Security under Message-Derived Keys: Signcryption in iMessage

Mihir Bellare
UCSD

Igors Stepanovs
ETH ⟵ UCSD

EUROCRYPT 2020
May 15, 2020

Even though some of these services are **encrypted**, the revelations about the **NSA's PRISM** and other **spy programs** showed that they were never really secure or private. Even today, new privacy controversies/scandals keep popping up related to these services.

The **lack of real privacy** and **security** on the big-name services has resulted in the development of newer messaging apps and services. These aim to provide secure communications that are actually secure. As of now (April 2020), there are **dozens of messaging apps available that claim to be secure**. In this article, we've surveyed the field and come up with what we consider to be the **5 best secure messaging apps of 2020**.

Characteristics we look for in a secure messaging app:

- Independence from the major tech companies
- End-to-end (E2E) encryption

Even though some of these services are **encrypted**, the revelations about the **NSA's PRISM** and other **spy programs** showed that they were never really secure or private. Even today, new privacy controversies/scandals keep popping up related to these services.

The **lack of real privacy** and **security** on the big-name services has resulted in the development of newer messaging apps and services. These aim to provide secure communications that are actually secure. As of now (April 2020), there are **dozens of messaging apps available that claim to be secure**. In this article, we've surveyed the field and come up with what we consider to be the **5 best secure messaging apps of 2020**.

Characteristics we look for in a secure messaging app:

- Independence from the major tech companies
- End-to-end (E2E) encryption

Widely used

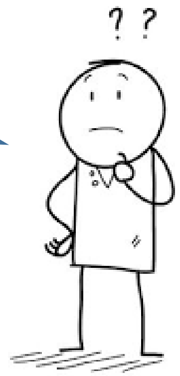
Secure messaging app	Number of messages per day
iMessage	40 billion
WhatsApp	65 billion
FB Messenger	1 billion

iMessage Telegram Signal WhatsApp

Viber Wire Skype Google Allo

Google Hangouts FB Messenger

Really?
How secure are they?



iMessage

Telegram

Signal

WhatsApp

Viber

Wire

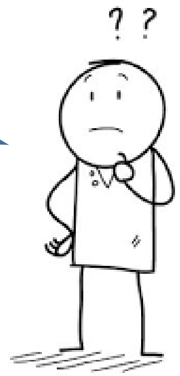
Skype

Google Allo

Google Hangouts

FB Messenger

Really?
How secure are they?



Not so easy to tell.
These apps include new cryptography.
This deserves analysis by cryptographers.
(The cryptography is often interesting in its own right ...)
Overall security involves a lot beyond the cryptography ...

iMessage | Telegram | Signal | WhatsApp

Viber | Wire | Skype | Google Allo

Google Hangouts | FB Messenger

Prior work has given definitions, schemes and analyses for **Ratcheting**

[CCDGS17], [BSJNS17], [JS18], [PR18], [ACD19], [JMM19], [DV19]

iMessage Telegram Signal WhatsApp

Viber Wire Skype Google Allo

Google Hangouts FB Messenger

This work is similarly motivated by iMessage

Estimated to have 1.3 billion active users in 2019

End-to-end encryption

End-to-end encryption protects your iMessage and FaceTime conversations across all your devices. With watchOS, iOS, and iPadOS, your messages are encrypted on your device so they can't be accessed without your passcode. iMessage and FaceTime are designed so that there's no way for Apple to read your messages when they're in transit between devices. You can choose to automatically delete your messages from your device after 30 days or a year or keep them on your device indefinitely.

iMessage

Sources

Apple Platform Security

Communities Contact

Security

Table of Contents ⊕

How iMessage sends and receives messages

Users start a new iMessage conversation by entering an address or name. If they enter a phone number or email address, the device contacts the Apple Identity Service (IDS) to retrieve the public keys and APNs addresses for all of the devices associated with the addressee. If the user enters a name, the device first uses the user's Contacts app to gather the phone numbers and email addresses associated with that name, then gets the public keys and APNs addresses from IDS.

The user's outgoing message is individually encrypted for each of the receiver's devices. The public encryption keys and signing keys of the receiving devices are retrieved from IDS. For each receiving device, the sending device generates a random 88-bit value and uses it as an HMAC-SHA256 key to construct a 40-bit value derived from the sender and receiver public key and the plaintext. The concatenation of the 88-bit and 40-bit values makes a 128-bit key, which encrypts the message with it using AES in CTR mode. The 40-bit value is used by the receiver side to verify the integrity of the decrypted plaintext. This per-message AES key is encrypted using RSA-OAEP to the public key of the receiving device. The combination of the encrypted message text and the encrypted message key is then hashed with SHA-1, and the hash is signed with ECDSA using the sending device's private signing key. Starting with iOS 13 and iPadOS 13.1, devices may use an ECIES encryption instead of RSA encryption.

The resulting messages, one for each receiving device, consist of the encrypted message text, the encrypted message key, and the sender's digital signature. They are then dispatched to the APNs for delivery. Metadata, such as the timestamp and APNs routing information, isn't encrypted. Communication with APNs is encrypted using a forward-secret TLS channel.

Protocol description at Apple iOS Security webpage

Some details are missing

Reverse engineering:

2012: OpenIM wiki https://wiki.imfreedom.org/wiki/IMessage
2013: Quarkslab https://blog.quarkslab.com/imessage-privacy.html
2016: [GGKMR16]

https://support.apple.com/guide/security/how-imessage-sends-and-receives-messages-sec70e68c949/1/web/1

iMsg1 : iOS 9 version

secret signing key of sender

public encryption key of receiver

message

$\underline{\text{iMsg1.Enc}(pk_r, sk_s, M)}$

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

ciphertext

iMsg1 : iOS 9 version

public encryption key of receiver

secret signing key of sender

message

iMsg1.Enc(pk_r, sk_s, M)

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

ciphertext

In 2016, Garman, Green, Kaptichuk, Miers, Rushanan [GGKMR16] gave chosen-ciphertext attacks on iMSg1 that succeeded in message recovery.

Dancing on the Lip of the Volcano: Chosen Ciphertext Attacks on Apple iMessage

Abstract

Apple's iMessage is one of the most widely-deployed end-to-end encrypted messaging protocols. Despite its broad deployment, the encryption protocols used by iMessage have never been subjected to rigorous cryptanalysis. In this paper, we conduct a thorough analysis of iMessage to determine the security of the protocol against a variety of attacks. Our analysis shows that iMessage has significant vulnerabilities that can be exploited by a sophisticated attacker. In particular, we outline a novel chosen ciphertext attack on Huffman compressed data, which allows *retrospective* decryption of some iMessage payloads in less than 2^{18} queries. The practical implication of these attacks is that any party who gains access to iMessage ciphertexts may potentially decrypt them remotely and after the fact. We additionally describe mitigations that will prevent these attacks on the protocol, without breaking backwards compatibility. Apple has deployed our mitigations in the latest iOS and OS X releases.

CVE-ID

CVE-2016-1788 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

Messages in Apple iOS before 9.3, OS X before 10.11.4, and watchOS before 2.2 does not properly implement a cryptographic protection mechanism, which allows remote attackers to read message attachments via vectors related to duplicate messages.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- APPLE:APPLE-SA-2016-03-21-1
- URL:http://lists.apple.com/archives/security-announce/2016/Mar/msg00000.html
- APPLE:APPLE-SA-2016-03-21-2
- URL:http://lists.apple.com/archives/security-announce/2016/Mar/msg00001.html
- APPLE:APPLE-SA-2016-03-21-5

iMsg2 : iOS 9.3 onwards version

secret signing key of sender

public encryption key of receiver

message

iMsg2.Enc(pk_r, sk_s, M)

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

ciphertext

iMsg2 : iOS 9.3 onwards version

secret signing key of sender

public encryption key of receiver

message

$\underline{\text{iMsg2.Enc}(pk_r, sk_s, M)}$

$L \leftarrow_\$ \{0,1\}^{88}$

$h \leftarrow \text{HMAC}(L, pk_s \| pk_r \| M)[1..40]$

$K \leftarrow L \| h$

$C_1 \leftarrow \text{AES-CTR.Enc}(K, M)$

$C_2 \leftarrow \text{RSA-OAEP.Enc}(pk_r, K)$

$H \leftarrow \text{SHA1}(C_1 \| C_2)$

$S \leftarrow \text{EC-DSA.Sign}(sk_s, H)$

Return $((C_1, C_2), S)$

ciphertext

Is iMsg2 secure?

Has the [GGKMR16] attack been (provably) thwarted?

iMsg2 : iOS 9.3 onwards version

Is iMsg2 secure?

Has the [GGKMR16] attack been (provably) thwarted?

public encryption key of receiver

secret signing key of sender

message

iMsg2.Enc(pk_r, sk_s, M)

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

ciphertext

Message *M* is being encrypted under a key *K* that is itself a function of *M*

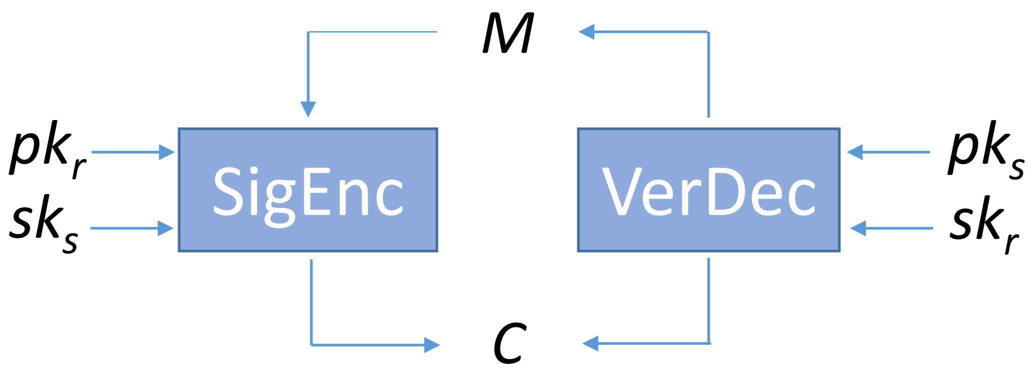
Intriguing technique:
Encryption under message-derived key

To answer this question meaningfully, we need to identify (and formalize) the GOAL underlying iMsg1 and iMsg2. ⟵

Is iMsg2 secure?
Has the [GGKMR16] attack been (provably) thwarted?

Question: What is the goal of iMsg1, iMsg2?

Our answer: Signcryption

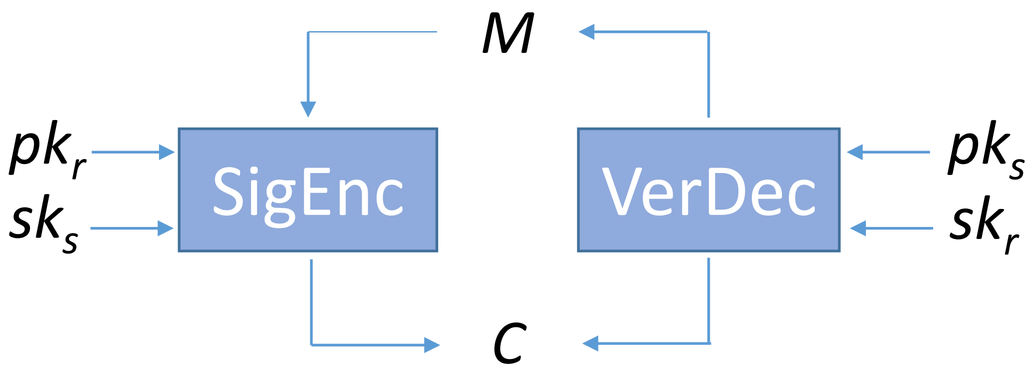


Zheng [Zh97]
An, Dodis, Rabin [ADR02]

Receiver has a public encryption key pk_r and secret decryption key sk_r
Sender has a secret signing key sk_s and public verification key pk_s
Signcryption aims to provide both privacy and authenticity of the message M

Question: What is the goal of iMsg1, iMsg2?

Our answer: Signcryption



Zheng [Zh97]
An, Dodis, Rabin [ADR02]

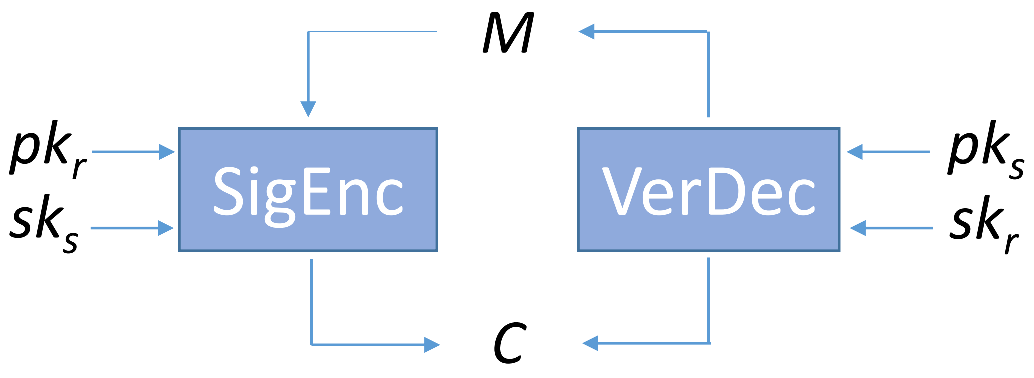
Receiver has a public encryption key pk_r and secret decryption key sk_r
Sender has a secret signing key sk_s and public verification key pk_s
Signcryption aims to provide both privacy and authenticity of the message M

[ADR02] define both

- Insider security: Adversary is a user with keys
- Outsider security: It is not

Question: What is the goal of iMsg1, iMsg2?

Our answer: Signcryption



Zheng [Zh97]
An, Dodis, Rabin [ADR02]

Receiver has a public encryption key pk_r and secret decryption key sk_r
Sender has a secret signing key sk_s and public verification key pk_s
Signcryption aims to provide both privacy and authenticity of the message M

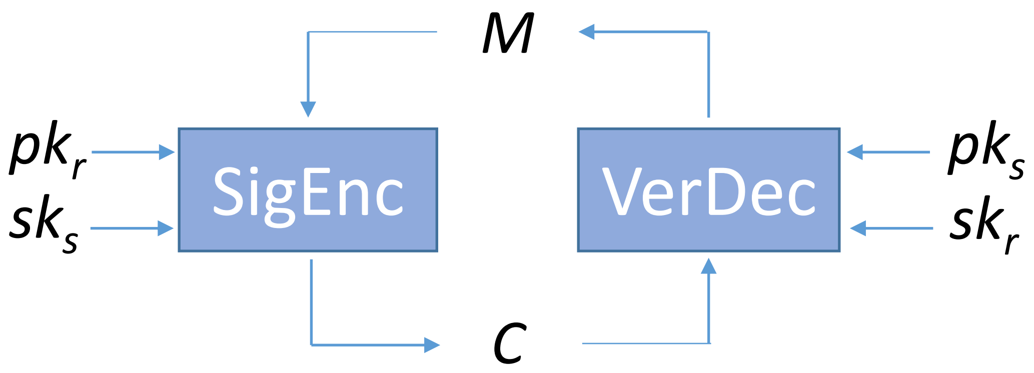
[ADR02] define both

- Insider security: Adversary is a user with keys
- Outsider security: It is not

Signcryption is the asymmetric (public-key setting) analogue of symmetric authenticated encryption.

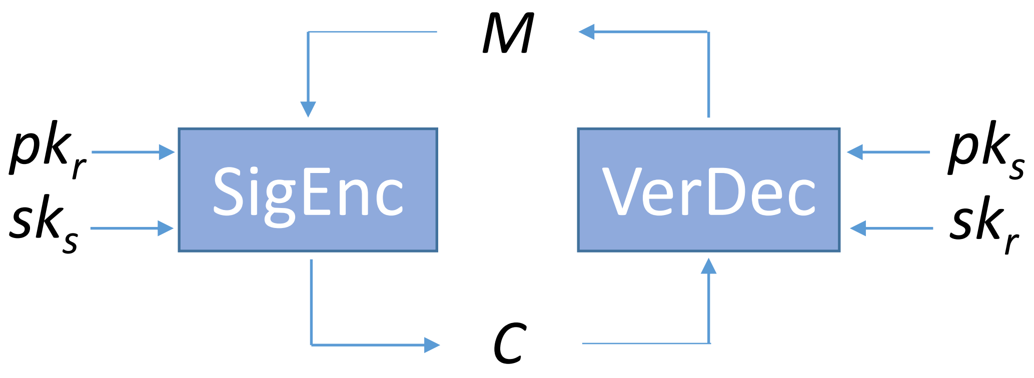
Question: What is the goal of iMsg1, iMsg2?

Our answer: Signcryption



Neither any Apple documents we found, nor [GGKMR16], explicitly identify Signcryption as the goal

Question: What is the goal of iMsg1, iMsg2?



Our answer: Signcryption

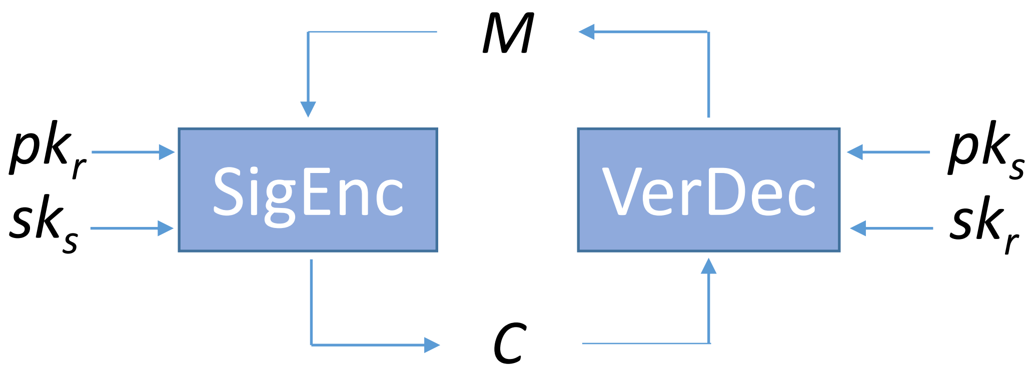
Neither any Apple documents we found, nor [GGKMR16], explicitly identify Signcryption as the goal

But identifying Signcryption as the goal yields some insight:

- iMsg1 is kind of Encrypt-then-Sign as per [ADR02]

Question: What is the goal of iMsg1, iMsg2?

Our answer: Signcryption



Neither any Apple documents we found, nor [GGKMR16], explicitly identify Signcryption as the goal

But identifying Signcryption as the goal yields some insight:

- iMsg1 is kind of Encrypt-then-Sign as per [ADR02]
- The [GGKMR16] attack on iMsg1 is a clever practical rendition of a generic attack on insider privacy from [ADR02]

So insider security should be the goal for iMsg2

Contributions in brief

Theoretical

Definitions and proofs

Give definitions for signcryption in Messaging setting

Introduce and **define encryption under message-derived keys** (EMDK)

Give general construction of signcryption from EMDK, encryption and signatures

Prove insider-security (priv, auth) of this general scheme

Practical

Analysis of security of iMsg2

Contributions in brief

Theoretical

Definitions and proofs

Give definitions for signcryption in Messaging setting

Introduce and **define encryption under message-derived keys** (EMDK)

Give general construction of signcryption from EMDK, encryption and signatures

Prove insider-security (priv, auth) of this general scheme

Instantiate →

Practical

Analysis of security of iMsg2

Obtain insider **security proof** for iMsg2

Derive **quantitative security** lower bounds

Contributions in brief

Theoretical

Definitions and proofs

Give definitions for signcryption in Messaging setting

Introduce and **define encryption under message-derived keys** (EMDK)

Give general construction of signcryption from EMDK, encryption and signatures

Prove insider-security (priv, auth) of this general scheme

Provide attacks matching the claimed lower bounds

Practical

Analysis of security of iMsg2

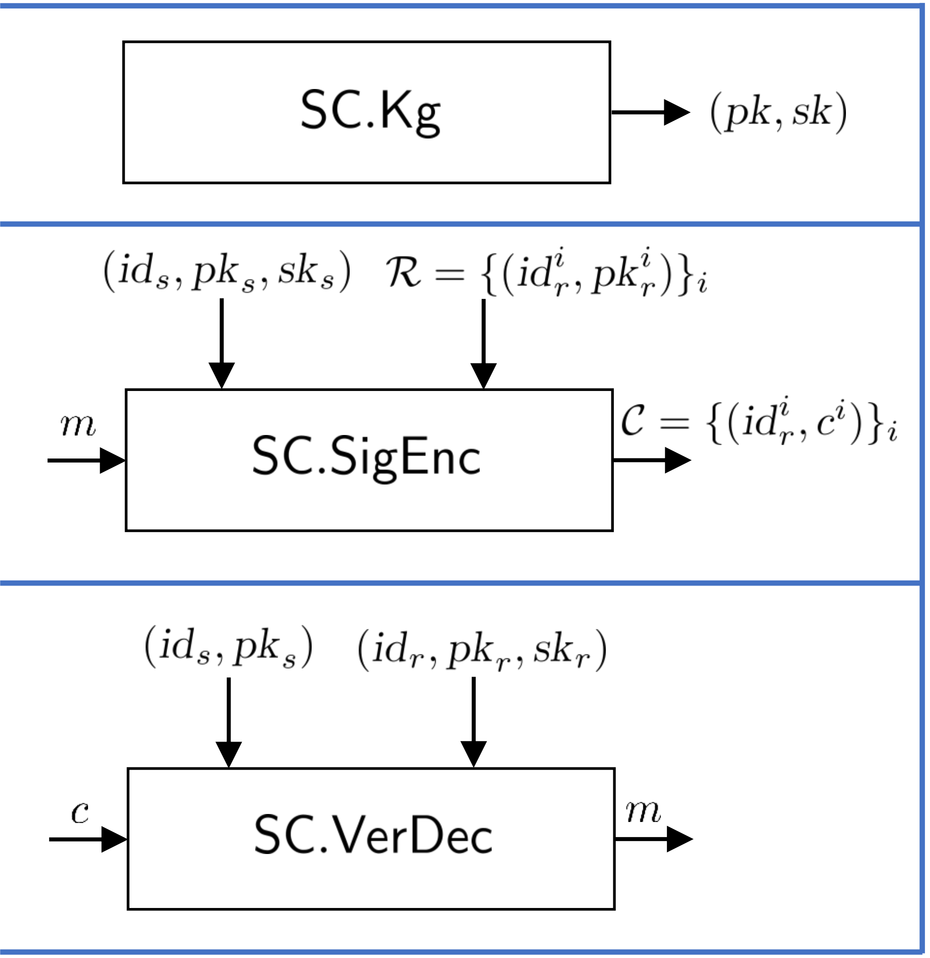
Instantiate →

Obtain insider **security proof** for iMsg2

Derive **quantitative security** lower bounds

Signcryption expanded

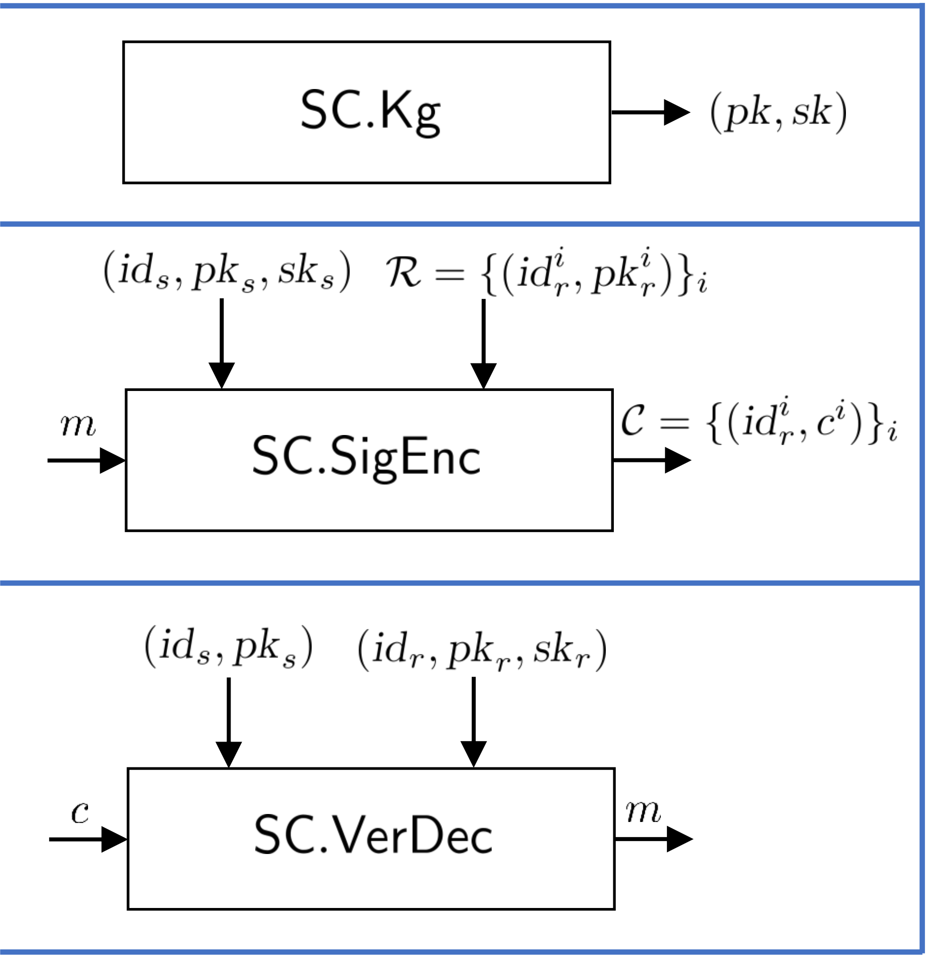
Syntax captures: Multiple recipients
Explicit user identities



Not displayed: associated data, public parameters.

Signcryption expanded

Syntax captures: Multiple recipients
Explicit user identities



Not displayed: associated data, public parameters.

Security definitions: *Start from the standard definitions by An, Dodis and Rabin.*

PRIV of SC

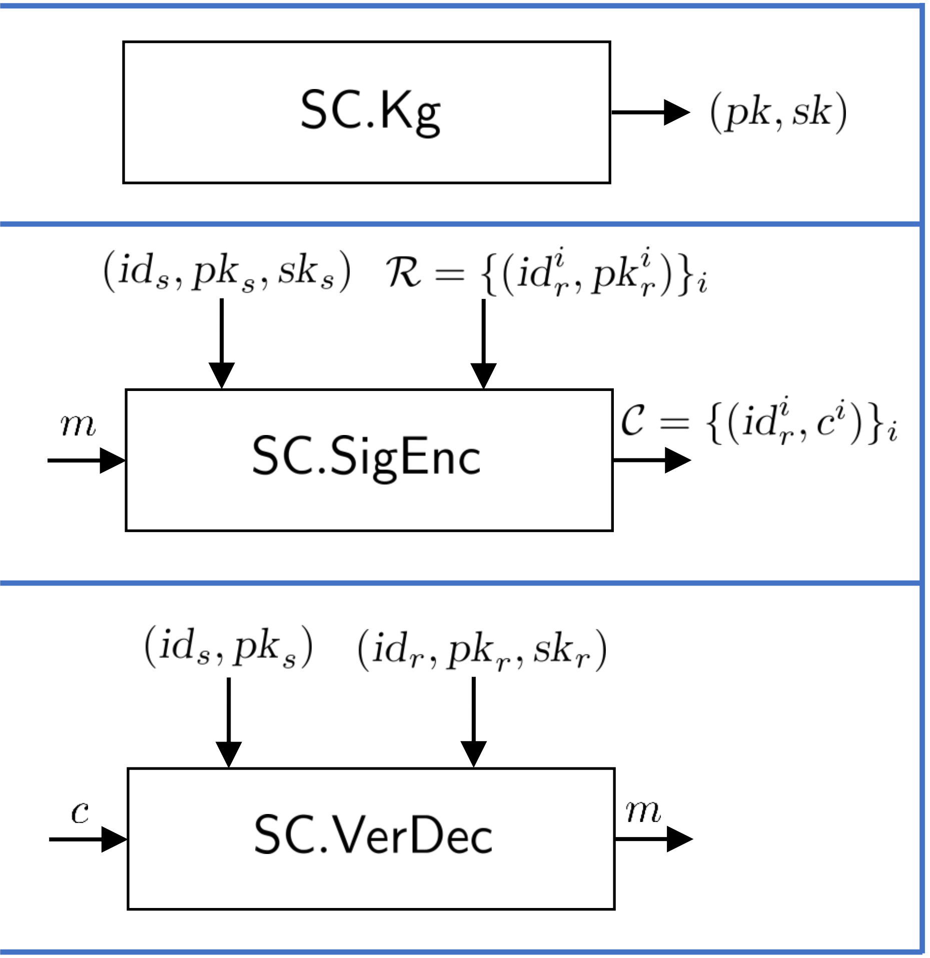
IND-CCA style definition.
Adversary has access to LR/VerDec oracles.
Need to guess the challenge bit used by the LR oracle.

AUTH of SC

UF-CMA/INT-CTXT style definition.
Adversary has access to SigEnc/VerDec oracles.
Need to forge a new ciphertext.

Signcryption expanded

Syntax captures: Multiple recipients
Explicit user identities



Not displayed: associated data, public parameters.

Security definitions: *Start from the standard definitions by An, Dodis and Rabin.*

PRIV of SC

IND-CCA style definition.
Adversary has access to LR/VerDec oracles.
Need to guess the challenge bit used by the LR oracle.

AUTH of SC

UF-CMA/INT-CTXT style definition.
Adversary has access to SigEnc/VerDec oracles.
Need to forge a new ciphertext.

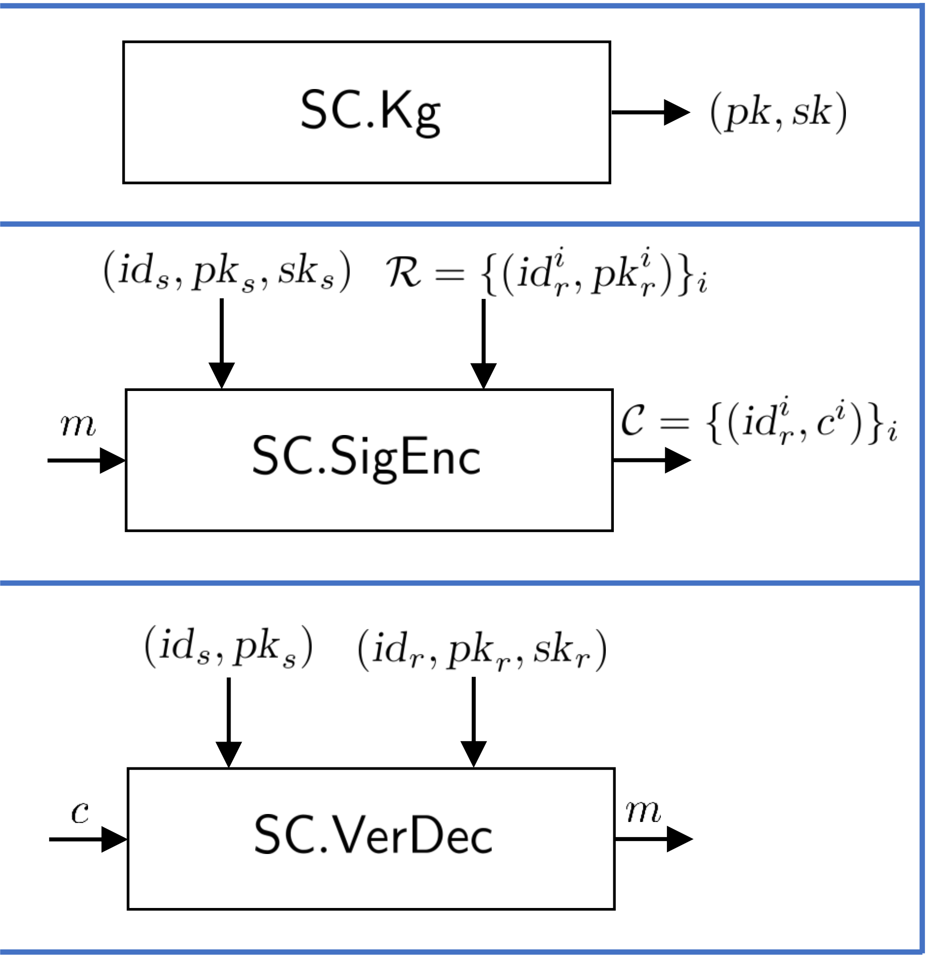
Definitional framework:

PRIV, AUTH separately
Unified PRIV+AUTH definition
Insider and outsider security
Parameterization by relations

Captures security in multi-user setting.
Adversary has full control over the network.
Secret key exposures are allowed.

Signcryption expanded

Syntax captures: Multiple recipients
Explicit user identities



Not displayed: associated data, public parameters.

Security definitions: *Start from the standard definitions by An, Dodis and Rabin.*

PRIV of SC

IND-CCA style definition.
Adversary has access to LR/VerDec oracles.
Need to guess the challenge bit used by the LR oracle.

AUTH of SC

UF-CMA/INT-CTXT style definition.
Adversary has access to SigEnc/VerDec oracles.
Need to forge a new ciphertext.

Definitional framework:

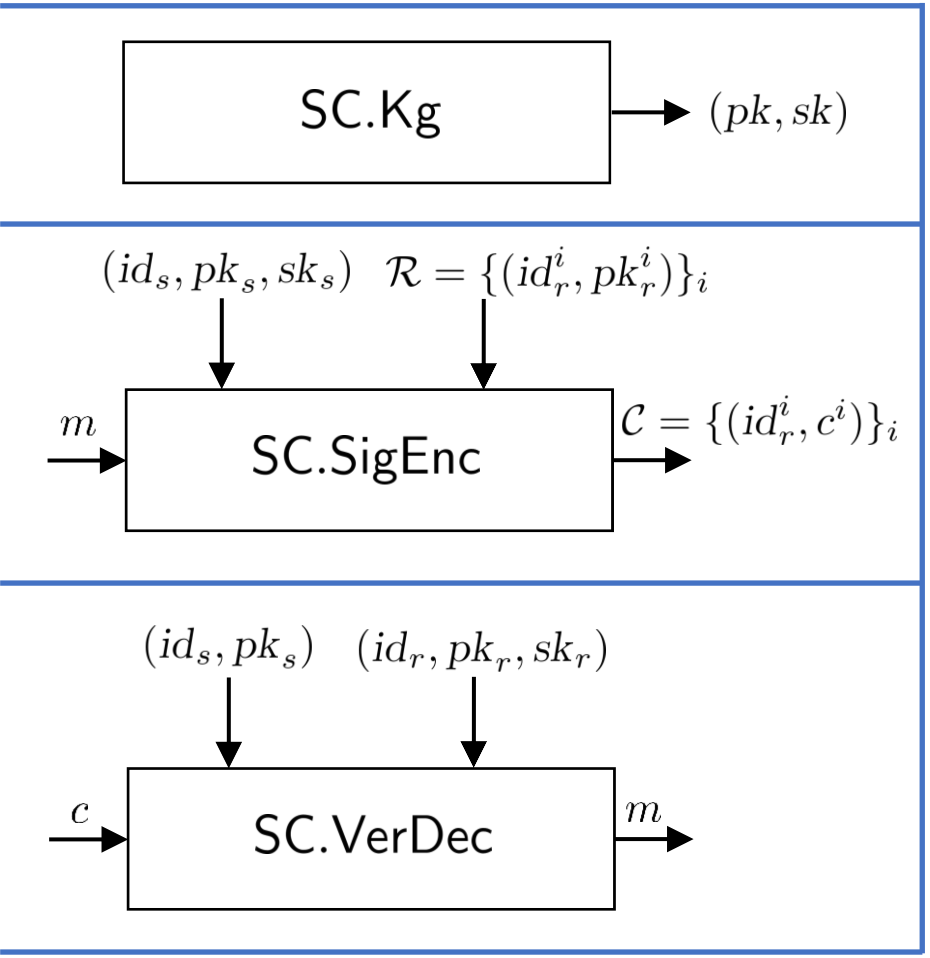
PRIV, AUTH separately
Unified PRIV+AUTH definition
Insider and outsider security
Parameterization by relations

Captures security in multi-user setting.
Adversary has full control over the network.
Secret key exposures are allowed.

Captured by considering different classes of adversaries.

Signcryption expanded

Syntax captures: Multiple recipients
Explicit user identities



Not displayed: associated data, public parameters.

Security definitions: *Start from the standard definitions by An, Dodis and Rabin.*

PRIV of SC

IND-CCA style definition.
Adversary has access to LR/VerDec oracles.
Need to guess the challenge bit used by the LR oracle.

AUTH of SC

UF-CMA/INT-CTXT style definition.
Adversary has access to SigEnc/VerDec oracles.
Need to forge a new ciphertext.

Definitional framework:

PRIV, AUTH separately
Unified PRIV+AUTH definition
Insider and outsider security
Parameterization by relations

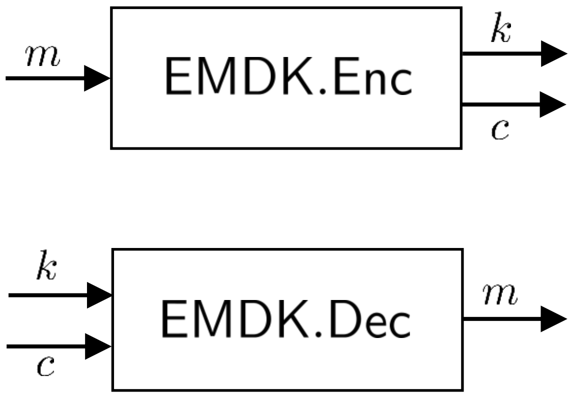
Captures security in multi-user setting.
Adversary has full control over the network.
Secret key exposures are allowed.

Captured by considering different classes of adversaries.

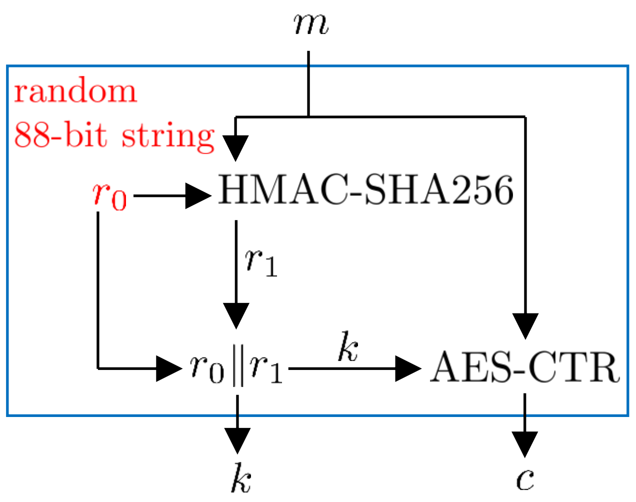
Use different relations to capture:
- standard unforgeability,
- strong unforgeability,
- RCCA/IND-gCCA2 security,
- …

Encryption under Message Derived Keys

Encryption under Message Derived Keys



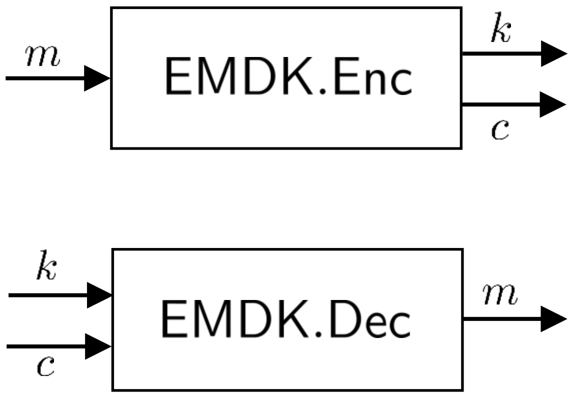
iMessage-based EMDK scheme



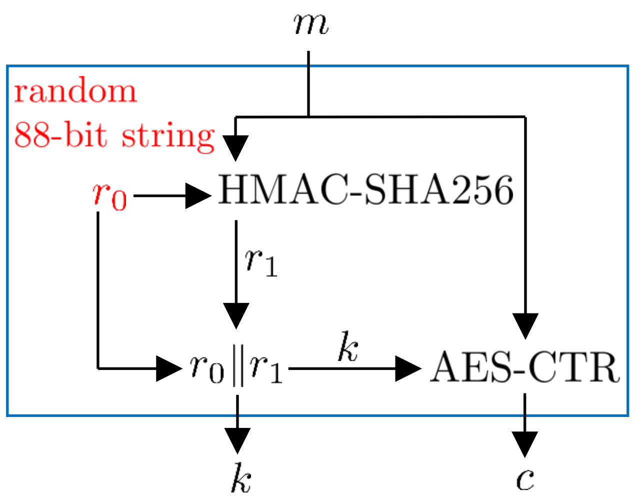
HMAC-SHA256 output truncated to 40 bits
AES-CTR uses 128-bit key

Our goal: analyze the security of the iMessage-based EMDK scheme.

Encryption under Message Derived Keys

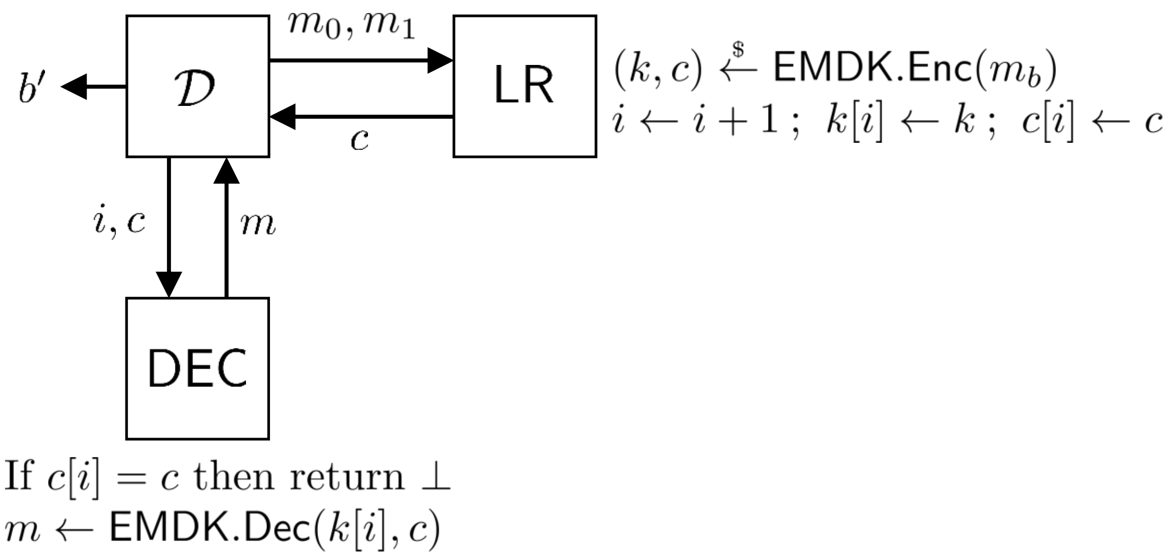


iMessage-based EMDK scheme



HMAC-SHA256 output truncated to 40 bits
AES-CTR uses 128-bit key

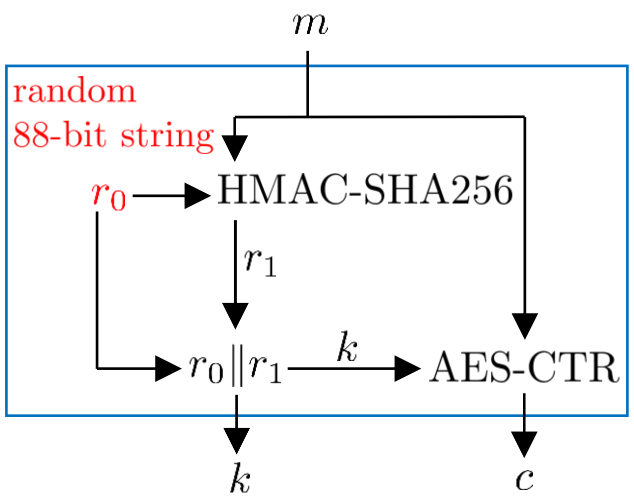
Authenticated Encryption security of EMDK (AE)



Our goal: analyze the security of the iMessage-based EMDK scheme.

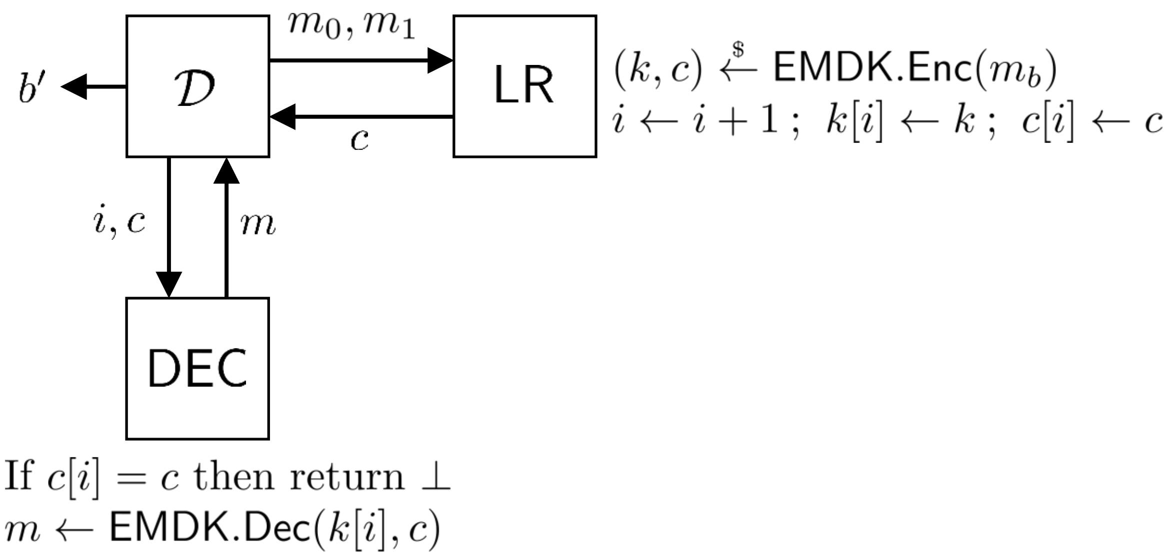
Encryption under Message Derived Keys

iMessage-based EMDK scheme

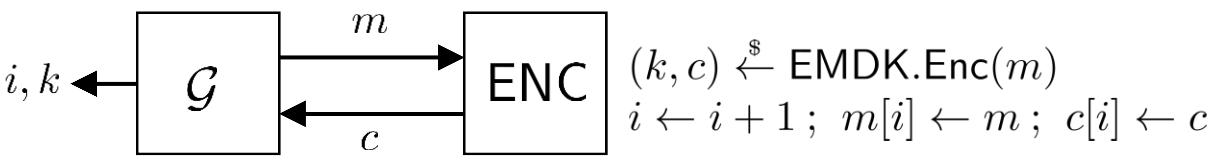


HMAC-SHA256 output truncated to 40 bits
AES-CTR uses 128-bit key

Authenticated Encryption security of EMDK (AE)



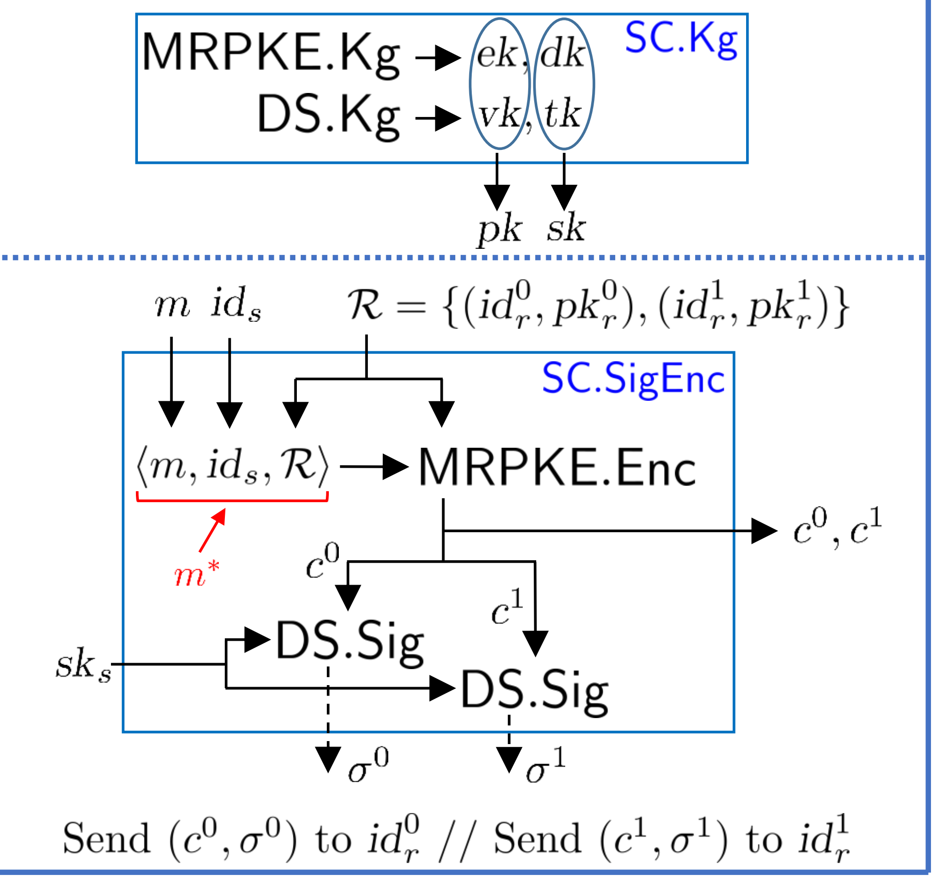
Robustness of EMDK (ROB)



$\mathcal{G}$ wins if $\text{EMDK.Dec}(k, c[i]) \neq m[i]$

Our goal: analyze the security of the iMessage-based EMDK scheme.

SignCryption

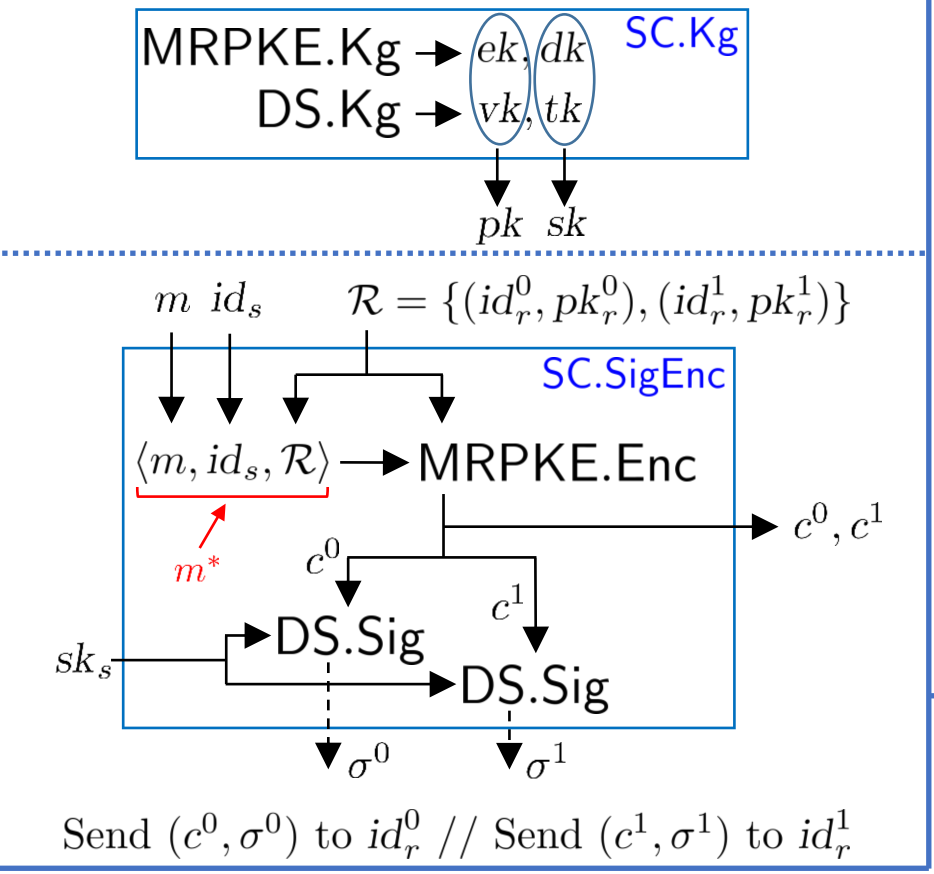


DS: ECDSA on NIST P-256 curve

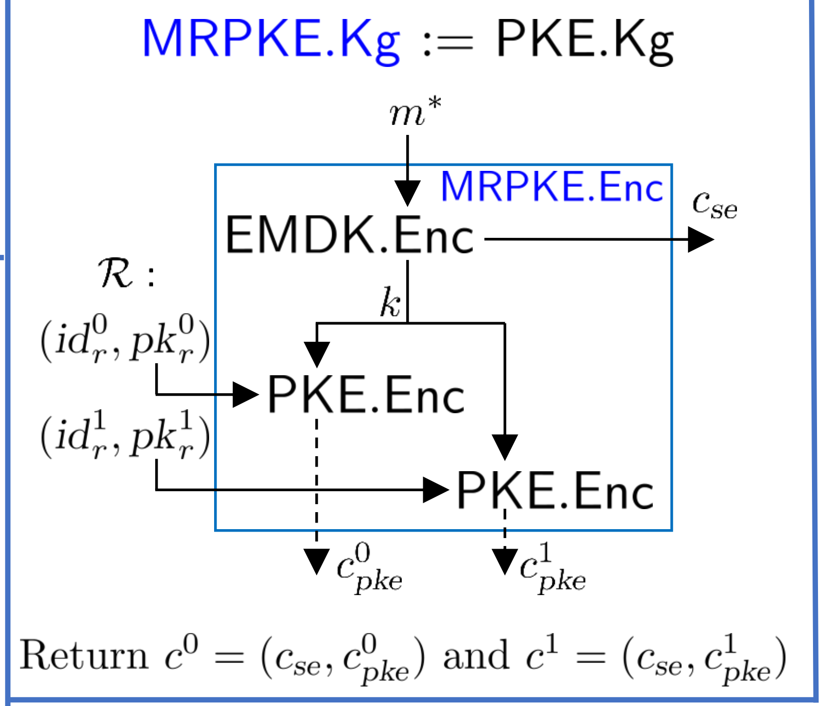
AUTH of SC
- → UF of DS
- → ROB of MRPKE

PRIV of SC
- → IND-CCA of MRPKE

SignCryption



Multi-Recipient Public-Key Encryption



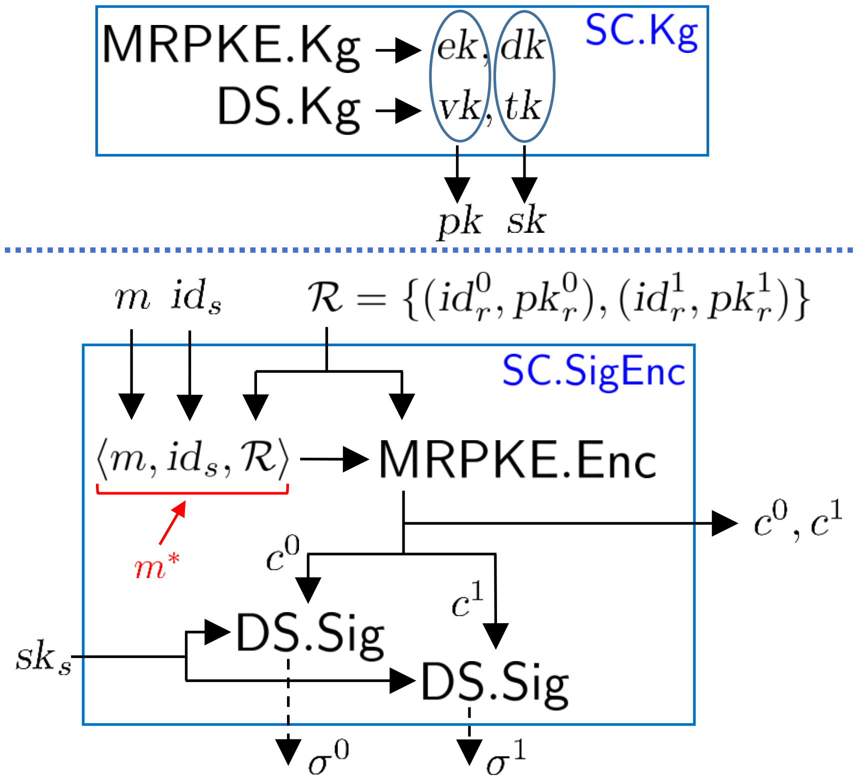
DS: ECDSA on NIST P-256 curve

PKE: RSA-OAEP with 1280-bit key

AUTH of SC
- → UF of DS
- → ROB of MRPKE
  - $\vee$ → ROB of PKE
  - → ROB of EMDK

PRIV of SC
- → IND-CCA of MRPKE
  - → IND-CCA of PKE
  - → AE of EMDK

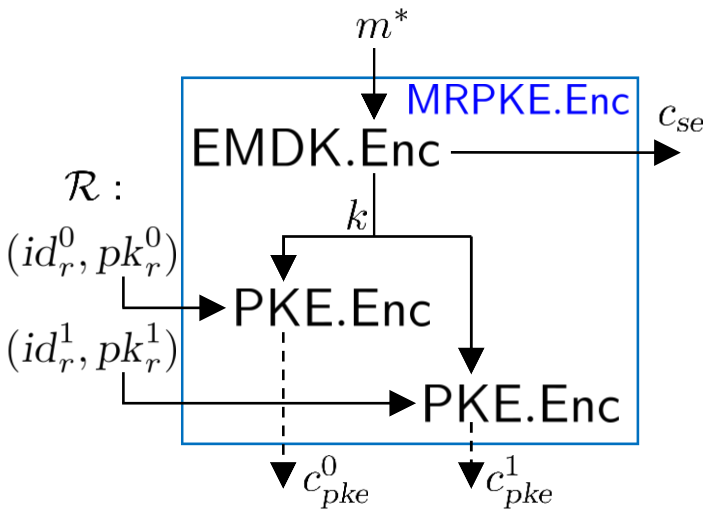
SignCryption



Send (c^0, σ^0) to id_r^0 // Send (c^1, σ^1) to id_r^1

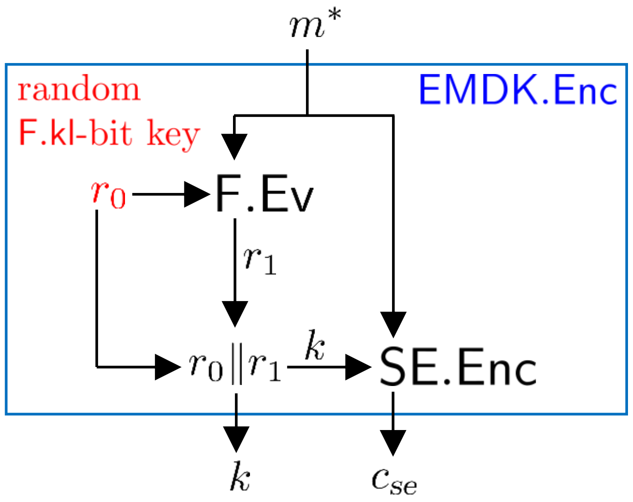
Multi-Recipient Public-Key Encryption

MRPKE.Kg := PKE.Kg



Return $c^0 = (c_{se}, c^0_{pke})$ and $c^1 = (c_{se}, c^1_{pke})$

Encryption under Message Derived Keys



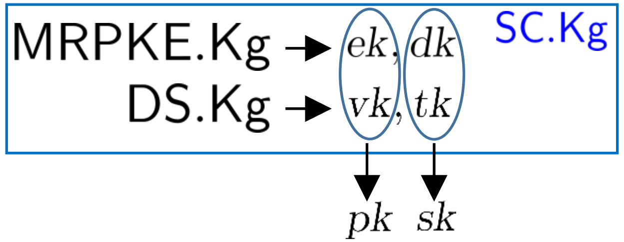
DS: ECDSA on NIST P-256 curve
PKE: RSA-OAEP with 1280-bit key
SE: AES-CTR with 128-bit key
F: HMAC-SHA256 (F.kl $= 88$, F.ol $= 40$)

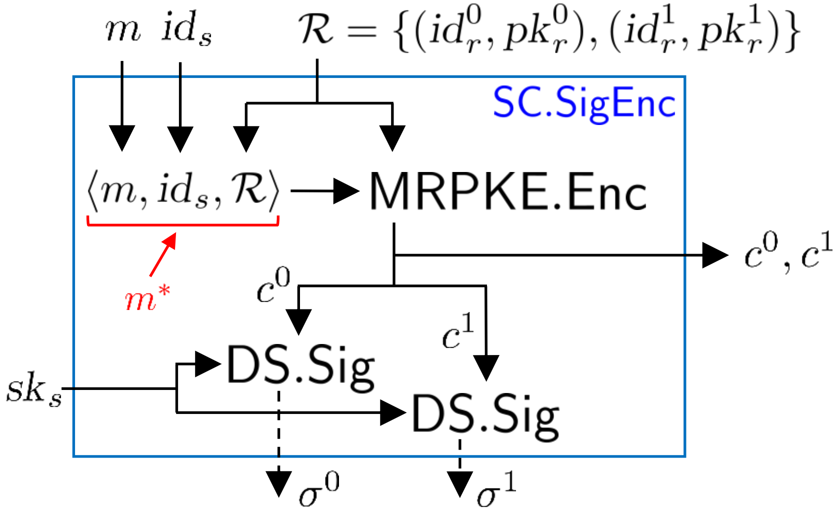
AUTH of SC
- UF of DS
- ROB of MRPKE
  - ROB of PKE
  - ∨ ROB of EMDK
    - ROM WROB of SE

