a new <u>c</u>orrupted identity *id*; it permits malformed keys, meaning *sk* should not necessarily be a valid secret key that matches with *pk*. Oracle EXP can be called to expose the secret key of any identity. The game maintains a table exp to mark which identities are exposed; all corrupted identities that were created by calling oracle NEWC are marked as exposed right away. The signcryption oracle SIGENC returns ciphertexts produced by sender identity id_s to each of the recipient identities contained in set $\mathcal{I}$, encrypting message m with associated data *ad*. Oracle VERDEC returns the plainext obtained as the result of unsigncrypting the ciphertext c sent from sender id_s to recipient id_r, with associated data *ad*. The goal of adversary $\mathcal{F}$ is to forge a valid signcryption ciphertext, and query it to oracle VERDEC. The game does not let adversary win by querying oracle VERDEC with a forgery that was produced for an exposed sender identity id_s, since the adversary could have trivially produced a valid ciphertext due to its knowledge of the sender's secret key. Certain choices of relaxing relation R can lead to another trivial attack.

A choice of relaxing relation for authenticity. When adversary $\mathcal{F}$ in game $\mathrm{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}}$ calls oracle SigEnc on inputs $id_s, \mathcal{I}, m, ad$, then for each ciphertext c produced for a recipient $id_r \in \mathcal{I}$ the game adds a tuple $((id_s, id_r, m, ad), c)$ to set Q. This set is then used inside oracle VerDec. Oracle VerDec constructs $z_0 = ((id_s, id_r, m, ad), c)$ and prevents the adversary from winning the game if $\mathsf{R}.\mathsf{Vf}(z_0, z_1)$ is true for any $z_1 \in Q$. If the relaxing relation is empty (meaning $\mathsf{R} = \emptyset$ and hence $\mathsf{R}.\mathsf{Vf}(z_0, z_1) = \mathsf{false}$ for all $z_0, z_1 \in \{0,1\}^*$) then an adversary is allowed to trivially win the game by calling oracle SigEnc and claiming any of the resulting ciphertexts as a forgery (without changing the sender and recipient identities). Let us call this a "ciphertext replay" attack.

In order to capture a meaningful security notion, the AUTH-security of SC should be considered with respect to a relaxing relation that prohibits the above trivial attack. The strongest such security notion is achieved by considering AUTH-security of SC with respect to the relaxing relation R_{id} that is defined in Fig. 4; this relaxing relation prevents *only* the ciphertext replay attack. The resulting security notion captures the strong authenticity (or unforgeability) of signcryption. Alternatively, one could think of this notion as capturing the ciphertext integrity of signcryption.

Note that a relaxing relation R prohibits the ciphertext replay attack iff $\mathsf{R}_{\mathsf{id}} \subseteq \mathsf{R}$. Now consider the relaxing relation R_{m} as defined in Fig. 4; it is a proper superset of R_{id}. The AUTH-security of SC with respect to R_{m} captures the standard authenticity (or unforgeability, or plaintext integrity) of signcryption. The resulting security notion does not let adversary win by merely replaying an encryption of (m, ad) from id_s to id_r for any fixed (id_s, id_r, m, ad), even if the adversary can produce a new ciphertext that was not seen before.

Capturing outsider authenticity. Game $\mathrm{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}}$ captures the authenticity of SC in the *insider* setting, because it allows adversary to win by producing a forgery from an honest sender identity to an *exposed* recipient identity. This, in particular, implies that SC assures non-repudiation, meaning that the sender cannot deny the validity of a ciphertext it sent to a recipient (since the knowledge of the recipient's secret key does not help to produce a forgery). In contrast, the *outsider* authenticity only requires SC to be secure when both the sender and the recipient are honest. Our definition can capture the notion of outsider authenticity by considering a class of *outsider* adversaries that never query $\text{VerDec}(id_s, id_r, c, ad)$ when $\mathsf{exp}[id_r] = \mathsf{true}$.

Privacy of signcryption. Consider game $\mathrm{G}^{\mathsf{priv}}$ of Fig. 6 associated to a signcryption scheme SC, a relaxing relation R and an adversary $\mathcal{D}$. The advantage of adversary $\mathcal{D}$ in breaking the PRIV-security of SC with respect to R is defined as $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}) = 2\Pr[\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R},\mathcal{D}}] - 1$. The game samples a challenge bit $b \in \{0,1\}$, and the adversary is required to guess it. Adversary $\mathcal{D}$ has access to oracles NewH, NewC, Exp, LR, and VerDec. The oracles can be called in any order. Oracles NewH, NewC, and Exp are the same as in the authenticity game (with the exception of oracle Exp also checking table ch, which is explained below). Oracle LR encrypts challenge message m_b with associated data ad, produced by sender identity id_s to each of the recipient identities contained in set $\mathcal{I}$. Oracle LR aborts if $m_0 \neq m_1$ and if the recipient set $\mathcal{I}$ contains an identity id_r that is exposed. Otherwise, the adversary would be able to trivially win the game by using the exposed recipient's secret key to decrypt a challenge ciphertext produced by this oracle. If $m_0 \neq m_1$ and none of the recipient identities is exposed, then oracle LR uses table ch to mark each of the recipient identities; the game will no longer allow to expose any of these identities by calling oracle Exp. Oracle VerDec returns the plaintext obtained as the result of unsigncrypting the ciphertext c sent from id_s to id_r with associated data ad. We discuss the choice of a relaxing relation R below. However, note that oracle LR updates the set Q (used by relaxing relation) only when $m_0 \neq m_1$. This is because the output of LR does not depend on the challenge bit when $m_0 = m_1$, and hence such queries should not affect the set of prohibited queries to oracle VerDec.

```
Game $G^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R},\mathcal{D}}$
$b \leftarrow_{\$} \{0,1\}$ ; $\pi \leftarrow_{\$} \mathsf{SC.Setup}$ ; $b' \leftarrow_{\$} \mathcal{D}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\textsc{LR},\textsc{VerDec}}(\pi)$ ; Return $b' = b$

$\textsc{NewH}(id)$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(pk, sk) \leftarrow_{\$} \mathsf{SC.Kg}(\pi)$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $pk$

$\textsc{NewC}(id, pk, sk)$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $\mathsf{true}$

$\textsc{Exp}(id)$
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\bot$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$

$\textsc{LR}(id_s, \mathcal{I}, m_0, m_1, ad)$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) or $|m_0| \neq |m_1|$ then return $\bot$
If $m_0 \neq m_1$ then
    If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\bot$
    For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{R} \leftarrow \emptyset$ ; For each $id \in \mathcal{I}$ do $\mathcal{R} \leftarrow \mathcal{R} \cup \{(id, \mathsf{pk}[id])\}$
$\mathcal{C} \leftarrow_{\$} \mathsf{SC.SigEnc}(\pi, id_s, \mathsf{pk}[id_s], \mathsf{sk}[id_s], \mathcal{R}, m_b, ad)$
For each $(id_r, c) \in \mathcal{C}$ do
    If $m_0 \neq m_1$ then
        $Q \leftarrow Q \cup \{((id_s, id_r, m_0, ad), c)\}$
        $Q \leftarrow Q \cup \{((id_s, id_r, m_1, ad), c)\}$
Return $\mathcal{C}$

$\textsc{VerDec}(id_s, id_r, c, ad)$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return ($\bot$, "init")
$m \leftarrow \mathsf{SC.VerDec}(\pi, id_s, \mathsf{pk}[id_s], id_r, \mathsf{pk}[id_r], \mathsf{sk}[id_r], c, ad)$
If $m = \bot$ then return ($\bot$, "dec")
$z_0 \leftarrow ((id_s, id_r, m, ad), c)$ ; If $\exists z_1 \in Q$: $\mathsf{R.Vf}(z_0, z_1)$ then return ($\bot$, "priv")
Return ($m$, "ok")
```

Figure 6: Games defining privacy of signcryption scheme SC with respect to relaxing relation R.

<u>Outputs of oracle VerDec.</u> The output of oracle VerDec in game G^{priv} is a pair containing the plaintext (or the incorrect decryption symbol $\bot$) as its first element, and the status message as its second element. This ensures that the adversary can distinguish whether VerDec returned $\bot$ because it failed to decrypt the ciphertext (yields error message "dec"), or because the relaxing relation prohibits the query (yields error message "priv"). Giving more information to the adversary results in a stronger security definition, and will help us prove equivalence between the joint and separate security notions of signcryption in Appendix C. Note that an adversary can distinguish between different output branches of all other oracles used in our authenticity and privacy games.

<u>A choice of relaxing relation for privacy.</u> Consider relaxing relations R_{id} and R_{m} that are defined in Fig. 4. We recover IND-CCA security of SC as the PRIV-security of SC with respect to R_{id}. And we capture the RCCA security of SC as the PRIV-security of SC with respect to R_{m}. Recall that the intuition behind the RCCA security [28, 53] is to prohibit the adversary from querying its decryption oracle with ciphertexts that encrypt a previously queried challenge message. In particular, this is the reason that two elements are added to set Q during each call to oracle

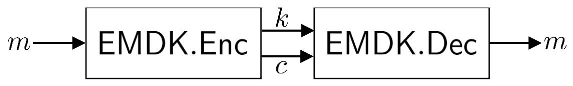

Figure 7: Constituent algorithms of encryption scheme under message derived keys EMDK.

LR, one for each of m_0 and m_1. Our definition of RCCA security for SC is very similar to that of IND-gCCA2 security as proposed by An, Dodis and Rabin [4]. The difference is that our definition passes the decrypted message as input to the relation, whereas IND-gCCA2 instead allows relations that take public keys of sender and recipient as input. It is not clear that having the relation take the public key would make our definition meaningfully stronger.

<u>Capturing outsider privacy.</u> Game $\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R},\mathcal{D}}$ captures the privacy of SC in the *insider* setting, meaning that the adversary is allowed to request challenge encryptions from id_s to id_r even when id_s is exposed. This implies some form of forward security because exposing the sender's key does not help the adversary win the indistinguishability game. To recover the notion of *outsider* privacy, consider a class of *outsider* adversaries that never query $\mathrm{LR}(id_s, \mathcal{I}, m_0, m_1, ad)$ when $\mathsf{exp}[id_s] = \mathsf{true}$.

4 Encryption under message derived keys

We now define Encryption under Message Derived Keys (EMDK). It can be thought of as a special type of symmetric encryption allowing to use keys that depend on the messages to be encrypted. This type of primitive will be at the core of analyzing the security of $\mathsf{iMessage}$-based signcryption scheme. In Section 4.1 we define syntax, correctness and basic security notions for EMDK schemes. In Section 4.2 we define the $\mathsf{iMessage}$-based EMDK scheme and analyse its security.

4.1 Syntax, correctness and security of EMDK

We start by defining the syntax and correctness of encryption schemes under message derived keys. The interaction between constituent algorithms of EMDK is shown in Fig. 7. The main security notions for EMDK schemes are AE (authenticated encryption) and ROB (robustness). We also define the IND (indistinguishability) notion that will be used in Section 4.2 for an intermediate result towards showing the AE-security of the $\mathsf{iMessage}$-based EMDK scheme.

<u>Encryption schemes under message derived keys.</u> An encryption scheme under message derived keys EMDK specifies algorithms $\mathsf{EMDK.Enc}$ and $\mathsf{EMDK.Dec}$, where $\mathsf{EMDK.Dec}$ is deterministic. Associated to EMDK is a key length $\mathsf{EMDK.kl} \in \mathbb{N}$. The encryption algorithm $\mathsf{EMDK.Enc}$ takes a message $m \in \{0,1\}^*$ to return a key $k \in \{0,1\}^{\mathsf{EMDK.kl}}$ and a ciphertext $c \in \{0,1\}^*$. The decryption algorithm $\mathsf{EMDK.Dec}$ takes k, c to return message $m \in \{0,1\}^* \cup \{\perp\}$, where $\perp$ denotes incorrect decryption. Decryption correctness requires that $\mathsf{EMDK.Dec}(k,c) = m$ for all $m \in \{0,1\}^*$, and all $(k,c) \in [\mathsf{EMDK.Enc}(m)]$.

<u>Indistinguishability of EMDK.</u> Consider game $\mathrm{G}^{\mathsf{ind}}$ of Fig. 8, associated to an encryption scheme under message derived keys EMDK, and to an adversary $\mathcal{D}$. The advantage of $\mathcal{D}$ in breaking the IND security of EMDK is defined as $\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{EMDK}}(\mathcal{D}) = 2 \cdot \Pr[\mathrm{G}^{\mathsf{ind}}_{\mathsf{EMDK},\mathcal{D}}] - 1$. The game samples a random challenge bit b and requires the adversary to guess it. The adversary has access to an encryption oracle LR that takes two challenge messages m_0, m_1 to return an EMDK encryption of m_b.

```
Game $\mathrm{G}^{\mathsf{ind}}_{\mathsf{EMDK},\mathcal{D}}$
$b \leftarrow_{\$} \{0,1\}$ ; $b' \leftarrow_{\$} \mathcal{D}^{\mathrm{LR}}$
Return $b = b'$

$\mathrm{LR}(m_0, m_1)$
If $|m_0| \neq |m_1|$ then return $\bot$
$(k, c) \leftarrow_{\$} \mathsf{EMDK.Enc}(m_b)$
Return $c$
```

```
Game $\mathrm{G}^{\mathsf{ae}}_{\mathsf{EMDK},\mathcal{D}}$
$b \leftarrow_{\$} \{0,1\}$ ; $b' \leftarrow_{\$} \mathcal{D}^{\mathrm{LR},\textsc{Dec}}$
Return $b = b'$

$\mathrm{LR}(m_0, m_1)$
If $|m_0| \neq |m_1|$ then return $\bot$
$n \leftarrow n + 1$
$(\mathsf{k}[n], \mathsf{c}[n]) \leftarrow_{\$} \mathsf{EMDK.Enc}(m_b)$
Return $(n, \mathsf{c}[n])$

$\textsc{Dec}(i, c)$
If $i \notin [n]$ or $\mathsf{c}[i] = c$ then return $\bot$
$m \leftarrow \mathsf{EMDK.Dec}(\mathsf{k}[i], c)$
If $b = 1$ then return $m$ else return $\bot$
```

```
Game $\mathrm{G}^{\mathsf{rob}}_{\mathsf{EMDK},\mathcal{G}}$
$(i, k) \leftarrow_{\$} \mathcal{G}^{\textsc{Enc}}$
If $i \notin [n]$ then return false
$m \leftarrow \mathsf{EMDK.Dec}(k, \mathsf{c}[i])$
Return $m \neq \bot$ and $m \neq \mathsf{m}[i]$

$\textsc{Enc}(m)$
$(k, c) \leftarrow_{\$} \mathsf{EMDK.Enc}(m)$
$n \leftarrow n + 1$ ; $\mathsf{m}[n] \leftarrow m$ ; $\mathsf{c}[n] \leftarrow c$
Return $(k, c)$
```

Figure 8: Games defining indistinguishability, authenticated encryption security, and robustness of encryption scheme under message derived keys EMDK.

```
$\mathsf{EMDK.Enc}^{\mathrm{RO}}(m)$
$k \leftarrow_{\$} \{0,1\}^{\mathsf{EMDK.kl}}$ ; $\ell \leftarrow |m|$
$x \leftarrow m \oplus \mathrm{RO}(k, \ell)$
$h \leftarrow \mathrm{RO}(k \,\|\, m, \ell)$
$c \leftarrow (x, h)$
Return $(k, c)$
```

```
$\mathsf{EMDK.Dec}^{\mathrm{RO}}(k, c)$
$(x, h) \leftarrow c$ ; $\ell \leftarrow |x|$
$m \leftarrow x \oplus \mathrm{RO}(k, \ell)$
$h' \leftarrow \mathrm{RO}(k \,\|\, m, \ell)$
If $h \neq h'$ then return $\bot$
Else return $m$
```

```
$\mathrm{RO}(z, \ell)$
If $T[z, \ell] = \bot$ then
    $T[z, \ell] \leftarrow_{\$} \{0,1\}^{\ell}$
Return $T[z, \ell]$
```

Figure 9: Sample EMDK scheme $\mathsf{EMDK} = \mathsf{SIMPLE\text{-}EMDK}$ in the ROM.

Authenticated encryption security of EMDK. Consider game G^{ae} of Fig. 8, associated to an encryption scheme under message derived keys EMDK, and to an adversary $\mathcal{D}$. The advantage of $\mathcal{D}$ in breaking the AE security of EMDK is defined as $\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{EMDK}}(\mathcal{D}) = 2 \cdot \Pr[\mathrm{G}^{\mathsf{ae}}_{\mathsf{EMDK},\mathcal{D}}] - 1$. Compared to the indistinguishability game from above, game G^{ae} saves the keys and ciphertexts produced by oracle LR, and also provides a decryption oracle $\textsc{Dec}$ to adversary $\mathcal{D}$. The decryption oracle allows to decrypt a ciphertext with any key that was saved by oracle Enc, returning either the actual decryption m (if $b = 1$) or the incorrect decryption symbol $\bot$ (if $b = 0$). To prevent trivial wins, the adversary is not allowed to query oracle $\textsc{Dec}$ with a key-ciphertext pair that were produced by the same LR query.

Robustness of EMDK. Consider game $\mathrm{G}^{\mathsf{rob}}$ of Fig. 8, associated to an encryption scheme under message derived keys EMDK, and to an adversary $\mathcal{G}$. The advantage of $\mathcal{G}$ in breaking the ROB security of EMDK is defined as $\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{EMDK}}(\mathcal{G}) = \Pr[\mathrm{G}^{\mathsf{rob}}_{\mathsf{EMDK},\mathcal{G}}]$. To win the game, adversary $\mathcal{G}$ is required to find (c, k_0, k_1, m_0, m_1) such that c decrypts to m_0 under key k_0, and c decrypts to m_1 under key k_1, but $m_0 \neq m_1$. Furthermore, the game requires that the ciphertext (along with one of the keys) was produced during a call to oracle $\textsc{Enc}$ that takes a message m as input to return the output (k, c) of running $\mathsf{EMDK.Enc}(m)$ with honestly generated random coins. The other key can be arbitrarily chosen by the adversary. In the symmetric encryption setting, a similar notion called *wrong-key detection* was previously defined by Canetti et al. [27]. The notion of robustness for public-key encryption was formalized by Abdalla et al. [1] and further extended by Farshim et al. [46].

Sample EMDK scheme SIMPLE-EMDK. It is easy to build an EMDK scheme that is both AE-

```
EMDK.Enc(m)
    $r_0 \leftarrow_\$ \{0,1\}^{\mathsf{F.kl}}$ ; $r_1 \leftarrow \mathsf{F.Ev}(r_0, m)$
    $k \leftarrow r_0 \| r_1$ ; $c_{se} \leftarrow_\$ \mathsf{SE.Enc}(k, m)$
    Return $(k, c_{se})$

EMDK.Dec(k, c_se)
    $m \leftarrow \mathsf{SE.Dec}(k, c_{se})$ ; If $m = \perp$ then return $\perp$
    $r_0 \leftarrow k[1 \ldots \mathsf{F.kl}]$ ; $r_1 \leftarrow k[\mathsf{F.kl}+1 \ldots \mathsf{SE.kl}]$
    If $r_1 \neq \mathsf{F.Ev}(r_0, m)$ then return $\perp$
    Return $m$
```

Figure 10: iMessage-based EMDK scheme $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$.

```
Game $\mathrm{G}^{\mathsf{pkr}}_{\mathsf{SE},\ell,\mathcal{P}}$
    $\mathcal{P}^{\textsc{Enc},\textsc{GuessKey}}$ ; Return win
Enc(m)
    $k \leftarrow_\$ \{0,1\}^{\mathsf{SE.kl}}$ ; $c \leftarrow_\$ \mathsf{SE.Enc}(k, m)$
    $n \leftarrow n + 1$ ; $\mathsf{k}[n] \leftarrow k[1 \ldots \ell]$ ; Return $c$
GuessKey(p)
    If $\exists i \in [n] \colon \mathsf{k}[i] = p$ then win $\leftarrow$ true

Game $\mathrm{G}^{\mathsf{wrob}}_{\mathsf{SE},\ell,\mathcal{G}}$
    $\mathcal{G}^{\textsc{Enc}}$ ; Return win
Enc(r_0, m)
    $r_1 \leftarrow_\$ \{0,1\}^{\ell}$ ; $k \leftarrow r_0 \| r_1$
    $c \leftarrow \mathsf{SE.Enc}(k, m)$
    If $\exists (m', c) \in W \colon m' \neq m$ then
        win $\leftarrow$ true
    $W \leftarrow W \cup \{(m, c)\}$ ; Return $r_1$
```

Figure 11: Games defining partial key recovery security of symmetric encryption scheme SE with respect to prefix length ℓ, and weak robustness of deterministic symmetric encryption scheme SE with respect to randomized key-suffix length ℓ.

secure and ROB-secure. One example of such scheme is the construction SIMPLE-EMDK in the random oracle model (ROM) that is defined in Fig. 9. In the next section we will define the EMDK scheme used iMessage; it looks convoluted, and its security is hard to prove even in the ideal models. In Appendix B we define the EMDK scheme that was initially used in iMessage; it was replaced with the current EMDK scheme in order to fix a security flaw in the iMessage design. We believe that the design of the currently used EMDK scheme was chosen based on a requirement to maintain backward-compatibility across the initial and the current versions of iMessage protocol.

4.2 iMessage-based EMDK scheme

In this section we define the EMDK scheme IMSG-EMDK that is used as the core building block in the construction of iMessage (we use it to specify the iMessage-based signcryption scheme in Section 5). We will provide reductions showing the AE-security and the ROB-security of IMSG-EMDK. These security reductions will first require us to introduce two new security notions for symmetric encryption schemes: *partial key recovery* and *weak robustness.*

EMDK SCHEME IMSG-EMDK. Let SE be a symmetric encryption scheme. Let F be a function family with $\mathsf{F.In} = \{0,1\}^*$ such that $\mathsf{F.kl} + \mathsf{F.ol} = \mathsf{SE.kl}$. Then $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$ is the EMDK scheme as defined in Fig. 10, with key length $\mathsf{EMDK.kl} = \mathsf{SE.kl}$.

Informally, the encryption algorithm $\mathsf{EMDK.Enc}(m)$ samples a hash function key r_0 and computes hash $r_1 \leftarrow_\$ \mathsf{F.Ev}(r_0, m)$. It then encrypts m by running $\mathsf{SE.Enc}(k, m)$, where $k = r_0 \| r_1$ is a message-derived key. The decryption algorithm splits k into r_0 and r_1 and – upon recovering m – checks that $r_1 = \mathsf{F.Ev}(r_0, m)$. In the iMessage construction, SE is instantiated with AES-CTR using 128-bit keys and a fixed IV=1, whereas F is instantiated with HMAC-SHA256 using $\mathsf{F.kl} = 88$ and $\mathsf{F.ol} = 40$.

PARTIAL KEY RECOVERY SECURITY OF SE. Consider game $\mathrm{G}^{\mathsf{pkr}}$ of Fig. 11, associated to a symmetric encryption scheme SE, a prefix length $\ell \in \mathbb{N}$ and an adversary $\mathcal{P}$. The advantage of $\mathcal{P}$ in breaking the PKR-security of SE with respect to ℓ is defined as $\mathsf{Adv}^{\mathsf{pkr}}_{\mathsf{SE},\ell}(\mathcal{P}) = \Pr[\mathrm{G}^{\mathsf{pkr}}_{\mathsf{SE},\ell,\mathcal{P}}]$. The adversary $\mathcal{P}$ has access to oracle $\textsc{Enc}$ that takes a message m and encrypts it under a uniformly random key k (independently sampled for each oracle call). The goal of the adversary is to recover the first ℓ bits of *any* secret key that was used in prior $\textsc{Enc}$ queries.

<u>WEAK ROBUSTNESS OF DETERMINISTIC SE.</u> Consider game $\mathrm{G}^{\mathsf{wrob}}$ of Fig. 11, associated to a deterministic symmetric encryption scheme SE, a randomized key-suffix length $\ell \in \mathbb{N}$, and an adversary $\mathcal{G}$. The advantage of $\mathcal{G}$ in breaking the WROB-security of SE with respect to ℓ is defined as $\mathsf{Adv}^{\mathsf{wrob}}_{\mathsf{SE},\ell}(\mathcal{G}) = \Pr[\mathrm{G}^{\mathsf{wrob}}_{\mathsf{SE},\ell,\mathcal{G}}]$. The adversary has access to oracle $\textsc{Enc}$. The oracle takes a prefix of an encryption key $r_0 \in \{0,1\}^{\mathsf{SE.kl}-\ell}$ and message m as input. It then randomly samples the suffix of the key $r_1 \in \{0,1\}^{\ell}$ and returns it to the adversary. The adversary wins if it succeeds to query $\textsc{Enc}$ on some inputs (r_0, m) and (r'_0, m') such that $m \neq m'$ yet the oracle mapped both queries to the same ciphertext c. In other words, the goal of the adversary is to find k_0, m_0, k_1, m_1 such that $\mathsf{SE.Enc}(k_0, m_0) = \mathsf{SE.Enc}(k_1, m_1)$ and $m_0 \neq m_1$ (which also implies $k_0 \neq k_1$), and the adversary has only a partial control over the choice of k_0 and k_1. Note that this assumption can be validated in the ideal cipher model.

<u>SECURITY REDUCTIONS FOR $\mathsf{IMSG\text{-}EMDK}$.</u> We now provide the reductions for AE-security and ROB-security of $\mathsf{IMSG\text{-}EMDK}$. The former is split into Theorem 4.1 and Theorem 4.2, whereas the latter is provided in Theorem 4.3. Note that in Appendix D we provide the standard definitions for the random oracle model, the UNIQUE-security and the OTIND-security of symmetric encryption, and the TCR-security of function families. The proofs of Theorem 4.1, Theorem 4.2 and Theorem 4.3 are in Appendix E, Appendix F, and Appendix G respectively.

Theorem 4.1 *Let* SE *be a symmetric encryption scheme. Let* F *be a function family with* $\mathsf{F.In} = \{0,1\}^*$*, such that* $\mathsf{F.kl} + \mathsf{F.ol} = \mathsf{SE.kl}$*. Let* $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$*. Let* $\mathcal{D}_{\mathrm{AE}}$ *be an adversary against the* AE*-security of* EMDK*. Then we build an adversary* $\mathcal{U}$ *against the* UNIQUE*-security of* SE*, an adversary* $\mathcal{H}$ *against the* TCR*-security of* F*, and an adversary* $\mathcal{D}_{\mathrm{IND}}$ *against the* IND*-security of* EMDK *such that*

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{EMDK}}(\mathcal{D}_{\mathrm{AE}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{unique}}_{\mathsf{SE}}(\mathcal{U}) + 2 \cdot \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{F}}(\mathcal{H}) + \mathsf{Adv}^{\mathsf{ind}}_{\mathsf{EMDK}}(\mathcal{D}_{\mathrm{IND}}).$$

Theorem 4.2 *Let* SE *be a symmetric encryption scheme. Let* F *be a function family with* $\mathsf{F.In} = \{0,1\}^*$ *and* $\mathsf{F.kl} + \mathsf{F.ol} = \mathsf{SE.kl}$*, defined by* $\mathsf{F.Ev}^{\mathrm{RO}}(r, m) = \mathrm{RO}(\langle r, m\rangle, \mathsf{F.ol})$ *in the random oracle model. Let* $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$*. Let* $\mathcal{D}_{\mathsf{EMDK}}$ *be an adversary against the* IND*-security of* EMDK *that makes* q_{LR} *queries to its* LR *oracle and* q_{RO} *queries to random oracle* RO*. Then we build an adversary* $\mathcal{P}$ *against the* PKR*-security of* SE *with respect to* $\mathsf{F.kl}$*, and an adversary* $\mathcal{D}_{\mathsf{SE}}$ *against the* OTIND*-security of* SE*, such that*

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{EMDK}}(\mathcal{D}_{\mathsf{EMDK}}) \leq 2 \cdot \gamma + 2 \cdot \mathsf{Adv}^{\mathsf{pkr}}_{\mathsf{SE},\mathsf{F.kl}}(\mathcal{P}) + \mathsf{Adv}^{\mathsf{otind}}_{\mathsf{SE}}(\mathcal{D}_{\mathsf{SE}}),$$

where

$$\gamma = \frac{(2 \cdot q_{\mathrm{RO}} + q_{\mathrm{LR}} - 1) \cdot q_{\mathrm{LR}}}{2^{\mathsf{F.kl}+1}}.$$

Theorem 4.3 *Let* SE *be a deterministic symmetric encryption scheme. Let* F *be a function family with* $\mathsf{F.In} = \{0,1\}^*$ *and* $\mathsf{F.kl} + \mathsf{F.ol} = \mathsf{SE.kl}$*, defined by* $\mathsf{F.Ev}^{\mathrm{RO}}(r, m) = \mathrm{RO}(\langle r, m\rangle, \mathsf{F.ol})$ *in the random oracle model. Let* $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$*. Let* $\mathcal{G}_{\mathsf{EMDK}}$ *be an adversary against the* ROB*-security of* EMDK*. Then we build an adversary* $\mathcal{U}$ *against the* UNIQUE*-security of* SE*, and an adversary* $\mathcal{G}_{\mathsf{SE}}$ *against the* WROB*-security of* SE *with respect to* $\mathsf{F.ol}$ *such that*

$$\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{EMDK}}(\mathcal{G}_{\mathsf{EMDK}}) \leq \mathsf{Adv}^{\mathsf{unique}}_{\mathsf{SE}}(\mathcal{U}) + \mathsf{Adv}^{\mathsf{wrob}}_{\mathsf{SE},\mathsf{F.ol}}(\mathcal{G}_{\mathsf{SE}}).$$

$$\begin{tikzcd}
\mathsf{F} \arrow[d] & \mathsf{SE} \arrow[dl] & \mathsf{PKE} \arrow[dddl] & \mathsf{DS} \arrow[dddddl] \\
\boxed{\mathsf{IMSG\text{-}EMDK}} \arrow[d] & & & \\
\mathsf{EMDK} \arrow[dr] & & & \\
& \boxed{\mathsf{IMSG\text{-}MRPKE}} \arrow[dr] & & \\
& & \mathsf{MRPKE} \arrow[d] & \\
& & \boxed{\mathsf{IMSG\text{-}SC}} \arrow[d] & \\
& & \mathsf{SC} &
\end{tikzcd}$$

Scheme	Construction	Figure
EMDK	IMSG-EMDK[F, SE]	10
MRPKE	IMSG-MRPKE[EMDK, PKE]	14
SC	IMSG-SC[MRPKE, DS]	13

Scheme	Instantiation
F	HMAC-SHA256 ($\mathsf{F.kl} = 88$, $\mathsf{F.ol} = 40$)
SE	AES-CTR with 128-bit key and IV=1
PKE	RSA-OAEP with 1280-bit key
DS	ECDSA with NIST P-256 curve

Figure 12: Modular design of iMessage-based signcryption scheme. The boxed nodes in the diagram denote transforms that build a new cryptographic scheme from two underlying primitives.

5 Design and security of iMessage

In this section we define a signcryption scheme that models the current design of iMessage protocol for end-to-end encrypted messaging, and we analyze its security. All publicly available information about the iMessage protocol is provided by Apple in *iOS Security Guide* [5] that is regularly updated but is very limited and vague. So in addition to the iOS Security Guide, we also reference work that attempted to reverse-engineer [74, 70] and attack [51] the prior versions of iMessage. A message-recovery attack against iMessage was previously found and implemented by Garman et al. [51] in 2016, and subsequently fixed by Apple starting from version 9.3 of iOS, and version 10.11.4 of Mac OS X. The implemented changes to the protocol prevented the attack, but also made the protocol design less intuitive. It appears that one of the goals of the updated protocol design was to preserve backward-compatibility, and that could be the reason why the current design is a lot more more sophisticated than otherwise necessary. Apple has not formalized any claims about the security achieved by the initial or the current iMessage protocol, or the assumptions that are required from the cryptographic primitives that serve as the building blocks. We fill in the gap by providing precise claims about the security of iMessage design when modeled by our signcryption scheme. In this section we focus only on the current protocol design of iMessage. In Appendix B we provide the design of the initial iMessage protocol, we explain the attack proposed by Garman et al. [51], and we introduce the goal of backward-compatibility for signcryption schemes.

5.1 iMessage-based signcryption scheme IMSG-SC

IDENTIFYING SIGNCRYPTION AS THE GOAL. The design of iMessage combines multiple cryptographic primitives to build an end-to-end encrypted messaging protocol. It uses HMAC-SHA256, AES-CTR, RSA-OAEP and ECDSA as the underlying primitives. Apple's *iOS Security Guide* [5] and prior work on reverse-engineering and analysis of iMessage [74, 70, 51] does not explicitly indicate what type of cryptographic scheme is built as the result of combining these primitives. We identify it as a signcryption scheme. We define the iMessage-based signcryption scheme IMSG-SC in a modular way that facilitates its security analysis. Fig. 12 shows the order in which the underlying primitives are combined to build IMSG-SC, while also providing intermediate constructions along the way. We now explain this step by step.

MODULAR DESIGN OF IMSG-SC. Our construction starts from choosing a function family F and

```
SC.Setup
    $\pi \leftarrow$ MRPKE.Setup ; Return $\pi$

SC.SigEnc($\pi, id_s, pk_s, sk_s, \mathcal{R}, m, ad$)
    $\mathcal{I} \leftarrow \emptyset$ ; $\mathcal{R}_{pke} \leftarrow \emptyset$ ; $\mathcal{C} \leftarrow \emptyset$
    For each $(id_r, pk_r) \in \mathcal{R}$ do
        $(vk_r, ek_r) \leftarrow pk_r$
        $\mathcal{I} \leftarrow \mathcal{I} \cup \{id_r\}$
        $\mathcal{R}_{pke} \leftarrow \mathcal{R}_{pke} \cup \{(id_r, ek_r)\}$
    $m_{pke} \leftarrow \langle m, id_s, \mathcal{I} \rangle$
    $\mathcal{C}_{pke} \leftarrow\$$ MRPKE.Enc($\pi, \mathcal{R}_{pke}, m_{pke}$)
    $(tk_s, dk_s) \leftarrow sk_s$
    For each $(id_r, c_{pke}) \in \mathcal{C}_{pke}$ do
        $\sigma \leftarrow\$$ DS.Sig($tk_s, \langle c_{pke}, ad \rangle$)
        $c \leftarrow (c_{pke}, \sigma)$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
    Return $\mathcal{C}$

SC.Kg($\pi$)
    $(vk, tk) \leftarrow\$$ DS.Kg
    $(ek, dk) \leftarrow\$$ MRPKE.Kg($\pi$)
    $pk \leftarrow (vk, ek)$ ; $sk \leftarrow (tk, dk)$
    Return $(pk, sk)$

SC.VerDec($\pi, id_s, pk_s, id_r, pk_r, sk_r, c, ad$)
    $(c_{pke}, \sigma) \leftarrow c$ ; $(vk_s, ek_s) \leftarrow pk_s$
    $(vk_r, ek_r) \leftarrow pk_r$ ; $(tk_r, dk_r) \leftarrow sk_r$
    $d \leftarrow$ DS.Ver($vk_s, \langle c_{pke}, ad \rangle, \sigma$)
    If not $d$ then return $\perp$
    $m_{pke} \leftarrow$ MRPKE.Dec($\pi, ek_r, dk_r, c_{pke}$)
    If $m_{pke} = \perp$ then return $\perp$
    $\langle m, id_s^*, \mathcal{I} \rangle \leftarrow m_{pke}$
    If $id_s \neq id_s^*$ or $id_r \notin \mathcal{I}$ then return $\perp$
    Return $m$
```

Figure 13: Signcryption scheme SC = IMSG-SC[MRPKE, DS].

a symmetric encryption scheme SE (instantiated with HMAC-SHA256 and AES-CTR in iMessage). It combines them to build an encryption scheme under message derived keys EMDK = IMSG-EMDK[F, SE]. The resulting EMDK scheme is combined with public-key encryption scheme PKE (instantiated with RSA-OAEP in iMessage) to build a multi-recipient public-key encryption scheme MRPKE = IMSG-MRPKE[EMDK, PKE] (syntax and correctness of MRPKE schemes is defined in Appendix D). Finally, MRPKE and digital signature scheme DS (instantiated with ECDSA in iMessage) are combined to build the iMessage-based signcryption scheme SC = IMSG-SC[MRPKE, DS]. The definition of IMSG-EMDK was provided in Section 4.2. We now define IMSG-SC and IMSG-MRPKE.

Signcryption scheme IMSG-SC. Let MRPKE be a multi-recipient public-key encryption scheme. Let DS be a digital signature scheme. Then SC = IMSG-SC[MRPKE, DS] is the signcryption scheme as defined in Fig. 13, with SC.ID $= \{0,1\}^*$. In order to produce a signcryption of message m with associated data ad, algorithm SC.SigEnc performs the following steps. It builds a new message $m_{pke} = \langle m, id_s, \mathcal{I} \rangle$ as the unique encoding of $m, id_s, \mathcal{I}$, where $\mathcal{I}$ is the set of recipients. It then calls MRPKE.Enc to encrypt the same message m_{pke} for every recipient. Algorithm MRPKE.Enc returns a set $\mathcal{C}_{pke}$ containing pairs (id_r, c_{pke}), each indicating that an MRPKE ciphertext c_{pke} was produced for recipient id_r. For each recipient, the corresponding ciphertext c_{pke} is then encoded with the associated data ad into $\langle c_{pke}, ad \rangle$ and signed using the signing key tk_s of sender identity id_s, producing a signature σ. The pair $(id_r, (c_{pke}, \sigma))$ is then added to the output set of algorithm SC.SigEnc. When running the unsigncryption of ciphertext c sent from id_s to id_r, algorithm SC.VerDec ensures that the recovered MRPKE plaintext $m_{pke} = \langle m, id_s^*, \mathcal{I} \rangle$ is consistent with $id_s = id_s^*$ and $id_r \in \mathcal{I}$.

Multi-recipient public-key encryption scheme IMSG-MRPKE. Let EMDK be an encryption scheme under message derived keys. Let PKE be a public-key encryption scheme with PKE.In $= \{0,1\}^{\mathsf{EMDK.kl}}$. Then MRPKE = IMSG-MRPKE[EMDK, PKE] is the multi-recipient public-key encryption scheme as defined in Fig. 14. Algorithm MRPKE.Enc first runs $(k, c_{se}) \leftarrow\$$ EMDK.Enc(m) to produce an EMDK ciphertext c_{se} that encrypts m under key k. The obtained key k is then independently encrypted for each recipient identity id_r using its PKE encryption key ek_r, and the corresponding tuple $(id_r, (c_{se}, c_{pke}))$ is added to the output set of algorithm MRPKE.Enc.

Combining everything together. Let SC be the iMessage-based signcryption scheme that is

```
MRPKE.Setup
    $\pi \leftarrow \varepsilon$ ; Return $\pi$

MRPKE.Enc($\pi, \mathcal{R}, m$)
    $\mathcal{C} \leftarrow \emptyset$ ; $(k, c_{se}) \leftarrow\$$ EMDK.Enc($m$)
    For each $(id_r, ek_r) \in \mathcal{R}$ do
        $c_{pke} \leftarrow\$$ PKE.Enc($ek_r, k$)
        $c \leftarrow (c_{se}, c_{pke})$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
    Return $\mathcal{C}$

MRPKE.Kg($\pi$)
    $(ek, dk) \leftarrow\$$ PKE.Kg ; Return $(ek, dk)$

MRPKE.Dec($\pi, ek, dk, c$)
    $(c_{se}, c_{pke}) \leftarrow c$
    $k \leftarrow$ PKE.Dec($ek, dk, c_{pke}$)
    If $k = \perp$ then return $\perp$
    $m \leftarrow$ EMDK.Dec($k, c_{se}$)
    Return $m$
```

Figure 14: Multi-recipient public-key encryption scheme MRPKE = IMSG-MRPKE[EMDK, PKE].

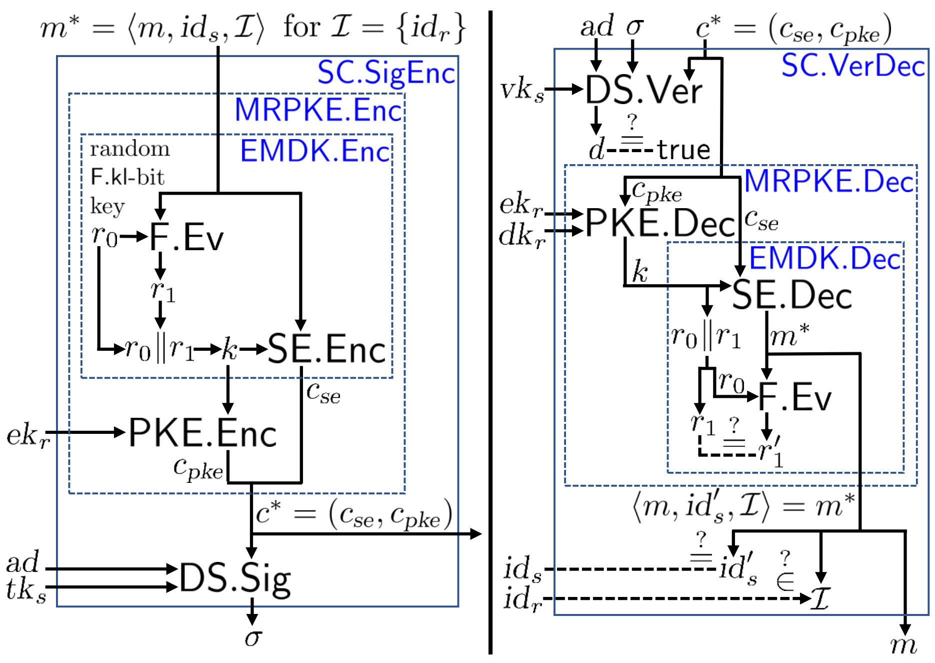

Figure 15: Algorithms SC.SigEnc (left panel) and SC.VerDec (right panel) for SC = IMSG-SC [MRPKE, DS], where MRPKE = IMSG-MRPKE[EMDK, PKE] and EMDK = IMSG-EMDK[F, SE]. For simplicity, we let id_r be the only recipient, and we do not show how to parse inputs and combine outputs for the displayed algorithms. The dotted lines inside SC.VerDec denote equality check, and the dotted arrow denotes membership check.

produced by combining all of the underlying primitives described above. Then the data flow within the fully expanded algorithms SC.SigEnc and SC.VerDec is schematically displayed in Fig. 15. For simplicity, the diagrams show the case when a message m is sent to a single recipient id_r.

Details not captured by our model. Our goal in this work was to capture the iMessage design as closely as possible. Garman et al. [51] provided the most detailed description of the iMessage design prior to the changes deployed in version 9.3 of iOS, and version 10.11.4 of Mac OS X. The only publicly available information about the current iMessage design is provided in *iOS Security Guide* [5] and is very vague. We used the former in order to fill in the gaps in our understanding whenever the latter was too ambiguous. This approach seems to be justified by Apple's apparent goal to preserve backward-compatibility between the two versions of iMessage's end-to-end encryption protocol. Based on the above, we now list the main differences that we believe exist between our constuction and the iMessage design.

In our construction, the message encrypted by the EMDK scheme is $\langle m, id_s, \mathcal{I} \rangle$, which is the uniquely decodable encoding of values $m, id_s, \mathcal{I}$. As per our requirements from Section 2, we assume that this encoding is trivially malleable. In the implementation of iMessage the values $m, id_s, \mathcal{I}$ are encoded into a binary plist key-value data structure, then compressed using the gzip compression format, and only then encrypted with the EMDK scheme (i.e. using AES-CTR). In order to implement a practical attack against the initial iMessage design, Garman et al. [51] had to come up with a novel, non-trival technique of exploiting the malleability of AES-CTR when it encrypts gzip compressed data.

Now let c_{mrpke} be the MRPKE ciphertext and let c_{pke} be the PKE ciphertext (to disambiguate between different meanings of c_{pke} in our schemes). As per the description of Garman et al. [51], iMessage splits the EMDK ciphertext c_{se} into $c_1 \,\|\, c_2$ such that c_1 contains the first 101 bytes of c_{se}. It then computes $c_{pke} \leftarrow\$ \mathsf{PKE.Enc}(ek_r, k \,\|\, c_1)$, and sets $c_{mrpke} = (c_2, c_{pke})$. This is clearly done in order to optimize the bandwidth. It could also significantly improve the security of the entire scheme whenever the length of EMDK ciphertext does not exceed 101 bytes. However, according to Garman et al. [51], the first 101 bytes of the EMDK ciphertext will normally contain only plist key values and gzip header information (such as Huffman table). So we chose to omit this detail in our construction, and instead not put any part of the EMDK ciphertext into the PKE ciphertext.

Our iMessage-based signcryption scheme allows to use arbitrary associated data ad. The iMessage implementation does not appear to use associated data. In particular, iMessage implementation uses DS to sign only the MRPKE ciphertext, and it was our own choice to also sign associated data (as opposed to appending ad to the EMDK message).

5.2 Parameter-choice induced attacks on privacy of iMessage

The iMessage-based signcryption scheme SC uses the EMDK scheme $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$ as one of its underlying primitives. Recall that in order to encrypt a payload $m' = \langle m, id_s, \mathcal{I} \rangle$, the EMDK scheme samples a function key $r_0 \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$, computes a hash of m' as $r_1 \leftarrow \mathsf{F.Ev}(r_0, m')$, sets the encryption key $k \leftarrow r_0 \,\|\, r_1$, and produces a ciphertext as $c_{se} \leftarrow\$ \mathsf{SE.Enc}(k, m')$. The implementation of iMessage uses parameters $\mathsf{F.kl} = 88$ and $\mathsf{F.ol} = 40$. In this section we provide three adversaries against the privacy of SC whose success depends on the choice of F.kl and F.ol. In next sections we will provide security proofs for SC. We will show that each adversary in this section arises from an attack against a different step in our security proofs. We will be able to conclude that these are roughly the best attacks that arise from the choice of EMDK parameters. We will also explain why it is hard to construct any adversaries against the authenticity of SC. Now consider the adversaries of Fig. 16. The formal claims about each adversary will be stated at the end of this section.

Exhaustive key search. Adversary $\mathcal{D}_{\mathsf{exhaustive},n}$ is parameterized by message length $n \in \mathbb{N}$. It calls oracle LR to produce a single challenge ciphertext (encrypting m_0 or m_1) from an honest sender id_s to an honest recipient id_r. Then for every possible value of $r_0 \in \{0,1\}^{\mathsf{F.kl}}$, it computes r_1 as the hash of payload $m'_1 = \langle m_1, id_s, \{id_r\} \rangle$, sets $k \leftarrow r_0 \,\|\, r_1$, and uses it to decrypt the EMDK-encrypted part of the challenge ciphertext. The adversary returns 1 iff the ciphertext decrypts to m'_1. $\mathcal{D}_{\mathsf{exhaustive},n}$ always returns 1 when $b = 1$, but it is very unlikely to return 1 when $b = 0$ because m_1 is uniformly random and independent of m_0.

Birthday attack. Adversary $\mathcal{D}_{\mathsf{birthday}}$ makes roughly $2^{\mathsf{F.kl}/2}$ queries to oracle LR, using distinct values for message m_0 and a fixed value for message m_1. The adversary returns 1 iff two different challenge ciphertexts produced by LR contain the same EMDK ciphertext. Due to the birthday

```
$\mathcal{D}_{\mathsf{exhaustive},n}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\mathrm{LR},\textsc{VerDec}}(\pi)$
$id_s \leftarrow$ "send" ; $pk_s \leftarrow\$ \textsc{NewH}(id_s)$
$id_r \leftarrow$ "recv" ; $pk_r \leftarrow\$ \textsc{NewH}(id_r)$
$\mathcal{I} \leftarrow \{id_r\}$ ; $ad \leftarrow \varepsilon$
$m_0 \leftarrow 0^n$ ; $m_1 \leftarrow\$ \{0,1\}^n$
$\mathcal{C} \leftarrow\$ \mathrm{LR}(id_s, \mathcal{I}, m_0, m_1, ad)$
$\{(id_r, c)\} \leftarrow \mathcal{C}$ ; $((c_{se}, c_{pke}), \sigma) \leftarrow c$
$m_1' \leftarrow \langle m_1, id_s, \mathcal{I}\rangle$
For each $r_0 \in \{0,1\}^{\mathsf{F.kl}}$ do
    $r_1 \leftarrow \mathsf{F.Ev}(r_0, m_1')$ ; $k \leftarrow r_0 \,\|\, r_1$
    If $\mathsf{SE.Dec}(k, c_{se}) = m_1'$ then return 1
Return 0
```

```
$\mathcal{D}_{\mathsf{birthday}}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\mathrm{LR},\textsc{VerDec}}(\pi)$
$id_s \leftarrow$ "send" ; $pk_s \leftarrow\$ \textsc{NewH}(id_s)$
$id_r \leftarrow$ "recv" ; $pk_r \leftarrow\$ \textsc{NewH}(id_r)$
$\mathcal{I} \leftarrow \{id_r\}$ ; $ad \leftarrow \varepsilon$
$S \leftarrow \emptyset$ ; $p \leftarrow \lceil \mathsf{F.kl}/2 \rceil$ ; $m_1 \leftarrow 0^p$
For each $m_0 \in \{0,1\}^p$ do
    $\mathcal{C} \leftarrow\$ \mathrm{LR}(id_s, \mathcal{I}, m_0, m_1, ad)$
    $\{(id_r, c)\} \leftarrow \mathcal{C}$ ; $((c_{se}, c_{pke}), \sigma) \leftarrow c$
    If $c_{se} \in S$ then return 1
    $S \leftarrow S \cup \{c_{se}\}$
Return 0
```

```
$\mathcal{D}_{\mathsf{ADR02}}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\mathrm{LR},\textsc{VerDec}}(\pi)$
$id_s \leftarrow 0^{128}$ ; $pk_s \leftarrow\$ \textsc{NewH}(id_s)$ ; $id_r \leftarrow 1^{128}$ ; $pk_r \leftarrow\$ \textsc{NewH}(id_r)$
$\mathcal{I} \leftarrow \{id_r\}$ ; $m_0 \leftarrow 0^{128}$ ; $m_1 \leftarrow 1^{128}$ ; $ad \leftarrow \varepsilon$
$\mathcal{C} \leftarrow\$ \mathrm{LR}(id_s, \mathcal{I}, m_0, m_1, ad)$ ; $\{(id_r, c)\} \leftarrow \mathcal{C}$ ; $((c_{se}, c_{pke}), \sigma) \leftarrow c$
$id_c \leftarrow 0^{64}1^{64}$ ; $(pk_c, sk_c) \leftarrow\$ \mathsf{SC.Kg}(\pi)$ ; $\textsc{NewC}(id_c, pk_c, sk_c)$ ; $(tk_c, dk_c) \leftarrow sk_c$
$m_1' \leftarrow \langle m_1, id_s, \{id_r\}\rangle$ ; $m_1'' \leftarrow \langle m_1, id_c, \{id_r\}\rangle$ ; $c_{se}' \leftarrow c_{se} \oplus (m_1' \oplus m_1'')$
$\sigma' \leftarrow\$ \mathsf{DS.Sig}(tk_c, \langle (c_{se}', c_{pke}), ad\rangle)$ ; $c' \leftarrow ((c_{se}', c_{pke}), \sigma')$
$(m, \mathsf{err}) \leftarrow \textsc{VerDec}(id_c, id_r, c', ad)$ ; If $m = m_1$ then return 1 else return 0
```

Figure 16: Adversaries $\mathcal{D}_{\mathsf{exhaustive},n}$, $\mathcal{D}_{\mathsf{birthday}}$ and $\mathcal{D}_{\mathsf{ADR02}}$ against the PRIV-security of $\mathsf{SC} = \mathsf{IMSG\text{-}SC}[\mathsf{MRPKE}, \mathsf{DS}]$, where $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE}[\mathsf{EMDK}, \mathsf{PKE}]$ and $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$. Adversary $\mathcal{D}_{\mathsf{ADR02}}$ requires that SE is AES-CTR with a fixed IV.

paradox, this is likely to occur if $b = 1$; it happens whenever the same value of $r_0 \in \{0,1\}^{\mathsf{F.kl}}$ is sampled twice. But this is a lot less likely to occur if $b = 0$; it happens whenever two distinct messages encrypted under two distinct EMDK keys produce the same EMDK ciphertext.

MULTI-USER ATTACK. Adversary $\mathcal{D}_{\mathsf{ADR02}}$ calls oracle LR to get a challenge ciphertext $c = ((c_{se}, c_{pke}), \sigma)$ encrypting message m_b from an honest sender id_s to an honest recipient id_r. It then creates a corrupted identity id_c, mauls the EMDK ciphertext c_{se} to replace the original sender identity id_s with id_c, resigns the MRPKE ciphertext (c_{se}, c_{pke}) with id_c's signing key tk_c, and passes the resulting ciphertext $c' = ((c_{se}', c_{pke}), \sigma')$ to oracle VerDec in order to get the message m_b in plain. This attack uses the malleability of AES-CTR, and the malleability of the encoding $\langle m_b, id_s, \{id_r\}\rangle$ (as required in Section 2). Furthermore, this attack only succeeds when $\mathsf{F.Ev}(r_0, \langle m_b, id_c, \{id_r\}\rangle) = r_1$, where $k = r_0 \,\|\, r_1$ is the EMDK key that was used to encrypt the challenge message between honest identities. The advantage of adversary $\mathcal{D}_{\mathsf{ADR02}}$ can be amplified by trying many distinct corrupted identities until the above condition is satisfied; but this comes at the expense of making a higher number of oracle queries (and hence a higher runtime complexity). This attack was initially proposed by An, Dodis and Rabin [4] to show that privacy of signcryption in two-user setting does not imply its privacy in multi-user setting. This also served as the starting point for the practical message-recovery attack implemented by Garman et al. [51] against the initial design of iMessage (see Appendix B for details).

FORMAL CLAIMS AND ANALYSIS. We provide the number of queries, the runtime complexity and the advantage of each adversary in Fig. 17. The assumptions necessary to prove the advantage are stated in Lemma 5.1 below. Note that $\mathcal{D}_{\mathsf{birthday}}$ represents a purely theoretical attack, but both $\mathcal{D}_{\mathsf{exhaustive}}$ and $\mathcal{D}_{\mathsf{ADR02}}$ can lead to practical message-recovery attacks (the latter used by Garman

Adversary	q_{LR}	$q_{\textsc{NewC}}$	$q_{\textsc{VerDec}}$	Runtime complexity	Advantage
$\mathcal{D}_{\mathsf{exhaustive},n}$	1	0	0	$2^{\mathsf{F.kl}}$ evaluations of $\mathsf{F.Ev}$, $\mathsf{SE.Enc}$	$\geq 1 - 2^{\mathsf{SE.kl}-n}$
$\mathcal{D}_{\mathsf{birthday}}$	$2^{\lceil \mathsf{F.kl}/2 \rceil}$	0	0	$2^p \cdot p$ for $p = \lceil \mathsf{F.kl}/2 \rceil$	$> 1/8 - 2^{\mathsf{F.kl}-128}$
$\mathcal{D}_{\mathsf{ADR02}}$	1	1	1	1 evaluation of $\mathsf{SC.Kg}$, $\mathsf{DS.Sig}$	$= 2^{-\mathsf{F.ol}}$

Figure 17: The resources used by adversaries $\mathcal{D}_{\mathsf{exhaustive},n}$, $\mathcal{D}_{\mathsf{birthday}}$ and $\mathcal{D}_{\mathsf{ADR02}}$, and the advantage achieved by each of them. Columns labeled q_{O} denote the number of queries an adversary makes to oracle O. All adversaries make 2 queries to oracle NewH, and 0 queries to oracle Exp. See Lemma 5.1 for necessary assumptions.

et al. [51]).

Let $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$. Adversary $\mathcal{D}_{\mathsf{ADR02}}$ shows that EMDK can have at most $\mathsf{F.ol}$ bits of security with respect to PRIV, and adversary $\mathcal{D}_{\mathsf{birthday}}$ shows that EMDK can have at most $\approx \mathsf{F.kl}/2 + \log_2 \mathsf{F.kl}$ bits of security with respect to PRIV. It follows that setting $\mathsf{F.ol} \approx \mathsf{F.kl}/2$ is a good initial guideline, and roughly corresponds to the parameter choices made in iMessage. We will provide a more detailed analysis in Section 5.5. The proof of Lemma 5.1 is in Appendix H.

Lemma 5.1 *Let* SE *be a symmetric encryption scheme. Let* F *be a function family with* $\mathsf{F.In} = \{0,1\}^*$ *such that* $\mathsf{F.kl} + \mathsf{F.ol} = \mathsf{SE.kl}$. *Let* $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}[\mathsf{F}, \mathsf{SE}]$. *Let* PKE *be a public-key encryption scheme with* $\mathsf{PKE.In} = \{0,1\}^{\mathsf{SE.kl}}$. *Let* $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE}[\mathsf{EMDK}, \mathsf{PKE}]$. *Let* DS *be a digital signature scheme. Let* $\mathsf{SC} = \mathsf{IMSG\text{-}SC}[\mathsf{MRPKE}, \mathsf{DS}]$. *Let* $\mathsf{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ *be any relaxing relation. Then for any* $n > \mathsf{SE.kl}$,

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}_{\mathsf{exhaustive},n}) \geq 1 - 2^{\mathsf{SE.kl}-n}.$$

Furthermore, for any $1 \leq \mathsf{F.kl} \leq 124$, *if* SE *is AES-CTR with a fixed IV, and if AES is modeled as the ideal cipher, then*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}_{\mathsf{birthday}}) > 1/8 - 2^{\mathsf{F.kl}-128}.$$

Let $\mathsf{R_m}$ *be the relaxing relation defined in Fig. 4. If* SE *is AES-CTR with a fixed IV, and if* F *is defined as* $\mathsf{F.Ev}^{\mathrm{RO}}(r, m) = \mathrm{RO}(\langle r, m \rangle, \mathsf{F.ol})$ *in the random oracle model, then*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R_m}}(\mathcal{D}_{\mathsf{ADR02}}) = 2^{-\mathsf{F.ol}}.$$

5.3 Authenticity of iMessage

In this section we reduce the authenticity of the iMessage-based signcryption scheme SC to the security of its underlying primitives. First we reduce the authenticity of $\mathsf{SC} = \mathsf{IMSG\text{-}SC}[\mathsf{MRPKE}, \mathsf{DS}]$ to the unforgeability of DS and to the robustness of MRPKE. And then we reduce the robustness of $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE}[\mathsf{EMDK}, \mathsf{PKE}]$ to the robustness of *either* PKE or EMDK; it is sufficient that only one of the two is robust.

<u>Reduction showing authenticity of IMSG-SC.</u> Recall that an SC ciphertext is a pair (c_{pke}, σ) that consists of an MRPKE ciphertext c_{pke} (encrypting some $\langle m, id_s, \mathcal{I} \rangle$) and a DS signature σ of $\langle c_{pke}, ad \rangle$. Intuitively, the authenticity of SC requires some type of unforgeability from DS in order to prevent the adversary from producing a valid signature on arbitrary c_{pke} and ad of its own choice. However, the unforgeability of DS is not a sufficient condition, because the adversary is allowed to win the game $\mathrm{G}^{\mathsf{auth}}$ by forging an SC ciphertext for a corrupted recipient identity that uses maliciously chosen SC keys. So an additional requirement is that the adversary should not be able to find an SC key pair (pk, sk) that succesfully decrypts an honestly produced SC

```
$\underline{\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*].\mathsf{Vf}(z, z^*)}$
$((id_s, id_r, m, ad), (c_{pke}, \sigma)) \leftarrow z$ ; $z_0 \leftarrow ((id_s, id_r, m, ad, c_{pke}), \sigma)$
$((id_s^*, id_r^*, m^*, ad^*), (c_{pke}^*, \sigma^*)) \leftarrow z^*$ ; $z_1 \leftarrow ((id_s^*, id_r^*, m^*, ad^*, c_{pke}^*), \sigma^*)$
Return $\mathsf{R}^*.\mathsf{Vf}(z_0, z_1)$
```

Figure 18: Relaxing relation $\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*]$.

ciphertext (c_{pke}, σ) to an *unintended* message. To ensure this, we require that MRPKE is robust, for a robustness notion that we define below. Note that finding a new key pair that decrypts the ciphertext to the *original* message will not help the adversary to win the game because then the decryption will fail by not finding the corrupted recipient's identity in recipient set $\mathcal{I}$.

The necessity for MRPKE to be robust is inherent in the construction of IMSG-SC, even if its authenticity was required only in the *outsider* setting. In the outsider setting, the authenticity game requires the adversary to use an honest recipient for the attack. But one would still need to make sure that an MRPKE ciphertext is unlikely to be decrypted to an unintended message using honestly generated keys of an unintended recipient. So this would still require a weak notion of robustness from MRPKE. This could have been avoided if the recipient's identity was directly signed by DS, e.g. by adding id_r to $\langle c_{pke}, ad\rangle$.

We define unforgeability UF of a digital signature scheme with respect to a relaxing relation R, such that the standard unforgeability is captured with respect to R_{m} and the strong unforgeability is captured with respect to R_{id}. We show that if DS is UF-secure with respect to a relaxing relation $\mathsf{R}^* \in \{\mathsf{R}_{\mathsf{m}}, \mathsf{R}_{\mathsf{id}}\}$ then SC is AUTH-secure with respect to the corresponding parameterized relaxing relation $\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*]$, which we define below. ECDSA signatures are not strongly unforgeable [49], so iMessage is AUTH-secure with respect to $\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}_{\mathsf{m}}]$.

<u>Relaxing relation IMSG-AUTH-REL.</u> Let R_{m} and R_{id} be the relaxing relations defined in Section 3. Let $\mathsf{R}^* \in \{\mathsf{R}_{\mathsf{m}}, \mathsf{R}_{\mathsf{id}}\}$. Then $\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*]$ is the relaxing relation as defined in Fig. 18. Note that

$$\mathsf{R}_{\mathsf{id}} = \mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}_{\mathsf{id}}] \subset \mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}_{\mathsf{m}}] \subset \mathsf{R}_{\mathsf{m}},$$

where AUTH-security with respect to R_{id} captures the stronger security definition due to imposing the least number of restrictions regarding which queries are permitted to oracle VerDec. Relaxing relation $\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}_{\mathsf{m}}]$ does not allow adversary to win the authenticity game by only mauling the signature σ and not changing anything else.

<u>Unforgeability of digital signatures.</u> Consider game G^{uf} of Fig. 19, associated to a digital signature scheme DS, a relaxing relation R, and an adversary $\mathcal{F}$. The advantage of $\mathcal{F}$ in breaking the UF-security of DS with respect to R is defined as $\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{DS},\mathsf{R}}(\mathcal{F}) = \Pr[\mathrm{G}^{\mathsf{uf}}_{\mathsf{DS},\mathsf{R},\mathcal{F}}]$. UF-security with respect to R_{m} captures the standard unforgeability, and UF-security with respect to R_{id} captures the strong unforgeability.

<u>Robustness of MRPKE.</u> Consider game $\mathrm{G}^{\mathsf{rob}}$ of Fig. 20, associated to a multi-recipient public-key encryption scheme MRPKE and an adversary $\mathcal{G}$. The advantage of $\mathcal{G}$ in breaking the ROB-security of MRPKE is defined as $\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{MRPKE}}(\mathcal{G}) = \Pr[\mathrm{G}^{\mathsf{rob}}_{\mathsf{MRPKE},\mathcal{G}}]$. Adversary $\mathcal{G}$ has access to oracle Enc that returns MRPKE ciphertexts encrypting message m to every recipient from set $\mathcal{R}$. Set $\mathcal{R}$ contains pairs (id, ek) each containing a user identity $id \in \{0,1\}^*$ and an MRPKE encryption key ek. In order to win the game, adversary $\mathcal{G}$ has to find a key pair (ek, dk) that decrypts a ciphertext returned by Enc to a message that is different from the originally encrypted message. Throughout the game, adversary is allowed to use arbitrary MRPKE keys (including maliciously generated key

```
Game $\mathrm{G}^{\mathsf{uf}}_{\mathsf{DS},\mathsf{R},\mathcal{F}}$
$(id, m, \sigma) \leftarrow\$ \mathcal{F}^{\textsc{NewUser},\textsc{Exp},\textsc{Sign}}$ ; If (not $\mathsf{initialized}[id]$) or $\mathsf{exp}[id]$ then return false
$z_0 \leftarrow ((id, m), \sigma)$ ; If $\exists z_1 \in Q$: $\mathsf{R.Vf}(z_0, z_1)$ then return false
$d \leftarrow \mathsf{DS.Ver}(\mathsf{vk}[id], m, \sigma)$ ; Return $d$

$\textsc{NewUser}(id)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(vk, tk) \leftarrow\$ \mathsf{DS.Kg}$ ; $\mathsf{vk}[id] \leftarrow vk$ ; $\mathsf{tk}[id] \leftarrow tk$ ; Return $vk$

$\textsc{Exp}(id)$
If not $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{tk}[id]$

$\textsc{Sign}(id, m)$
If not $\mathsf{initialized}[id]$ then return $\perp$
$\sigma \leftarrow\$ \mathsf{DS.Sig}(\mathsf{tk}[id], m)$ ; $Q \leftarrow Q \cup \{((id, m), \sigma)\}$ ; Return $\sigma$
```

Figure 19: Game defining unforgeability of digital signature scheme DS with respect to relaxing relation R.

```
Game $\mathrm{G}^{\mathsf{rob}}_{\mathsf{MRPKE},\mathcal{G}}$
$\pi \leftarrow \mathsf{MRPKE.Setup}$ ; $(ek, dk, c) \leftarrow\$ \mathcal{G}^{\textsc{Enc}}(\pi)$
$m_1 \leftarrow \mathsf{MRPKE.Dec}(\pi, ek, dk, c)$
Return $m_1 \neq \perp$ and $(\exists (m_0, c) \in W : m_0 \neq m_1)$

$\textsc{Enc}(\mathcal{R}, m)$
$\mathcal{C} \leftarrow\$ \mathsf{MRPKE.Enc}(\pi, \mathcal{R}, m)$
For each $(id, c) \in \mathcal{C}$ do $W \leftarrow W \cup \{(m, c)\}$
Return $\mathcal{C}$
```

```
Game $\mathrm{G}^{\mathsf{rob}}_{\mathsf{PKE},\mathcal{G}}$
$(i, ek, dk) \leftarrow\$ \mathcal{G}^{\textsc{Enc}}$
If $i \notin [n]$ then return false
$m \leftarrow \mathsf{PKE.Dec}(ek, dk, \mathsf{c}[i])$
Return $m \neq \perp$ and $m \neq \mathsf{m}[i]$

$\textsc{Enc}(ek, m)$
$c \leftarrow\$ \mathsf{PKE.Enc}(ek, m)$ ; $n \leftarrow n + 1$
$\mathsf{m}[n] \leftarrow m$ ; $\mathsf{c}[n] \leftarrow c$ ; Return $c$
```

Figure 20: Games defining robustness of multi-recipient public-key encryption scheme MRPKE, and robustness of public-key encryption scheme PKE.

pairs that do not provide decryption correctness). However, oracle Enc runs algorithm MRPKE.Enc with honestly sampled random coins.

Theorem 5.2 *Let* MRPKE *be a multi-recipient public-key encryption scheme. Let* DS *be a digital signature scheme. Let* $\mathsf{SC} = \mathsf{IMSG\text{-}SC}[\mathsf{MRPKE}, \mathsf{DS}]$. *Let* $\mathsf{R}^* \in \{\mathsf{R_m}, \mathsf{R_{id}}\}$. *Let* $\mathcal{F}_{\mathsf{SC}}$ *be an adversary against the* AUTH*-security of* SC *with respect to relaxing relation* $\mathsf{R} = \mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*]$. *Then we build an adversary* $\mathcal{F}_{\mathsf{DS}}$ *against the* UF*-security of* DS *with respect to* R^**, and an adversary* $\mathcal{G}$ *against the* ROB*-security of* MRPKE *such that*

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}}(\mathcal{F}_{\mathsf{SC}}) \leq \mathsf{Adv}^{\mathsf{uf}}_{\mathsf{DS},\mathsf{R}^*}(\mathcal{F}_{\mathsf{DS}}) + \mathsf{Adv}^{\mathsf{rob}}_{\mathsf{MRPKE}}(\mathcal{G}).$$

The proof of Theorem 5.2 is in Appendix I.

Reduction showing robustness of MRPKE. The ciphertext of $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE}[\mathsf{EMDK}, \mathsf{PKE}]$ is a pair (c_{se}, c_{pke}), where c_{se} is an EMDK ciphertext encrypting some $m^* = \langle m, id_s, \mathcal{I}\rangle$, and c_{pke} is a PKE ciphertext encrypting the corresponding EMDK key k. The decryption algorithm of MRPKE first uses the PKE key pair (ek, dk) to decrypt c_{pke}, and then uses the recovered EMDK key k to decrypt c_{se}. We show that just one of PKE and EMDK being robust implies that MRPKE is also robust. Our definition of robustness for public-key encryption requires that it is hard to find a key pair (ek, dk) that decrypts an honestly produced ciphertext to a plaintext that is different from the originally encrypted message. If this condition holds for PKE, then clearly MRPKE is robust regardless of whether EMDK is robust. On the other hand, if PKE is not robust, then the robustness of EMDK (as defined in Section 4) would guarantee that the adversary is unlikely to decrypt c_{se} to a message other than m^* even if it has full control over the choice of EMDK key k. It is not known whether RSA-OAEP is robust, so our concrete security analysis of $\mathsf{iMessage}$ in Section 5.5 will rely entirely on the robustness of $\mathsf{EMDK} = \mathsf{IMSG\text{-}EMDK}$.

<u>Robustness of PKE.</u> Consider game $\mathrm{G}^{\mathsf{rob}}$ of Fig. 20, associated to a public-key encryption scheme PKE and an adversary $\mathcal{G}$. The advantage of $\mathcal{G}$ in breaking the ROB-security of PKE is defined as $\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{PKE}}(\mathcal{G}) = \Pr[\mathrm{G}^{\mathsf{rob}}_{\mathsf{PKE},\mathcal{G}}]$. The adversary has access to oracle $\textsc{Enc}$ that takes ek, m as input to return a ciphertext c computed by running $\mathsf{PKE.Enc}(ek, m)$ with honestly sampled random coins; the adversary can call this oracle many times. In order to win the game, the adversary has to provide a key pair ek, dk that decrypts any of these ciphertexts (produced in $\textsc{Enc}$) to a message that is different from the message that was used to produce this specific ciphertext. Note that the key pair ek, dk is allowed to be maliciously generated and even malformed.

The notions of robustness for public-key encryption were first formalized by Abdalla et al. [1], and later extended by Farshim et al. [46]. Our security notion requires the adversary to come up with a ciphertext that is mapped to two *distinct* messages; this was not required by prior security notions. The prior security notions require the adversary to come up with two *different* key pairs that map some ciphertext to valid messages. In contrast, our security notion also allows adversary to win by coming up with a single malformed key pair that can encrypt some message m into a ciphertext c that decrypts to a different message $m' \neq m$ (since decryption correctness is only required for honestly generated keys). So our security notion is incomparable to prior notions.

Theorem 5.3 *Let* EMDK *be an encryption scheme under message derived keys. Let* PKE *be a public-key encryption scheme with* $\mathsf{PKE.In} = \{0,1\}^{\mathsf{EMDK.kl}}$. *Let* $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE}[\mathsf{EMDK}, \mathsf{PKE}]$. *Let* $\mathcal{G}_{\mathsf{MRPKE}}$ *be an adversary against the* ROB*-security of* MRPKE. *Then we build an adversary* $\mathcal{G}_{\mathsf{EMDK}}$ *against the* ROB*-security of* EMDK *such that*

$$\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{MRPKE}}(\mathcal{G}_{\mathsf{MRPKE}}) \leq \mathsf{Adv}^{\mathsf{rob}}_{\mathsf{EMDK}}(\mathcal{G}_{\mathsf{EMDK}}),$$

and an adversary $\mathcal{G}_{\mathsf{PKE}}$ *against the* ROB*-security of* PKE *such that*

$$\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{MRPKE}}(\mathcal{G}_{\mathsf{MRPKE}}) \leq \mathsf{Adv}^{\mathsf{rob}}_{\mathsf{PKE}}(\mathcal{G}_{\mathsf{PKE}}).$$

The proof of Theorem 5.3 is in Appendix J.

5.4 Privacy of iMessage

In this section we reduce the PRIV-security of $\mathsf{SC} = \mathsf{IMSG\text{-}SC}[\mathsf{MRPKE}, \mathsf{DS}]$ to the INDCCA-security of MRPKE, then reduce the INDCCA-security of $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE}[\mathsf{EMDK}, \mathsf{PKE}]$ to the AE-security of EMDK and the INDCCA-security of PKE. The reductions are straightforward.

An adversary attacking the PRIV-security of SC is allowed to query oracle LR and get a challenge ciphertext from an exposed sender as long as the recipient is honest. This means that the adversary can use the sender's DS signing key to arbitrarily change associated data ad and signature σ of any challenge ciphertext prior to querying it to oracle $\textsc{VerDec}$. Our security reduction for PRIV-security of SC will be with respect to a relation that prohibits the adversary from trivially winning this way. Note that if $\mathsf{IMSG\text{-}SC}$ was defined to instead put ad inside $\langle m, id_s, \mathcal{I}\rangle$, then our security reduction would be able to show the PRIV-security of SC with respect to $\mathsf{R_{id}}$ assuming DS

```
$\underline{\mathsf{IMSG\text{-}PRIV\text{-}REL.Vf}(z, z^*)}$
$((id_s, id_r, m, ad), (c_{pke}, \sigma)) \leftarrow z$ ; $((id_s^*, id_r^*, m^*, ad^*), (c_{pke}^*, \sigma^*)) \leftarrow z^*$
Return $(id_s, id_r, m, c_{pke}) = (id_s^*, id_r^*, m^*, c_{pke}^*)$
```

Figure 21: Relaxing relation IMSG-PRIV-REL.

had *unique* signatures. However, ECDSA does not have this property (for the same reason it is not strongly unforgeable, as explained in [49]).

<u>Relaxing relation IMSG-PRIV-REL.</u> Let IMSG-PRIV-REL be the relaxing relation defined in Fig. 21. It first discards the associated data ad and the signature σ, and then compares the resulting tuples against each other. This reflects the intuition that an adversary can trivially change the values of ad and σ in any challenge ciphertext when attacking the PRIV-security of IMSG-SC.

Theorem 5.4 *Let* MRPKE *be a multi-recipient public-key encryption scheme. Let* DS *be a digital signature scheme. Let* $\mathsf{SC} = \mathsf{IMSG\text{-}SC}[\mathsf{MRPKE}, \mathsf{DS}]$. *Let* $\mathcal{D}_{\mathsf{SC}}$ *be an adversary against the* PRIV-*security of* SC *with respect to the relaxing relation* $\mathsf{R} = \mathsf{IMSG\text{-}PRIV\text{-}REL}$. *Then we build an adversary* $\mathcal{D}_{\mathsf{MRPKE}}$ *against the* INDCCA-*security of* MRPKE *such that*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}_{\mathsf{SC}}) \leq \mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{MRPKE}}(\mathcal{D}_{\mathsf{MRPKE}}).$$

The proof of Theorem 5.4 is in Appendix K.

Theorem 5.5 *Let* EMDK *be an encryption scheme under message derived keys. Let* PKE *be a public-key encryption scheme with input set* $\mathsf{PKE.In} = \{0,1\}^{\mathsf{EMDK.kl}}$. *Let* $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE}[\mathsf{EMDK}, \mathsf{PKE}]$. *Let* $\mathcal{D}_{\mathsf{MRPKE}}$ *be an adversary against the* INDCCA-*security of* MRPKE. *Then we build an adversary* $\mathcal{D}_{\mathsf{PKE}}$ *against the* INDCCA-*security of* PKE, *and an adversary* $\mathcal{D}_{\mathsf{EMDK}}$ *against the* AE-*security of* EMDK *such that*

$$\mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{MRPKE}}(\mathcal{D}_{\mathsf{MRPKE}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{PKE}}(\mathcal{D}_{\mathsf{PKE}}) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{EMDK}}(\mathcal{D}_{\mathsf{EMDK}}).$$

The proof of Theorem 5.5 is in Appendix L.

<u>Privacy in the outsider setting.</u> In the outsider setting, the PRIV-security of SC with respect to relaxing relation $\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*]$ can be shown assuming UF-security of DS with respect to R^* (same as in Section 5.3) and the INDCPA-security of MRPKE. The latter can be further reduced to the INDCPA-security of PKE and the IND-security of EMDK.

5.5 Concrete security of iMessage

In this section we summarize the results concerning the security of our iMessage-based signcryption scheme. For simplicity, we use the constructions and primitives from all across our work without formally redefining each of them.

<u>Corollary for abstract schemes.</u> Let SC be the iMessage-based signcryption scheme, defined based on the appropriate underlying primitives. Let $\mathsf{R}_{\mathsf{auth}} = \mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*]$ and $\mathsf{R}_{\mathsf{priv}} = \mathsf{IMSG\text{-}PRIV\text{-}REL}$. Then for any adversary $\mathcal{F}_{\mathsf{SC}}$ attacking the AUTH-security of SC we can build new adversaries such that:

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}}}(\mathcal{F}_{\mathsf{SC}}) \leq \mathsf{Adv}^{\mathsf{uf}}_{\mathsf{DS},\mathsf{R}^*}(\mathcal{F}_{\mathsf{DS}}) + \min(\mathsf{Adv}^{\mathsf{rob}}_{\mathsf{PKE}}(\mathcal{G}_{\mathsf{PKE}}), \alpha),$$

where

$$\alpha = \mathsf{Adv}^{\mathsf{unique}}_{\mathsf{SE}}(\mathcal{U}_0) + \mathsf{Adv}^{\mathsf{wrob}}_{\mathsf{SE},\mathsf{F.ol}}(\mathcal{G}_{\mathsf{SE}}).$$

For any adversary $\mathcal{D}_{\mathsf{SC}}$ attacking the PRIV-security of SC, making q_{LR} queries to LR oracle and q_{RO} queries to RO oracle, we build new adversaries such that:

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}_{\mathsf{priv}}}(\mathcal{D}_{\mathsf{SC}}) \leq 2 \cdot (\beta + \gamma) + \mathsf{Adv}^{\mathsf{otind}}_{\mathsf{SE}}(\mathcal{D}_{\mathsf{SE}}),$$

where

$$\beta = \mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{PKE}}(\mathcal{D}_{\mathsf{PKE}}) + \mathsf{Adv}^{\mathsf{unique}}_{\mathsf{SE}}(\mathcal{U}_1) + \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{F}}(\mathcal{H}) + \mathsf{Adv}^{\mathsf{pkr}}_{\mathsf{SE},\mathsf{F.kl}}(\mathcal{P}),$$

$$\gamma = \frac{(2 \cdot q_{\mathrm{RO}} + q_{\mathrm{LR}} - 1) \cdot q_{\mathrm{LR}}}{2^{\mathsf{F.kl}+1}}.$$

<u>Bit-security of iMessage.</u> We now assess the concrete security of $\mathsf{iMessage}$ when the abstract schemes that constitute SC are instantiated with real-world primitives. First, note that $\mathsf{Adv}^{\mathsf{unique}}_{\mathsf{SE}}(\mathcal{U}) = 0$ for any $\mathcal{U}$ when SE is AES-CTR. We will approximate the bit-security of SC based on the other terms above.

We assume that ECDSA with 256-bit keys (on the NIST P-256 curve) has 128 bits of UF-security with respect to $\mathsf{R_m}$ [34, 10]. We assume that RSA-OAEP with 1280-bit keys has 80 bits of INDCCA-security [34, 57]. SE is AES-CTR with key length $\mathsf{SE.kl}$; we assume that SE has $\mathsf{SE.kl}$ bits of OTIND-security.

For every other term used above, we approximate the corresponding bit-security based on the advantage ϵ and the runtime T of the best adversary we can come up with. For simplicity, we model F as the random oracle and we model SE as the ideal cipher. This simplifies the task of finding the "best possible" adversary against each security notion and then calculating its advantage. In each case we consider either a constant-time adversary making a single guess in its security game (achieving some advantage ϵ in time $T \approx 1$), or an adversary that runs a birthday attack (achieving advantage $\epsilon \geq 0.3 \cdot \frac{q \cdot (q-1)}{N}$ in time $T \approx q \cdot \log_2 q$ for $q = \sqrt{2N}$). We use the following adversaries:

(i) Assume SE is AES-CTR where AES modeled as the ideal cipher with block length 128. In game $\mathrm{G}^{\mathsf{wrob}}_{\mathsf{SE},\mathsf{F.ol},\mathcal{G}}$ consider an adversary $\mathcal{G}$ that repeatedly queries its oracle $\textsc{Enc}$ on inputs (r_0, m) where all $r_0 \in \{0,1\}^{\mathsf{F.kl}}$ are distinct and all $m \in \{0,1\}^{128}$ are distinct. The adversary wins if a collision occurs across the 128-bit outputs of $\mathsf{SE.Enc}$. Then $\epsilon = \mathsf{Adv}^{\mathsf{wrob}}_{\mathsf{SE},\mathsf{F.ol}}(\mathcal{G}_{\mathsf{SE}}) \geq 0.3 \cdot \frac{q_{\textsc{Enc}} * (q_{\textsc{Enc}} - 1)}{2^{128}}$ and $T = q_{\textsc{Enc}} \cdot \log_2 q_{\textsc{Enc}}$ for $q_{\textsc{Enc}} = \sqrt{2^{128+1}}$.

(ii) In game $\mathrm{G}^{\mathsf{tcr}}_{\mathsf{F},\mathcal{H}}$ consider an adversary $\mathcal{H}$ that queries its oracle $\textsc{NewKey}(x_0)$ for any $x_0 \in \{0,1\}^*$ and then makes a guess $(1, x_1)$ for any $x_0 \neq x_1$. Then $\epsilon = \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{F}}(\mathcal{H}) = 2^{-\mathsf{F.ol}}$ and $T \approx 1$ in the random oracle model.

(iii) In game $\mathrm{G}^{\mathsf{pkr}}_{\mathsf{SE},\mathsf{F.kl},\mathcal{P}}$ consider an adversary $\mathcal{P}$ that makes a single call to $\textsc{Enc}$ and then randomly guesses any key prefix $p \in \{0,1\}^{\mathsf{F.kl}}$. Then $\epsilon = \mathsf{Adv}^{\mathsf{pkr}}_{\mathsf{SE},\mathsf{F.kl}}(\mathcal{P}) = 2^{-\mathsf{F.kl}}$ and $T \approx 1$ in the ideal cipher model.

(iv) The term γ upper bounds the probability of an adversary finding a collision when running the birthday attack (in the random oracle model). The corresponding lower bound (for $q_{\mathrm{RO}} = 0$) is $\epsilon \geq 0.3 \cdot \frac{q_{\mathrm{LR}} \cdot (q_{\mathrm{LR}} - 1)}{2^{\mathsf{F.kl}}}$ with $T = q_{\mathrm{LR}} \cdot \log_2 q_{\mathrm{LR}}$ and $q_{\mathrm{LR}} = \sqrt{2^{\mathsf{F.kl}+1}}$.

We wrote a script that combines all of the above to find the lower bound for the bit-security of SC (with respect to PRIV and AUTH security notions) for different choices of $\mathsf{SE.kl}$, $\mathsf{F.kl}$ and $\mathsf{F.ol}$.

SE.kl	F.kl	F.ol	PRIV bit-security	AUTH bit-security
128	88	40	39	71
	80	48	45	
	72	56	41	
192	128	64	63	
	120	72	66	
	112	80	62	
256	168	88	79	
	160	96	79	

Figure 22: Lower bounds for bit-security of SC across different parameter choices.

This assumes that the above adversaries are optimal, and computes the lower bound according to Section 2. Fig. 2 (in Section 1) shows the bit-security lower bounds with respect to privacy, depending on the choice of symmetric key length SE.kl and authentication tag length F.ol. Fig. 22 shows the choices of F.kl and F.ol that yield the best lower bounds for the bit-security of PRIV for each $\mathsf{SE.kl} \in \{128, 192, 256\}$. According to our results, the security of the iMessage-based signcryption scheme would slightly improve if the value of F.ol was chosen to be 48 instead of 40. The bit-security of SC with respect to AUTH is constant because it does not depend on the values of SE.kl, F.kl, F.ol. The assumption that RSA-OAEP with 1280-bit long keys has 80 bits of INDCCA-security limits the bit-security that can be achieved when $\mathsf{SE.kl} = 256$; otherwise, the PRIV bit-security for $\mathsf{SE.kl} = 256$ would allow a lower bound of 86 bits. But note that using $\mathsf{SE.kl} \in \{192, 256\}$ is likely not possible while maintaining the backward-compatibility of iMessage.

Acknowledgments

Igors Stepanovs's work was done while at UCSD. We thank Adina Wollner, Wei Dai and Joseph Jaeger for discussions and insights.

References

[1] M. Abdalla, M. Bellare, and G. Neven. Robust encryption. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Heidelberg, Feb. 2010. 4, 13, 24

[2] J. Alwen, S. Coretti, and Y. Dodis. The double ratchet: Security notions, proofs, and modularization for the signal protocol. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 129–158. Springer, Heidelberg, May 2019. 2

[3] J. H. An. Authenticated encryption in the public-key setting: Security notions and analyses. Cryptology ePrint Archive, Report 2001/079, 2001. `http://eprint.iacr.org/2001/079`. 32

[4] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 83–107. Springer, Heidelberg, Apr. / May 2002. 2, 3, 12, 20, 32, 33, 34

[5] Apple. iOS security: iOS 12.3. Technical whitepaper, `https://www.apple.com/business/docs/site/iOS_Security_Guide.pdf`, May 2019. Accessed: 2019-09-26. 2, 16, 18

[6] C. Badertscher, F. Banfi, and U. Maurer. A constructive perspective on signcryption security. In D. Catalano and R. De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 102–120. Springer, Heidelberg, Sept. 2018. 33

[7] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. In D. Naccache and P. Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 80–98. Springer, Heidelberg, Feb. 2002. 32

[8] F. Bao and R. H. Deng. A signcryption scheme with signature directly verifiable by public key. In H. Imai and Y. Zheng, editors, *PKC'98*, volume 1431 of *LNCS*, pages 55–59. Springer, Heidelberg, Feb. 1998. 33

[9] M. Barbosa and P. Farshim. Certificateless signcryption. In M. Abe and V. Gligor, editors, *ASIACCS 08*, pages 369–372. ACM Press, Mar. 2008. 33

[10] E. Barker. Recommendation for key management part 1: General (revision 5). *NIST special publication*, 800(57):1–174, 2019. 26

[11] P. S. L. M. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In B. K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 515–532. Springer, Heidelberg, Dec. 2005. 33

[12] M. Bellare, A. Boldyreva, K. Kurosawa, and J. Staddon. Multi-recipient encryption schemes: Efficient constructions and their security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, 2007. 42

[13] M. Bellare, A. Boldyreva, K. Kurosawa, and J. Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Trans. Information Theory*, 53(11):3927–3943, 2007. 4

[14] M. Bellare, A. Boldyreva, and J. Staddon. Randomness re-use in multi-recipient encryption schemeas. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 85–99. Springer, Heidelberg, Jan. 2003. 42

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

[16] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, Dec. 2000. 3

[17] M. Bellare, R. Ng, and B. Tackmann. Nonces are noticed: AEAD revisited. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 235–265. Springer, Heidelberg, Aug. 2019. 3

[18] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 6, 42

[19] M. Bellare and P. Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 470–484. Springer, Heidelberg, Aug. 1997. 41

[20] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 7, 37, 44, 47, 48, 53

[21] M. Bellare, A. C. Singh, J. Jaeger, M. Nyayapati, and I. Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 619–650. Springer, Heidelberg, Aug. 2017. 2

[22] T. E. Bjørstad and A. W. Dent. Building better signcryption schemes with tag-KEMs. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 491–507. Springer, Heidelberg, Apr. 2006. 33

[23] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In K. Nyberg and H. M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, Aug. 2003. 4

[24] P. Bose, V. T. Hoang, and S. Tessaro. Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 468–499. Springer, Heidelberg, Apr. / May 2018. 3

[25] X. Boyen. Multipurpose identity-based signcryption (a swiss army knife for identity-based cryptography). In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 383–399. Springer, Heidelberg, Aug. 2003. 33

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

[27] R. Canetti, Y. T. Kalai, M. Varia, and D. Wichs. On symmetric encryption and point obfuscation. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 52–71. Springer, Heidelberg, Feb. 2010. 4, 13

[28] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Heidelberg, Aug. 2003. 3, 8, 11

[29] L. Chen and J. Malone-Lee. Improved identity-based signcryption. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 362–379. Springer, Heidelberg, Jan. 2005. 33

[30] D. Chiba, T. Matsuda, J. C. N. Schuldt, and K. Matsuura. Efficient generic constructions of signcryption with insider security in the multi-user setting. In J. Lopez and G. Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 220–237. Springer, Heidelberg, June 2011. 33

[31] S. S. M. Chow, S.-M. Yiu, L. C. K. Hui, and K. P. Chow. Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity. In J. I. Lim and D. H. Lee, editors, *ICISC 03*, volume 2971 of *LNCS*, pages 352–369. Springer, Heidelberg, Nov. 2004. 33

[32] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the Signal messaging protocol. In *Proc. IEEE European Symposium on Security and Privacy (EuroS&P) 2017*. IEEE, April 2017. To appear. 2

[33] Common Vulnerabilities and Exposures system. Cve-2016-1788. Available at `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-1788`. Accessed: 2020-2-20. 2

[34] Damien Giry. Cryptographic key length recommendation. Available at `https://www.keylength.com`. Accessed: 2019-11-20. 26

[35] P. Datta, R. Dutta, and S. Mukhopadhyay. Compact attribute-based encryption and signcryption for general circuits from multilinear maps. In A. Biryukov and V. Goyal, editors, *INDOCRYPT 2015*, volume 9462 of *LNCS*, pages 3–24. Springer, Heidelberg, Dec. 2015. 33

[36] P. Datta, R. Dutta, and S. Mukhopadhyay. Functional signcryption: Notion, construction, and applications. In M. H. Au and A. Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 268–288. Springer, Heidelberg, Nov. 2015. 33

[37] A. W. Dent. Hybrid signcryption schemes with insider security. In C. Boyd and J. M. G. Nieto, editors, *ACISP 05*, volume 3574 of *LNCS*, pages 253–266. Springer, Heidelberg, July 2005. 33

[38] A. W. Dent. Hybrid signcryption schemes with outsider security. In J. Zhou, J. Lopez, R. H. Deng, and F. Bao, editors, *ISC 2005*, volume 3650 of *LNCS*, pages 203–217. Springer, Heidelberg, Sept. 2005. 33

[39] A. W. Dent. Aggregate signcryption. Cryptology ePrint Archive, Report 2012/200, 2012. `http://eprint.iacr.org/2012/200`. 33

[40] A. W. Dent, M. Fischlin, M. Manulis, M. Stam, and D. Schröder. Confidential signatures and deterministic signcryption. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 462–479. Springer, Heidelberg, May 2010. 33

[41] A. W. Dent and Y. Zheng, editors. *Practical Signcryption*. ISC. Springer, Heidelberg, 2010. 32

[42] Y. Dodis, M. J. Freedman, S. Jarecki, and S. Walfish. Optimal signcryption from any trapdoor permutation. Cryptology ePrint Archive, Report 2004/020, 2004. `http://eprint.iacr.org/2004/020`. 33

[43] S. Duan and Z. Cao. Efficient and provably secure multi-receiver identity-based signcryption. In L. M. Batten and R. Safavi-Naini, editors, *ACISP 06*, volume 4058 of *LNCS*, pages 195–206. Springer, Heidelberg, July 2006. 33

[44] F. B. Durak and S. Vaudenay. Bidirectional asynchronous ratcheted key agreement with linear complexity. In N. Attrapadung and T. Yagi, editors, *IWSEC 19*, volume 11689 of *LNCS*, pages 343–362. Springer, Heidelberg, Aug. 2019. 2

[45] L. El Aimani. Generic constructions for verifiable signcryption. In H. Kim, editor, *ICISC 11*, volume 7259 of *LNCS*, pages 204–218. Springer, Heidelberg, Nov. / Dec. 2012. 33

[46] P. Farshim, B. Libert, K. G. Paterson, and E. A. Quaglia. Robust encryption, revisited. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 352–368. Springer, Heidelberg, Feb. / Mar. 2013. 4, 13, 24

[47] P. Farshim, C. Orlandi, and R. Roşie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symm. Cryptol.*, 2017(1):449–473, 2017. 4

[48] V. Fehr and M. Fischlin. Sanitizable signcryption: Sanitization over encrypted data (full version). Cryptology ePrint Archive, Report 2015/765, 2015. `http://eprint.iacr.org/2015/765`. 33

[49] M. Fersch, E. Kiltz, and B. Poettering. On the provable security of (EC)DSA signatures. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1651–1662. ACM Press, Oct. 2016. 22, 25

[50] M. Gagné, S. Narayan, and R. Safavi-Naini. Threshold attribute-based signcryption. In J. A. Garay and R. D. Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 154–171. Springer, Heidelberg, Sept. 2010. 33

[51] C. Garman, M. Green, G. Kaptchuk, I. Miers, and M. Rushanan. Dancing on the lip of the volcano: Chosen ciphertext attacks on apple iMessage. In T. Holz and S. Savage, editors, *USENIX Security 2016*, pages 655–672. USENIX Association, Aug. 2016. 2, 3, 5, 16, 18, 19, 20, 21, 33, 34

[52] M. C. Gorantla, C. Boyd, and J. M. González Nieto. On the connection between signcryption and one-pass key establishment. In S. D. Galbraith, editor, *11th IMA International Conference on Cryptography and Coding*, volume 4887 of *LNCS*, pages 277–301. Springer, Heidelberg, Dec. 2007. 33

[53] J. Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 152–170. Springer, Heidelberg, Feb. 2004. 8, 11

[54] J. Jaeger and I. Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 33–62. Springer, Heidelberg, Aug. 2018. 2

[55] D. Jost, U. Maurer, and M. Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 159–188. Springer, Heidelberg, May 2019. 2

[56] K. Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In D. Naccache and P. Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 48–63. Springer, Heidelberg, Feb. 2002. 42

[57] A. K. Lenstra. Key length. Contribution to the handbook of information security. 2004. 26

[58] F. Li, M. Shirase, and T. Takagi. Certificateless hybrid signcryption. In *International Conference on Information Security Practice and Experience*, pages 112–123. Springer, 2009. 33

[59] B. Libert and J.-J. Quisquater. A new identity based signcryption scheme from pairings. In *Proceedings 2003 IEEE Information Theory Workshop (Cat. No. 03EX674)*, pages 155–158. IEEE, 2003. 33

[60] B. Libert and J.-J. Quisquater. Efficient signcryption with key privacy from gap Diffie-Hellman groups. In F. Bao, R. Deng, and J. Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 187–200. Springer, Heidelberg, Mar. 2004. 33

[61] B. Libert and J.-J. Quisquater. Improved signcryption from q-Diffie-Hellman problems. In C. Blundo and S. Cimato, editors, *SCN 04*, volume 3352 of *LNCS*, pages 220–234. Springer, Heidelberg, Sept. 2005. 33

[62] J. K. Liu, J. Baek, and J. Zhou. Online/offline identity-based signcryption revisited. In *Proceedings of the 6th International Conference on Information Security and Cryptology*, Inscrypt'10, pages 36–51, Berlin, Heidelberg, 2011. Springer-Verlag. 33

[63] J. Malone-Lee. Identity-based signcryption. Cryptology ePrint Archive, Report 2002/098, 2002. `http://eprint.iacr.org/2002/098`. 33

[64] J. Malone-Lee. A general construction for simultaneous signing and encrypting. In N. P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 116–135. Springer, Heidelberg, Dec. 2005. 33

[65] J. Malone-Lee and W. Mao. Two birds one stone: Signcryption using RSA. In M. Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 211–225. Springer, Heidelberg, Apr. 2003. 33

[66] T. Matsuda, K. Matsuura, and J. C. N. Schuldt. Efficient constructions of signcryption schemes and signcryption composability. In B. K. Roy and N. Sendrier, editors, *INDOCRYPT 2009*, volume 5922 of *LNCS*, pages 321–342. Springer, Heidelberg, Dec. 2009. 33

[67] U. Maurer. Constructive cryptography–a new paradigm for security definitions and proofs. In *Joint Workshop on Theory of Security and Applications*, pages 33–56. Springer, 2011. 33

[68] D. Micciancio and M. Walter. On the bit security of cryptographic primitives. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 3–28. Springer, Heidelberg, Apr. / May 2018. 7

[69] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. 41

[70] OpenIM wiki. iMessage. Available at `https://wiki.imfreedom.org/wiki/IMessage`. Accessed: 2019-11-21. 16

[71] T. Pandit, S. K. Pandey, and R. Barua. Attribute-based signcryption : Signer privacy, strong unforgeability and IND-CCA2 security in adaptive-predicates attack. In S. S. M. Chow, J. K. Liu, L. C. K. Hui, and S.-M. Yiu, editors, *ProvSec 2014*, volume 8782 of *LNCS*, pages 274–290. Springer, Heidelberg, Oct. 2014. 33

[72] J. Pieprzyk and D. Pointcheval. Parallel authentication and public-key encryption. In R. Safavi-Naini and J. Seberry, editors, *ACISP 03*, volume 2727 of *LNCS*, pages 387–401. Springer, Heidelberg, July 2003. 33

[73] B. Poettering and P. Rösler. Towards bidirectional ratcheted key exchange. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 3–32. Springer, Heidelberg, Aug. 2018. 2

[74] Quarkslab. iMessage privacy. Available at `https://blog.quarkslab.com/imessage-privacy.html`, Oct. 2013. Accessed: 2019-11-21. 16

[75] P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, Nov. 2002. 3

[76] S. S. D. Selvi, S. S. Vivek, and C. P. Rangan. Certificateless kem and hybrid signcryption schemes revisited. In *International Conference on Information Security Practice and Experience*, pages 294–307. Springer, 2010. 33

[77] S. S. D. Selvi, S. S. Vivek, and C. P. Rangan. Identity based public verifiable signcryption scheme. In S.-H. Heng and K. Kurosawa, editors, *ProvSec 2010*, volume 6402 of *LNCS*, pages 244–260. Springer, Heidelberg, Oct. 2010. 33

[78] S. S. D. Selvi, S. S. Vivek, J. Shriram, S. Kalaivani, and C. P. Rangan. Identity based aggregate signcryption schemes. In B. K. Roy and N. Sendrier, editors, *INDOCRYPT 2009*, volume 5922 of *LNCS*, pages 378–397. Springer, Heidelberg, Dec. 2009. 33

[79] S. S. D. Selvi, S. S. Vivek, D. Vinayagamurthy, and C. P. Rangan. ID based signcryption scheme in standard model. In T. Takagi, G. Wang, Z. Qin, S. Jiang, and Y. Yu, editors, *ProvSec 2012*, volume 7496 of *LNCS*, pages 35–52. Springer, Heidelberg, Sept. 2012. 33

[80] C. E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949. 42

[81] J.-B. Shin, K. Lee, and K. Shim. New DSA-verifiable signcryption schemes. In P. J. Lee and C. H. Lim, editors, *ICISC 02*, volume 2587 of *LNCS*, pages 35–47. Springer, Heidelberg, Nov. 2003. 33

[82] R. Steinfeld and Y. Zheng. A signcryption scheme based on integer factorization. In J. Pieprzyk, E. Okamoto, and J. Seberry, editors, *ISW 2000*, volume 1975 of *LNCS*, pages 308–322. Springer, Heidelberg, Dec. 2000. 32, 33

[83] Y. Wang, M. Manulis, M. H. Au, and W. Susilo. Relations among privacy notions for signcryption and key invisible "Sign-then-Encrypt". In C. Boyd and L. Simpson, editors, *ACISP 13*, volume 7959 of *LNCS*, pages 187–202. Springer, Heidelberg, July 2013. 33

[84] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 165–179. Springer, Heidelberg, Aug. 1997. 2, 32, 33

A Prior work on signcryption

Signcryption combines public-key encryption and signatures into a single cryptographic primitive that performs both functionalities simultaneously. The main goal of signcryption is to provide a better efficiency than by performing the encryption and signing separately. The concept and the first scheme were proposed by Zheng [84] in 1997, and over the years led to a wide body of research. In this section we summarize the prior work on signcryption, outlining the main research contributions in the literature. We note that a comprehensive analysis of early signcryption work was also provided by Dent and Zheng [41] in 2010.

SECURITY NOTIONS. The initial paper by Zheng [84] did not propose a security definition. The first security notion for authenticity of signcryption (but not for privacy) was defined by Steinfeld and Zheng [82]. Full security of signcryption (i.e. both authenticity and privacy) was concurrently defined in 2002 by An, Dodis and Rabin [4], and by Baek, Steinfeld and Zheng [7] (the former partially encompasses [3]) These papers define separate security notions for authenticity and privacy of signcryption. Furthermore, either security notion can be defined in *two-user* or *multi-user* setting, and in *outsider* or *insider* setting. These papers show that the security of signcryption in multi-user setting is not implied by its security in two-user setting, so the subsequent work uses multi-user definitions.

In the insider setting, the adversary has access either to sender's secret key (when attacking privacy), or to recipient's secret key (when attacking authenticity). In the outsider setting, the adversary does not have access to sender's and recipient's secret keys. The security definitions in the insider setting are strictly stronger than in the outsider setting. In the insider setting, the privacy of signcryption guarantees some form of forward security, and the authenticity guarantees non-repudiation; neither property is guaranteed in the outsider setting. But it might not be clear

```
EMDK.Enc(m)                                          | EMDK.Dec(k, c_se)
$k \leftarrow_\$ \{0,1\}^{\mathsf{SE.kl}}$ ; $c_{se} \leftarrow_\$ \mathsf{SE.Enc}(k, m)$ | $m \leftarrow \mathsf{SE.Dec}(k, c_{se})$
Return $(k, c_{se})$                                 | Return $m$
```

Figure 23: Basic EMDK scheme $\mathsf{EMDK} = \mathsf{BASIC\text{-}EMDK}[\mathsf{SE}]$.

that insider security is necessary to model the real-world requirements, so some of the subsequent work is done in the outsider setting. In a recent work, Badertscher, Banfi and Maurer [6] analyzed the security notions for signcryption in the constructive cryptography framework [67] and concluded that the insider security notions should be considered as the standard for signcryption.

Boyen [25] proposed additional security notions for signcryption called ciphertext unlinkability and ciphertext anonymity (the later also called key privacy). Libert and Quisquater [60] defined key invisibility of signcryption. Wang et al. [83] showed that key invisibility implies all other security notions for confidentiality of signcryption.

CONSTRUCTIONS. Prior work proposes a variety of generic constructions to build signcryption schemes. This includes generic compositions of encryption and signature schemes [4, 72, 66, 40, 45], encompassing "encrypt-then-sign", "sign-then-encrypt", "encrypt-and-sign", "commit-then-encrypt-and-sign", "encrypt-then-sign-then-encrypt", and more. More generic constructions are known from hybrid KEM-DEM techniques [38, 37, 64, 22, 52, 66, 58, 30] and from any trapdoor permutation [42].

Direct constructions of signcryption schemes are known from DH assumptions [84, 8, 81], from RSA problem or integer factorization [82, 65], and from hard problems in groups with bilinear maps [59, 60, 61].

EXTENSIONS. In order to address certificate management challenges in a public-key infrastructure, prior work considers different types of signcryption schemes. The solutions include identity-based signcryption [63, 25, 59, 31, 29, 11, 62, 77, 79] and certificateless signcryption [9, 58, 76]. This can be further generalized to obtain attribute-based signcryption [50, 71, 35] and functional signcryption [36].

Orthogonal to the certificate management problem, some of the prior work aims to build signcryption schemes with advanced functionality, such as deterministic signcryption [40], aggregate signcryption [78, 39], sanitizable signcryption [48], publicly verifiable signcryption [8, 31, 77, 45], and multi-recipient signcryption [43].

B Legacy design of iMessage

An attack against the privacy of iMessage was found in 2016 by Garman et al. [51]. In response, Apple updated the protocol design of iMessage and deployed the fix starting from iOS version 9.3 and Mac OS X version 10.11.4. In Section 5 we analyzed the current version of iMessage. In this section we provide the initial design of iMessage, we explain the attack by Garman et al. [51], and we introduce the notion of backward-compatibility for signcryption schemes.

The initial design of iMessage was the same as the current design (defined in Section 5.1), except it used a different EMDK scheme. We call it BASIC-EMDK and define it below. This EMDK scheme samples the encryption key k for SE uniformly at random, independently of the message. Note that instantiating SE with AES-CTR allows trivial attacks against both the AE-security and the ROB-security of BASIC-EMDK.

EMDK SCHEME BASIC-EMDK. Let SE be a symmetric encryption scheme. Then $\mathsf{EMDK} =$

$\mathsf{BASIC\text{-}EMDK[SE]}$ is the encryption scheme under message derived keys as defined in Fig. 23, with key length $\mathsf{EMDK.kl} = \mathsf{SE.kl}$.

<u>The GGKMR16 attack.</u> Let $\mathsf{SC} = \mathsf{IMSG\text{-}SC[MRPKE, DS]}$ for $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE[EMDK, PKE]}$ and $\mathsf{EMDK} = \mathsf{BASIC\text{-}EMDK}$. Then $\mathsf{BASIC\text{-}EMDK}$ is used to encrypt $m^* = \langle m, id_s, \mathcal{I}\rangle$. When instantiated, AES-CTR is used to encrypt m^* using a uniformly random key k. Due to the malleability of AES-CTR, an adversary can maul the $\mathsf{BASIC\text{-}EMDK}$ ciphertext to replace id_s with an arbitrary corrupted identity id_c of its own choice. Adversary then replaces the DS signature of MRPKE ciphertext with id_c's signature of the mauled ciphertext, and queries the VerDec oracle (in privacy game) to get the decryption of the originally encrypted message. This trivally breaks the privacy of the scheme, yielding a message-recovery attack. We formalized this adversary as $\mathcal{D}_{\mathsf{ADR02}}$ defined in Fig. 16 of Section 5.2, where it was used to attack the new $\mathsf{iMessage}$ design. However, this adversary has a significantly better advantage against the initial $\mathsf{iMessage}$ design: it achieves advantage 1. We formalize this claim as follows.

Proposition B.1 *Let* SE *be a symmetric encryption scheme. Let* $\mathsf{EMDK} = \mathsf{BASIC\text{-}EMDK[SE]}$. *Let* PKE *be a public-key encryption scheme with* $\mathsf{PKE.In} = \{0,1\}^{\mathsf{SE.kl}}$. *Let* $\mathsf{MRPKE} = \mathsf{IMSG\text{-}MRPKE[EMDK, PKE]}$. *Let* DS *be a digital signature scheme. Let* $\mathsf{SC} = \mathsf{IMSG\text{-}SC[MRPKE, DS]}$. *Consider the relaxing relation* $\mathsf{R_m}$ *defined in Section 3, and the adversary* $\mathcal{D}_{\mathsf{ADR02}}$ *defined in Section 5.2. If* SE *is AES-CTR with a fixed IV, then* $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R_m}}(\mathcal{D}_{\mathsf{ADR02}}) = 1$.

The basic attack idea used by this adversary was proposed by An, Dodis and Rabin [4] to show that two-user security of signcryption does not imply multi-user security of signcryption. In order to attack $\mathsf{iMessage}$, Garman et al. [51] adapted the attack to also use the malleability of AES-CTR. However, the implementation of $\mathsf{iMessage}$ encodes $m^* = \langle m, id_s, \mathcal{I}\rangle$ into a binary plist key-value data structure, and then compresses the result using the gzip compression format. In order to implement a practical attack against $\mathsf{iMessage}$, Garman et al. [51] had to develop a novel attack technique to deal with this encoding. Also note that in the practical (message-recovery) attack implemented by Garman et al. [51], there is no ciphertext decryption oracle, but instead they are able to use an oracle that returns a single bit indicating whether a ciphertext could be decrypted correctly.

<u>Backward-compatibility of signcryption.</u> We believe that in response to the attack by Garman et al. [51], the $\mathsf{iMessage}$ protocol was changed in a way that its new design is backward-compatible with the initial design. We now formalize the requirement of backward-compatility for signcryption schemes.

Let $\mathsf{SC}_0, \mathsf{SC}_1$ be any signcryption schemes. We say that SC_1 is *backward-compatible* with SC_0 if algorithm $\mathsf{SC}_0.\mathsf{VerDec}$ can be used to unsigncrypt ciphertexts produced by scheme SC_1. Formally, consider a new signcryption scheme SC_2 such that $\mathsf{SC}_2.\mathsf{Setup} = \mathsf{SC}_1.\mathsf{Setup}$, $\mathsf{SC}_2.\mathsf{Kg} = \mathsf{SC}_1.\mathsf{Kg}$, $\mathsf{SC}_2.\mathsf{SigEnc} = \mathsf{SC}_1.\mathsf{SigEnc}$, $\mathsf{SC}_2.\mathsf{VerDec} = \mathsf{SC}_0.\mathsf{VerDec}$, and $\mathsf{SC}_2.\mathsf{ID} = \mathsf{SC}_1.\mathsf{ID}$. We say that SC_1 is backward-compatible with SC_0 if the correctness condition holds for signcryption scheme SC_2. The scheme SC_2 in this claim models the interaction between a sender device using scheme SC_1 and a recipient device using scheme SC_0.

We stress that backward-compatibility does not guaranteee that schemes SC_1 and SC_2 satisfy any security notions, regardless of the guarantees provided by SC_0. It is easy to construct contrived examples where $\mathsf{SC}_1.\mathsf{SigEnc}$ appends all of its inputs (including secret key and plaintext) to the produced ciphertext. The scheme SC_1 will be backward-compatible with SC_0 if the latter correctly decrypts ciphertexts with arbitrary appended suffix strings, but schemes $\mathsf{SC}_1, \mathsf{SC}_2$ will not provide any security.

```
Game $\mathrm{G}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}},\mathcal{D}}$
$b \leftarrow_{\$} \{0,1\}$ ; $\pi \leftarrow_{\$} \mathsf{SC.Setup}$ ; $b' \leftarrow_{\$} \mathcal{D}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\textsc{LR},\textsc{VerDec}}(\pi)$ ; Return $b' = b$

$\textsc{NewH}(id)$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(pk, sk) \leftarrow_{\$} \mathsf{SC.Kg}(\pi)$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $pk$

$\textsc{NewC}(id, pk, sk)$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $\mathsf{true}$

$\textsc{Exp}(id)$
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\bot$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$

$\textsc{LR}(id_s, \mathcal{I}, m_0, m_1, ad)$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) or $|m_0| \neq |m_1|$ then return $\bot$
If $m_0 \neq m_1$ then
  If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\bot$
  For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{R} \leftarrow \emptyset$ ; For each $id \in \mathcal{I}$ do $\mathcal{R} \leftarrow \mathcal{R} \cup \{(id, \mathsf{pk}[id])\}$
$\mathcal{C} \leftarrow_{\$} \mathsf{SC.SigEnc}(\pi, id_s, \mathsf{pk}[id_s], \mathsf{sk}[id_s], \mathcal{R}, m_b, ad)$
For each $(id_r, c) \in \mathcal{C}$ do
  If $m_0 \neq m_1$ then
    $Q_{\mathsf{priv}} \leftarrow Q_{\mathsf{priv}} \cup \{((id_s, id_r, m_0, ad), c)\}$
    $Q_{\mathsf{priv}} \leftarrow Q_{\mathsf{priv}} \cup \{((id_s, id_r, m_1, ad), c)\}$
  Else $Q_{\mathsf{auth}} \leftarrow Q_{\mathsf{auth}} \cup \{((id_s, id_r, m_0, ad), c)\}$
Return $\mathcal{C}$

$\textsc{VerDec}(id_s, id_r, c, ad)$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return $(\bot$, "init"$)$
$m \leftarrow \mathsf{SC.VerDec}(\pi, id_s, \mathsf{pk}[id_s], id_r, \mathsf{pk}[id_r], \mathsf{sk}[id_r], c, ad)$
If $m = \bot$ then return $(\bot$, "dec"$)$
$z_0 \leftarrow ((id_s, id_r, m, ad), c)$
If $\exists z_1 \in Q_{\mathsf{priv}}$: $\mathsf{R}_{\mathsf{priv}}.\mathsf{Vf}(z_0, z_1)$ then return $(\bot$, "priv"$)$
If $\exists z_1 \in Q_{\mathsf{auth}}$: $\mathsf{R}_{\mathsf{auth}}.\mathsf{Vf}(z_0, z_1)$ then return $(m$, "auth"$)$
$\mathsf{cheated} \leftarrow \mathsf{exp}[id_s]$ ; If $\mathsf{cheated}$ then return $(m$, "cheat"$)$
If $b = 1$ then return $(m$, "ok"$)$ else return $(\bot$, "chal"$)$
```

Figure 24: Games defining the combined security of signcryption scheme SC, simultaneously capturing the authenticity of SC with respect to relaxing relation $\mathsf{R}_{\mathsf{auth}}$ and the privacy of SC with respect to relaxing relation $\mathsf{R}_{\mathsf{priv}}$.

C Combined security of signcryption

In this section we define a combined security notion for signcryption that simultaneously captures the authenticity and the privacy notions. We show that the combined security notion implies each of the two separate security notions. We also show that the separate security notions jointly imply the combined security notion for certain choices of relaxing relations.

COMBINED SECURITY OF SIGNCRYPTION. Consider game $\mathrm{G}^{\mathsf{sec}}$ of Fig. 24 associated to a signcryption scheme SC, relaxing relations $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$, and an adversary $\mathcal{D}$. The advantage of adversary $\mathcal{D}$ in breaking the SEC-security of SC with respect to $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$ is defined as $\mathsf{Adv}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}}}(\mathcal{D}) =$

$2\Pr[\mathrm{G}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}},\mathcal{D}}] - 1$.

The combined security notion is defined with respect to two relaxing relations: $\mathsf{R}_{\mathsf{auth}}$ specifying the restrictions for the authenticity purposes, and $\mathsf{R}_{\mathsf{priv}}$ specifying the restrictions for the privacy purposes. The game accordingly builds a separate set for each of the relaxing relations, where Q_{priv} contains information relevant to the privacy, and Q_{auth} contains information relevant to the authenticity. The oracles of the combined security game can be thought of as merging the corresponding oracles of the separate security games for authenticity and privacy; only the definition of oracle VerDec requires some caution. Specifically, the relaxing relation for privacy has to be checked before checking the relaxing relation for authenticity. And the last line of the oracle should only be reached if the adversary did not attempt to cheat by using an exposed sender.

Combined security in the outsider setting. To capture the notion of combined security in the *outsider* setting, consider the class of outsider adversaries that (i) never query oracle $\mathrm{LR}(id_s, \mathcal{I}, m_0, m_1, ad)$ when $\mathsf{exp}[id_s] = \mathsf{true}$, and (ii) never query oracle $\mathrm{VerDec}(id_s, id_r, c, ad)$ when $\mathsf{exp}[id_r] = \mathsf{true}$.

Relations between security notions. We reduce the SEC-security of SC with respect to $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$ (i) to the AUTH-security of SC with respect to $\mathsf{R}_{\mathsf{auth}}$ (for *any* $\mathsf{R}_{\mathsf{priv}}$), and (ii) to the PRIV-security of SC with respect to $\mathsf{R}_{\mathsf{priv}}$ (for *any* $\mathsf{R}_{\mathsf{auth}}$). The other direction is more complicated. For any relaxing relations $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$, we provide a reduction from the AUTH-security of SC with respect to $\mathsf{R}_{\mathsf{auth}} \cap \mathsf{R}_{\mathsf{priv}}$ jointly with the PRIV-security of SC with respect to $\mathsf{R}_{\mathsf{priv}}$, to the SEC-security of SC with respect to $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$.

So we can claim that SEC-security with respect to $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$ is equivalent to AUTH-security with respect to $\mathsf{R}_{\mathsf{auth}}$ and PRIV-security with respect to $\mathsf{R}_{\mathsf{priv}}$ whenever $\mathsf{R}_{\mathsf{auth}} \subseteq \mathsf{R}_{\mathsf{priv}}$ (which implies $\mathsf{R}_{\mathsf{auth}} \cap \mathsf{R}_{\mathsf{priv}} = \mathsf{R}_{\mathsf{auth}}$). Note that this condition holds for the relaxing relations with respect to which we proved the security of our iMessage-based signcryption scheme, meaning $\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*] \subseteq \mathsf{IMSG\text{-}PRIV\text{-}REL}$ for any $\mathsf{R}^* \in \{\mathsf{R}_{\mathsf{m}}, \mathsf{R}_{\mathsf{id}}\}$.

Lemma C.1 *Let* SC *be a signcryption scheme. Let* $\mathsf{R}_{\mathsf{auth}}$*,* $\mathsf{R}_{\mathsf{priv}}$ *be relaxing relations. Let* $\mathcal{F}$ *be an adversary against the* AUTH*-security of* SC *with respect to* $\mathsf{R}_{\mathsf{auth}}$*. Then we build an adversary* $\mathcal{D}_{\mathrm{SEC}}$ *against the* SEC*-security of* SC *with respect to* $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$ *such that*

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}}}(\mathcal{F}) \leq \mathsf{Adv}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}}}(\mathcal{D}_{\mathrm{SEC}}).$$

Proof of Lemma C.1: We build adversary $\mathcal{D}_{\mathrm{SEC}}$ against the SEC-security of SC with respect to $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$ as defined in Fig. 25. It simulates game $\mathrm{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}}}$ for adversary $\mathcal{F}$. Note that $\mathcal{F}$'s oracle calls to SigEnc are simulated by calling $\mathcal{D}_{\mathrm{SEC}}$'s oracle LR with $m_0 = m_1$ as input, so table ch and set Q_{priv} are always empty in game $\mathrm{G}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}},\mathcal{D}_{\mathrm{SEC}}}$. Meaning that $\mathcal{D}_{\mathrm{SEC}}$'s oracle VerDec never returns $(\bot, \text{“priv”})$. This allows $\mathcal{D}_{\mathrm{SEC}}$ to answer most of $\mathcal{F}$'s oracle calls by directly calling its own oracles. Only the simulation of $\mathcal{F}$'s calls to oracle VerDec requires some care.

Adversary $\mathcal{F}$ wins in game $\mathrm{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}}}$ if it manages to set win flag to true. Whenever this happens, adversary $\mathcal{D}_{\mathrm{SEC}}$ gets a return value (m, err) from its own oracle VerDec with either $\mathsf{err} = \text{“ok”}$ or $\mathsf{err} = \text{“chal”}$, thus being able to deduce that the challenge bit value in its own game is $b' = 1$ or $b' = 0$, respectively. In either case, adversary $\mathcal{D}_{\mathrm{SEC}}$ immediately calls $\mathbf{abort}(b')$ to halt with b' as its return value. ▌

Lemma C.2 *Let* SC *be a signcryption scheme. Let* $\mathsf{R}_{\mathsf{auth}}$*,* $\mathsf{R}_{\mathsf{priv}}$ *be relaxing relations. Let* $\mathcal{D}_{\mathrm{PRIV}}$ *be an adversary against the* PRIV*-security of* SC *with respect to* $\mathsf{R}_{\mathsf{priv}}$*. Then we build an adversary*

```
Adversary $\mathcal{D}_{\text{SEC}}^{\text{NewH},\text{NewC},\text{Exp},\text{LR},\text{VerDec}}(\pi)$
$\mathcal{F}^{\text{NewH},\text{NewC},\text{Exp},\text{SigEncSim},\text{VerDecSim}}(\pi)$
Return 0

SigEncSim$(id_s, \mathcal{I}, m, ad)$
Return LR$(id_s, \mathcal{I}, m, m, ad)$

VerDecSim$(id_s, id_r, c, ad)$
$(m, \mathsf{err}) \leftarrow$ VerDec$(id_s, id_r, c, ad)$
If $\mathsf{err} \notin \{$"ok", "chal"$\}$ then return $m$
If $\mathsf{err} =$ "ok" then $b' \leftarrow 1$ else $b' \leftarrow 0$
abort$(b')$
```

```
Adversary $\mathcal{D}_{\text{SEC}}^{\text{NewH},\text{NewC},\text{Exp},\text{LR},\text{VerDec}}(\pi)$
$b' \leftarrow_{\$} \mathcal{D}_{\text{PRIV}}^{\text{NewH},\text{NewC},\text{Exp},\text{LR},\text{VerDecSim}}(\pi)$
Return $b'$

VerDecSim$(id_s, id_r, c, ad)$
$(m, \mathsf{err}) \leftarrow$ VerDec$(id_s, id_r, c, ad)$
If $\mathsf{err} \in \{$"init", "dec", "priv"$\}$ then
    Return $(m, \mathsf{err})$
If $\mathsf{err} \in \{$"auth", "cheat"$\}$ then
    Return $(m,$ "ok"$)$
If $\mathsf{err} =$ "ok" then $b' \leftarrow 1$ else $b' \leftarrow 0$
abort$(b')$
```

Figure 25: **Left:** Adversary $\mathcal{D}_{\text{SEC}}$ for proof of Lemma C.1. $\mathcal{D}_{\text{SEC}}$ simulates game $\text{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}}}$ for adversary $\mathcal{F}$. **Right:** Adversary $\mathcal{D}_{\text{SEC}}$ for proof of Lemma C.2. $\mathcal{D}_{\text{SEC}}$ simulates game $\text{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}_{\mathsf{priv}}}$ for adversary $\mathcal{D}_{\text{PRIV}}$.

$\mathcal{D}_{\text{SEC}}$ *against the* SEC-*security of* SC *with respect to* $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$ *such that*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}_{\mathsf{priv}}}(\mathcal{D}_{\text{PRIV}}) \leq \mathsf{Adv}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}}}(\mathcal{D}_{\text{SEC}}).$$

Proof of Lemma C.2: We build adversary $\mathcal{D}_{\text{SEC}}$ against the SEC-security of SC with respect to $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$ as defined in Fig. 25. It simulates game $\text{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}_{\mathsf{priv}}}$ for adversary $\mathcal{D}_{\text{PRIV}}$. The simulation is perfect unless $\mathcal{D}_{\text{SEC}}$ gets a response with $\mathsf{err} \in \{$"ok", "chal"$\}$ from oracle VerDec, while attempting to simulate $\mathcal{D}_{\text{PRIV}}$'s call to its corresponding oracle. However, whenever this happens, $\mathcal{D}_{\text{SEC}}$ deduces the challenge bit b' in its own game without waiting for $\mathcal{D}_{\text{PRIV}}$'s guess, and calls **abort**(b') to immediately halt with b' as its output. ▌

Theorem C.3 *Let* SC *be a signcryption scheme. Let* $\mathsf{R}_{\mathsf{auth}}$, $\mathsf{R}_{\mathsf{priv}}$ *be relaxing relations. Let* $\mathcal{D}_{\text{SEC}}$ *be an adversary against the* SEC-*security of* SC *with respect to* $\mathsf{R}_{\mathsf{auth}}, \mathsf{R}_{\mathsf{priv}}$*. Then we build an adversary* $\mathcal{F}$ *against the* AUTH-*security of* SC *with respect to* $\mathsf{R}_{\mathsf{auth}} \cap \mathsf{R}_{\mathsf{priv}}$*, and an adversary* $\mathcal{D}_{\text{PRIV}}$ *against the* PRIV-*security of* SC *with respect to* $\mathsf{R}_{\mathsf{priv}}$ *such that*

$$\mathsf{Adv}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}}}(\mathcal{D}_{\text{SEC}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}} \cap \mathsf{R}_{\mathsf{priv}}}(\mathcal{F}) + \mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}_{\mathsf{priv}}}(\mathcal{D}_{\text{PRIV}}).$$

Proof of Theorem C.3: Consider games G_0–G_1 in Fig. 26. Lines not annotated with comments are common to both games. Game G_0 is equivalent to $\text{G}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}},\mathcal{D}_{\text{SEC}}}$, so

$$\mathsf{Adv}^{\mathsf{sec}}_{\mathsf{SC},\mathsf{R}_{\mathsf{auth}},\mathsf{R}_{\mathsf{priv}}}(\mathcal{D}_{\text{SEC}}) = 2 \cdot \Pr[\text{G}_0] - 1.$$

Games G_0 and G_1 are identical until bad_0. According to the *Fundamental Lemma of Game Playing* [20] we have

$$\Pr[\text{G}_0] - \Pr[\text{G}_1] \leq \Pr[\mathsf{bad}_0^{\text{G}_0}],$$

where $\Pr[\mathsf{bad}^{\mathsf{Q}}]$ denotes the probability of setting bad flag in game Q.

Adversary $\mathcal{D}_{\text{SEC}}$ setting flag bad_0 in oracle VerDec means that the sender id_s is not exposed, and the ciphertext c is not detected as trivial forgery according to $\mathsf{R}_{\mathsf{auth}}$. We build an adversary $\mathcal{F}$ against the AUTH-security of SC with respect to $\mathsf{R} = \mathsf{R}_{\mathsf{auth}} \cap \mathsf{R}_{\mathsf{priv}}$ as defined in Fig. 27, such that

$$\Pr[\mathsf{bad}_0^{\text{G}_0}] \leq \Pr[\text{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}}].$$

```
Games $G_0$–$G_1$
$b \leftarrow\!\$ \{0,1\}$ ; $\pi \leftarrow\!\$ \mathsf{SC.Setup}$ ; $b' \leftarrow\!\$ \mathcal{D}_{\mathrm{SEC}}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\mathrm{LR},\textsc{VerDec}}(\pi)$ ; Return $b' = b$

$\textsc{NewH}(id)$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(pk, sk) \leftarrow\!\$ \mathsf{SC.Kg}(\pi)$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $pk$

$\textsc{NewC}(id, pk, sk)$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $\mathsf{true}$

$\textsc{Exp}(id)$
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\bot$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$

$\mathrm{LR}(id_s, \mathcal{I}, m_0, m_1, ad)$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) or $|m_0| \neq |m_1|$ then return $\bot$
If $m_0 \neq m_1$ then
  If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\bot$
  For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{R} \leftarrow \emptyset$ ; For each $id \in \mathcal{I}$ do $\mathcal{R} \leftarrow \mathcal{R} \cup \{(id, \mathsf{pk}[id])\}$
$\mathcal{C} \leftarrow\!\$ \mathsf{SC.SigEnc}(\pi, id_s, \mathsf{pk}[id_s], \mathsf{sk}[id_s], \mathcal{R}, m_b, ad)$
For each $(id_r, c) \in \mathcal{C}$ do
  If $m_0 \neq m_1$ then
    $Q_{\mathsf{priv}} \leftarrow Q_{\mathsf{priv}} \cup \{((id_s, id_r, m_0, ad), c)\}$
    $Q_{\mathsf{priv}} \leftarrow Q_{\mathsf{priv}} \cup \{((id_s, id_r, m_1, ad), c)\}$
  Else $Q_{\mathsf{auth}} \leftarrow Q_{\mathsf{auth}} \cup \{((id_s, id_r, m_0, ad), c)\}$
Return $\mathcal{C}$

$\textsc{VerDec}(id_s, id_r, c, ad)$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return $(\bot, \text{"init"})$
$m \leftarrow \mathsf{SC.VerDec}(\pi, id_s, \mathsf{pk}[id_s], id_r, \mathsf{pk}[id_r], \mathsf{sk}[id_r], c, ad)$
If $m = \bot$ then return $(\bot, \text{"dec"})$
$z_0 \leftarrow ((id_s, id_r, m, ad), c)$
If $\exists z_1 \in Q_{\mathsf{priv}}$: $\mathsf{R_{priv}.Vf}(z_0, z_1)$ then return $(\bot, \text{"priv"})$
If $\exists z_1 \in Q_{\mathsf{auth}}$: $\mathsf{R_{auth}.Vf}(z_0, z_1)$ then return $(m, \text{"auth"})$
$\mathsf{cheated} \leftarrow \mathsf{exp}[id_s]$ ; If $\mathsf{cheated}$ then return $(m, \text{"cheat"})$
$\mathsf{bad}_0 \leftarrow \mathsf{true}$
Return $(\bot, \text{"bad}_0\text{"})$                                    // $G_1$
If $b = 1$ then return $(m, \text{"ok"})$ else return $(\bot, \text{"chal"})$
```

Figure 26: Games G_0–G_1 for proof of Theorem C.3. The code added for the transitions between games is highlighted in green.

Adversary $\mathcal{F}$ simulates game G_0 for adversary $\mathcal{D}_{\mathrm{SEC}}$. It maintains its own copies of $\mathsf{initialized}$, ch, exp, Q_{priv}, and Q_{auth}. Every time adversary $\mathcal{D}_{\mathrm{SEC}}$ makes a query to oracle LR, adversary $\mathcal{F}$ produces a response using its own oracle $\textsc{SigEnc}$. As a result of this, the set Q in game $G^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}}$ at any moment is a subset of $Q^* = Q_{\mathsf{priv}} \cup Q_{\mathsf{auth}}$ in game G_0. So setting bad_0 in G_0 means that $\nexists z_1 \in Q_{\mathsf{priv}}$: $\mathsf{R_{priv}.Vf}(z_0, z_1)$ and $\nexists z_1 \in Q_{\mathsf{auth}}$: $\mathsf{R_{auth}.Vf}(z_0, z_1)$, where z_0 is the tuple constructed during the corresponding call to $\mathcal{D}_{\mathrm{SEC}}$'s oracle $\textsc{VerDec}$. The two statements imply (but are not equivalent to) $\nexists z_1 \in Q^*$: $\mathsf{R.Vf}(z_0, z_1)$ for $\mathsf{R} = \mathsf{R_{auth}} \cap \mathsf{R_{priv}}$. This means that in game $G^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}}$ the statement $\nexists z_1 \in Q$: $\mathsf{R.Vf}(z_0, z_1)$ for $\mathsf{R} = \mathsf{R_{auth}} \cap \mathsf{R_{priv}}$ is also true, justifying the upper bound above.

We now build an adversary $\mathcal{D}_{\mathrm{PRIV}}$ against the PRIV-security of SC with respect to $\mathsf{R_{priv}}$ as defined

```
Adversary $\mathcal{F}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\textsc{SigEnc},\textsc{VerDec}}(\pi)$
$b \leftarrow_{\$} \{0,1\}$ ; $b' \leftarrow_{\$} \mathcal{D}_{\mathrm{SEC}}^{\textsc{NewHSim},\textsc{NewCSim},\textsc{ExpSim},\textsc{LRSim},\textsc{VerDecSim}}(\pi)$

$\textsc{NewHSim}(id)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; Return $\textsc{NewH}(id)$

$\textsc{NewCSim}(id, pk, sk)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\textsc{NewC}(id, pk, sk)$

$\textsc{ExpSim}(id)$
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\perp$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\textsc{Exp}(id)$

$\textsc{LRSim}(id_s, \mathcal{I}, m_0, m_1, ad)$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) or $|m_0| \neq |m_1|$ then return $\perp$
If $m_0 \neq m_1$ then
   If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\perp$
   For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{C} \leftarrow_{\$} \textsc{SigEnc}(id_s, \mathcal{I}, m_b, ad)$
For each $(id_r, c) \in \mathcal{C}$ do
   If $m_0 \neq m_1$ then
      $Q_{\mathsf{priv}} \leftarrow Q_{\mathsf{priv}} \cup \{((id_s, id_r, m_0, ad), c)\}$
      $Q_{\mathsf{priv}} \leftarrow Q_{\mathsf{priv}} \cup \{((id_s, id_r, m_1, ad), c)\}$
   Else $Q_{\mathsf{auth}} \leftarrow Q_{\mathsf{auth}} \cup \{((id_s, id_r, m_0, ad), c)\}$
Return $\mathcal{C}$

$\textsc{VerDecSim}(id_s, id_r, c, ad)$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return $(\perp, \text{“init”})$
$m \leftarrow \textsc{VerDec}(id_s, id_r, c, ad)$ ; If $m = \perp$ then return $(\perp, \text{“dec”})$
$z_0 \leftarrow ((id_s, id_r, m, ad), c)$
If $\exists z_1 \in Q_{\mathsf{priv}}$ : $\mathsf{R}_{\mathsf{priv}}.\mathsf{Vf}(z_0, z_1)$ then return $(\perp, \text{“priv”})$
If $\exists z_1 \in Q_{\mathsf{auth}}$ : $\mathsf{R}_{\mathsf{auth}}.\mathsf{Vf}(z_0, z_1)$ then return $(m, \text{“auth”})$
$\mathsf{cheated} \leftarrow \mathsf{exp}[id_s]$ ; If $\mathsf{cheated}$ then return $(m, \text{“cheat”})$
If $b = 1$ then return $(m, \text{“ok”})$ else return $(\perp, \text{“chal”})$
```

Figure 27: Adversary $\mathcal{F}$ for proof of Theorem C.3. $\mathcal{F}$ simulates game G_0 for adversary $\mathcal{D}_{\mathrm{SEC}}$. The highlighted lines mark the code of G_0's simulated oracles changed by $\mathcal{F}$.

in Fig. 28 such that

$$\Pr[\mathrm{G}_1] \leq \Pr[\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}_{\mathsf{priv}},\mathcal{D}_{\mathrm{PRIV}}}].$$

Adversary $\mathcal{D}_{\mathrm{PRIV}}$ simulates game G_1 for adversary $\mathcal{D}_{\mathrm{SEC}}$. It maintains its own copies of exp and Q_{auth}, and uses them to check the two conditions in G_1's simulated oracle $\textsc{VerDecSim}$ that are not checked during the corresponding call to oracle $\textsc{VerDec}$ in game $\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}_{\mathsf{priv}},\mathcal{D}_{\mathrm{PRIV}}}$.

Together, all of the above produces the claim in the theorem statement. ▮

```
Adversary $\mathcal{D}_{\text{PRIV}}^{\text{NewH},\text{NewC},\text{Exp},\text{LR},\text{VerDec}}(\pi)$
$b' \leftarrow\$ \mathcal{D}_{\text{SEC}}^{\text{NewH},\text{NewCSim},\text{ExpSim},\text{LRSim},\text{VerDecSim}}(\pi)$ ; Return $b'$

NewCSim($id, pk, sk$)
$d \leftarrow$ NewC($id, pk, sk$) ; If $d \neq \perp$ then $\mathsf{exp}[id] \leftarrow \mathsf{true}$
Return $d$

ExpSim($id$)
$sk \leftarrow$ Exp($id$) ; If $sk \neq \perp$ then $\mathsf{exp}[id] \leftarrow \mathsf{true}$
Return $sk$

LRSim($id_s, \mathcal{I}, m_0, m_1, ad$)
$\mathcal{C} \leftarrow\$$ LRSim($id_s, \mathcal{I}, m_0, m_1, ad$)
For each $(id_r, c) \in \mathcal{C}$ do
   If $m_0 = m_1$ then $Q_{\mathsf{auth}} \leftarrow Q_{\mathsf{auth}} \cup \{((id_s, id_r, m_0, ad), c)\}$
Return $\mathcal{C}$

VerDecSim($id_s, id_r, c, ad$)
$(m, \mathsf{err}) \leftarrow$ VerDec($id_s, id_r, c, ad$) ; If $m = \perp$ then return $(\perp, \mathsf{err})$
$z_0 \leftarrow ((id_s, id_r, m, ad), c)$
If $\exists z_1 \in Q_{\mathsf{auth}} : \mathsf{R}_{\mathsf{auth}}.\mathsf{Vf}(z_0, z_1)$ then return $(m, \text{“auth”})$
$\mathsf{cheated} \leftarrow \mathsf{exp}[id_s]$ ; If $\mathsf{cheated}$ then return $(m, \text{“cheat”})$
Return $(\perp, \text{“bad}_0\text{”})$
```

Figure 28: Adversary $\mathcal{D}_{\text{PRIV}}$ for proof of Theorem C.3. $\mathcal{D}_{\text{PRIV}}$ simulates game G_1 for adversary $\mathcal{D}_{\text{SEC}}$.

```
Game $\text{G}^{\mathsf{tcr}}_{\mathsf{F},\mathcal{H}}$
$(i, x_1) \leftarrow\$ \mathcal{H}^{\text{NewKey}}$
If $i \notin [n]$ then return $\mathsf{false}$
$y_1 \leftarrow \mathsf{F.Ev}(\mathsf{fk}[i], x_1)$
$\mathsf{win}_1 \leftarrow (x_1 \neq \mathsf{x}_0[i])$
$\mathsf{win}_2 \leftarrow (y_1 = \mathsf{y}_0[i])$
Return $\mathsf{win}_1$ and $\mathsf{win}_2$

NewKey($x_0$)
$n \leftarrow n + 1$ ; $\mathsf{fk}[n] \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$
$\mathsf{x}_0[n] \leftarrow x_0$
$\mathsf{y}_0[n] \leftarrow \mathsf{F.Ev}(\mathsf{fk}[n], x_0)$
Return $\mathsf{fk}[n]$
```

```
Game $\text{G}^{\mathsf{unique}}_{\mathsf{SE},\mathcal{U}}$
$(i, c) \leftarrow\$ \mathcal{U}^{\text{Enc}}$
If $i \notin [n]$ then return $\mathsf{false}$
$m \leftarrow \mathsf{SE.Dec}(\mathsf{k}[i], c)$
$\mathsf{win}_1 \leftarrow (c \neq \mathsf{c}[i])$
$\mathsf{win}_2 \leftarrow (m = \mathsf{m}[i])$
Return $\mathsf{win}_1$ and $\mathsf{win}_2$

Enc($k, m$)
$c \leftarrow\$ \mathsf{SE.Enc}(k, m)$
$n \leftarrow n + 1$ ; $\mathsf{k}[n] \leftarrow k$
$\mathsf{m}[n] \leftarrow m$ ; $\mathsf{c}[n] \leftarrow c$
Return $c$
```

```
Game $\text{G}^{\mathsf{otind}}_{\mathsf{SE},\mathcal{D}}$
$b \leftarrow\$ \{0,1\}$ ; $b' \leftarrow\$ \mathcal{D}^{\text{LR}}$
Return $b' = b$

LR($m_0, m_1$)
If $|m_0| \neq |m_1|$ then
   Return $\perp$
$k \leftarrow\$ \{0,1\}^{\mathsf{SE.kl}}$
$c \leftarrow\$ \mathsf{SE.Enc}(k, m_b)$
Return $c$
```

Figure 29: Games defining target collision resistance of function family F, ciphertext uniqueness of symmetric encryption scheme SE, and one-time indistinguishability of symmetric encryption scheme SE.

D Standard definitions

Function families. A family of functions F specifies a deterministic algorithm $\mathsf{F.Ev}$. Associated to F is a key length $\mathsf{F.kl} \in \mathbb{N}$, an input set $\mathsf{F.In}$, and an output length $\mathsf{F.ol} \in \mathbb{N}$. The evaluation algorithm $\mathsf{F.Ev}$ takes a function key $fk \in \{0,1\}^{\mathsf{F.kl}}$ and an input $x \in \mathsf{F.In}$ to return an output $y \in \{0,1\}^{\mathsf{F.ol}}$.

TARGET COLLISION RESISTANCE OF FUNCTION FAMILY. Consider game $\mathrm{G}^{\mathsf{tcr}}$ of Fig. 29, associated to a function family F and an adversary $\mathcal{H}$. The advantage of $\mathcal{H}$ in breaking the TCR-security of F is defined as $\mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{F}}(\mathcal{H}) = \Pr[\mathrm{G}^{\mathsf{tcr}}_{\mathsf{F},\mathcal{H}}]$. To win the game, adversary $\mathcal{H}$ must find inputs x_0, x_1 such that $\mathsf{F.Ev}(fk, x_0) = \mathsf{F.Ev}(fk, x_1)$ but $x_0 \neq x_1$. The adversary has to choose x_0 prior to receiving the function key fk. It can choose multiple values of x_0, but a new uniformly random key fk is independently sampled for every choice of x_0. Target collision resistant hash functions were introduced by Naor and Yung [69] under the name of Universal One-Way Hash Functions (UOWHF). Bellare and Rogaway [19] redefined the corresponding security notion under the name of target collision resistance.

SYMMETRIC ENCRYPTION SCHEMES. A symmetric encryption scheme SE specifies algorithms $\mathsf{SE.Enc}$ and $\mathsf{SE.Dec}$, where $\mathsf{SE.Dec}$ is deterministic. Associated to SE is a key length $\mathsf{SE.kl} \in \mathbb{N}$ and a ciphertext length function $\mathsf{SE.cl}\colon \mathbb{N} \to \mathbb{N}$. The encryption algorithm $\mathsf{SE.Enc}$ takes a key $k \in \{0,1\}^{\mathsf{SE.kl}}$ and a message $m \in \{0,1\}^*$ to return a ciphertext $c \in \{0,1\}^{\mathsf{SE.cl}(|m|)}$. The decryption algorithm $\mathsf{SE.Dec}$ takes k, c to return message $m \in \{0,1\}^* \cup \{\bot\}$, where $\bot$ denotes incorrect decryption. Decryption correctness requires that $\mathsf{SE.Dec}(k, \mathsf{SE.Enc}(k, m)) = m$ for all $k \in \{0,1\}^{\mathsf{SE.kl}}$ and all $m \in \{0,1\}^*$.

CIPHERTEXT UNIQUENESS OF SE. Consider game $\mathrm{G}^{\mathsf{unique}}$ of Fig. 29, associated to a symmetric encryption scheme SE and an adversary $\mathcal{U}$. The advantage of $\mathcal{U}$ in breaking the UNIQUE-security of SE is defined as $\mathsf{Adv}^{\mathsf{unique}}_{\mathsf{SE}}(\mathcal{U}) = \Pr[\mathrm{G}^{\mathsf{unique}}_{\mathsf{SE},\mathcal{U}}]$. The game requires adversary $\mathcal{H}$ to (roughly) find distinct ciphertexts c_0, c_1 and some key k such that both ciphertexts decrypt to the same message under key k. However, the adversary does not have a full control over the choice of c_0. Instead, it has to query its ENC oracle on inputs k, m to get a ciphertext that is computed by runnning $\mathsf{SE.Enc}(k, m)$ with honestly sampled random coins. The adversary can call this oracle many times.

ONE-TIME INDISTINGUISHABILITY OF SE. Consider game $\mathrm{G}^{\mathsf{otind}}$ of Fig. 29, associated to a symmetric encryption scheme SE and an adversary $\mathcal{D}$. The advantage of $\mathcal{D}$ in breaking the OTIND-security of SE is defined as $\mathsf{Adv}^{\mathsf{otind}}_{\mathsf{SE}}(\mathcal{D}) = 2 \cdot \Pr[\mathrm{G}^{\mathsf{otind}}_{\mathsf{SE},\mathcal{D}}] - 1$. The adversary has access to oracle LR and can query it on messages m_0, m_1 to get back $\mathsf{SE.Enc}(k, m_b)$ for the challenge bit b. It can query the oracle many times, but for each of them the key k is sampled uniformly at random, independently of other LR calls. The adversary wins the game by guessing the challenge bit b.

DIGITAL SIGNATURE SCHEMES. A digital signature scheme DS specifies algorithms $\mathsf{DS.Kg}$, $\mathsf{DS.Sig}$, $\mathsf{DS.Ver}$, where $\mathsf{DS.Ver}$ is deterministic. The key generation algorithm $\mathsf{DS.Kg}$ returns a verification key vk and a signing key tk. The signing algorithm $\mathsf{DS.Sig}$ takes tk and a message $m \in \{0,1\}^*$ to return a signature σ. The verification algorithm $\mathsf{DS.Ver}$ takes vk, m, σ to return a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ regarding whether σ is a valid signature of m under vk. Correctness condition requires that $\mathsf{DS.Ver}(vk, m, \sigma) = \mathsf{true}$ for all $(tk, vk) \in [\mathsf{DS.Kg}]$, all $m \in \{0,1\}^*$, and all $\sigma \in [\mathsf{DS.Sig}(tk, m)]$.

PUBLIC-KEY ENCRYPTION SCHEMES. A public-key encryption scheme PKE specifies algorithms $\mathsf{PKE.Kg}$, $\mathsf{PKE.Enc}$ and $\mathsf{PKE.Dec}$, where $\mathsf{PKE.Dec}$ is deterministic. Associated to PKE is an input set $\mathsf{PKE.In}$. The key generation algorithm $\mathsf{PKE.Kg}$ returns a key pair (ek, dk), where ek is an encryption key and dk is a decryption key. The encryption algorithm $\mathsf{PKE.Enc}$ takes ek and a message $m \in \mathsf{PKE.In}$ to return a ciphertext c. The decryption algorithm $\mathsf{PKE.Dec}$ takes ek, dk, c to return $m \in \mathsf{PKE.In} \cup \{\bot\}$, where $\bot$ denotes incorrect decryption. Correctness condition requires that $\mathsf{PKE.Dec}(ek, dk, c) = m$ for all $(ek, dk) \in [\mathsf{PKE.Kg}]$, all $m \in \mathsf{PKE.In}$, and all $c \in [\mathsf{PKE.Enc}(ek, m)]$.

INDCCA SECURITY OF PKE. Consider game $\mathrm{G}^{\mathsf{indcca}}$ of Fig. 31, associated to a public-key encryption scheme PKE and an adversary $\mathcal{D}$. The advantage of $\mathcal{D}$ in breaking the INDCCA-security of PKE is defined as $\mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{PKE}}(\mathcal{D}) = 2 \cdot \Pr[\mathrm{G}^{\mathsf{indcca}}_{\mathsf{PKE},\mathcal{D}}] - 1$. The adversary can create an arbitrary number of user identities (key pairs) by calling oracle NEWUSER. For each user, the adversary can call

```
$\pi \leftarrow\!\!\$\ \mathsf{MRPKE.Setup}$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{MRPKE.Kg}(\pi)$
$\mathcal{C} \leftarrow\!\!\$\ \mathsf{MRPKE.Enc}(\pi, \mathcal{R}, m)$
$m \leftarrow \mathsf{MRPKE.Dec}(\pi, ek, dk, c)$
```

Figure 30: Syntax of the constituent algorithms of multi-recipient public-key encryption scheme MRPKE.

LR to get a challenge ciphertext, call DEC to decrypt a ciphertext, and call EXP to get this user's decryption key. The adversary wins if it can guess the challenge bit b. To avoid trivial attacks, the game does not allow adversary to use DEC for decrypting challenge ciphertexts produced by LR, and it also enforces that oracles EXP and LR can be called for the same user simultaneously.

MULTI-RECIPIENT PUBLIC-KEY ENCRYPTION SCHEMES. A multi-recipient public-key encryption scheme MRPKE specifies algorithms MRPKE.Setup, MRPKE.Kg, MRPKE.Enc and MRPKE.Dec, where MRPKE.Dec is deterministic. The setup algorithm MRPKE.Setup returns public parameters π. The key generation algorithm MRPKE.Kg takes π to return a key pair (ek, dk), where ek is an encryption key and dk is a decryption key. The encryption algorithm MRPKE.Enc takes π, a set $\mathcal{R}$ of pairs (id, ek) each containing recipient identity $id \in \{0,1\}^*$ and encryption key ek, and a message $m \in \{0,1\}^*$ to return a set $\mathcal{C}$ of pairs (id, c), each denoting that ciphertext c should be sent to recipient with identity id. The decryption algorithm MRPKE.Dec takes π, ek_r, dk_r, c to return $m \in \{0,1\}^* \cup \{\bot\}$ where $\bot$ denotes incorrect decryption. The syntax used for the constituent algorithms of MRPKE is summarized in Fig. 30. Multi-recipient public-key encryption was defined by Bellare et al. [12] (see also [56, 14]).

Decryption correctness of MRPKE requires that for all $\pi \in [\mathsf{MRPKE.Setup}]$, all $n \in \mathbb{N}$, all $(ek_1, dk_1), \ldots, (ek_n, dk_n) \in [\mathsf{MRPKE.Kg}(\pi)]$, all *distinct* $id_1, \ldots, id_n \in \{0,1\}^*$, and all $m \in \{0,1\}^*$ the following conditions hold. Let $\mathcal{R} = \{(id_i, ek_i)\}_{1 \le i \le n}$. We require that for all $\mathcal{C} \in [\mathsf{MRPKE.Enc}(\pi, \mathcal{R}, m)]$: (i) $|\mathcal{C}| = |\mathcal{R}|$; (ii) for each $i \in \{1, \ldots, n\}$ there exists a unique $c \in \{0,1\}^*$ such that $(id_i, c) \in \mathcal{C}$; (iii) for each $i \in \{1, \ldots, n\}$ and each c such that $(id_i, c) \in \mathcal{C}$ we have $m = \mathsf{MRPKE.Dec}(\pi, ek_i, dk_i, c)$.

INDCCA SECURITY OF MRPKE. Consider game $\mathrm{G}^{\mathsf{indcca}}$ in the right panel of Fig. 31, associated to a multi-recipient public-key encryption scheme MRPKE and an adversary $\mathcal{D}$. The advantage of $\mathcal{D}$ in breaking the INDCCA-security of MRPKE is defined as $\mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{MRPKE}}(\mathcal{D}) = 2 \cdot \Pr[\mathrm{G}^{\mathsf{indcca}}_{\mathsf{MRPKE},\mathcal{D}}] - 1$. The game is defined in the same way as the INDCCA game for PKE, except now the LR oracle takes a *set* of recipients, and returns a set of pairs each containing a recipient identity and the corresponding ciphertext.

RANDOM ORACLE MODEL. In the random oracle model (ROM) [18], the random oracle RO models a truly random function $f\colon \mathcal{D} \to \mathcal{R}$ with some domain $\mathcal{D}$ and range $\mathcal{R}$. In this paper we use RO to model functions with $\mathcal{D} = \{0,1\}^*$ and $\mathcal{R} = \{0,1\}^\ell$ for some $\ell \in \mathbb{N}$. So we let $f(z) = \mathrm{RO}(z, \ell)$ for the random oracle RO defined as follows:

```
$\underline{\mathrm{RO}(z, \ell)}$
If $T[z, \ell] = \bot$ then $T[z, \ell] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $T[z, \ell]$
```

It takes a string $z \in \{0,1\}^*$ and an output length $\ell \in \mathbb{N}$ as input, to return an element from $\{0,1\}^\ell$.

IDEAL CIPHER MODEL. In the ideal cipher model [80], a block cipher is modeled by a random permutation for every key in its key space. Formally, a block cipher $E\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$

```
Game $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{PKE},\mathcal{D}}$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{D}^{\textsc{NewUser},\textsc{Exp},\textsc{LR},\textsc{Dec}}$
Return $b' = b$

$\textsc{NewUser}(id)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$
$(ek, dk) \leftarrow\$ \mathsf{PKE.Kg}$ ; $\mathsf{ek}[id] \leftarrow ek$
$\mathsf{dk}[id] \leftarrow dk$ ; Return $ek$

$\textsc{Exp}(id)$
If not $\mathsf{initialized}[id]$ then return $\perp$
If $\mathsf{ch}[id]$ then return $\perp$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{dk}[id]$

$\mathrm{LR}(id, m_0, m_1)$
If not $\mathsf{initialized}[id]$ then return $\perp$
If $|m_0| \neq |m_1|$ then return $\perp$
If $m_0 \neq m_1$ then
  If $\mathsf{exp}[id]$ then return $\perp$
  $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$c \leftarrow\$ \mathsf{PKE.Enc}(\mathsf{ek}[id], m_b)$
$Q \leftarrow Q \cup \{(id, c)\}$
Return $c$

$\textsc{Dec}(id, c)$
If not $\mathsf{initialized}[id]$ then return $\perp$
If $(id, c) \in Q$ then return $\perp$
$m \leftarrow \mathsf{PKE.Dec}(\mathsf{ek}[id], \mathsf{dk}[id], c)$
Return $m$
```

```
Game $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{MRPKE},\mathcal{D}}$
$b \leftarrow\$ \{0,1\}$ ; $\pi \leftarrow\$ \mathsf{MRPKE.Setup}$
$b' \leftarrow\$ \mathcal{D}^{\textsc{NewUser},\textsc{Exp},\textsc{LR},\textsc{Dec}}(\pi)$
Return $b' = b$

$\textsc{NewUser}(id)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$
$(ek, dk) \leftarrow\$ \mathsf{MRPKE.Kg}(\pi)$ ; $\mathsf{ek}[id] \leftarrow ek$
$\mathsf{dk}[id] \leftarrow dk$ ; Return $ek$

$\textsc{Exp}(id)$
If not $\mathsf{initialized}[id]$ then return $\perp$
If $\mathsf{ch}[id]$ then return $\perp$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{dk}[id]$

$\mathrm{LR}(\mathcal{I}, m_0, m_1)$
If $\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$ then return $\perp$
$\mathcal{R} \leftarrow \emptyset$ ; If $|m_0| \neq |m_1|$ then return $\perp$
If $m_0 \neq m_1$ then
  If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\perp$
  For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
For each $id \in \mathcal{I}$ do $\mathcal{R} \leftarrow \mathcal{R} \cup \{(id, \mathsf{ek}[id])\}$
$\mathcal{C} \leftarrow\$ \mathsf{MRPKE.Enc}(\pi, \mathcal{R}, m_b)$
For each $(id, c) \in \mathcal{C}$ do $Q \leftarrow Q \cup \{(id, c)\}$
Return $\mathcal{C}$

$\textsc{Dec}(id, c)$
If not $\mathsf{initialized}[id]$ then return $\perp$
If $(id, c) \in Q$ then return $\perp$
$m \leftarrow \mathsf{MRPKE.Dec}(\pi, \mathsf{ek}[id], \mathsf{dk}[id], c)$
Return $m$
```

Figure 31: Games defining indistinguishability of public-key encryption scheme PKE under chosen ciphertext attack, and indistinguishability of multi-recipient public-key encryption scheme MRPKE under chosen ciphertext attack.

is modeled as $E(k,x) = \mathrm{EO}(k,x)$ and its inverse is modeled as $E^{-1}(k,y) = \mathrm{EO}^{-1}(k,y)$ for all $k \in \{0,1\}^\kappa$ and all $x, y \in \{0,1\}^n$, where oracles EO and EO^{-1} are defined as follows:

```
$\mathrm{EO}(k, x)$
$\ell \leftarrow |x|$
If $T[k, \ell, x] = \perp$ then
  $y \leftarrow\$ \{0,1\}^\ell \setminus R[k, \ell]$
  $D[k, \ell] \leftarrow D[k, \ell] \cup \{x\}$
  $R[k, \ell] \leftarrow R[k, \ell] \cup \{y\}$
  $T[k, \ell, x] \leftarrow y$ ; $T^{-1}[k, \ell, y] \leftarrow x$
Return $T[k, \ell, x]$
```

```
$\mathrm{EO}^{-1}(k, y)$
$\ell \leftarrow |y|$
If $T^{-1}[k, \ell, y] = \perp$ then
  $x \leftarrow\$ \{0,1\}^\ell \setminus D[k, \ell]$
  $D[k, \ell] \leftarrow D[k, \ell] \cup \{x\}$
  $R[k, \ell] \leftarrow R[k, \ell] \cup \{y\}$
  $T^{-1}[k, \ell, y] \leftarrow x$ ; $T[k, \ell, x] \leftarrow y$
Return $T[k, \ell, y]$
```

Let ℓ be the input/output length of the block cipher. Then the oracles maintain (shared) sets $D[k,\ell]$ and $R[k,\ell]$ with unused elements in the domain and range of the random permutation for key k, respectively. Furthermore, table entries $T[k,\ell,\cdot]$ and $T^{-1}[k,\ell,\cdot]$ are used to save the mapping between the input and output elements of the random permutation for key k.

```
Games G_0–G_2
$b \leftarrow_\$ \{0,1\}$ ; $b' \leftarrow_\$ \mathcal{D}_{\mathrm{AE}}^{\mathrm{LR,DEC}}$ ; Return $b = b'$

LR($m_0, m_1$)
If $|m_0| \neq |m_1|$ then return $\bot$
$n \leftarrow n+1$ ; $r_0 \leftarrow_\$ \{0,1\}^{\mathsf{F.kl}}$ ; $r_1 \leftarrow \mathsf{F.Ev}(r_0, m_b)$ ; $\mathsf{k}[n] \leftarrow r_0 \| r_1$
$\mathsf{c}[n] \leftarrow_\$ \mathsf{SE.Enc}(\mathsf{k}[n], m_b)$ ; $\mathsf{m}[n] \leftarrow m_b$ ; Return $(n, \mathsf{c}[n])$

DEC($i, c$)
If $i \notin [n]$ or $\mathsf{c}[i] = c$ then return $\bot$
$m \leftarrow \mathsf{SE.Dec}(\mathsf{k}[i], c)$ ; If $m = \bot$ then return $\bot$
If $m = \mathsf{m}[i]$ then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$
    Return $\bot$                                              // G_1, G_2
$s \leftarrow \mathsf{k}[i]$ ; $r_0 \leftarrow s[1 \ldots \mathsf{F.kl}]$ ; $r_1 \leftarrow s[\mathsf{F.kl}+1 \ldots \mathsf{SE.kl}]$
If $r_1 \neq \mathsf{F.Ev}(r_0, m)$ then return $\bot$
$\mathsf{bad}_1 \leftarrow \mathsf{true}$
Return $\bot$                                                  //      G_2
If $b = 1$ then return $m$ else return $\bot$
```

Figure 32: Games G_0–G_2 for proof of Theorem 4.1. The code added by expanding the algorithms of EMDK in game $G^{\mathsf{ae}}_{\mathsf{EMDK},\mathcal{D}_{\mathrm{AE}}}$ is highlighted in gray. The code added for the transitions between games is highlighted in green.

