WhatsUpp with Sender Keys?

Analysis, Improvements and Security Proofs

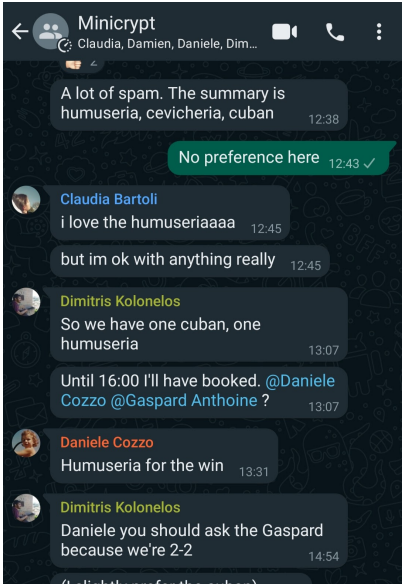
David Balbás[1,2], Daniel Collins[3], Phillip Gajland[4,5]

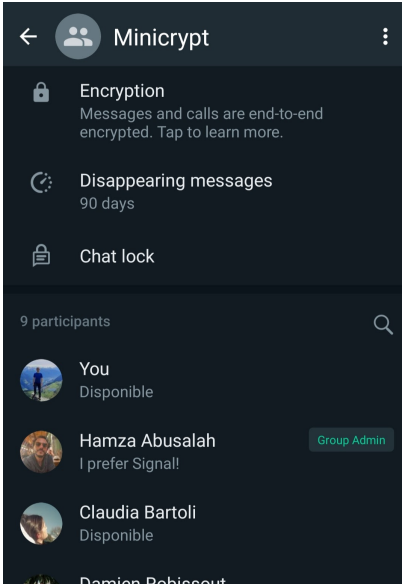
7th December 2023

[1]IMDEA Software Institute, Madrid, Spain
[2]Universidad Politécnica de Madrid, Spain
[3]EPFL, Lausanne, Switzerland
[4]Max Planck Institute for Security and Privacy, Bochum, Germany
[5]Ruhr University Bochum, Germany

ASIACRYPT 2023, Guangzhou, China

Group Messaging





Group Messaging

- **Messaging protocols** are used by billions daily. Many apps claim **security** + **end-to-end encryption**.

Group Messaging

- **Messaging protocols** are used by billions daily. Many apps claim **security** + **end-to-end encryption**.
- Formal protocol analysis is crucial.

Group Messaging

- **Messaging protocols** are used by billions daily. Many apps claim **security** + **end-to-end encryption**.
- Formal protocol analysis is crucial.
- **MLS**: Lots of theoretical analysis. *Secure, efficient, complex.*

Group Messaging

- **Messaging protocols** are used by billions daily. Many apps claim **security** + **end-to-end encryption**.
- Formal protocol analysis is crucial.
- **MLS**: Lots of theoretical analysis. *Secure, efficient, complex.*
- **Sender Keys**: WhatsApp, Signal. *No formal analysis so far.*

Sender Keys

- Sender Keys is a **simple, efficient** group messaging protocol used in WhatsApp, Signal, Matrix...

B

A

C

Sender Keys

- