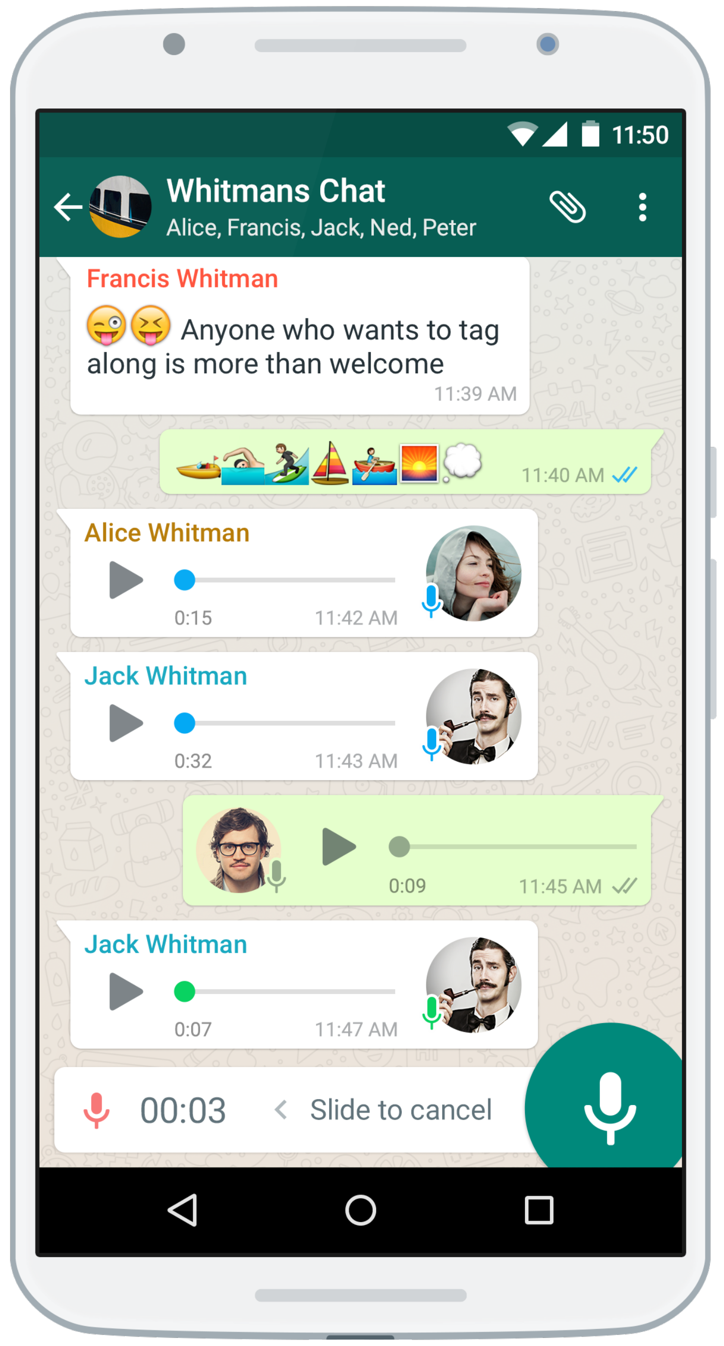
Optimal Channel Security Against Fine-Grained State Compromise: The Safety of Messaging

Joseph Jaeger
Igors Stepanovs

Alice and Bob want E2E secure communication

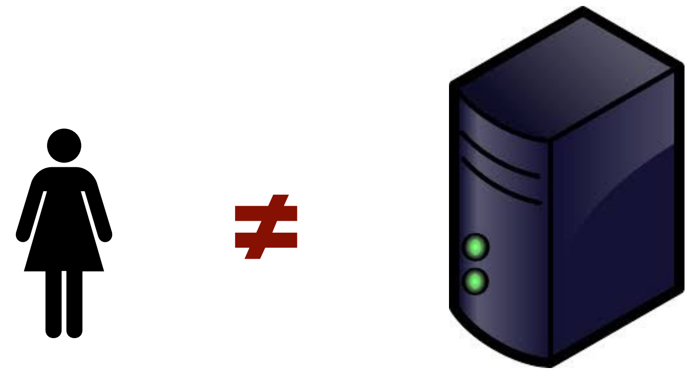
Plenty of Theory …

Symmetric encryption
Asymmetric encryption
Session key exchange
Signatures
…

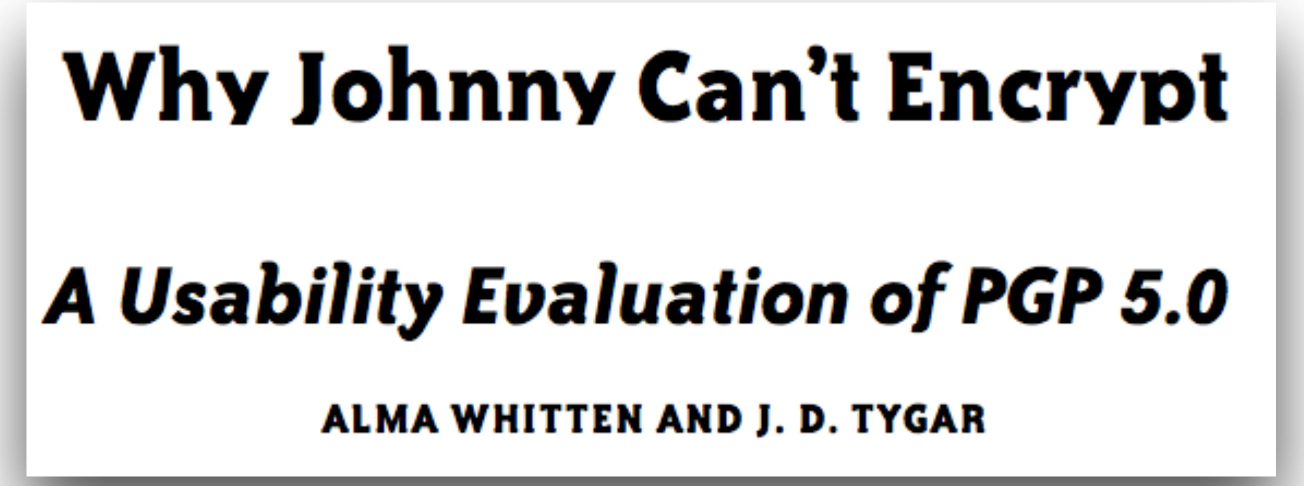


But what about E2E Tools?

✗ TLS is for web servers!



✗ PGP is a pain!


Why Johnny Can't Encrypt

A Usability Evaluation of PGP 5.0

ALMA WHITTEN AND J. D. TYGAR

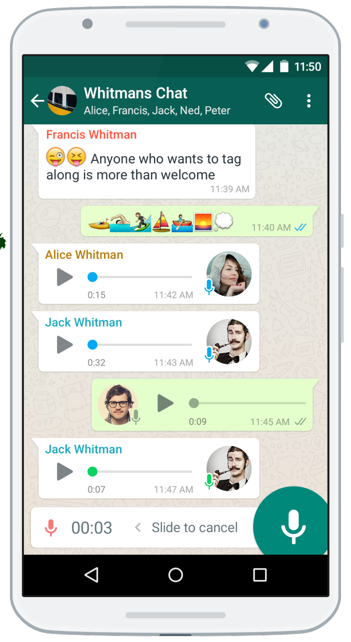
✓ Messaging Apps

Emerging as most convenient & usable.

E2E Messaging Apps

Apps Targeting E2E Security:

- Signal
- WhatsApp
- Skype (Private Conversations)
- Facebook Messenger (Secret Conversations)
- iMessage
- Telegram (Secret Chats)
- Viber
- Many more…



Whatsapp alone encrypts ~55 billion messages/day.

~700 million **iPhones** in use.

Understanding their security is important!

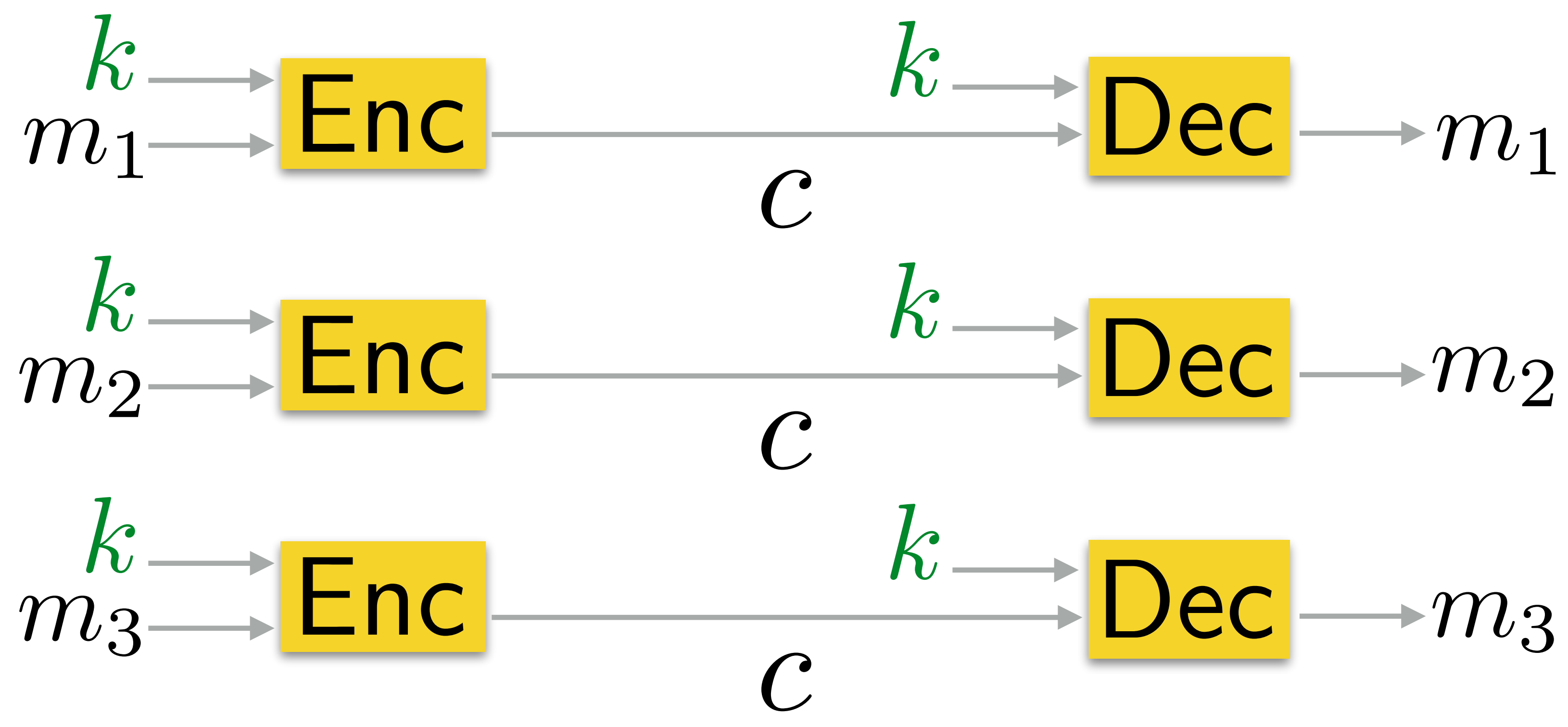
These are all based on Open Whisper System's **Double Ratchet Algorithm**. (i.e. the techniques of Signal)

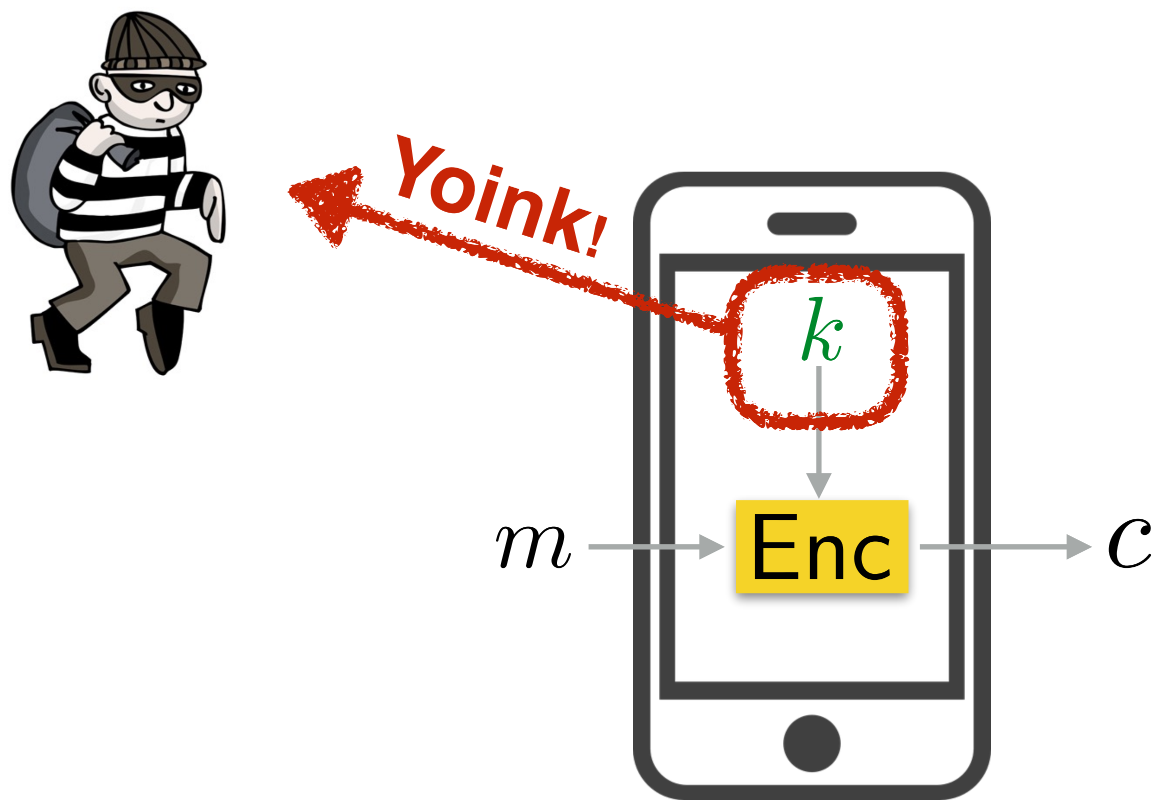
We aim to better understand its goal:

Security against state compromise

Traditional Encryption

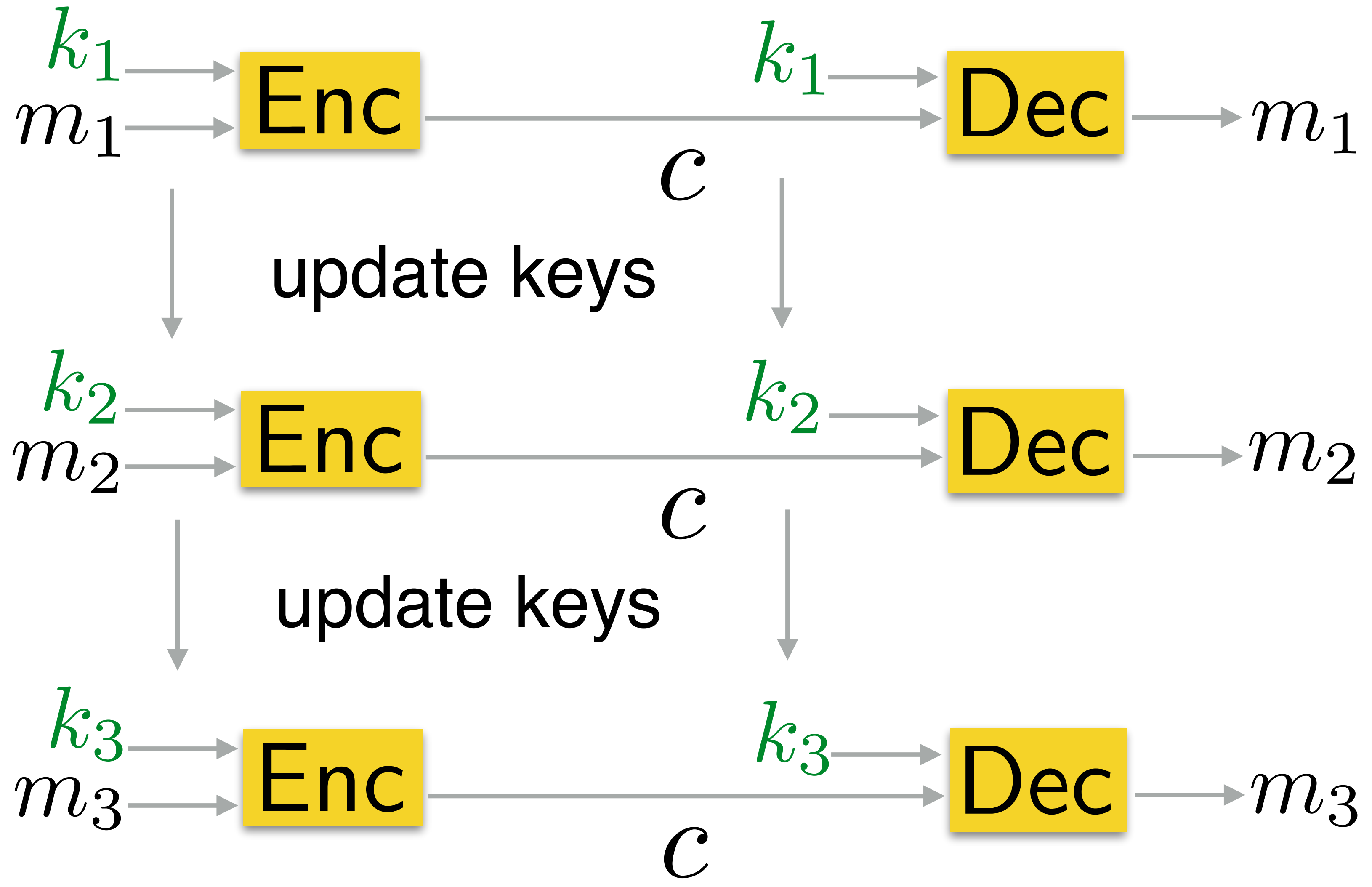
$k \in \vec{\mathbf{K}}_A \cap \vec{\mathbf{K}}_B$





Key Updating Encryption

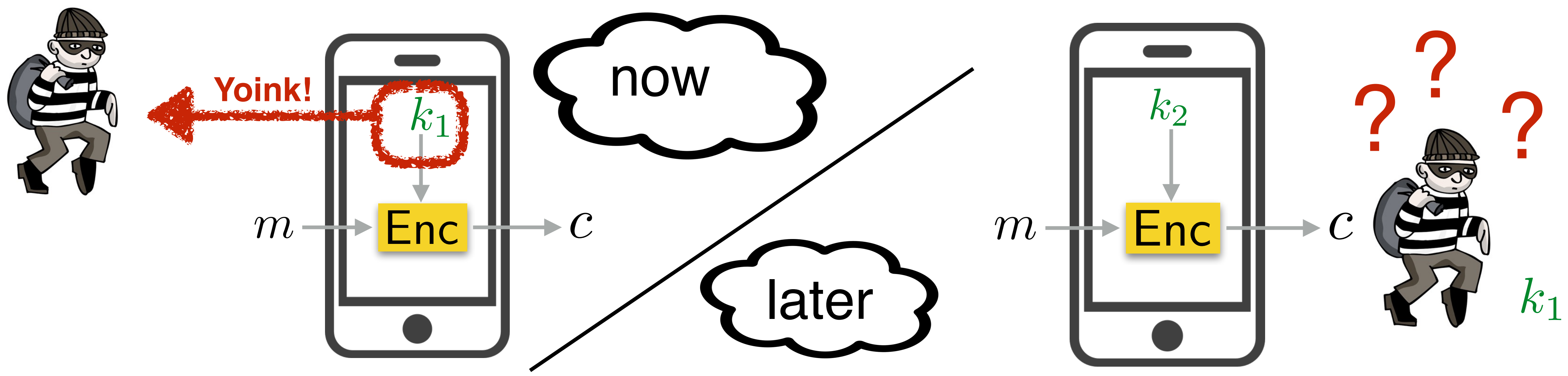
$k \in \vec{\mathbf{K}}_A \cap \vec{\mathbf{K}}_B$



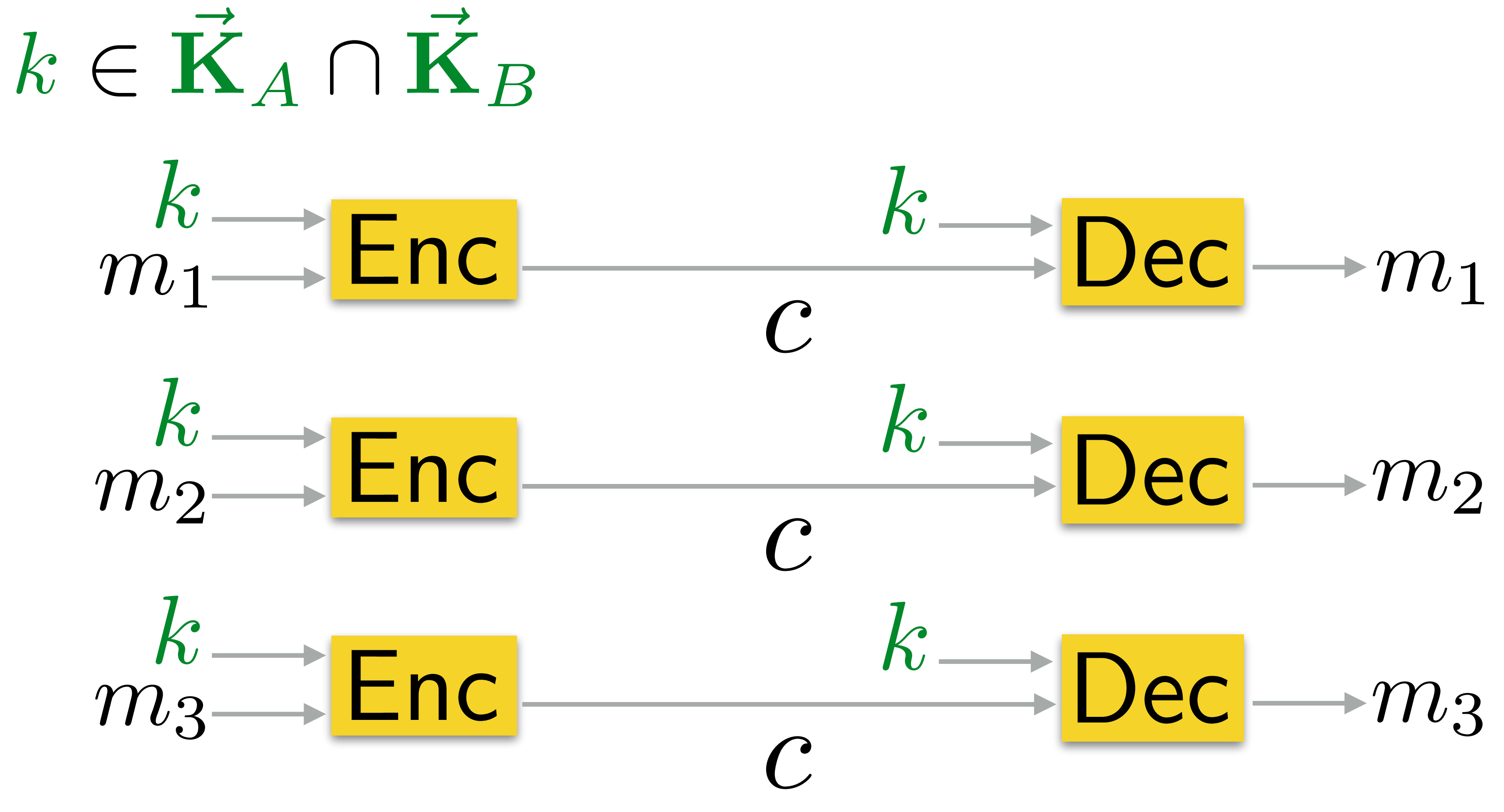
Addressed in practice:
Messaging app designers in practice are trying to protect against this threat by updating the secret key using ratcheting.

How does attacker access secrets?

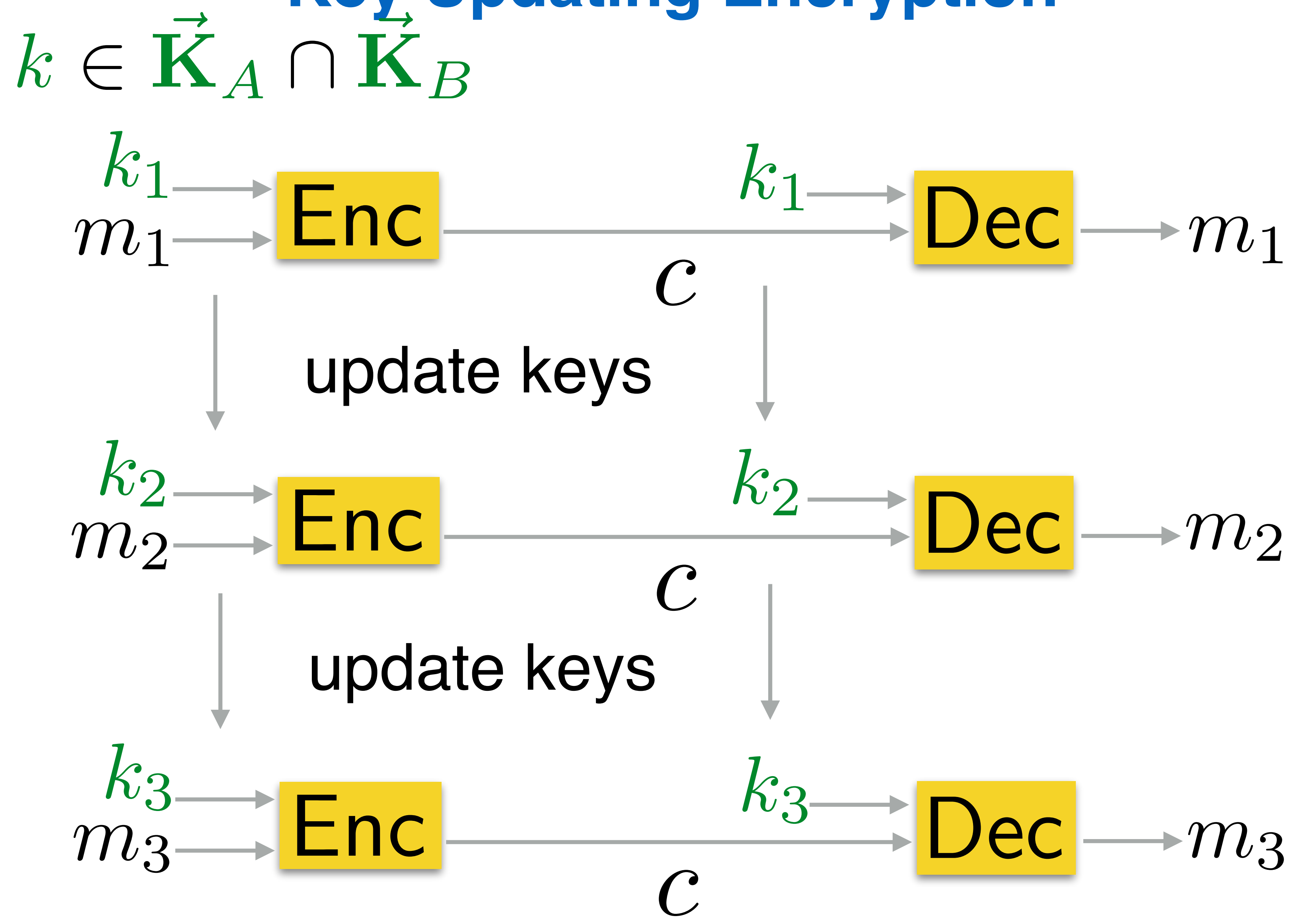
- Steals physical device
- Malware
- Border searches
- Unpatched vulnerabilities
- …



Traditional Encryption



Key Updating Encryption



Informal goals:

- **Forward security:** prior keys or communications remain secure
- **Backward security:** future keys or communications remain secure

Exactly what threat these goals prevent in practice needs careful consideration …

- **Less useful when** threat is persistent malware than can directly exfiltrate messages.
- **More useful when** users delete old messages, malware exfiltrates keys instead of messages, malware's presence limited by software security.

Forward and backward security are of particular interest for secure messaging because conversations can be very long lived … a chat session can stay open for a year …

Prior Work

What security goal does ratcheting achieve?

Formalize it

Show that ratcheting achieves it.

A Formal Security Analysis of the Signal Messaging Protocol

Katriel Cohn-Gordon[1], Cas Cremers[1], Benjamin Dowling[2], Luke Garratt[1], and Douglas Stebila[3]

Analyzed entirety of Signal key exchange and ratcheting

Does not model encryption

Ratcheted Encryption and Key Exchange: The Security of Messaging

Mihir Bellare[1(✉)], Asha Camper Singh[2], Joseph Jaeger[1], Maya Nyayapati[2], and Igors Stepanovs[1]

Introduced ratcheted key exchange and ratcheted encryption.

One-directional communication

Only sender's state vulnerable

Towards Bidirectional Ratcheted Key Exchange

Bertram Poettering[1] and Paul Rösler[2(✉)]

Extended ratcheted key exchange to be bidirectional

Does not model encryption

Prior Work

What security goal does ratcheting achieve?

Formalize it

Show that ratcheting achieves it.

Our Work

What is the BEST POSSIBLE messaging security we can achieve in the face of fine-grained state compromise?

Formalize it

Show how to achieve it.

NOT ratcheting!

Our Contributions

Optimal Channel Security Against Fine-Grained State Compromise: The Safety of Messaging

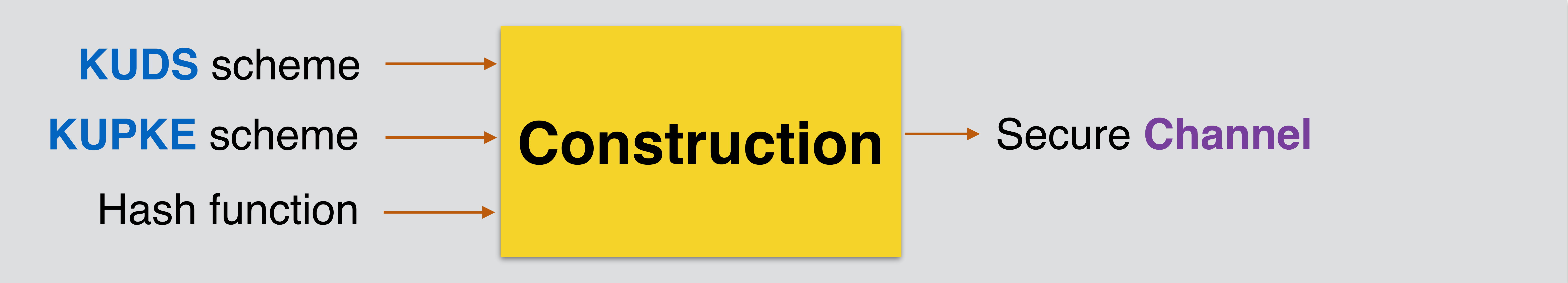
Joseph Jaeger[1] and Igors Stepanovs[1]

1. Define strongest possible security of a **channel** against fine-grained state compromise.

2. Define
   - Key-Updatable Digital Signatures (**KUDS**)
   - Key-Updatable Public-Key Encryption (**KUPKE**)

   Changed from proceedings version due to bugs in security proofs.

3. Constructions of **KUDS** and **KUPKE**.

4. 

5. **Proofs** that our constructions achieve our strong definitions of security.

Our Threat Model

all preventable attacks should be prevented

We want the best achievable integrity and privacy.

Our **Adversary** has:

- Complete control of communication.
- Ability to arbitrarily and repeatedly expose secrets.

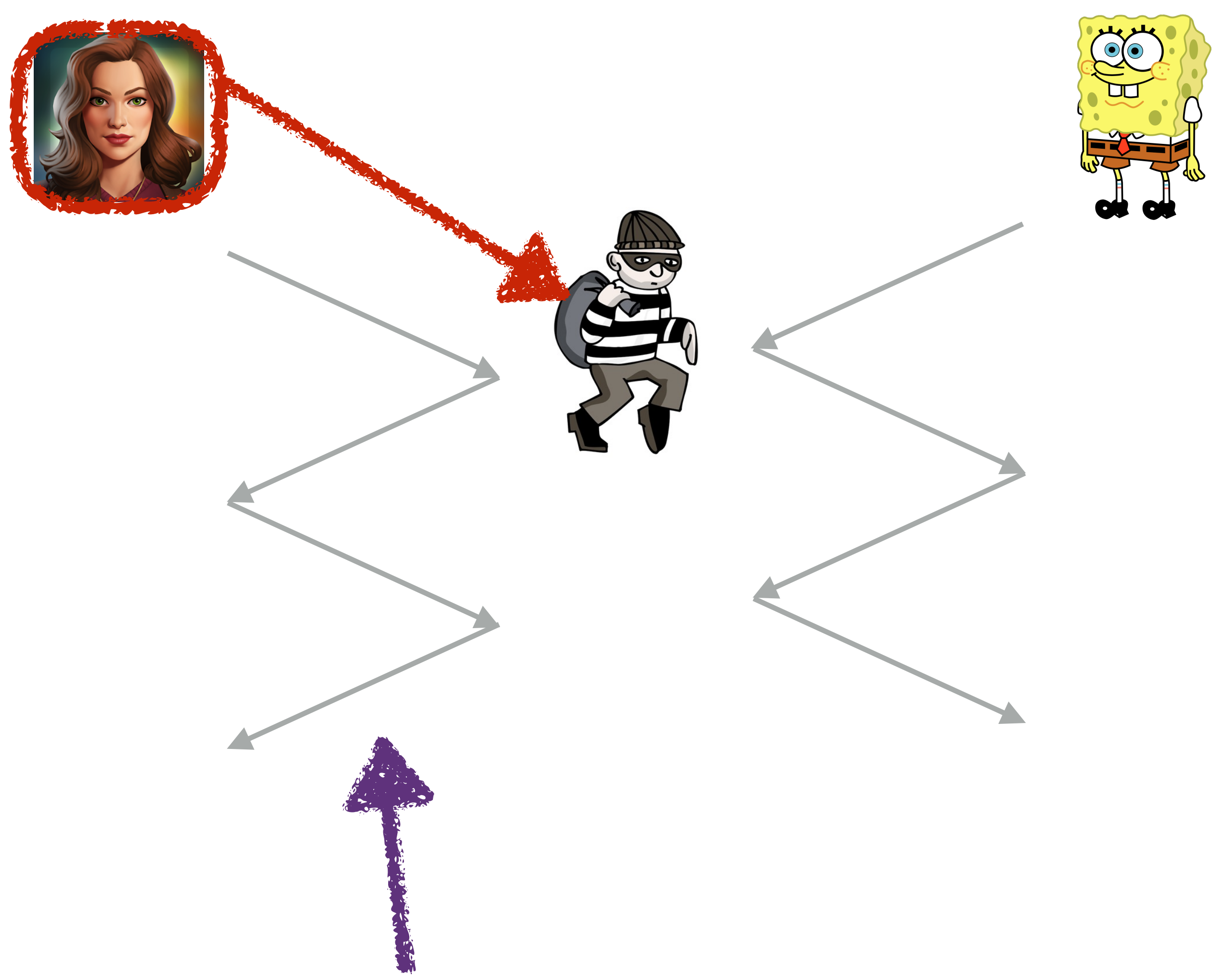
Does the Double Ratchet Algorithm (Signal) achieve this?

Answer: No. For example an attacker can,

- forge messages **to** an exposed user
- read ciphertexts **from** an exposed user
- and more

An Implication:

One exposure allows perfect MITM



This part is preventable.

Does this matter in practice?

Hard to say - requires better knowledge of attacks occurring in practice

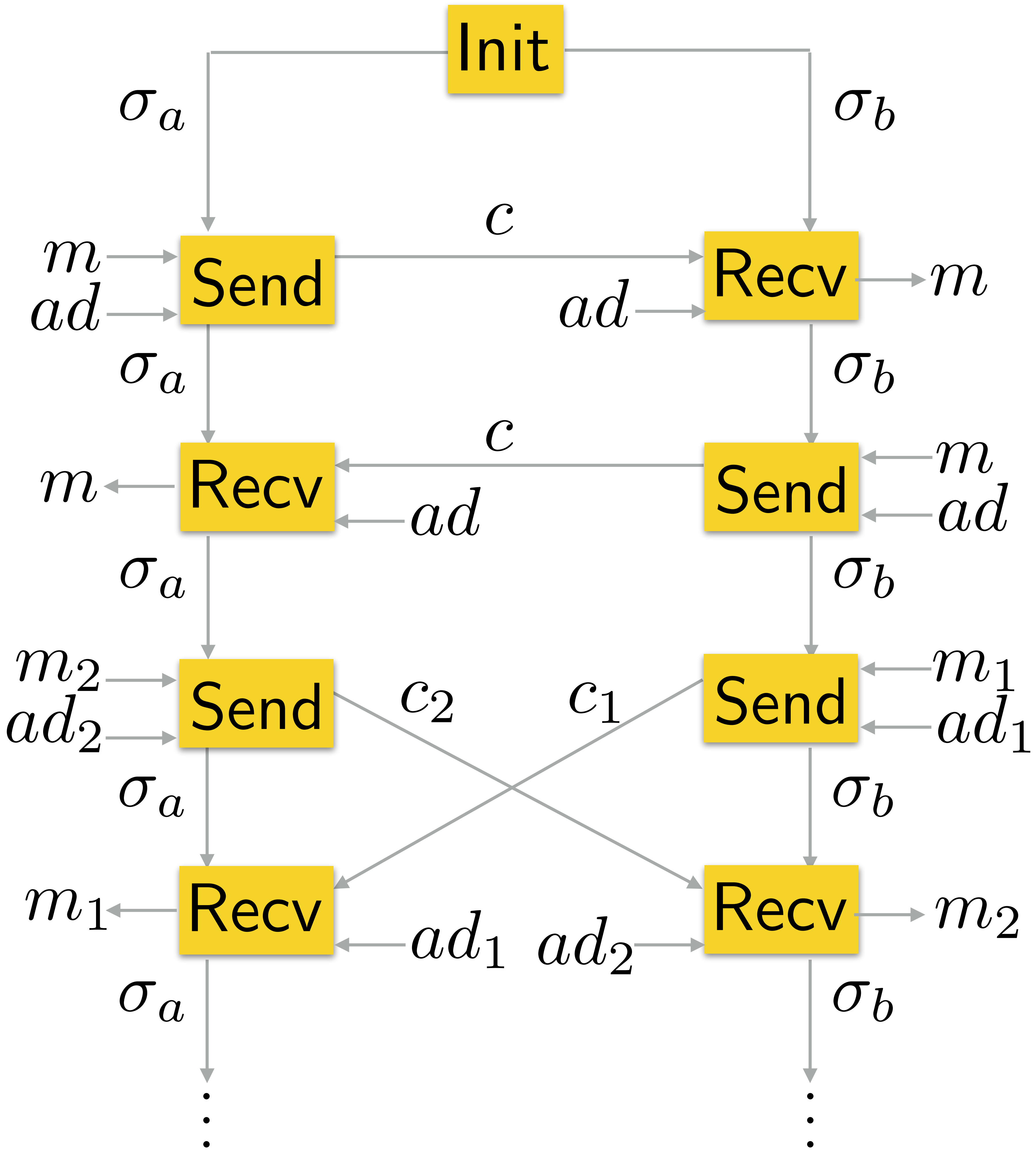
Can we just tweak Signal?

Probably not - seems to require fundamentally different techniques

(Bidirectional) Channel Syntax

From **Security Notions for Bidirectional Channels**

Giorgia Azzurra Marson and Bertram Poettering



Stateful encryption…

Allowing bidirectional communication.

Allows messages to cross "on the wire",
But preserves message order in either direction.

Defining security

Step 1: specify interface

$\textsc{Send}(u, m_0, m_1, ad)$

party u encrypts one of two messages

$\textsc{Recv}(u, c, ad)$

party u receives a ciphertext

$\textsc{Exp}(u, \mathsf{rand})$

party u exposes secret state

Attacker Goals:

- Tell which message is encrypted or
- Forge a new ciphertext

```
Game INTER^D_Ch
b <-$ {0,1}
s_I <- r_I <- s_R <- r_R <- 0
(σ_I, σ_R) <-$ Ch.Init
(z_I, z_R) <-$ (Ch.SendRS)^2
(η_I, η_R) <-$ (Ch.RecvRS)^2
b' <-$ D^{Send,Recv,Exp}
Return (b' = b)

Send(u, m_0, m_1, ad)
If nextop ≠ (u, "send")
   and nextop ≠ ⊥ then return ⊥
If |m_0| ≠ |m_1| then return ⊥
(σ_u, c) <- Ch.Send(σ_u, ad, m_b; z_u)
nextop <- ⊥
s_u <- s_u + 1 ; z_u <-$ Ch.SendRS
ctable_ū[s_u] <- (c, ad)
Return c

Recv(u, c, ad)
If nextop ≠ (u, "recv")
   and nextop ≠ ⊥ then return ⊥
(σ_u, m) <- Ch.Recv(σ_u, ad, c; η_u)
nextop <- ⊥ ; η_u <-$ Ch.RecvRS
If m ≠ ⊥ then r_u <- r_u + 1
If b = 0 and (c, ad) ≠ ctable_u[r_u] then
   Return m
Return ⊥

Exp(u, rand)
If nextop ≠ ⊥ then return ⊥
(z, η) <- (ε, ε)
If rand = "send" then
   nextop <- (u, "send") ; z <- z_u
Else if rand = "recv" then
   nextop <- (u, "recv") ; η <- η_u
Return (σ_u, z, η)
```

Adversary has:

Complete control of communication.

Ability to expose secrets.

Defining security

Step 2: generic attacks

We specified eight attacks that would break security of ***any*** channel.

Adversary $\mathcal{D}_1^{\text{SEND,RECV,EXP}}$	Adversary $\mathcal{D}_{3.1}^{\text{SEND,RECV,EXP}}$	Adversary $\mathcal{D}_4^{\text{SEND,RECV,EXP}}$
$(\sigma, z, \eta) \leftarrow \text{EXP}(\mathcal{I}, \varepsilon)$ $n \leftarrow \max_{c \in [\mathsf{Ch.Send}(\sigma,\varepsilon,1)]} \lvert c \rvert$ $m \leftarrow\!\!\$ \{0,1\}^{n+2}$ $c \leftarrow \text{SEND}(\mathcal{I}, m, 1, \varepsilon)$ If $\lvert c \rvert \leq n$ then return 1 Return 0	$(\sigma, z, \eta) \leftarrow \text{EXP}(\mathcal{I}, \varepsilon)$ $(\sigma, c) \leftarrow\!\!\$ \mathsf{Ch.Send}(\sigma, \varepsilon, 1)$ $m \leftarrow \text{RECV}(\mathcal{R}, c, \varepsilon)$ $(\sigma, c) \leftarrow\!\!\$ \mathsf{Ch.Send}(\sigma, \varepsilon, 1)$ $m \leftarrow \text{RECV}(\mathcal{R}, c, \varepsilon)$ If $m = \perp$ then return 1 Return 0	$c \leftarrow \text{SEND}(\mathcal{I}, 0, 1, \varepsilon)$ $(\sigma, z, \eta) \leftarrow \text{EXP}(\mathcal{R}, \varepsilon)$ $(\sigma, m) \leftarrow\!\!\$ \mathsf{Ch.Recv}(\sigma, \varepsilon, c)$ If $m = 1$ then return 1 Return 0
Adversary $\mathcal{D}_2^{\text{SEND,RECV,EXP}}$	**Adversary $\mathcal{D}_{3.2}^{\text{SEND,RECV,EXP}}$**	**Adversary $\mathcal{D}_5^{\text{SEND,RECV,EXP}}$**
$c \leftarrow \text{SEND}(\mathcal{I}, 1, 1, \varepsilon)$ $m \leftarrow \text{RECV}(\mathcal{R}, c, \varepsilon)$ If $m = \perp$ then return 1 Return 0	$(\sigma, z, \eta) \leftarrow \text{EXP}(\mathcal{I}, \varepsilon)$ $(\sigma, c) \leftarrow\!\!\$ \mathsf{Ch.Send}(\sigma, \varepsilon, 1)$ $m \leftarrow \text{RECV}(\mathcal{R}, c, \varepsilon)$ $c \leftarrow \text{SEND}(\mathcal{R}, 0, 1, \varepsilon)$ $(\sigma, m) \leftarrow\!\!\$ \mathsf{Ch.Recv}(\sigma, \varepsilon, c)$ If $m = 1$ then return 1 Return 0	$(\sigma, z, \eta) \leftarrow \text{EXP}(\mathcal{R}, \varepsilon)$ $c \leftarrow \text{SEND}(\mathcal{I}, 0, 1, \varepsilon)$ $(\sigma, m) \leftarrow\!\!\$ \mathsf{Ch.Recv}(\sigma, \varepsilon, c)$ If $m = 1$ then return 1 Return 0
Adversary $\mathcal{D}_3^{\text{SEND,RECV,EXP}}$		**Adversary $\mathcal{D}_6^{\text{SEND,RECV,EXP}}$**
$(\sigma, z, \eta) \leftarrow \text{EXP}(\mathcal{I}, \varepsilon)$ $(\sigma, c) \leftarrow\!\!\$ \mathsf{Ch.Send}(\sigma, \varepsilon, 1)$ $m \leftarrow \text{RECV}(\mathcal{R}, c, \varepsilon)$ If $m = \perp$ then return 1 Return 0		$(\sigma, z, \eta) \leftarrow \text{EXP}(\mathcal{I}, \text{“send”})$ $(\sigma, c) \leftarrow \mathsf{Ch.Send}(\sigma, \varepsilon, 1; z)$ $c' \leftarrow \text{SEND}(\mathcal{I}, 0, 1, \varepsilon)$ If $c' = c$ then return 1 Return 0

<u>For Example</u>

Expose state of one user and create forgery to other

Expose state of one user and decrypt ciphertext from other

Expose sending randomness of user to know which message is encrypted

<u>**NOT** generic attacks</u>
(i.e. attacks we require security against)

Expose state of user and create forgery to same

Expose state of user and decrypt ciphertext from same

Defining security

Step 3: augment interface

Our security definition AEAC:
(Authenticated encryption against compromise)

Added minimal restrictions to disallow generic attacks

<u>Game $\mathrm{AEAC}^{\mathcal{D}}_{\mathsf{Ch}}$</u>

$b \leftarrow_{\$} \{0,1\}$; $s_{\mathcal{I}} \leftarrow r_{\mathcal{I}} \leftarrow s_{\mathcal{R}} \leftarrow r_{\mathcal{R}} \leftarrow 0$
$\mathsf{restricted}_{\mathcal{I}} \leftarrow \mathsf{false}$; $\mathsf{restricted}_{\mathcal{R}} \leftarrow \mathsf{false}$
$\mathbf{forg}_{\mathcal{I}}[\cdot] \leftarrow$ "nontriv" ; $\mathbf{forg}_{\mathcal{R}}[\cdot] \leftarrow$ "nontriv"
$\mathcal{X}_{\mathcal{I}} \leftarrow \mathcal{X}_{\mathcal{R}} \leftarrow 0$; $(st_{\mathcal{I}}, st_{\mathcal{R}}) \leftarrow_{\$} \mathsf{Ch.Init}$
$(z_{\mathcal{I}}, z_{\mathcal{R}}) \leftarrow_{\$} (\mathsf{Ch.SendRS})^2$
$(\eta_{\mathcal{I}}, \eta_{\mathcal{R}}) \leftarrow_{\$} (\mathsf{Ch.RecvRS})^2$
$b' \leftarrow_{\$} \mathcal{D}^{\text{SEND,RECV,EXP}}$
Return $(b' = b)$

<u>$\text{SEND}(u, m_0, m_1, ad)$</u>

Require $\mathsf{nextop} \in \{(u, \text{"send"}), \bot\}$
Require $|m_0| = |m_1|$
If $r_u < \mathcal{X}_u$ or $\mathsf{restricted}_u$ or $\mathbf{ch}_u[s_u + 1] =$ "forb":
  Require $m_0 = m_1$
$(st_u, c) \leftarrow \mathsf{Ch.Send}(st_u, ad, m_b; z)$
$\mathsf{nextop} \leftarrow \bot$; $s_u \leftarrow s_u + 1$; $z_u \leftarrow_{\$} \mathsf{Ch.SendRS}$
If $\neg\mathsf{restricted}_u$: $\mathbf{cad}_{\overline{u}}[s_u] \leftarrow (c, ad)$
If $m_0 \neq m_1$: $\mathbf{ch}_u[s_u] \leftarrow$ "done"
Return c

<u>$\text{RECV}(u, c, ad)$</u>

Require $\mathsf{nextop} \in \{(u, \text{"recv"}), \bot\}$
$(st_u, m) \leftarrow \mathsf{Ch.Recv}(st_u, ad, c; \eta_u)$
$\mathsf{nextop} \leftarrow \bot$; $\eta_u \leftarrow_{\$} \mathsf{Ch.RecvRS}$
If $m = \bot$: return $\bot$
$r_u \leftarrow r_u + 1$
If $\mathbf{forg}_u[r_u] =$ "triv" and $(c, ad) \neq \mathbf{cad}_u[r_u]$:
  $\mathsf{restricted}_u \leftarrow \mathsf{true}$
If $\mathsf{restricted}_u$ or $(b = 0$ and $(c, ad) \neq \mathbf{cad}_u[r_u])$:
  Return m
Return $\bot$

<u>$\text{EXP}(u, \mathsf{rand})$</u> // $\mathsf{rand} \in \{\varepsilon, \text{"send"}, \text{"recv"}\}$

Require $\mathsf{nextop} = \bot$
If $\mathsf{restricted}_u$: Return (st_u, z_u, η_u)
If $\exists i \in (r_u, s_{\overline{u}}]$ s.t. $\mathbf{ch}_{\overline{u}}[i] =$ "done":
  Return $\bot$
$\mathbf{forg}_{\overline{u}}[s_u + 1] \leftarrow$ "triv" ; $(z, \eta) \leftarrow (\varepsilon, \varepsilon)$; $\mathcal{X}_{\overline{u}} \leftarrow s_u + 1$
If $\mathsf{rand} =$ "send" then
  $\mathsf{nextop} \leftarrow (u, \text{"send"})$; $z \leftarrow z_u$; $\mathcal{X}_{\overline{u}} \leftarrow s_u + 2$
  $\mathbf{forg}_{\overline{u}}[s_u + 2] \leftarrow$ "triv" ; $\mathbf{ch}_u[s_u + 1] \leftarrow$ "forb"
Else if $\mathsf{rand} =$ "recv" then
  $\mathsf{nextop} \leftarrow (u, \text{"recv"})$; $\eta \leftarrow \eta_u$
Return (st_u, z, η)

<u>Some implications</u>

Users can't forge to self

Users can't read own ciphertexts

After decrypting a ciphertext lose ability to decrypt it

After sending ciphertext lose ability to authenticate it

After decrypting a forged ciphertext unable to encrypt to / decrypt from valid partner.

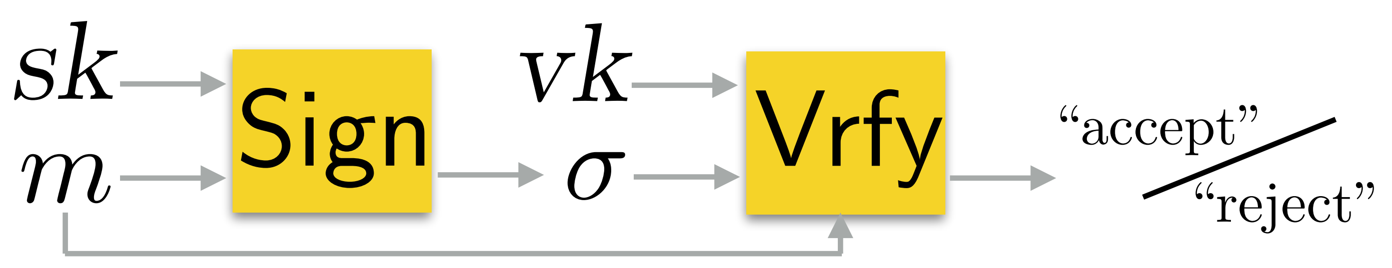
<u>We achieve with new forms of</u>

DS

PKE

Key-Updatable Digital Signature Schemes

New Public Key Primitives



Syntax Augment DS scheme with algorithms to update keys with respect to arbitrary strings.

Security Variant of (one-time) strong unforgeability.

$\textsc{Sign}(m)$ $\textsc{Upd}(\Delta)$ $\textsc{Exp}()$

Forgery to a sequence of updates $\vec{\Delta}_1$ disallowed if exposed key for $\vec{\Delta}_2 \sqsubseteq \vec{\Delta}_1$

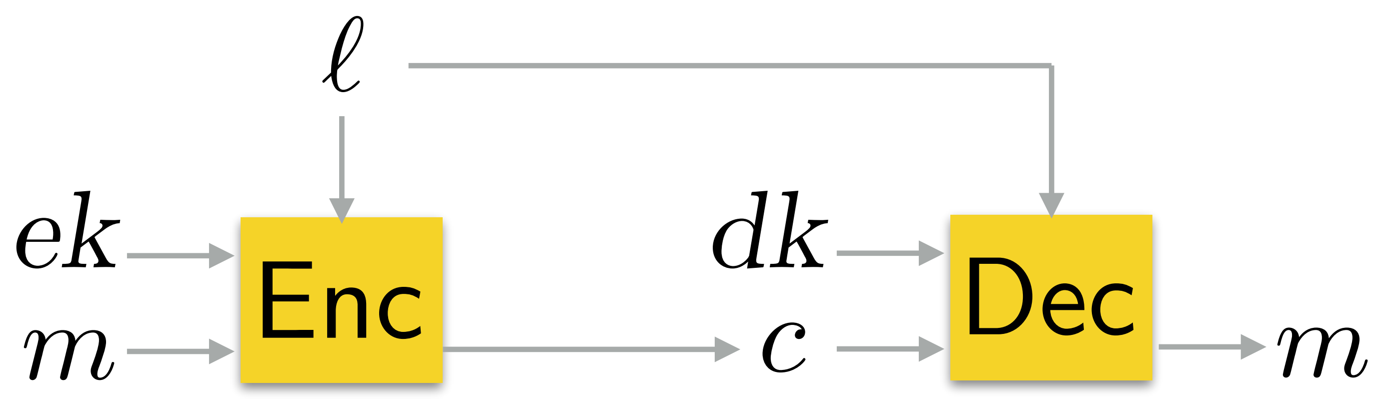
Construction From a forward-secure DS scheme.

To update key, sign update string then evolve to future key.

Algorithm $\mathsf{DS}_{\mathrm{KU}}.\mathsf{UpdSk}(sk, \Delta)$

$(sk_{\mathrm{KE}}, i, \mathbf{\Sigma}) \leftarrow sk$
$\mathbf{\Sigma}[i] \leftarrow\!\!\$\ \mathsf{DS}_{\mathrm{KE}}.\mathsf{Sign}(sk_{\mathrm{KE}}, 0 \,\|\, \Delta)$
$sk_{\mathrm{KE}} \leftarrow\!\!\$\ \mathsf{DS}_{\mathrm{KE}}.\mathsf{Up}(sk_{\mathrm{KE}})$
$sk \leftarrow (sk_{\mathrm{KE}}, i+1, \mathbf{\Sigma})$
Return sk

Key-Updatable Public Key Encryption



Syntax Augment PKE scheme with algorithms to update keys with respect to arbitrary strings.

Security Variant of CCA-security with labels

$\textsc{Enc}(m_0, m_1, \ell)$ $\textsc{Dec}(c, \ell)$ $\textsc{UpdDk}()$ $\textsc{UpdEk}()$ $\textsc{Exp}()$

Challenge query to sequence of updates $\vec{\Delta}_1$disallowed if exposed key for $\vec{\Delta}_2 \sqsubseteq \vec{\Delta}_1$

Construction Immediate from a hierarchical identity-based encryption scheme.
(Update strings correspond to HIBE identities.)

Our Construction

KUPKE scheme	PKE
KUDS scheme	DS
Hash function	H

→ Channel CH[DS, PKE, H]

State stored by each party

s / r	Sent/Received Counters
sk / ek	Signing/Encryption Keys
$vk / \vec{dk}$	Verification/Decryption Keys
$\tau_r / \vec{\tau}_s$	Sent/Received "Transcripts"
hk	Hash Function Key

Algorithm $\mathsf{SCh.Send}(\sigma, ad, m)$

$(s, r, r^{ack}, sk, vk, ek, \mathbf{dk}, hk, \tau_r, \vec{\tau}_s) \leftarrow \sigma \; ; \; s \leftarrow s + 1$

$(sk', vk') \leftarrow_{\$} \mathsf{DS.Kg} \; ; \; (ek', \mathbf{dk}[s]) \leftarrow_{\$} \mathsf{PKE.Kg}$

$\ell \leftarrow (s, r, ad, vk', ek', \tau_r, \vec{\tau}_s[s-1])$

$(ek', c') \leftarrow_{\$} \mathsf{PKE.Enc}(ek, \ell, m, \vec{\tau}_s[r^{ack}+1, \ldots, s-1])$

$v \leftarrow (c', \ell) \; ; \; \sigma \leftarrow_{\$} \mathsf{DS.Sign}(sk, v)$

$c \leftarrow (\sigma, v) \; ; \; \vec{\tau}_s[s] \leftarrow \mathsf{H.Ev}(hk, c)$

$\sigma \leftarrow (s, r, r^{ack}, sk', vk, ek, \mathbf{dk}, hk, \tau_r, \vec{\tau}_s)$

Return (σ, c)

Privacy from PKE.

Integrity from DS

New keys with every message (**Forward/Backward security**)

Key-updates (**Forward security**)

- ek/vk updated with sent transcripts
- dk/sk update with received transcripts

Counter prevents **reordering**

Paper has 9 attacks against variants

Security of our Bidirectional Channel

Theorem: Suppose

- H is collision-resistant.
- DS is a UFEXP-secure and UNIQ-secure KUDS scheme.
- PKE is an INDEXP-secure KUPKE scheme.

Then our channel, $\mathsf{SCh} = \mathsf{SCH}[\mathsf{DS}, \mathsf{PKE}, \mathsf{H}]$ is AEAC-secure.

Tight reduction to multi-user security of underlying primitives.

Concretely: Given adversary $\mathcal{D}$ (making q queries) we build adversaries $\mathcal{A}_{\mathsf{H}}, \mathcal{A}_{\mathsf{DS}}, \mathcal{B}_{\mathsf{DS}}, \mathcal{A}_{\mathsf{PKE}}$ such that

$$\mathsf{Adv}^{\mathsf{aeac}}_{\mathsf{SCh},\mathcal{D}} \leq 2(q2^{-\mu} + \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{H},\mathcal{A}_{\mathsf{H}}} + \mathsf{Adv}^{\mathsf{ufexp}}_{\mathsf{DS},\mathcal{A}_{\mathsf{DS}}} + \mathsf{Adv}^{\mathsf{uniq}}_{\mathsf{DS},\mathcal{B}_{\mathsf{DS}}}) + \mathsf{Adv}^{\mathsf{indexp}}_{\mathsf{PKE},\mathcal{A}_{\mathsf{PKE}}}$$

Where $\mu = \mathrm{H}_\infty(\mathsf{DS.Kg}) + \mathrm{H}_\infty(\mathsf{PKE.Kg}) + \mathrm{H}_\infty(\mathsf{PKE.Enc})$

Proof

Step 1: Integrity.

Substep 1.1: Cannot predict future ciphertext. (Min-entropy)

Substep 1.2: Cannot cause transcript collision. (HF security)

Substep 1.3: Cannot forge signatures. (DS security)

Subtle proof step missed by some related papers

Step 2: Privacy.

Substep 2.1: Cannot send ciphertext with new signature. (DS uniqueness)

Substep 2.2: Encryption is secure. (PKE security)

Our Contributions

Optimal Channel Security Against Fine-Grained State Compromise: The Safety of Messaging

Joseph Jaeger[1] and Igors Stepanovs[1]

1. Define strongest possible security of a **channel** against fine-grained state compromise.
2. Define Key-Updatable Digital Signatures (**KUDS**)
Key-Updatable Public-Key Encryption (**KUPKE**)
3. Constructions of **KUDS** and **KUPKE**.
4. 

5. **Proofs** that our constructions achieve our strong definitions of security.

Thanks! Any questions?