PRIV of SC
- IND-CCA of MRPKE
  - IND-CCA of PKE
  - AE of EMDK
    - TCR of F
    - IND of EMDK
      - ROM OT-IND of SE
      - PKR of SE

Partial Key Recovery

SignCryption

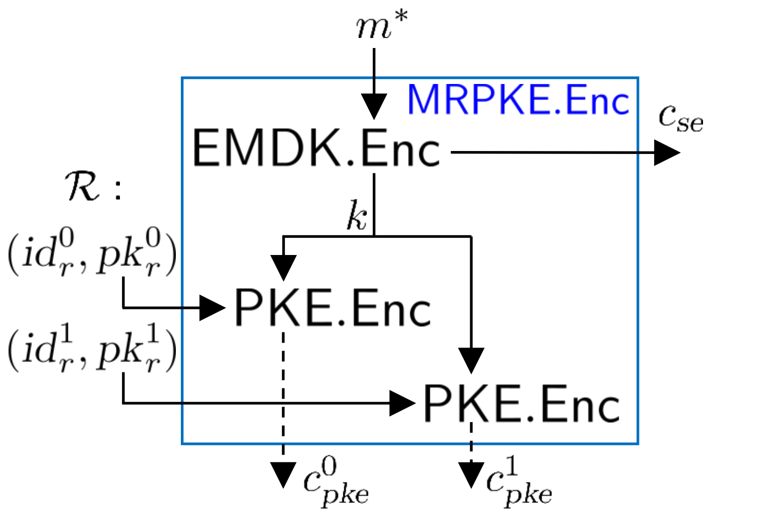




Send (c^0, σ^0) to id_r^0 // Send (c^1, σ^1) to id_r^1

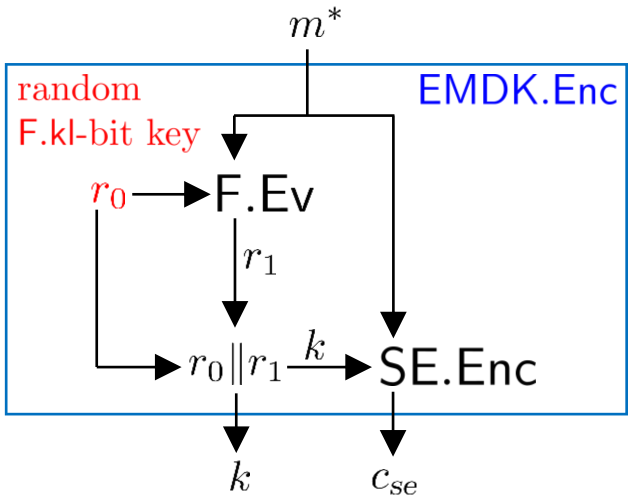
Multi-Recipient Public-Key Encryption

MRPKE.Kg := PKE.Kg

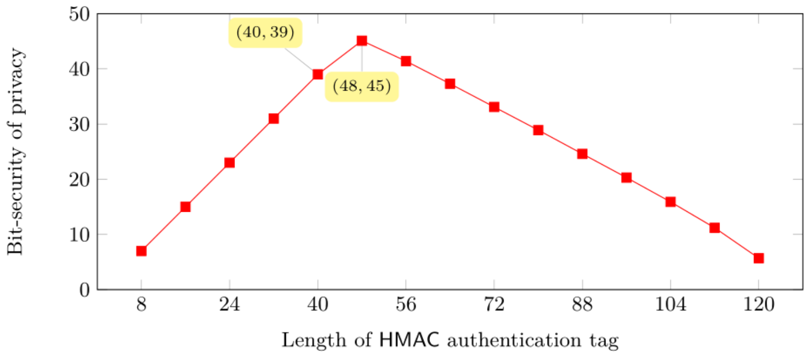


Return $c^0 = (c_{se}, c_{pke}^0)$ and $c^1 = (c_{se}, c_{pke}^1)$

Encryption under Message Derived Keys



F random oracle; SE ideal cipher $\Longrightarrow$
AUTH bit-security of SC is *at least* 71 bits
PRIV bit-security of SC is *at least* 39 bits



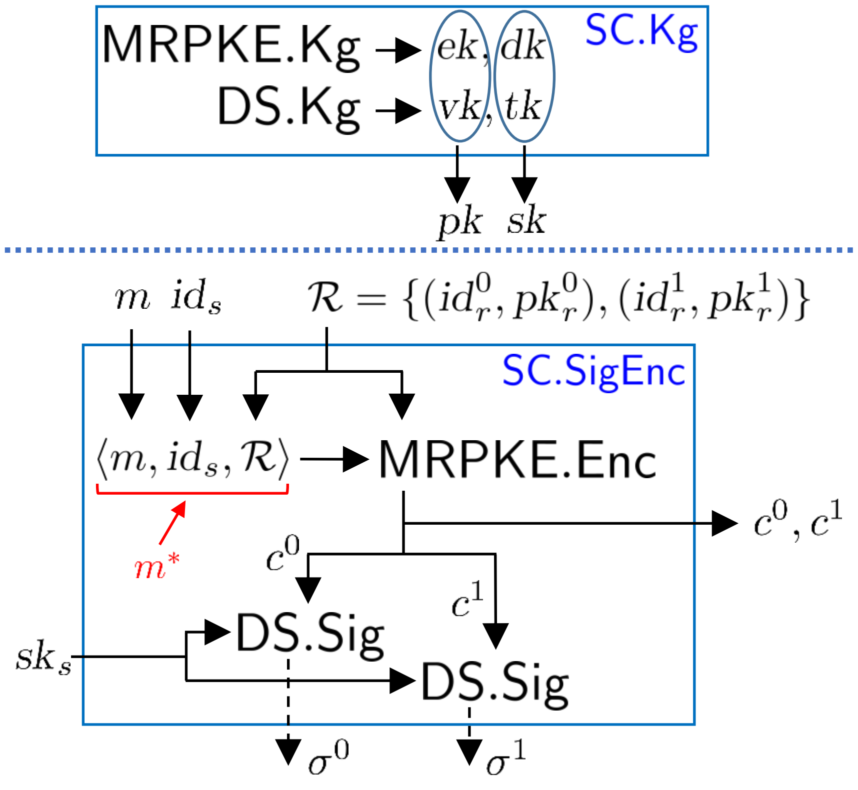
DS: ECDSA on NIST P-256 curve
PKE: RSA-OAEP with 1280-bit key
SE: AES-CTR with 128-bit key
F: HMAC-SHA256 (F.kl = 88, F.ol = 40)

AUTH of SC
- UF of DS
- ROB of MRPKE
  - ∨ ROB of PKE
  - ROB of EMDK
    - ROM WROB of SE

PRIV of SC
- IND-CCA of MRPKE
  - IND-CCA of PKE
  - AE of EMDK
    - TCR of F
    - IND of EMDK
      - ROM OT-IND of SE
      - PKR of SE

Partial Key Recovery

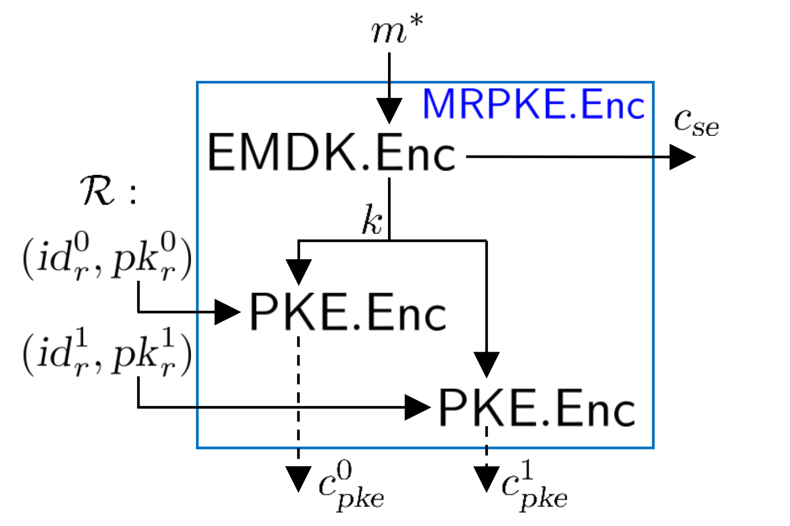
SignCryption



Send (c^0, σ^0) to id_r^0 // Send (c^1, σ^1) to id_r^1

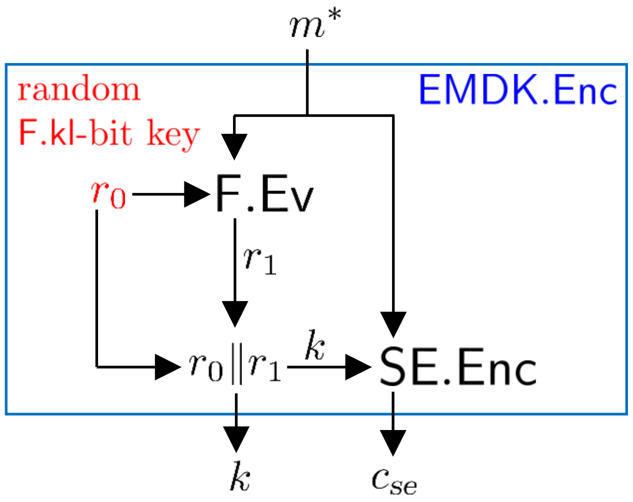
Multi-Recipient Public-Key Encryption

MRPKE.Kg := PKE.Kg

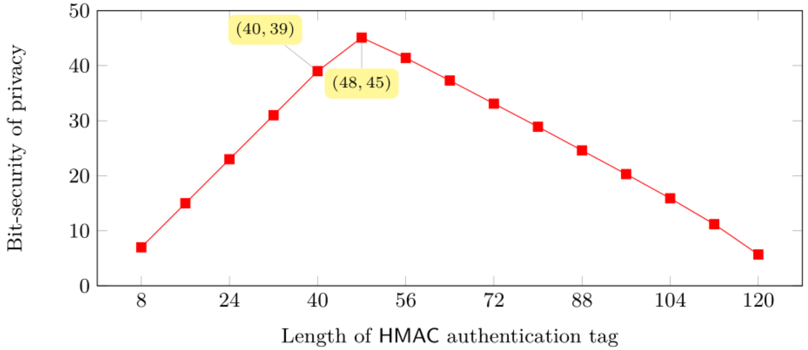


Return $c^0 = (c_{se}, c_{pke}^0)$ and $c^1 = (c_{se}, c_{pke}^1)$

Encryption under Message Derived Keys



F random oracle; SE ideal cipher $\Longrightarrow$
AUTH bit-security of SC is *at least* 71 bits
PRIV bit-security of SC is *at least* 39 bits



DS: ECDSA on NIST P-256 curve
PKE: RSA-OAEP with 1280-bit key
SE: AES-CTR with 128-bit key
F: HMAC-SHA256 (F.kl = 88, F.ol = 40)

AUTH of SC
- UF of DS
- ROB of MRPKE
  - ROB of PKE
  - ∨ ROB of EMDK
    - ROM WROB of SE

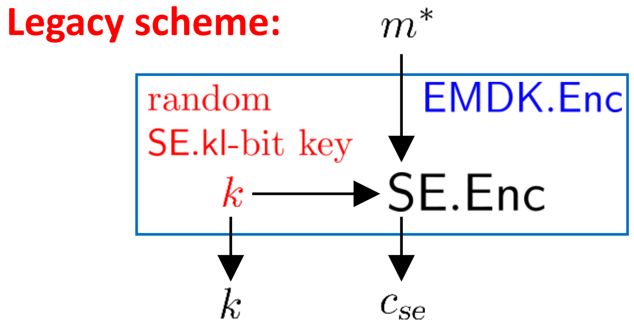
PRIV of SC
- IND-CCA of MRPKE
  - IND-CCA of PKE
  - AE of EMDK
    - TCR of F
    - IND of EMDK
      - ROM OT-IND of SE
      - PKR of SE

Partial Key Recovery

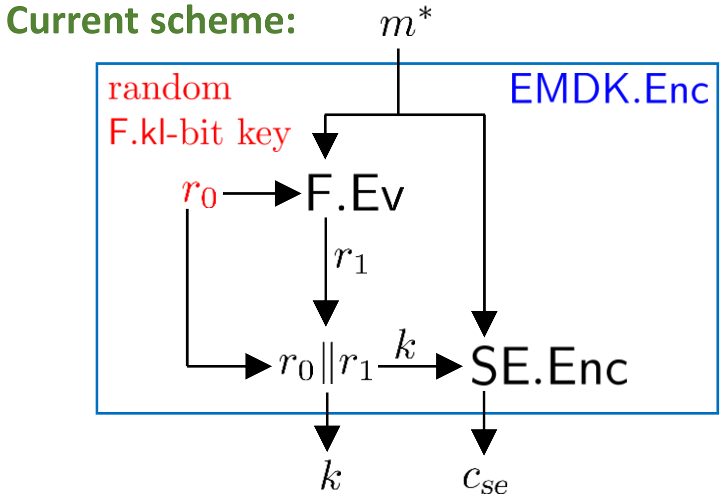
Attacks:

Guess m^{**} such that $r_1 = \mathsf{F.Ev}(r_0, m^{**})$.
Birthday attack on $r_0 \in \{0,1\}^{\mathsf{F.kl}}$.
Exhaustive key search over $r_0 \in \{0,1\}^{\mathsf{F.kl}}$.

Encryption under Message Derived Keys



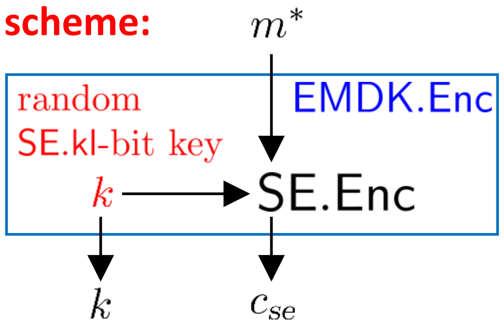
Used in iMSg1 scheme, attacked by [GGKMR16]



Used in iMSg2 scheme, starting from iOS 9.3

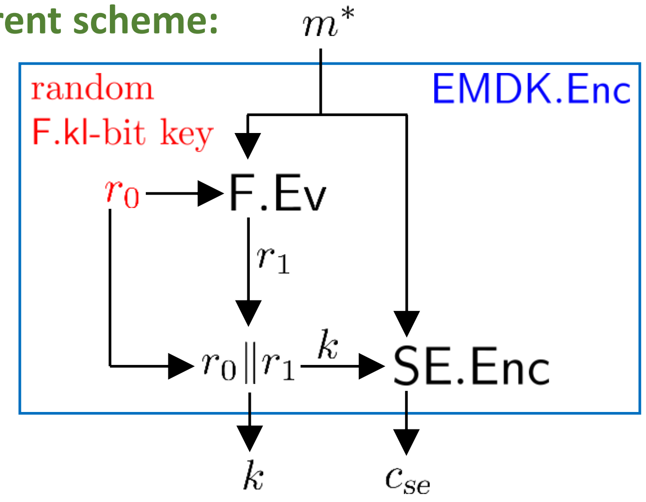
Encryption under Message Derived Keys

Legacy scheme:



Used in iMSg1 scheme, attacked by [GGKMR16]

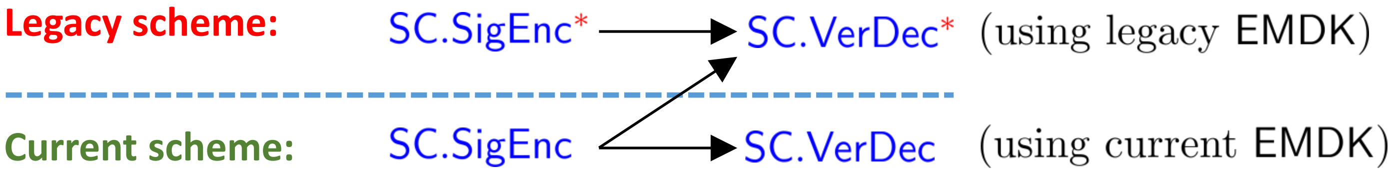
Current scheme:



Used in iMSg2 scheme, starting from iOS 9.3

Goal: backward compatibility. *confirmed by Apple*

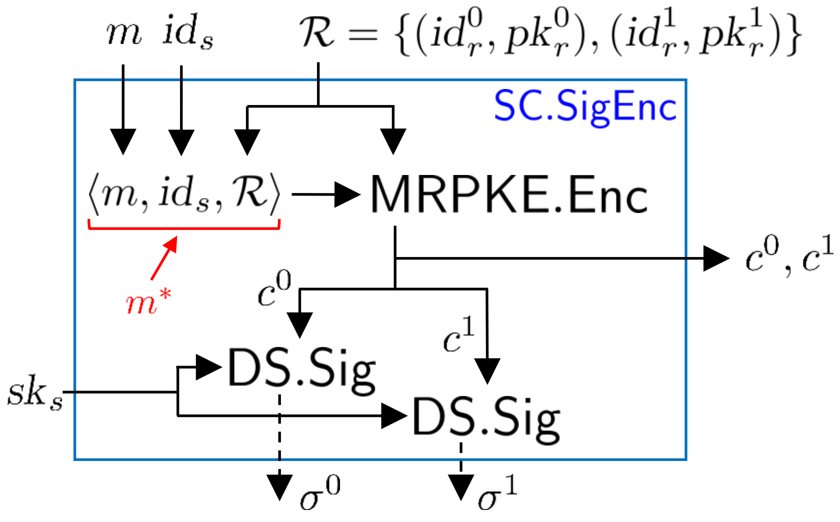
Decryption correctness:



Communication with Apple

We confirmed that our theoretical construction captures the design of iMessage, with a minor difference

Our signcryption scheme



iMessage implementation

Encrypted payload contains a uniformly random seed r.

$$\underbrace{\langle m, id_s, \mathcal{R}, r \rangle}_{m^*}$$

Makes two attacks harder

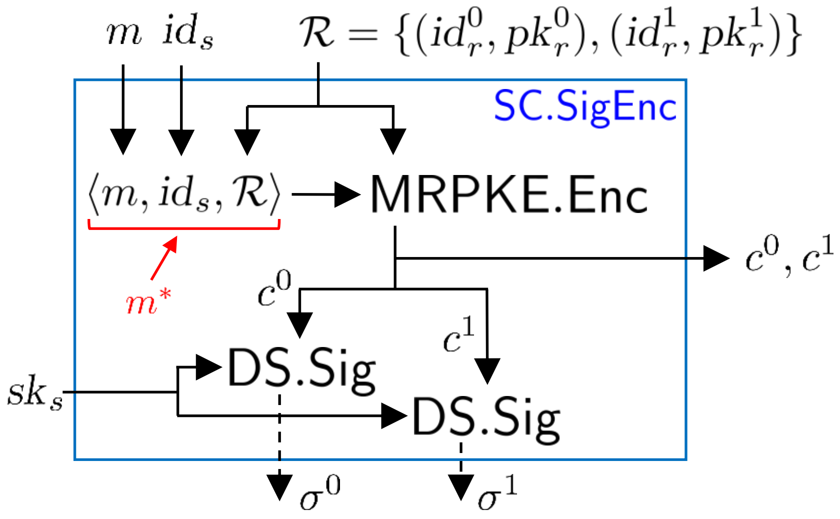
but

concrete security bounds remain the same.

Communication with Apple

We confirmed that our theoretical construction captures the design of iMessage, with a minor difference

Our signcryption scheme



iMessage implementation

Encrypted payload contains a uniformly random seed r.

$\langle m, id_s, \mathcal{R}, r \rangle$

m^*

Makes two attacks harder

but

concrete security bounds remain the same.

Practical security of iMessage

	Theoretical	Practical
GGKMR16	O(1)	2^{18} queries, 35 hours
BS16	2^{39}	?

According to GGKMR16, iOS 9.3 implemented additional implementation-level attack mitigations.

Thank you!

https://eprint.iacr.org/2020/224.pdf