BIRTHDAY PROBLEM. Let $q, N \in \mathbb{N}$. Consider an experiment that samples q values $x_1, \ldots, x_q$ from a set S of size $|S| = N$. The values are sampled uniformly at random and independent of each other. Let

$$C(N,q) = \Pr[x_1, \ldots, x_q \text{ are } \textbf{not} \text{ all distinct}].$$

Then

$$0.3 \cdot \frac{q \cdot (q-1)}{N} \leq C(N,q) \leq 0.5 \cdot \frac{q \cdot (q-1)}{N},$$

where the lower bound holds for $1 \leq q \leq \sqrt{2N}$.

E Proof of Theorem 4.1

Consider games G_0–G_2 in Fig. 32. Lines not annotated with comments are common to all games. Game G_0 is equivalent to $G^{\mathsf{ae}}_{\mathsf{EMDK},\mathcal{D}_{\mathrm{AE}}}$, so

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{EMDK}}(\mathcal{D}_{\mathrm{AE}}) = 2 \cdot \Pr[G_0] - 1.$$

Games G_0 and G_1 are identical until bad_0; games G_1 and G_2 are identical until bad_1. According to the Fundamental Lemma of Game Playing [20] we have

$$\Pr[G_0] - \Pr[G_1] \leq \Pr[\mathsf{bad}_0^{G_0}] \text{ and } \Pr[G_1] - \Pr[G_2] \leq \Pr[\mathsf{bad}_1^{G_1}],$$

where $\Pr[\mathsf{bad}^{\mathsf{Q}}]$ denotes the probability of setting bad flag in game Q. Setting flag bad_0 in game G_0 means that $\mathsf{SE.Dec}(\mathsf{k}[i], \mathsf{c}[i]) = \mathsf{SE.Dec}(\mathsf{k}[i], c)$ for some $c \neq \mathsf{c}[i]$. We use this to build an adversary $\mathcal{U}$ against the UNIQUE-security of SE, simulating game G_0 for $\mathcal{D}_{\mathrm{AE}}$ as defined in Fig. 33, such that

$$\Pr[\mathsf{bad}_0^{G_0}] \leq G^{\mathsf{unique}}_{\mathsf{SE},\mathcal{U}}.$$

Setting flag bad_1 in game G_1 means that $\mathsf{F.Ev}(r_0, \mathsf{m}[i]) = \mathsf{F.Ev}(r_0, m)$ for some $m \neq \mathsf{m}[i]$, such that

```
Adversary $\mathcal{U}^{\text{Enc}}$
$b \leftarrow\!\!\$\ \{0,1\}$ ; $b' \leftarrow\!\!\$\ \mathcal{D}_{\text{AE}}^{\text{LRSim,DecSim}}$ ; Return $\mathsf{out}$
LRSim$(m_0, m_1)$
If $|m_0| \neq |m_1|$ then return $\perp$
$n \leftarrow n+1$ ; $r_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$ ; $r_1 \leftarrow \mathsf{F.Ev}(r_0, m_b)$ ; $\mathsf{k}[n] \leftarrow r_0 \,\|\, r_1$
$\mathsf{c}[n] \leftarrow\!\!\$\ \text{Enc}(\mathsf{k}[n], m_b)$ ; $\mathsf{m}[n] \leftarrow m_b$ ; Return $(n, \mathsf{c}[n])$
DecSim$(i, c)$
If $i \notin [n]$ or $\mathsf{c}[i] = c$ then return $\perp$
$m \leftarrow \mathsf{SE.Dec}(\mathsf{k}[i], c)$ ; If $m = \perp$ then return $\perp$
If $m = \mathsf{m}[i]$ then $\mathsf{out} \leftarrow (i, c)$
$s \leftarrow \mathsf{k}[i]$ ; $r_0 \leftarrow s[1 \ldots \mathsf{F.kl}]$ ; $r_1 \leftarrow s[\mathsf{F.kl}+1 \ldots \mathsf{SE.kl}]$
If $r_1 \neq \mathsf{F.Ev}(r_0, m)$ then return $\perp$
If $b = 1$ then return $m$ else return $\perp$
```

Figure 33: Adversary $\mathcal{U}$ for proof of Theorem 4.1. $\mathcal{U}$ simulates game G_0 for adversary $\mathcal{D}_{\text{AE}}$. The highlighted lines mark the code of G_0's simulated oracles changed by $\mathcal{U}$.

```
Adversary $\mathcal{H}^{\text{NewKey}}$
$b \leftarrow\!\!\$\ \{0,1\}$ ; $b' \leftarrow\!\!\$\ \mathcal{D}_{\text{AE}}^{\text{LRSim,DecSim}}$ ; Return $\mathsf{out}$
LRSim$(m_0, m_1)$
If $|m_0| \neq |m_1|$ then return $\perp$
$n \leftarrow n+1$ ; $r_0 \leftarrow\!\!\$\ \text{NewKey}(m_b)$ ; $r_1 \leftarrow \mathsf{F.Ev}(r_0, m_b)$ ; $\mathsf{k}[n] \leftarrow r_0 \,\|\, r_1$
$\mathsf{c}[n] \leftarrow\!\!\$\ \mathsf{SE.Enc}(\mathsf{k}[n], m_b)$ ; $\mathsf{m}[n] \leftarrow m_b$ ; Return $(n, \mathsf{c}[n])$
DecSim$(i, c)$
If $i \notin [n]$ or $\mathsf{c}[i] = c$ then return $\perp$
$m \leftarrow \mathsf{SE.Dec}(\mathsf{k}[i], c)$ ; If $m = \perp$ then return $\perp$
If $m = \mathsf{m}[i]$ then return $\perp$
$s \leftarrow \mathsf{k}[i]$ ; $r_0 \leftarrow s[1 \ldots \mathsf{F.kl}]$ ; $r_1 \leftarrow s[\mathsf{F.kl}+1 \ldots \mathsf{SE.kl}]$
If $r_1 \neq \mathsf{F.Ev}(r_0, m)$ then return $\perp$
$\mathsf{out} \leftarrow (i, m)$
If $b = 1$ then return $m$ else return $\perp$
```

Figure 34: Adversary $\mathcal{H}$ for proof of Theorem 4.1. $\mathcal{H}$ simulates game G_1 for adversary $\mathcal{D}_{\text{AE}}$. The highlighted lines mark the code of G_1's simulated oracles changed by $\mathcal{H}$.

$r_0 \in \{0,1\}^{\mathsf{F.kl}}$ was sampled uniformly at random and $\mathsf{m}[i]$ was chosen independently of it. We use this to build an adversary $\mathcal{H}$ against the TCR-security of F, simulating game G_1 for $\mathcal{D}_{\text{AE}}$ as defined in Fig. 34, such that

$$\Pr[\mathsf{bad}_1^{G_0}] \leq \mathrm{G}^{\mathsf{tcr}}_{\mathsf{F},\mathcal{H}}.$$

In game G_2 the oracle Dec always returns $\perp$. We use this to build an adversary $\mathcal{D}_{\text{IND}}$ against the IND-security of EMDK, simulating game G_2 for $\mathcal{D}_{\text{AE}}$ as defined in Fig. 35, such that

$$\Pr[G_2] = \mathrm{G}^{\mathsf{ind}}_{\mathsf{EMDK},\mathcal{D}_{\text{IND}}}.$$

Together, all of the above produce the claim in the theorem statement.

```
$\underline{\text{Adversary } \mathcal{D}_{\text{IND}}^{\text{LR}}}$
$b' \leftarrow\$ \mathcal{D}_{\text{AE}}^{\text{LRSim,DecSim}}$ ; Return $b'$
$\underline{\text{LRSim}(m_0, m_1)}$
If $|m_0| \neq |m_1|$ then return $\bot$
$n \leftarrow n + 1$ ; $c \leftarrow\$ \text{LR}(m_0, m_1)$ ; Return $(n, c)$
$\underline{\text{DecSim}(i, c)}$
Return $\bot$
```

Figure 35: Adversary $\mathcal{D}_{\text{IND}}$ for proof of Theorem 4.1. $\mathcal{D}_{\text{IND}}$ simulates game G_2 for adversary $\mathcal{D}_{\text{AE}}$. The highlighted lines mark the code of G_2's simulated oracles changed by $\mathcal{D}_{\text{IND}}$.

```
$\underline{\text{Games G}_0\text{–G}_3}$
$b \leftarrow\$ \{0,1\}$ ; $b' \leftarrow\$ \mathcal{D}_{\mathsf{EMDK}}^{\text{LR,RO}}$ ; Return $b = b'$
$\underline{\text{LR}(m_0, m_1)}$
If $|m_0| \neq |m_1|$ then return $\bot$
$r_0 \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$
$r_1 \leftarrow\$ \text{RO}(\langle r_0, m_b\rangle, \mathsf{F.ol})$                 // G_0
$r_1 \leftarrow\$ \{0,1\}^{\mathsf{F.ol}}$                                //     G_1, G_2, G_3
If $T[\langle r_0, m_b\rangle, \mathsf{F.ol}] \neq \bot$ then
    $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    $r_1 \leftarrow T[\langle r_0, m_b\rangle, \mathsf{F.ol}]$                   //     G_1
$T[\langle r_0, m_b\rangle, \mathsf{F.ol}] \leftarrow r_1$                       //     G_1, G_2, G_3
$k \leftarrow r_0 \| r_1$ ; $c \leftarrow\$ \mathsf{SE.Enc}(k, m_b)$ ; Return $c$
$\underline{\text{RO}(z, \ell)}$
If $\mathsf{local}[z,\ell]$ then return $T[z,\ell]$
$\mathsf{local}[z,\ell] \leftarrow \mathsf{true}$ ; $h \leftarrow\$ \{0,1\}^{\ell}$
If $T[z,\ell] \neq \bot$ then
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$
    $h \leftarrow T[z,\ell]$ ; $\mathsf{local}[z,\ell] \leftarrow \mathsf{false}$          // G_0, G_1, G_2
$T[z,\ell] \leftarrow h$ ; Return $T[z,\ell]$
```

Figure 36: Games G_0–G_3 for proof of Theorem 4.2. The code added by expanding the algorithms of EMDK in game $\text{G}_{\mathsf{EMDK},\mathcal{D}}^{\mathsf{ind}}$ is highlighted in gray. The code added for the transitions between games is highlighted in green.

F Proof of Theorem 4.2

Consider games G_0–G_3 in Fig. 36. Lines not annotated with comments are common to all games. Game G_0 is equivalent to $\text{G}_{\mathsf{EMDK},\mathcal{D}_{\mathsf{EMDK}}}^{\mathsf{ind}}$ when F is modeled as the random oracle, so

$$\mathsf{Adv}_{\mathsf{EMDK}}^{\mathsf{ind}}(\mathcal{D}_{\mathsf{EMDK}}) = 2 \cdot \Pr[\text{G}_0] - 1.$$

Game G_1 expands the code of the RO call inside oracle LR, so games G_0 and G_1 are equivalent, and

$$\Pr[\text{G}_0] = \Pr[\text{G}_1].$$

```
Adversary $\mathcal{P}^{\text{Enc},\text{GuessKey}}$
$b \leftarrow_\$ \{0,1\}$ ; $b' \leftarrow_\$ \mathcal{D}_{\mathsf{EMDK}}^{\text{LRSim},\text{ROSim}}$
LRSim$(m_0, m_1)$
If $|m_0| \neq |m_1|$ then return $\perp$
$c \leftarrow_\$ \text{Enc}(m_b)$ ; Return $c$
ROSim$(z, \ell)$
If $\mathsf{local}[z,\ell]$ then return $T[z,\ell]$
$\mathsf{local}[z,\ell] \leftarrow \mathsf{true}$ ; $h \leftarrow_\$ \{0,1\}^\ell$ ; If $|z| \geq \mathsf{F.kl}$ then GuessKey$(z[1\ldots\mathsf{F.kl}])$
$T[z,\ell] \leftarrow h$ ; Return $T[z,\ell]$
```

Figure 37: Adversary $\mathcal{P}$ for proof of Theorem 4.2. $\mathcal{P}$ simulates game G_3 for adversary $\mathcal{D}_{\mathsf{EMDK}}$. The highlighted lines mark the code of G_3's simulated oracles changed by $\mathcal{P}$.

Games G_1 and G_2 are identical until bad_1; games G_2 and G_3 are identical until bad_2. According to the Fundamental Lemma of Game Playing [20] we have

$$\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_2] \leq \Pr[\mathsf{bad}_1^{\mathrm{G}_1}] \text{ and } \Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_3] \leq \Pr[\mathsf{bad}_2^{\mathrm{G}_3}],$$

where $\Pr[\mathsf{bad}^{\mathsf{Q}}]$ denotes the probability of setting bad flag in game Q.

The probability of setting flag bad_1 in game G_1 can be upper bounded by assuming that (in the worst case) adversary will make q_{RO} queries to oracle RO *prior* to its first query to oracle LR. Specifically, consider an adversary that queries its oracles only on inputs that require the game to access table T at indices $z = \langle r_0, \mathsf{m}\rangle$ and $\ell = \mathsf{F.ol}$ for some fixed message m. All values of r_0 used for calls to oracle RO should be chosen to be distinct, whereas oracle LR chooses r_0 uniformly at random and there is a chance that it matches one of the values that was used before. Then the probability of the condition $T[\langle r_0, m_b\rangle, \mathsf{F.ol}] \neq \perp$ being true (setting flag bad_1) during some call to oracle LR in game G_1 can be upper bounded as follows:

$$\Pr[\mathsf{bad}_1^{\mathrm{G}_1}] \leq \sum_{i=0}^{q_{\mathrm{LR}}-1} \frac{q_{\mathrm{RO}} + i}{2^{\mathsf{F.kl}}} = \frac{(2 \cdot q_{\mathrm{RO}} + q_{\mathrm{LR}} - 1) \cdot q_{\mathrm{LR}}}{2^{\mathsf{F.kl}+1}}.$$

Setting flag bad_2 in game G_3 means that table entry $T[z,\ell]$ was initialized during adversary's call to $\mathrm{LR}(m_0, m_1)$, so $z = \langle r_0, m_b\rangle$ for some r_0 that is the prefix of the secret key k that was subsequently used to run $\mathsf{SE.Enc}(k, m_b)$ at the end of this call to oracle LR. We use this observation to build an adversary $\mathcal{P}$ against PKR-security of SE with respect to $\mathsf{F.kl}$, simulating game G_3 for $\mathcal{D}_{\mathsf{EMDK}}$ as defined in Fig. 37, such that

$$\Pr[\mathsf{bad}_2^{\mathrm{G}_3}] \leq \mathrm{G}^{\mathsf{pkr}}_{\mathsf{SE},\mathsf{F.kl},\mathcal{P}}.$$

Note that adversary $\mathcal{P}$ makes a query to its oracle GuessKey regardless of whether $T[z,\ell] \neq \perp$ would be true in game G_3, because it cannot check this condition itself. However, any time this condition would be true while $\mathcal{D}_{\mathsf{EMDK}}$ is playing in (simulated) game G_3 – adversary $\mathcal{P}$ would succeed to break the PKR-security of SE accordingly.

In G_3 the consistency between oracles LR and RO is no longer maintained (both oracles only write to table T and never read from it). We use this to build an adversary $\mathcal{D}_{\mathsf{SE}}$ against OTIND-security of SE, simulating game G_3 for $\mathcal{D}_{\mathsf{EMDK}}$ as defined in Fig. 38, such that

$$\Pr[\mathrm{G}_3] \leq \mathrm{G}^{\mathsf{otind}}_{\mathsf{SE},\mathcal{D}_{\mathsf{SE}}}.$$

Together, all of the above produce the claim in the theorem statement.

```
Adversary $\mathcal{D}_{\mathsf{SE}}^{\mathrm{LR}}$
$b' \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{EMDK}}^{\mathrm{LRSim},\mathrm{ROSim}}$ ; Return $b'$
LRSim($m_0, m_1$)
Return LR($m_0, m_1$)
ROSim($z, \ell$)
If $\mathsf{local}[z,\ell]$ then return $T[z,\ell]$
$\mathsf{local}[z,\ell] \leftarrow \mathsf{true}$ ; $T[z,\ell] \leftarrow\!\!\$\ \{0,1\}^{\ell}$ ; Return $T[z,\ell]$
```

Figure 38: Adversary $\mathcal{D}_{\mathsf{SE}}$ for proof of Theorem 4.2. $\mathcal{D}_{\mathsf{SE}}$ simulates game G_3 for adversary $\mathcal{D}_{\mathsf{EMDK}}$. The highlighted lines mark the code of G_3's simulated oracles changed by $\mathcal{D}_{\mathsf{SE}}$.

```
Games $\mathrm{G}_0$–$\mathrm{G}_2$
$\mathsf{win} \leftarrow \mathsf{true}$ ; $(i,k) \leftarrow\!\!\$\ \mathcal{G}_{\mathsf{EMDK}}^{\mathrm{Enc},\mathrm{RO}}$ ; If $i \notin [n]$ then return false
$m \leftarrow \mathsf{SE.Dec}(k, \mathsf{c}[i])$ ; If $m = \perp$ then return false
If $\mathsf{c}[i] \neq \mathsf{SE.Enc}(k,m)$ then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$
    Return false                                                    // $\mathrm{G}_1$, $\mathrm{G}_2$
$r_0 \leftarrow k[1 \dots \mathsf{F.kl}]$ ; $r_1 \leftarrow k[\mathsf{F.kl}+1 \dots \mathsf{SE.kl}]$
If $r_1 \neq \mathrm{RO}(\langle r_0, m\rangle, \mathsf{F.ol})$ then return false
Return $m \neq \perp$ and $m \neq \mathsf{m}[i]$ and win
Enc($m$)
$r_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$ ; $r_1 \leftarrow\!\!\$\ \mathrm{RO}(\langle r_0, m\rangle, \mathsf{F.ol})$ ; $k \leftarrow r_0 \,\|\, r_1$ ; $c \leftarrow \mathsf{SE.Enc}(k,m)$
$n \leftarrow n+1$ ; $\mathsf{m}[n] \leftarrow m$ ; $\mathsf{c}[n] \leftarrow c$ ; Return $(k,c)$
RO($z, \ell$)
If $T[z,\ell] = \perp$ then
    $\langle r_0, m\rangle \leftarrow z$
    $T[z,\ell] \leftarrow\!\!\$\ \{0,1\}^{\ell}$
    $r_1 \leftarrow T[z,\ell]$ ; $k \leftarrow r_0 \,\|\, r_1$ ; $c \leftarrow \mathsf{SE.Enc}(k,m)$
    If $\exists (m',c) \in W : m' \neq m$ then
        $\mathsf{bad}_1 \leftarrow \mathsf{true}$
        $\mathsf{win} \leftarrow \mathsf{false}$                                        //       $\mathrm{G}_2$
    $W \leftarrow W \cup \{(m,c)\}$
Return $T[z,\ell]$
```

Figure 39: Games G_0–G_3 for proof of Theorem 4.3. The code added by expanding the algorithms of EMDK in game $\mathrm{G}_{\mathsf{EMDK},\mathcal{G}_{\mathsf{EMDK}}}^{\mathsf{rob}}$ is highlighted in gray. The code added for the transitions between games is highlighted in green.

G Proof of Theorem 4.3

Consider games G_0–G_2 in Fig. 39. Lines not annotated with comments are common to all games. Game G_0 is equivalent to $\mathrm{G}_{\mathsf{EMDK},\mathcal{G}_{\mathsf{EMDK}}}^{\mathsf{rob}}$ when F is modeled as a random oracle, so

$$\mathsf{Adv}_{\mathsf{EMDK}}^{\mathsf{rob}}(\mathcal{G}_{\mathsf{EMDK}}) = \Pr[\mathrm{G}_0].$$

Games G_0 and G_1 are identical until bad_0; games G_1 and G_2 are identical until bad_1. According to the Fundamental Lemma of Game Playing [20] we have

$$\Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] \leq \Pr[\mathsf{bad}_0^{\mathrm{G}_0}] \text{ and } \Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_2] \leq \Pr[\mathsf{bad}_1^{\mathrm{G}_1}]$$

```
Adversary $\mathcal{U}^{\text{Enc}}$
$(i,k) \leftarrow\$ \mathcal{G}_{\mathsf{EMDK}}^{\text{EncSim},\text{ROSim}}$ ; If $i \notin [n]$ then return false
$m \leftarrow \mathsf{SE.Dec}(k,\mathsf{c}[i])$ ; If $m = \perp$ then return false
$c' \leftarrow \text{Enc}(k,m)$ ; Return $(1,\mathsf{c}[i])$
EncSim($m$)
$r_0 \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$ ; $r_1 \leftarrow\$ \text{RO}(\langle r_0, m\rangle, \mathsf{F.ol})$ ; $k \leftarrow r_0 \| r_1$ ; $c \leftarrow \mathsf{SE.Enc}(k,m)$
$n \leftarrow n+1$ ; $\mathsf{m}[n] \leftarrow m$ ; $\mathsf{c}[n] \leftarrow c$ ; Return $(k,c)$
ROSim($z,\ell$)
If $T[z,\ell] = \perp$ then $T[z,\ell] \leftarrow\$ \{0,1\}^{\ell}$
Return $T[z,\ell]$
```

Figure 40: Adversary $\mathcal{U}$ for proof of Theorem 4.3. $\mathcal{U}$ simulates game G_0 for adversary $\mathcal{G}_{\mathsf{EMDK}}$.

```
Adversary $\mathcal{G}_{\mathsf{SE}}^{\text{Enc}}$
$(i,k) \leftarrow\$ \mathcal{G}_{\mathsf{EMDK}}^{\text{EncSim},\text{ROSim}}$
EncSim($m$)
$r_0 \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$ ; $r_1 \leftarrow\$ \text{RO}(\langle r_0, m\rangle, \mathsf{F.ol})$ ; $k \leftarrow r_0 \| r_1$ ; $c \leftarrow \mathsf{SE.Enc}(k,m)$
$n \leftarrow n+1$ ; $\mathsf{m}[n] \leftarrow m$ ; $\mathsf{c}[n] \leftarrow c$ ; Return $(k,c)$
ROSim($z,\ell$)
If $T[z,\ell] = \perp$ then
    $\langle r_0, m\rangle \leftarrow z$ ; $T[z,\ell] \leftarrow\$ \text{Enc}(r_0,m)$
Return $T[z,\ell]$
```

Figure 41: Adversary $\mathcal{G}_{\mathsf{SE}}$ for proof of Theorem 4.3. $\mathcal{G}_{\mathsf{SE}}$ simulates game G_1 for adversary $\mathcal{G}_{\mathsf{EMDK}}$. The highlighted lines mark the code of G_1's simulated oracle changed by $\mathcal{G}_{\mathsf{SE}}$.

where $\Pr[\mathsf{bad}^{\mathsf{Q}}]$ denotes the probability of setting bad flag in game Q.

Setting flag bad_0 in game G_0 means that $\mathsf{c}[i] \neq \mathsf{SE.Enc}(k,m)$ but $m = \mathsf{SE.Dec}(k,\mathsf{c}[i])$ meaning there exists $c' = \mathsf{SE.Enc}(k,m)$ such that $c' \neq \mathsf{c}[i]$ and both decrypt to m under key k. We use this to build an adversary $\mathcal{U}$ against UNIQUE-security of SE, simulating game G_0 for $\mathcal{G}_{\mathsf{EMDK}}$ as defined in Fig. 40, such that

$$\Pr[\mathsf{bad}_0^{\mathrm{G}_0}] \le \mathrm{G}_{\mathsf{SE},\mathcal{U}}^{\mathsf{unique}}.$$

Note that our definition of symmetric encryption schemes requires the decryption correctness to hold for all keys in $\{0,1\}^{\mathsf{SE.kl}}$, so the attack works even though the adversary $\mathcal{G}_{\mathsf{EMDK}}$ can return an arbitrary key k. This step of the proof also uses the assumption that SE is deterministic.

Intuitively, the ciphertext uniqueness property of SE (used in transition from G_0 to G_1) ensures that $\mathcal{G}_{\mathsf{EMDK}}$ winning in game G_1 is equivalent to finding k_1 and $m_0 \neq m_1$ such that $m_1 = \mathsf{SE.Dec}(k_1, \mathsf{SE.Enc}(k_0,m_0))$ for a k_0 that is generated during some call to Enc (and depends on m_0). The next step of the proof will reduce this to breaking the weak robustness of SE.

Setting flag bad_1 in game G_1 means that (r_0,m) and (r_0',m') for some $m \neq m'$ are two inputs to oracle RO that both happened to produce the same ciphertext c. We use this to build an adversary $\mathcal{G}_{\mathsf{SE}}$ against WROB-security of SE with respect to F.ol, simulating game G_1 for $\mathcal{G}_{\mathsf{EMDK}}$ as defined in Fig. 41. Adversary $\mathcal{G}_{\mathsf{SE}}$ sets the falg win in game $\mathrm{G}_{\mathsf{SE},\ell}^{\mathsf{wrob}}$ whenever adversary $\mathcal{G}_{\mathsf{EMDK}}$ sets flag bad_1 in game G_1, so

$$\Pr[\mathsf{bad}_1^{\mathrm{G}_1}] \le \mathrm{G}_{\mathsf{SE},\mathsf{F.ol},\mathcal{G}_{\mathsf{SE}}}^{\mathsf{wrob}}.$$

Finally, G_2's random oracle RO is programmed to set $\mathsf{win} \leftarrow \mathsf{false}$ whenever its output would otherwise enable adversary $\mathcal{G}_{\mathsf{EMDK}}$ to win in game G_2, meaning

$$\Pr[\mathrm{G}_2] = 0.$$

Together, all of the above produce the claim in the theorem statement.

H Proof of Lemma 5.1

First, we prove that $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}_{\mathsf{exhaustive},n}) \geq 1 - 2^{\mathsf{SE.kl}-n}$. Let b denote the challenge bit in game $\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R},\mathcal{D}_{\mathsf{exhaustive},n}}$, and let b' denote the corresponding output value of $\mathcal{D}_{\mathsf{exhaustive},n}$. Adversary $\mathcal{D}_{\mathsf{exhaustive},n}$ runs an exhaustive search over each of the $2^{\mathsf{F.kl}}$ possible keys that can be used by $\mathsf{IMSG\text{-}EMDK}$ for message $m'_1 = \langle m_1, id_s, \mathcal{I}\rangle$. If $b = 1$ then the adversary will always find the real key k that satisfies $\mathsf{SE.Dec}(k, c_{se}) = m'_1$, and hence $\Pr[\, b' = 1 \,|\, b = 1\,] = 1$. If $b = 0$ then m_1 is uniformly random and independent of c_{se}. The adversary returns 1 if the condition $\mathsf{SE.Dec}(k, c_{se}) = m'_1$ checks to be true. There are 2^n distinct values that m'_1 can take, and there are $2^{\mathsf{SE.kl}}$ different keys that can be used to decrypt c_{se}. So the probability that there exists a key that decrypts c_{se} into m'_1 is at most $2^{\mathsf{SE.kl}}/2^n$, meaning $\Pr[\, b' = 1 \,|\, b = 0\,] \leq 2^{\mathsf{SE.kl}-n}$. This proves the claim about $\mathcal{D}_{\mathsf{exhaustive},n}$.

Next, we prove that $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}_{\mathsf{birthday}}) \geq 1/8 - 2^{\mathsf{F.kl}-128}$. Let b denote the challenge bit in game $\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R},\mathcal{D}_{\mathsf{birthday}}}$, and let b' denote the corresponding output value of $\mathcal{D}_{\mathsf{birthday}}$. If $b = 1$ then adversary $\mathcal{D}_{\mathsf{birthday}}$ makes 2^p calls to oracle LR, each time encrypting the same message $m_1 = 0^{128}$. If the value of $r_0 \in \{0,1\}^{\mathsf{F.kl}}$ (sampled in $\mathsf{IMSG\text{-}EMDK.Enc}$) is the same across any two calls to LR, then both calls will return the same SE ciphertext c_{se}, and the adversary will return 1. According to the birthday attack bounds from Appendix D, we have

$$\Pr[\, b' = 1 \,|\, b = 1\,] \geq C(2^{\mathsf{F.kl}}, 2^p) \geq 0.3 \cdot \frac{2^p \cdot (2^p - 1)}{2^{\mathsf{F.kl}}} > 2^{2p-\mathsf{F.kl}-3},$$

because $0.3 \cdot 2^p \cdot (2^p - 1) > \frac{1}{8} \cdot 2^{2p}$ for all $p \geq 1$. The lower bound can be used because $2^p = 2^{\lceil \mathsf{F.kl}/2 \rceil} \leq \sqrt{2 \cdot 2^{\mathsf{F.kl}}}$ holds. If $b = 0$ then adversary $\mathcal{D}_{\mathsf{birthday}}$ calls oracle LR to encrypt 2^p distinct messages. However, two distinct messages might get encrypted into the same $\mathsf{IMSG\text{-}EMDK}$ ciphertext c_{se} since each message could get encrypted under a different key. We use the ideal cipher model to upper bound the probability that any two calls to LR result in the same ciphertext c_{se}. Let AES be the ideal cipher with 128-bit block length. Let $m' = \langle m_0, id_s, \{id_r\}\rangle$. Without loss of generality, assume that $|m'_0| \geq 128$ when $|m_0| = p$, and assume that the entirety of m_0 is encoded in the first 128-bits of m'_0. Then the probability that two different calls to oracle LR returned ciphertexts that start with the same 128-bit block is as follows:

$$\Pr[\, b' = 1 \,|\, b = 0\,] \leq C(2^{128}, 2^p) \leq 0.5 \cdot \frac{2^p \cdot (2^p - 1)}{2^{128}} \leq 2^{2p-129}.$$

The above bounds for $p = \lceil \mathsf{F.kl}/2 \rceil$ give the following:

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}_{\mathsf{birthday}}) > 2^{2p-\mathsf{F.kl}-3} - 2^{2p-129} \geq 1/8 - 2^{\mathsf{F.kl}-128}.$$

Adversary $\mathcal{D}_{\mathsf{birthday}}$ has to maintain a set S that contains up to 2^p elements, giving a runtime complexity roughly $2^p \cdot \log_2 2^p = 2^p \cdot p$.

Finally, we prove that $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R_m}}(\mathcal{D}_{\mathsf{ADR02}}) \geq 2^{-\mathsf{F.ol}}$. Let b denote the challenge bit in game $\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R_m},\mathcal{D}_{\mathsf{ADR02}}}$, and let b' denote the corresponding output value of $\mathcal{D}_{\mathsf{ADR02}}$. The response from oracle LR contains $\mathsf{IMSG\text{-}EMDK}$ ciphertext c_{se} that encrypts $m'_b = \langle m_b, id_s, \{id_r\}\rangle$ under some key $k = r_0 \,\|\, r_1$ such that $r_1 = \mathrm{RO}(\langle r_0, m'_b\rangle, \mathsf{F.ol})$. Scheme $\mathsf{IMSG\text{-}EMDK}$ is defined to use AES-CTR with a fixed IV as the underlying encryption scheme. Let $m''_1 = \langle m_1, id_c, \{id_r\}\rangle$, as defined in the code of adversary $\mathcal{D}_{\mathsf{ADR02}}$, where $|id_s| = |id_c|$. The adversary mauls c_{se}, producing a new ciphertext $c'_{se} = c_{se} \oplus m'_1 \oplus m''_1$. According to the properties of the uniquely decodable encoding $\langle \dots \rangle$ as defined in Section 2, and according to malleability of AES-CTR, the mauled AES-CTR ciphertext c'_{se} decrypts to $m''_b = \langle m_b, id_c, \{id_r\}\rangle$ under the originally used key $k = r_0 \,\|\, r_1$. This causes the initial sender's identity id_s to be replaced with the corrupted sender's identity id_c. However, the $\mathsf{IMSG\text{-}EMDK}$ scheme also checks the integrity of the key (after decrypting the ciphertext with AES-CTR), meaning the decryption fails unless $r_1 = \mathrm{RO}(\langle r_0, m''_b\rangle, \mathsf{F.ol})$ is true. We know that $m''_b \neq m'_b$, so the probability of this condition being true is $2^{-\mathsf{F.ol}}$ in the ROM. It follow that $\Pr[\, b' = 1 \,|\, b = 1\,] = 2^{-\mathsf{F.ol}}$ and $\Pr[\, b' = 1 \,|\, b = 0\,] = 0$. The latter is because even if the $\mathsf{IMSG\text{-}EMDK}$ decryption succeeds, the oracle VerDec would return message m_0 that is not equal to m_1.

```
Games $G_0$–$G_2$
$\pi \leftarrow \mathsf{MRPKE.Setup}$ ; $\mathcal{F}_{\mathsf{SC}}^{\textsc{NewH},\textsc{NewC},\textsc{Exp},\textsc{SigEnc},\textsc{VerDec}}(\pi)$ ; Return win

$\textsc{NewH}(id)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(vk, tk) \leftarrow\$ \mathsf{DS.Kg}$ ; $(ek, dk) \leftarrow\$ \mathsf{MRPKE.Kg}(\pi)$ ; $pk \leftarrow (vk, ek)$
$\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow (tk, dk)$ ; Return $pk$

$\textsc{NewC}(id, pk, sk)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return true

$\textsc{Exp}(id)$
If not $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$

$\textsc{SigEnc}(id_s, \mathcal{I}, m, ad)$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) then return $\perp$
$\mathcal{R}_{pke} \leftarrow \emptyset$ ; $\mathcal{C} \leftarrow \emptyset$
For each $id_r \in \mathcal{I}$ do
    $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $\mathcal{R}_{pke} \leftarrow \mathcal{R}_{pke} \cup \{(id_r, ek_r)\}$
$m_{pke} \leftarrow \langle m, id_s, \mathcal{I}\rangle$ ; $\mathcal{C}_{pke} \leftarrow\$ \mathsf{MRPKE.Enc}(\pi, \mathcal{R}_{pke}, m_{pke})$ ; $(tk_s, dk_s) \leftarrow \mathsf{sk}[id_s]$
For each $(id_r, c_{pke}) \in \mathcal{C}_{pke}$ do
    $\sigma \leftarrow\$ \mathsf{DS.Sig}(tk_s, \langle c_{pke}, ad\rangle)$ ; $c \leftarrow (c_{pke}, \sigma)$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
    $Q^* \leftarrow Q^* \cup \{((id_s, id_r, m, ad, c_{pke}), \sigma)\}$ ; $Q_{\mathsf{DS}} \leftarrow Q_{\mathsf{DS}} \cup \{((id_s, c_{pke}, ad), \sigma)\}$
    $W \leftarrow W \cup \{(m_{pke}, c_{pke})\}$
Return $\mathcal{C}$

$\textsc{VerDec}(id_s, id_r, c, ad)$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return $\perp$
$(c_{pke}, \sigma) \leftarrow c$ ; $(vk_s, ek_s) \leftarrow \mathsf{pk}[id_s]$ ; $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $(tk_r, dk_r) \leftarrow \mathsf{sk}[id_r]$
$d \leftarrow \mathsf{DS.Ver}(vk_s, \langle c_{pke}, ad\rangle, \sigma)$ ; If not $d$ then return $\perp$
$z_0^* \leftarrow ((id_s, c_{pke}, ad), \sigma)$
If $\nexists z_1^* \in Q_{\mathsf{DS}}$ : $\mathsf{R}^*.\mathsf{Vf}(z_0^*, z_1^*)$ and not $\mathsf{exp}[id_s]$ then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$
    Return $\perp$                                                    // $G_1$, $G_2$
$m_{pke} \leftarrow \mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$ ; If $m_{pke} = \perp$ then return $\perp$
If $\exists (m_0, c_{pke}) \in W$ : $m_0 \neq m_{pke}$ then
    $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    Return $\perp$                                                    //       $G_2$
$\langle m, id_s^*, \mathcal{I}\rangle \leftarrow m_{pke}$ ; If $id_s \neq id_s^*$ or $id_r \notin \mathcal{I}$ then return $\perp$
If $m = \perp$ then return $\perp$
$z_0 \leftarrow ((id_s, id_r, m, ad, c_{pke}), \sigma)$ ; If $\exists z_1 \in Q^*$ : $\mathsf{R}^*.\mathsf{Vf}(z_0, z_1)$ then return $m$
$\mathsf{cheated} \leftarrow \mathsf{exp}[id_s]$ ; If not cheated then win $\leftarrow$ true
Return $m$
```

Figure 42: Games G_0–G_2 for proof of Theorem 5.2. The code added by expanding $\mathsf{R.Vf}$ and the algorithms of SC in game $G^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}_{\mathsf{SC}}}$ is highlighted in gray. The code added for the transitions between games is highlighted in green.

I Proof of Theorem 5.2

Consider games G_0–G_2 in Fig. 42. Lines not annotated with comments are common to all games. All games expand the code of $\mathsf{R.Vf}$ (for $\mathsf{R} = \mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*]$) and that of algorithms of SC.

The former means that oracle VerDec now evaluates $\mathsf{R}^*.\mathsf{Vf}$, with respect to an adjusted set Q^* that is built according to the definition of $\mathsf{IMSG\text{-}AUTH\text{-}REL}[\mathsf{R}^*]$. Game G_0 is equivalent to $\mathrm{G}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R},\mathcal{F}_{\mathsf{SC}}}$, so

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}}(\mathcal{F}_{\mathsf{SC}}) = \Pr[\mathrm{G}_0].$$

Games G_0 and G_1 are identical until bad_0; games G_1 and G_2 are identical until bad_1. According to the Fundamental Lemma of Game Playing [20] we have

$$\Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] \le \Pr[\mathsf{bad}_0^{\mathrm{G}_0}] \qquad \text{and} \qquad \Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_2] \le \Pr[\mathsf{bad}_1^{\mathrm{G}_1}],$$

where $\Pr[\mathsf{bad}^{\mathsf{Q}}]$ denotes the probability of setting bad flag in game Q. We will build an adversary $\mathcal{F}_{\mathsf{DS}}$ against the UF-security of DS with respect to R^*, and an adversary $\mathcal{G}$ against the ROB-security of MRPKE such that

$$\Pr[\mathsf{bad}_0^{\mathrm{G}_0}] \le \Pr[\mathrm{G}^{\mathsf{uf}}_{\mathsf{DS},\mathsf{R}^*,\mathcal{F}_{\mathsf{DS}}}], \tag{1}$$

and

$$\Pr[\mathsf{bad}_1^{\mathrm{G}_1}] \le \Pr[\mathrm{G}^{\mathsf{rob}}_{\mathsf{MRPKE},\mathcal{G}}]. \tag{2}$$

Finally, we will show that adversary $\mathcal{F}_{\mathsf{SC}}$ is unable to win in game G_2, so

$$\Pr[\mathrm{G}_2] = 0. \tag{3}$$

Together, all of the above justifies the claim in the theorem statement:

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{SC},\mathsf{R}}(\mathcal{F}_{\mathsf{SC}}) \le \mathsf{Adv}^{\mathsf{uf}}_{\mathsf{DS},\mathsf{R}^*}(\mathcal{F}_{\mathsf{DS}}) + \mathsf{Adv}^{\mathsf{rob}}_{\mathsf{MRPKE}}(\mathcal{G}).$$

We now explain each of the steps and build the corresponding adversaries.

The transition from G_0 to G_1 uses set Q_{DS}. This set is populated with an element $((id_s, c_{pke}, ad), \sigma)$ each time the DS signing key of identity id_s is used to sign a message $\langle c_{pke}, ad\rangle$ to produce a signature σ. If adversary $\mathcal{F}_{\mathsf{SC}}$ sets flag bad_0 in game G_0, then it produced a valid forgery $((id_s, c_{pke}, ad), \sigma)$ for DS with respect to R^*. It means that the signature σ verifies with respect to sender id_s and message $\langle c_{pke}, ad\rangle$, while the sender id_s is not exposed, and checking the forgery with respect to Q_{DS} and R^* does not invalidate it as a trivial attack. To justify Equation (1), we build an adversary $\mathcal{F}_{\mathsf{DS}}$ against the UF-security of DS with respect to R^* as defined in Fig. 43. Adversary $\mathcal{F}_{\mathsf{DS}}$ simulates game G_0 for adversary $\mathcal{F}_{\mathsf{SC}}$. As soon as $\mathcal{F}_{\mathsf{DS}}$ detects a forgery (during the simulation of oracle VerDecSim for adversary $\mathcal{F}_{\mathsf{SC}}$), it calls **abort**, meaning it immediately halts the simulation of $\mathcal{F}_{\mathsf{SC}}$ and returns the forgery as its own output.

The transition from G_1 to G_2 requires that for any ciphertext c_{pke} produced in oracle SigEnc using arbitrary MRPKE key pair and message m_{pke} of adversary's choice, it is hard to find another MRPKE key pair that decrypts c_{pke} to a different message. This should hold even for MRPKE keys that are maliciously chosen to be malformed. In particular, the key pairs used for SigEnc do not need to satisfy the decryption correctness, whereas the key pairs used for VerDec do not have to work with the signcryption algorithm. In games $\mathrm{G}_1, \mathrm{G}_2$ the set W is used to save every message-ciphertext pair (m_{pke}, c_{pke}) produced inside oracle SigEnc. Adversary $\mathcal{F}_{\mathsf{SC}}$ setting bad_1 in game G_1 means that it found a key pair that decrypts one of the ciphertexts from W into a different message (note that if two calls to SigEnc map different messages to the same ciphertext, and one of the used key pairs can decrypt this ciphertext, then adversary can use that to win the game). To justify Equation (1), we build an adversary $\mathcal{G}$ against the ROB-security of MRPKE, as defined in Fig. 44. Adversary $\mathcal{G}$ simulates game G_1 for adversary $\mathcal{F}_{\mathsf{SC}}$. Whenever adversary $\mathcal{F}_{\mathsf{SC}}$ triggers bad_1, adversary $\mathcal{G}$ saves (in its output variable out) the ciphertext c_{pke} and the key pair ek_r, dk_r that decrypts c_{pke} to a message different than the one that produced this ciphertext during

```
Adversary $\mathcal{F}_{\mathsf{DS}}^{\textsc{NewUser},\textsc{Exp},\textsc{Sign}}$
$\pi \leftarrow \mathsf{MRPKE.Setup}$ ; $\mathcal{F}_{\mathsf{SC}}^{\textsc{NewHSim},\textsc{NewCSim},\textsc{ExpSim},\textsc{SigEncSim},\textsc{VerDecSim}}(\pi)$
$\textsc{NewHSim}(id)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $vk \leftarrow\$ \textsc{NewUser}(id)$ ; $(ek, dk) \leftarrow\$ \mathsf{MRPKE.Kg}(\pi)$
$pk \leftarrow (vk, ek)$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow (\perp, dk)$ ; Return $pk$
$\textsc{NewCSim}(id, pk, sk)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $\mathsf{true}$
$\textsc{ExpSim}(id)$
If not $\mathsf{initialized}[id]$ then return $\perp$
If not $\mathsf{exp}[id]$ then
   $tk \leftarrow \textsc{Exp}(id)$ ; $(\perp, dk) \leftarrow \mathsf{sk}[id]$ ; $\mathsf{sk}[id] \leftarrow (tk, dk)$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$
$\textsc{SigEncSim}(id_s, \mathcal{I}, m, ad)$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) then return $\perp$
$\mathcal{R}_{pke} \leftarrow \emptyset$ ; $\mathcal{C} \leftarrow \emptyset$
For each $id_r \in \mathcal{I}$ do
   $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $\mathcal{R}_{pke} \leftarrow \mathcal{R}_{pke} \cup \{(id_r, ek_r)\}$
$m_{pke} \leftarrow \langle m, id_s, \mathcal{I}\rangle$ ; $\mathcal{C}_{pke} \leftarrow\$ \mathsf{MRPKE.Enc}(\pi, \mathcal{R}_{pke}, m_{pke})$ ; $(tk_s, dk_s) \leftarrow \mathsf{sk}[id_s]$
For each $(id_r, c_{pke}) \in \mathcal{C}_{pke}$ do
   If $\mathsf{exp}[id_s]$ then $\sigma \leftarrow\$ \mathsf{DS.Sig}(tk_s, \langle c_{pke}, ad\rangle)$ else $\sigma \leftarrow\$ \textsc{Sign}(id_s, \langle c_{pke}, ad\rangle)$
   $c \leftarrow (c_{pke}, \sigma)$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
   $Q^* \leftarrow Q^* \cup \{((id_s, id_r, m, ad, c_{pke}), \sigma)\}$ ; $Q_{\mathsf{DS}} \leftarrow Q_{\mathsf{DS}} \cup \{((id_s, c_{pke}, ad), \sigma)\}$
Return $\mathcal{C}$
$\textsc{VerDecSim}(id_s, id_r, c, ad)$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return $\perp$
$(c_{pke}, \sigma) \leftarrow c$ ; $(vk_s, ek_s) \leftarrow \mathsf{pk}[id_s]$ ; $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $(tk_r, dk_r) \leftarrow \mathsf{sk}[id_r]$
$d \leftarrow \mathsf{DS.Ver}(vk_s, \langle c_{pke}, ad\rangle, \sigma)$ ; If not $d$ then return $\perp$
$z_0^* \leftarrow ((id_s, c_{pke}, ad), \sigma)$
If $\nexists z_1^* \in Q_{\mathsf{DS}}$: $\mathsf{R}^*.\mathsf{Vf}(z_0^*, z_1^*)$ and not $\mathsf{exp}[id_s]$ then
   $\mathsf{bad}_0 \leftarrow \mathsf{true}$ ; $\mathbf{abort}(id_s, \langle c_{pke}, ad\rangle, \sigma)$
$m_{pke} \leftarrow \mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$ ; If $m_{pke} = \perp$ then return $\perp$
$\langle m, id_s^*, \mathcal{I}\rangle \leftarrow m_{pke}$ ; If $id_s \neq id_s^*$ or $id_r \notin \mathcal{I}$ then return $\perp$
If $m = \perp$ then return $\perp$
$z_0 \leftarrow ((id_s, id_r, m, ad, c_{pke}), \sigma)$ ; If $\exists z_1 \in Q^*$: $\mathsf{R}^*.\mathsf{Vf}(z_0, z_1)$ then return $m$
$\mathsf{cheated} \leftarrow \mathsf{exp}[id_s]$ ; If not $\mathsf{cheated}$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $m$
```

Figure 43: Adversary $\mathcal{F}_{\mathsf{DS}}$ for proof of Theorem 5.2. $\mathcal{F}_{\mathsf{DS}}$ simulates game G_0 for adversary $\mathcal{F}_{\mathsf{SC}}$. The highlighted lines mark the code of G_0's simulated oracles changed by $\mathcal{F}_{\mathsf{DS}}$.

a prior call to oracle SigEnc.

We now justify Equation (3). The adversary $\mathcal{F}_{\mathsf{SC}}$ can only win in game G_2 by setting win in oracle VerDec. We show that this can never happen for any $\mathsf{R}^* \in \{\mathsf{R_m}, \mathsf{R_{id}}\}$. Consider any $\mathcal{F}_{\mathsf{SC}}$'s call to VerDec that reached the instruction $z_0 \leftarrow ((id_s, id_r, m, ad, c_{pke}), \sigma)$. We assume that $\mathsf{exp}[id_s]$ is not true, otherwise VerDec will not set win. Then we can deduce that $((id_s, c_{pke}, ad), \sigma') \in Q_{\mathsf{DS}}$ for some σ', where $\sigma = \sigma'$ is required if $\mathsf{R}^* = \mathsf{R_{id}}$, and $\sigma \neq \sigma'$ is allowed if $\mathsf{R}^* = \mathsf{R_m}$; otherwise, ad-

```
$\underline{\text{Adversary } \mathcal{G}^{\text{Enc}}(\pi)}$
$\mathcal{F}_{\mathsf{SC}}^{\text{NewHSim,NewCSim,ExpSim,SigEncSim,VerDecSim}}(\pi)$ ; Return out
$\underline{\text{NewHSim}(id)}$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(vk, tk) \leftarrow\$ \mathsf{DS.Kg}$ ; $(ek, dk) \leftarrow\$ \mathsf{MRPKE.Kg}(\pi)$
$pk \leftarrow (vk, ek)$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow (tk, dk)$ ; Return $pk$
$\underline{\text{NewCSim}(id, pk, sk)}$
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $\mathsf{true}$
$\underline{\text{ExpSim}(id)}$
If not $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$
$\underline{\text{SigEncSim}(id_s, \mathcal{I}, m, ad)}$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) then return $\bot$
$\mathcal{R}_{pke} \leftarrow \emptyset$ ; $\mathcal{C} \leftarrow \emptyset$
For each $id_r \in \mathcal{I}$ do
    $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $\mathcal{R}_{pke} \leftarrow \mathcal{R}_{pke} \cup \{(id_r, ek_r)\}$
$m_{pke} \leftarrow \langle m, id_s, \mathcal{I} \rangle$ ; $\mathcal{C}_{pke} \leftarrow\$ \text{Enc}(\mathcal{R}_{pke}, m_{pke})$ ; $(tk_s, dk_s) \leftarrow \mathsf{sk}[id_s]$
For each $(id_r, c_{pke}) \in \mathcal{C}_{pke}$ do
    $\sigma \leftarrow\$ \mathsf{DS.Sig}(tk_s, \langle c_{pke}, ad \rangle)$ ; $c \leftarrow (c_{pke}, \sigma)$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
    $Q^* \leftarrow Q^* \cup \{((id_s, id_r, m, ad, c_{pke}), \sigma)\}$ ; $Q_{\mathsf{DS}} \leftarrow Q_{\mathsf{DS}} \cup \{((id_s, c_{pke}, ad), \sigma)\}$
    $W \leftarrow W \cup \{(m_{pke}, c_{pke})\}$
Return $\mathcal{C}$
$\underline{\text{VerDecSim}(id_s, id_r, c, ad)}$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return $\bot$
$(c_{pke}, \sigma) \leftarrow c$ ; $(vk_s, ek_s) \leftarrow \mathsf{pk}[id_s]$ ; $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $(tk_r, dk_r) \leftarrow \mathsf{sk}[id_r]$
$d \leftarrow \mathsf{DS.Ver}(vk_s, \langle c_{pke}, ad \rangle, \sigma)$ ; If not $d$ then return $\bot$
$z_0^* \leftarrow ((id_s, c_{pke}, ad), \sigma)$
If $\nexists z_1^* \in Q_{\mathsf{DS}}$: $\mathsf{R}^*.\mathsf{Vf}(z_0^*, z_1^*)$ and not $\mathsf{exp}[id_s]$ then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$ ; Return $\bot$
$m_{pke} \leftarrow \mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$ ; If $m_{pke} = \bot$ then return $\bot$
If $\exists (m_0, c_{pke}) \in W$: $m_0 \neq m_{pke}$ then
    $\mathsf{bad}_1 \leftarrow \mathsf{true}$ ; $\mathsf{out} \leftarrow (ek_r, dk_r, c_{pke})$
$\langle m, id_s^*, \mathcal{I} \rangle \leftarrow m_{pke}$ ; If $id_s \neq id_s^*$ or $id_r \notin \mathcal{I}$ then return $\bot$
If $m = \bot$ then return $\bot$
$z_0 \leftarrow ((id_s, id_r, m, ad, c_{pke}), \sigma)$ ; If $\exists z_1 \in Q^*$: $\mathsf{R}^*.\mathsf{Vf}(z_0, z_1)$ then return $m$
$\mathsf{cheated} \leftarrow \mathsf{exp}[id_s]$ ; If not $\mathsf{cheated}$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $m$
```

Figure 44: Adversary $\mathcal{G}$ for proof of Theorem 5.2. $\mathcal{G}$ simulates game G_1 for adversary $\mathcal{F}_{\mathsf{SC}}$. The highlighted lines mark the code of G_1's simulated oracles changed by $\mathcal{G}$.

versary would set bad_0 and get $\bot$ as output. Furthermore, we know that c_{pke} was decrypted to the unique m_{pke} for which (m_{pke}, c_{pke}) was inserted into set W during an earlier call to oracle SigEnc; otherwise, adversary would set bad_1 and get $\bot$ as output. Message m_{pke} can be uniquely decoded into $\langle m, id_s^*, \mathcal{I} \rangle$, binding (id_s, c_{pke}, ad) to m and to each recipient's identity from $\mathcal{I}$. It follows that $z_1 = ((id_s, id_r, m, ad, c_{pke}), \sigma') \in Q^*$ for σ' as specified above. Then $\exists z_1 \in Q^*$: $\mathsf{R}^*.\mathsf{Vf}(z_0, z_1)$ is true, so oracle VerDec returns m and does not reach the condition that sets win if $\mathsf{cheated}$ is false. This concludes the proof.

```
Adversary $\mathcal{G}_{\mathsf{EMDK}}^{\textsc{Enc}}$
$(ek, dk, c) \leftarrow\$ \mathcal{G}_{\mathsf{MRPKE}}^{\textsc{EncSim}}(\varepsilon)$ ; $(c_{se}, c_{pke}) \leftarrow c$
$k_1 \leftarrow \mathsf{PKE.Dec}(ek, dk, c_{pke})$
If $k_1 =\perp$ then return $\perp$
$m_1 \leftarrow \mathsf{EMDK.Dec}(k_1, c_{se})$
If $m_1 =\perp$ then return $\perp$
For $i = 1, \ldots, n$ do
  If $\mathsf{c}[i] = c$ and $\mathsf{m}[i] \neq m_1$ then
    return $(i, k_1)$
Return $\perp$

$\textsc{EncSim}(\mathcal{R}, m_0)$
$\mathcal{C} \leftarrow \emptyset$ ; $(k_0, c_{se}) \leftarrow\$ \textsc{Enc}(m_0)$
$n \leftarrow n+1$ ; $\mathsf{m}[n] \leftarrow m_0$ ; $\mathsf{c}[n] \leftarrow c_{se}$
For each $(id_r, ek_r) \in \mathcal{R}$ do
  $c_{pke} \leftarrow\$ \mathsf{PKE.Enc}(ek_r, k_0)$
  $c \leftarrow (c_{se}, c_{pke})$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
  $W \leftarrow W \cup \{(m_0, c)\}$
Return $\mathcal{C}$
```

```
Adversary $\mathcal{G}_{\mathsf{PKE}}^{\textsc{Enc}}$
$(ek, dk, c) \leftarrow\$ \mathcal{G}_{\mathsf{MRPKE}}^{\textsc{EncSim}}(\varepsilon)$ ; $(c_{se}, c_{pke}) \leftarrow c$
$k_1 \leftarrow \mathsf{PKE.Dec}(ek, dk, c_{pke})$
If $k_1 =\perp$ then return $\perp$
Fir $i = 1, \ldots, n$ do
  If $\mathsf{c}[i] = c$ and $\mathsf{m}[i] \neq k_1$ then
    return $(i, ek, dk)$
Return $\perp$

$\textsc{EncSim}(\mathcal{R}, m_0)$
$\mathcal{C} \leftarrow \emptyset$ ; $(k_0, c_{se}) \leftarrow\$ \mathsf{EMDK.Enc}(m_0)$
For each $(id_r, ek_r) \in \mathcal{R}$ do
  $c_{pke} \leftarrow\$ \textsc{Enc}(ek_r, k_0)$ ; $n \leftarrow n+1$
  $\mathsf{m}[n] \leftarrow k_0$ ; $\mathsf{c}[n] \leftarrow c_{pke}$
  $c \leftarrow (c_{se}, c_{pke})$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
  $W \leftarrow W \cup \{(m_0, c)\}$
Return $\mathcal{C}$
```

Figure 45: Adversaries $\mathcal{G}_{\mathsf{EMDK}}$ and $\mathcal{G}_{\mathsf{PKE}}$ for proof of Theorem 5.3. Both adversaries simulate game $\mathrm{G}_{\mathsf{MRPKE}}^{\mathsf{rob}}$ for adversary $\mathcal{G}_{\mathsf{MRPKE}}$. The highlighted lines mark the code of $\mathrm{G}_{\mathsf{MRPKE}}^{\mathsf{rob}}$'s simulated oracle changed by the corresponding adversary.

J Proof of Theorem 5.3

We build an adversary $\mathcal{G}_{\mathsf{EMDK}}$ against the ROB-security of EMDK, and an adversary $\mathcal{G}_{\mathsf{PKE}}$ against the ROB-security of PKE as defined in Fig. 45. Both adversaries simulate game $\mathrm{G}_{\mathsf{MRPKE}}^{\mathsf{rob}}$ for adversary $\mathcal{G}_{\mathsf{MRPKE}}$, replacing the encryption algorithm of EMDK or PKE by a call to the corresponding scheme's encryption oracle.

Recall that adversary $\mathcal{G}_{\mathsf{MRPKE}}$ wins in game $\mathrm{G}_{\mathsf{MRPKE}}^{\mathsf{rob}}$ if it returns (ek, dk, c) such that decrypting $c = (c_{se}, c_{pke})$ produces a plaintext $m_1 \neq \perp$ that is different from some other plaintext m_0 that was used to build the ciphertext c. The latter condition means that $\exists (m_0, c) \in W : m_0 \neq m_1$. Adversary $\mathcal{G}_{\mathsf{EMDK}}$ in game $\mathrm{G}_{\mathsf{EMDK}}^{\mathsf{rob}}$ can directly use it to produce a ciphertext c_{se} that correctly decrypts to plaintext m_1 that is different from the plaintext m_0 that was used to obtain c_{se}, so

$$\mathsf{Adv}_{\mathsf{MRPKE}}^{\mathsf{rob}}(\mathcal{G}_{\mathsf{MRPKE}}) \leq \mathsf{Adv}_{\mathsf{EMDK}}^{\mathsf{rob}}(\mathcal{G}_{\mathsf{EMDK}}).$$

Now consider adversary $\mathcal{G}_{\mathsf{PKE}}$ in game $\mathrm{G}_{\mathsf{PKE}}^{\mathsf{rob}}$. Adversary $\mathcal{G}_{\mathsf{PKE}}$ runs $(k_0, c_{se}) \leftarrow\$ \mathsf{EMDK.Enc}(m_0)$ at the beginning of the oracle $\textsc{EncSim}$ that it simulates for adverasry $\mathcal{G}_{\mathsf{MRPKE}}$, meaning that decryption correctness holds for key k_0. In particular, it implies $\mathsf{EMDK.Dec}(k_0, c_{se}) = m_0$. As per above, adversary $\mathcal{G}_{\mathsf{MRPKE}}$ wins its game by returning ek, dk, c for $c = (c_{se}, c_{pke})$ that correctly decrypts to a different message m_1. This means that the key k_1 obtained by by decrypting c_{pke} under ek, dk is different from the key k_0 used to produce c_{se} in an earlier call to $\textsc{EncSim}$. So we have

$$\mathsf{Adv}_{\mathsf{MRPKE}}^{\mathsf{rob}}(\mathcal{G}_{\mathsf{MRPKE}}) \leq \mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{rob}}(\mathcal{G}_{\mathsf{PKE}}).$$

```
Games $\mathrm{G}_0$–$\mathrm{G}_2$
$b \leftarrow\!\!\$\; \{0,1\}$ ; $\pi \leftarrow \mathsf{MRPKE.Setup}$ ; $b' \leftarrow\!\!\$\; \mathcal{D}_{\mathsf{SC}}^{\mathrm{NewH,NewC,Exp,LR,VerDec}}(\pi)$ ; Return $b' = b$
NewH($id$)
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(vk, tk) \leftarrow\!\!\$\; \mathsf{DS.Kg}$ ; $(ek, dk) \leftarrow\!\!\$\; \mathsf{MRPKE.Kg}(\pi)$ ; $pk \leftarrow (vk, ek)$
$\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow (tk, dk)$ ; Return $pk$
NewC($id, pk, sk$)
If $\mathsf{initialized}[id]$ then return $\bot$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $\mathsf{true}$
Exp($id$)
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\bot$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$
LR($id_s, \mathcal{I}, m_0, m_1, ad$)
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) or $|m_0| \neq |m_1|$ then return $\bot$
If $m_0 \neq m_1$ then
   If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\bot$
   For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{R}_{pke} \leftarrow \emptyset$ ; $\mathcal{C} \leftarrow \emptyset$
For each $id_r \in \mathcal{I}$ do
   $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $\mathcal{R}_{pke} \leftarrow \mathcal{R}_{pke} \cup \{(id_r, ek_r)\}$
$m_{pke} \leftarrow \langle m_b, id_s, \mathcal{I}\rangle$ ; $\mathcal{C}_{pke} \leftarrow\!\!\$\; \mathsf{MRPKE.Enc}(\pi, \mathcal{R}_{pke}, m_{pke})$ ; $(tk_s, dk_s) \leftarrow \mathsf{sk}[id_s]$
For each $(id_r, c_{pke}) \in \mathcal{C}_{pke}$ do
   $\sigma \leftarrow\!\!\$\; \mathsf{DS.Sig}(tk_s, \langle c_{pke}, ad\rangle)$ ; $c \leftarrow (c_{pke}, \sigma)$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
   $Q_{\mathsf{MRPKE}} \leftarrow Q_{\mathsf{MRPKE}} \cup \{(id_r, c_{pke})\}$
   If $m_0 \neq m_1$ then
      $Q^* \leftarrow Q^* \cup \{(id_s, id_r, m_0, c_{pke})\}$
      $Q^* \leftarrow Q^* \cup \{(id_s, id_r, m_1, c_{pke})\}$
   Else $\mathsf{m}[id_r, c_{pke}] \leftarrow \langle m_0, id_s, \mathcal{I}\rangle$
Return $\mathcal{C}$
VerDec($id_s, id_r, c, ad$)
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return ($\bot$, "init")
$(c_{pke}, \sigma) \leftarrow c$ ; $(vk_s, ek_s) \leftarrow \mathsf{pk}[id_s]$ ; $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $(tk_r, dk_r) \leftarrow \mathsf{sk}[id_r]$
$d \leftarrow \mathsf{DS.Ver}(vk_s, \langle c_{pke}, ad\rangle, \sigma)$ ; If not $d$ then return ($\bot$, "dec")
If $\mathsf{exp}[id_r]$ then $m_{pke} \leftarrow \mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$                // $\mathrm{G}_1$, $\mathrm{G}_2$
If (not $\mathsf{exp}[id_r]$) and $\mathsf{m}[id_r, c_{pke}] \neq \bot$ then $m_{pke} \leftarrow \mathsf{m}[id_r, c_{pke}]$          // $\mathrm{G}_1$, $\mathrm{G}_2$
If (not $\mathsf{exp}[id_r]$) and $\mathsf{m}[id_r, c_{pke}] = \bot$ and $(id_r, c_{pke}) \in Q_{\mathsf{MRPKE}}$ then // $\mathrm{G}_1$, $\mathrm{G}_2$
   $m_{pke} \leftarrow \mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$                                   // $\mathrm{G}_1$
   Return ($\bot$, "priv")                                                                              //       $\mathrm{G}_2$
If (not $\mathsf{exp}[id_r]$) and $\mathsf{m}[id_r, c_{pke}] = \bot$ and $(id_r, c_{pke}) \notin Q_{\mathsf{MRPKE}}$ then // $\mathrm{G}_1$, $\mathrm{G}_2$
   $m_{pke} \leftarrow \mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$
If $m_{pke} = \bot$ then return ($\bot$, "dec")
$\langle m, id_s^*, \mathcal{I}\rangle \leftarrow m_{pke}$ ; If $id_s \neq id_s^*$ or $id_r \notin \mathcal{I}$ then return ($\bot$, "dec")
If $m = \bot$ then return ($\bot$, "dec")
If $(id_s, id_r, m, c_{pke}) \in Q^*$ then return ($\bot$, "priv")
Return ($m$, "ok")
```

Figure 46: Games G_0–G_2 for proof of Theorem 5.4. The code added by expanding $\mathsf{R.Vf}$ and the algorithms of SC in game $\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R},\mathcal{D}_{\mathsf{SC}}}$ is highlighted in gray. The code added for the transition between games is highlighted in green.

K Proof of Theorem 5.4

Consider games G_0–G_2 of Fig. 46. Lines not annotated with comments are common to all games. Game G_0 is equivalent to $\mathrm{G}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R},\mathcal{D}_{\mathsf{SC}}}$, so

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}_{\mathsf{SC}}) = 2 \cdot \Pr[\mathrm{G}_0] - 1.$$

Game G_1 expands oracle VerDec to obtain plaintext m_{pke} depending on multiple different conditions. In three of four cases m_{pke} is obtained by decrypting c_{pke} as intended; only the second condition differs. For the second condition, note that if c_{pke} was produced as a result of calling oracle LR for some $m_0 = m_1$ then the value of m_{pke} is saved in $\mathsf{m}[id_r, c_{pke}]$, and is used directly in VerDec, without having to decrypt c_{pke}. This is only done if the recipient's identity is not exposed, which guarantees that the recipient's keys were honestly generated and hence the decryption correctness holds (in contrast, keys added by calling oracle NewC might not provide decryption correctness). It follows that games G_0 and G_1 are equivalent, so

$$\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1].$$

Below we will also show that games G_1 and G_2 are equivalent, meaning

$$\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2]. \tag{4}$$

Next we will build an adversary $\mathcal{D}_{\mathsf{MRPKE}}$ against the INDCCA-security of MRPKE such that

$$\Pr[\mathrm{G}_2] \le \mathrm{G}^{\mathsf{indcca}}_{\mathsf{MRPKE},\mathcal{D}_{\mathsf{MRPKE}}}. \tag{5}$$

Together, all of the above proves the claim in the theorem statement:

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{SC},\mathsf{R}}(\mathcal{D}_{\mathsf{SC}}) \le \mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{MRPKE}}(\mathcal{D}_{\mathsf{MRPKE}}).$$

We now justify Equation (4) and build adversary $\mathcal{D}_{\mathsf{MRPKE}}$.

Games G_1 and G_2 are different only in cases when adversary $\mathcal{D}_{\mathsf{SC}}$ queries its oracle VerDec on input that triggers the following condition:

$$(\text{not } \mathsf{exp}[id_r]) \text{ and } \mathsf{m}[id_r, c_{pke}] = \perp \text{ and } (id_r, c_{pke}) \in Q_{\mathsf{MRPKE}}$$

Since the recipient id_r is not exposed, it means its keys were created by calling oracle NewH, and hence provide decryption correctness. According to the decryption correctness of MRPKE, we know $m_{pke} = \mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$ is the same plaintext that was used in an earlier call to LR during which (id_r, c_{pke}) was added to Q_{MRPKE}. The plaintext uniquely encodes some $m_b, id_s, \mathcal{I}$, and these values were also used in LR to populate set Q^* (as guaranteed by $\mathsf{m}[id_r, c_{pke}] = \perp$). It follows that Q^* contains the tuple $(id_s, id_r, m_b, c_{pke})$, and hence the current call to oracle VerDec will eventually return $(\perp, \text{“priv”})$. In this case, game G_2 is defined to immediately return $(\perp, \text{“priv”})$, and is hence functionally equivalent to G_1. This justifies Equation (4).

Next we construct an adversary $\mathcal{D}_{\mathsf{MRPKE}}$ against the INDCCA-security of MRPKE as defined in Fig. 47. Adversary $\mathcal{D}_{\mathsf{MRPKE}}$ simulates game G_2 for adversary $\mathcal{D}_{\mathsf{SC}}$. Note that adversary $\mathcal{D}_{\mathsf{MRPKE}}$ only calls its decryption oracle Dec in game $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{MRPKE}}$ when $\mathsf{initialized}[id_r]$ and $(id_r, c_{pke}) \notin Q_{\mathsf{MRPKE}}$. This guarantees that Dec always returns the result of evaluating $\mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$, and hence ensures perfect simulation. It follows that $\mathcal{D}_{\mathsf{MRPKE}}$ wins in game $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{MRPKE}}$ whenever $\mathcal{D}_{\mathsf{SC}}$ wins in game G_2, justifying Equation (5).

```
Adversary $\mathcal{D}_{\mathsf{MRPKE}}^{\textsc{NewUser},\textsc{Exp},\textsc{LR},\textsc{Dec}}(\pi)$
$b' \leftarrow\$ \mathcal{D}_{\mathsf{SC}}^{\textsc{NewHSim},\textsc{NewCSim},\textsc{ExpSim},\textsc{LRSim},\textsc{VerDecSim}}(\pi)$ ; Return $b'$

$\textsc{NewHSim}(id)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(vk, tk) \leftarrow\$ \mathsf{DS.Kg}$ ; $ek \leftarrow\$ \textsc{NewUser}(id)$ ; $pk \leftarrow (vk, ek)$
$\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow (tk, \perp)$ ; Return $pk$

$\textsc{NewCSim}(id, pk, sk)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; $\mathsf{pk}[id] \leftarrow pk$ ; $\mathsf{sk}[id] \leftarrow sk$ ; Return $\mathsf{true}$

$\textsc{ExpSim}(id)$
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\perp$
If not $\mathsf{exp}[id]$ then
    $dk \leftarrow \textsc{Exp}(id)$ ; $(tk, \perp) \leftarrow \mathsf{sk}[id]$ ; $\mathsf{sk}[id] \leftarrow (tk, dk)$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{sk}[id]$

$\textsc{LRSim}(id_s, \mathcal{I}, m_0, m_1, ad)$
If (not $\mathsf{initialized}[id_s]$) or ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) or $|m_0| \neq |m_1|$ then return $\perp$
If $m_0 \neq m_1$ then
    If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\perp$
    For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{C} \leftarrow \emptyset$ ; $m'_0 \leftarrow \langle m_0, id_s, \mathcal{I}\rangle$ ; $m'_1 \leftarrow \langle m_1, id_s, \mathcal{I}\rangle$ ; $\mathcal{C}_{pke} \leftarrow\$ \textsc{LR}(\mathcal{I}, m'_0, m'_1)$
$(tk_s, dk_s) \leftarrow \mathsf{sk}[id_s]$
For each $(id_r, c_{pke}) \in \mathcal{C}_{pke}$ do
    $\sigma \leftarrow\$ \mathsf{DS.Sig}(tk_s, \langle c_{pke}, ad\rangle)$ ; $c \leftarrow (c_{pke}, \sigma)$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id_r, c)\}$
    $Q_{\mathsf{MRPKE}} \leftarrow Q_{\mathsf{MRPKE}} \cup \{(id_r, c_{pke})\}$
    If $m_0 \neq m_1$ then
        $Q^* \leftarrow Q^* \cup \{(id_s, id_r, m_0, c_{pke})\}$
        $Q^* \leftarrow Q^* \cup \{(id_s, id_r, m_1, c_{pke})\}$
    Else $\mathsf{m}[id_r, c_{pke}] \leftarrow \langle m_0, id_s, \mathcal{I}\rangle$
Return $\mathcal{C}$

$\textsc{VerDecSim}(id_s, id_r, c, ad)$
If (not $\mathsf{initialized}[id_s]$) or (not $\mathsf{initialized}[id_r]$) then return $(\perp, \text{“init”})$
$(c_{pke}, \sigma) \leftarrow c$ ; $(vk_s, ek_s) \leftarrow \mathsf{pk}[id_s]$ ; $(vk_r, ek_r) \leftarrow \mathsf{pk}[id_r]$ ; $(tk_r, dk_r) \leftarrow \mathsf{sk}[id_r]$
$d \leftarrow \mathsf{DS.Ver}(vk_s, \langle c_{pke}, ad\rangle, \sigma)$ ; If not $d$ then return $(\perp, \text{“dec”})$
If $\mathsf{exp}[id_r]$ then $m_{pke} \leftarrow \mathsf{MRPKE.Dec}(\pi, ek_r, dk_r, c_{pke})$
If (not $\mathsf{exp}[id_r]$) and $\mathsf{m}[id_r, c_{pke}] \neq \perp$ then $m_{pke} \leftarrow \mathsf{m}[id_r, c_{pke}]$
If (not $\mathsf{exp}[id_r]$) and $\mathsf{m}[id_r, c_{pke}] = \perp$ and $(id_r, c_{pke}) \in Q_{\mathsf{MRPKE}}$ then return $(\perp, \text{“priv”})$
If (not $\mathsf{exp}[id_r]$) and $\mathsf{m}[id_r, c_{pke}] = \perp$ and $(id_r, c_{pke}) \notin Q_{\mathsf{MRPKE}}$ then
    $m_{pke} \leftarrow \textsc{Dec}(id_r, c_{pke})$
If $m_{pke} = \perp$ then return $(\perp, \text{“dec”})$
$\langle m, id_s^*, \mathcal{I}\rangle \leftarrow m_{pke}$ ; If $id_s \neq id_s^*$ or $id_r \notin \mathcal{I}$ then return $(\perp, \text{“dec”})$
If $m = \perp$ then return $(\perp, \text{“dec”})$
If $(id_s, id_r, m, c_{pke}) \in Q^*$ then return $(\perp, \text{“priv”})$
Return $(m, \text{“ok”})$
```

Figure 47: Adversary $\mathcal{D}_{\mathsf{MRPKE}}$ for proof of Theorem 5.4. $\mathcal{D}_{\mathsf{MRPKE}}$ simulates game G_2 for adversary $\mathcal{D}_{\mathsf{SC}}$. The highlighted lines mark the code of G_2's simulated oracles changed by $\mathcal{D}_{\mathsf{MRPKE}}$.

```
Games G_0–G_2
$b \leftarrow\$ \{0,1\}$ ; $\pi \leftarrow\$ \varepsilon$ ; $b' \leftarrow\$ \mathcal{D}_{\mathsf{MRPKE}}^{\textsc{NewUser},\textsc{Exp},\text{LR},\textsc{Dec}}(\pi)$ ; Return $b' = b$

NewUser(id)
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(ek, dk) \leftarrow\$ \mathsf{PKE.Kg}$ ; $\mathsf{ek}[id] \leftarrow ek$ ; $\mathsf{dk}[id] \leftarrow dk$ ; Return $ek$

Exp(id)
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\perp$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{dk}[id]$

LR(I, m_0, m_1)
If ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) or $|m_0| \neq |m_1|$ then return $\perp$
If $m_0 \neq m_1$ then
    If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\perp$
    For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{C} \leftarrow \emptyset$
If $m_0 \neq m_1$ then
    $(k_1, c_{se}) \leftarrow\$ \mathsf{EMDK.Enc}(m_b)$ ; $k_0 \leftarrow 0^{\mathsf{EMDK.kl}}$
Else
    $(k_1, c_{se}) \leftarrow\$ \mathsf{EMDK.Enc}(m_0)$ ; $k_0 \leftarrow k_1$
For each $id \in \mathcal{I}$ do
    $c_{pke} \leftarrow\$ \mathsf{PKE.Enc}(\mathsf{ek}[id], k_1)$                    // G_0, G_1
    $c_{pke} \leftarrow\$ \mathsf{PKE.Enc}(\mathsf{ek}[id], k_0)$                    //       G_2
    $c \leftarrow (c_{se}, c_{pke})$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id, c)\}$ ; $Q \leftarrow Q \cup \{(id, c)\}$
    $Q_{\mathsf{PKE}} \leftarrow Q_{\mathsf{PKE}} \cup \{(id, c_{pke})\}$ ; $\mathsf{k}[id, c_{pke}] \leftarrow k_1$
Return $\mathcal{C}$

Dec(id, c)
If (not $\mathsf{initialized}[id]$) or $(id, c) \in Q$ then return $\perp$
$(c_{se}, c_{pke}) \leftarrow c$
If $(id, c_{pke}) \in Q_{\mathsf{PKE}}$ then
    $k \leftarrow \mathsf{PKE.Dec}(\mathsf{ek}[id], \mathsf{dk}[id], c_{pke})$ ; If $k = \perp$ then return $\perp$      // G_0
    $k \leftarrow \mathsf{k}[id, c_{pke}]$                                             //    G_1, G_2
    $m \leftarrow \mathsf{EMDK.Dec}(k, c_{se})$
Else
    $k \leftarrow \mathsf{PKE.Dec}(\mathsf{ek}[id], \mathsf{dk}[id], c_{pke})$ ; If $k = \perp$ then return $\perp$
    $m \leftarrow \mathsf{EMDK.Dec}(k, c_{se})$
Return $m$
```

Figure 48: Games G$_0$–G$_2$ for proof of Theorem 5.5. The code added by expanding the algorithms of MRPKE in game $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{MRPKE},\mathcal{D}_{\mathsf{MRPKE}}}$ is highlighted in gray. The code added for the transitions between games is highlighted in green.

L Proof of Theorem 5.5

Consider games G_0–G_2 of Fig. 48. Lines not annotated with comments are common to all games. Game G_0 is equivalent to $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{MRPKE},\mathcal{D}_{\mathsf{MRPKE}}}$, so

$$\mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{MRPKE}}(\mathcal{D}_{\mathsf{MRPKE}}) = 2 \cdot \Pr[\mathrm{G}_0] - 1.$$

Every time a ciphertext c_{pke} is computed by running $\mathsf{PKE.Enc}(\mathsf{ek}[id], k_1)$ in oracle LR of game G_0, the encrypted key k_1 is saved in table entry $\mathsf{k}[id, c_{pke}]$. Game G_1 differs from game G_0 in oracle DEC. Whenever possible, it recovers the value k_1 directly from $\mathsf{k}[id, c_{pke}]$ instead of explicitly running the decryption $\mathsf{PKE.Dec}(\mathsf{ek}[id], \mathsf{dk}[id], c_{pke})$. The two games are equivalent according to the decryption correctness of PKE, so

$$\Pr[\mathrm{G}_0] = \Pr[\mathrm{G}_1].$$

We will build an adversary $\mathcal{D}_{\mathsf{PKE}}$ against the INDCCA-security of PKE as follows. Let g denote the challenge bit in game $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{PKE},\mathcal{D}_{\mathsf{PKE}}}$, and let g' denote the bit returned by $\mathcal{D}_{\mathsf{PKE}}$ in this game. We claim that

$$\Pr[\mathrm{G}_1] = \Pr[\, g' = 1 \mid g = 1\,] \qquad \text{and} \qquad \Pr[\mathrm{G}_2] = \Pr[\, g' = 1 \mid g = 0\,], \tag{6}$$

meaning that

$$\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_2] = \mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{PKE}}(\mathcal{D}_{\mathsf{PKE}}).$$

Finally, we will build an adversary $\mathcal{D}_{\mathsf{EMDK}}$ against the AE-security of EMDK such that

$$\Pr[\mathrm{G}_2] \leq \Pr[\mathrm{G}^{\mathsf{ae}}_{\mathsf{EMDK},\mathcal{D}_{\mathsf{EMDK}}}]. \tag{7}$$

Together, all of the above proves the claim in the theorem statement:

$$\mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{MRPKE}}(\mathcal{D}_{\mathsf{MRPKE}}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{indcca}}_{\mathsf{PKE}}(\mathcal{D}_{\mathsf{PKE}}) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{EMDK}}(\mathcal{D}_{\mathsf{EMDK}}).$$

We now build adversaries $\mathcal{D}_{\mathsf{PKE}}$ and $\mathcal{D}_{\mathsf{EMDK}}$.

To justify the claims in Equation (6), we build an adversary $\mathcal{D}_{\mathsf{PKE}}$ against the INDCCA-security of PKE as defined in Fig. 49. Let g denote the challenge bit in game $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{PKE},\mathcal{D}_{\mathsf{PKE}}}$. Adversary $\mathcal{D}_{\mathsf{PKE}}$ simulates game G_1 for adversary $\mathcal{D}_{\mathsf{MRPKE}}$ when $g = 1$, and it simulates game G_2 for adversary $\mathcal{D}_{\mathsf{MRPKE}}$ when $g = 0$. Adversary $\mathcal{D}_{\mathsf{PKE}}$ is defined to return 1 only when adversary $\mathcal{D}_{\mathsf{MRPKE}}$ would have won in its own game. Note that adversary $\mathcal{D}_{\mathsf{MRPKE}}$ is allowed to expose the decryption keys of any recipients for which it called oracle LR only on challenge messages m_0, m_1 such that $m_0 = m_1$. In this case the outputs of challenge queries do not depend on the challenge bit b, so even exposing the key does not allow adversary $\mathcal{D}_{\mathsf{MRPKE}}$ to trivially win the game. In a similar vein, adversary $\mathcal{D}_{\mathsf{PKE}}$ ensures that $k_0 = k_1$ whenever it has to simulate oracle LR for adversary $\mathcal{D}_{\mathsf{MRPKE}}$ on an input for which $m_0 = m_1$. This subsequently allows adversary $\mathcal{D}_{\mathsf{PKE}}$ to use its EXP oracle to simulate adversary's $\mathcal{D}_{\mathsf{MRPKE}}$ exposure queries. Finally, note that games G_1 and G_2 do not allow adversary $\mathcal{D}_{\mathsf{MRPKE}}$ to call oracle DEC on inputs id, c such that c was previously produced by calling oracle LR to encrypt some message for recipient id. This avoids trivial attacks. However, games G_1 and G_2 do allow adversary $\mathcal{D}_{\mathsf{MRPKE}}$ to call oracle LR to produce a ciphertext $c = (c_{se}, c_{pke})$ for recipient id, and subsequently call oracle DEC on input id, c' for $c' = (c'_{se}, c_{pke})$ such that c'_{se} was not produced during an earlier call to LR. The earlier transition from game G_0 to game G_1 ensures adversary $\mathcal{D}_{\mathsf{PKE}}$ is able to simulate such calls to $\mathcal{D}_{\mathsf{MRPKE}}$'s oracle DEC by doing a lookup in $\mathsf{k}[id, c_{pke}]$ instead of calling its own decryption oracle.

To justify the claim in Equation (7), we build an adversary $\mathcal{D}_{\mathsf{EMDK}}$ against the AE-security

```
Adversary $\mathcal{D}_{\mathsf{PKE}}^{\textsc{NewUser},\textsc{Exp},\text{LR},\textsc{Dec}}$
$b \leftarrow\!\!\$\ \{0,1\}$ ; $\pi \leftarrow\!\!\$\ \varepsilon$ ; $b' \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{MRPKE}}^{\textsc{NewUserSim},\textsc{ExpSim},\text{LRSim},\textsc{DecSim}}(\pi)$
If $b = b'$ then return 1 else return 0

$\textsc{NewUserSim}(id)$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $ek \leftarrow\!\!\$\ \textsc{NewUser}(id)$ ; $\mathsf{ek}[id] \leftarrow ek$ ; $\mathsf{dk}[id] \leftarrow \perp$ ; Return $ek$

$\textsc{ExpSim}(id)$
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\perp$
If not $\mathsf{exp}[id]$ then $\mathsf{dk}[id] \leftarrow \textsc{Exp}(id)$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{dk}[id]$

$\text{LRSim}(\mathcal{I}, m_0, m_1)$
If $(\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id])$ or $|m_0| \neq |m_1|$ then return $\perp$
If $m_0 \neq m_1$ then
  If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\perp$
  For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{C} \leftarrow \emptyset$
If $m_0 \neq m_1$ then
  $(k_1, c_{se}) \leftarrow\!\!\$\ \mathsf{EMDK.Enc}(m_b)$ ; $k_0 \leftarrow 0^{\mathsf{EMDK.kl}}$
Else
  $(k_1, c_{se}) \leftarrow\!\!\$\ \mathsf{EMDK.Enc}(m_0)$ ; $k_0 \leftarrow k_1$
For each $id \in \mathcal{I}$ do
  $c_{pke} \leftarrow\!\!\$\ \text{LR}(id, k_0, k_1)$ ; $c \leftarrow (c_{se}, c_{pke})$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id, c)\}$
  $Q \leftarrow Q \cup \{(id, c)\}$ ; $Q_{\mathsf{PKE}} \leftarrow Q_{\mathsf{PKE}} \cup \{(id, c_{pke})\}$ ; $\mathsf{k}[id, c_{pke}] \leftarrow k_1$
Return $\mathcal{C}$

$\textsc{DecSim}(id, c)$
If (not $\mathsf{initialized}[id]$) or $(id, c) \in Q$ then return $\perp$
$(c_{se}, c_{pke}) \leftarrow c$
If $(id, c_{pke}) \in Q_{\mathsf{PKE}}$ then
  $k \leftarrow \mathsf{k}[id, c_{pke}]$ ; $m \leftarrow \mathsf{EMDK.Dec}(k, c_{se})$
Else
  $k \leftarrow \textsc{Dec}(id, c_{pke})$ ; If $k = \perp$ then return $\perp$
  $m \leftarrow \mathsf{EMDK.Dec}(k, c_{se})$
Return $m$
```

Figure 49: Adversary $\mathcal{D}_{\mathsf{PKE}}$ for proof of Theorem 5.5. Depending on the challenge bit in $\mathrm{G}^{\mathsf{indcca}}_{\mathsf{PKE},\mathcal{D}_{\mathsf{PKE}}}$, adversary $\mathcal{D}_{\mathsf{PKE}}$ simulates game G_1 or G_2 for adversary $\mathcal{D}_{\mathsf{MRPKE}}$. The highlighted lines mark the code of G_1's (or G_2's) simulated oracles changed by $\mathcal{D}_{\mathsf{PKE}}$.

of EMDK as defined in Fig. 50. Adversary $\mathcal{D}_{\mathsf{EMDK}}$ simulates game G_2 for adversary $\mathcal{D}_{\mathsf{MRPKE}}$. To simulate $\mathcal{D}_{\mathsf{MRPKE}}$'s queries to oracle LR, adversary $\mathcal{D}_{\mathsf{EMDK}}$ uses its own oracle LR if $m_0 \neq m_1$, and otherwise encrypts m_0 by calling $\mathsf{EMDK.Enc}(m_0)$ on its own. Adversary $\mathcal{D}_{\mathsf{EMDK}}$ saves either the key identity k_1^{id} returned by its LR oracle, or the key k_1 used for encrypting m_0, accordingly. The saved information is then used to simulate $\mathcal{D}_{\mathsf{MRPKE}}$'s queries to oracle DEC; either by calling $\mathcal{D}_{\mathsf{EMDK}}$'s oracle DEC for key identity k_1^{id}, or by explicitly running the decryption algorithm EMDK.Dec with key k_1.

```
$\underline{\text{Adversary } \mathcal{D}_{\mathsf{EMDK}}^{\text{LR,Dec}}}$
$\pi \leftarrow\!\!\$\ \varepsilon$ ; $b' \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{MRPKE}}^{\text{NewUserSim,ExpSim,LRSim,DecSim}}(\pi)$ ; Return $b'$
$\underline{\text{NewUserSim}(id)}$
If $\mathsf{initialized}[id]$ then return $\perp$
$\mathsf{initialized}[id] \leftarrow \mathsf{true}$ ; $(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}$ ; $\mathsf{ek}[id] \leftarrow ek$ ; $\mathsf{dk}[id] \leftarrow dk$ ; Return $ek$
$\underline{\text{ExpSim}(id)}$
If (not $\mathsf{initialized}[id]$) or $\mathsf{ch}[id]$ then return $\perp$
$\mathsf{exp}[id] \leftarrow \mathsf{true}$ ; Return $\mathsf{dk}[id]$
$\underline{\text{LRSim}(\mathcal{I}, m_0, m_1)}$
If ($\exists id \in \mathcal{I}$: not $\mathsf{initialized}[id]$) or $|m_0| \neq |m_1|$ then return $\perp$
If $m_0 \neq m_1$ then
    If $\exists id \in \mathcal{I}$: $\mathsf{exp}[id]$ then return $\perp$
    For each $id \in \mathcal{I}$ do $\mathsf{ch}[id] \leftarrow \mathsf{true}$
$\mathcal{C} \leftarrow \emptyset$ ; $k_1 \leftarrow \perp$ ; $k_1^{\mathsf{id}} \leftarrow \perp$
If $m_0 \neq m_1$ then
    $(k_1^{\mathsf{id}}, c_{se}) \leftarrow\!\!\$\ \text{LR}(m_0, m_1)$ ; $k_0 \leftarrow 0^{\mathsf{EMDK.kl}}$
Else
    $(k_1, c_{se}) \leftarrow\!\!\$\ \mathsf{EMDK.Enc}(m_0)$ ; $k_0 \leftarrow k_1$
For each $id \in \mathcal{I}$ do
    $c_{pke} \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{ek}[id], k_0)$ ; $c \leftarrow (c_{se}, c_{pke})$ ; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(id, c)\}$ ; $Q \leftarrow Q \cup \{(id, c)\}$
    $Q_{\mathsf{PKE}} \leftarrow Q_{\mathsf{PKE}} \cup \{(id, c_{pke})\}$ ; $\mathsf{k}[id, c_{pke}] \leftarrow (k_1, k_1^{\mathsf{id}})$
Return $\mathcal{C}$
$\underline{\text{DecSim}(id, c)}$
If (not $\mathsf{initialized}[id]$) or $(id, c) \in Q$ then return $\perp$
$(c_{se}, c_{pke}) \leftarrow c$
If $(id, c_{pke}) \in Q_{\mathsf{PKE}}$ then
    $(k_1, k_1^{\mathsf{id}}) \leftarrow \mathsf{k}[id, c_{pke}]$
    If $k_1^{\mathsf{id}} \neq \perp$ then $m \leftarrow \text{Dec}(k_1^{\mathsf{id}}, c_{se})$ else $m \leftarrow \mathsf{EMDK.Dec}(k_1, c_{se})$
Else
    $k \leftarrow \mathsf{PKE.Dec}(\mathsf{ek}[id], \mathsf{dk}[id], c_{pke})$ ; If $k = \perp$ then return $\perp$
    $m \leftarrow \mathsf{EMDK.Dec}(k, c_{se})$
Return $m$
```

Figure 50: Adversary $\mathcal{D}_{\mathsf{EMDK}}$ for proof of Theorem 5.5. $\mathcal{D}_{\mathsf{EMDK}}$ simulates game G_2 for adversary $\mathcal{D}_{\mathsf{MRPKE}}$. The highlighted lines mark the code of G_2's simulated oracles changed by $\mathcal{D}_{\mathsf{EMDK}}$.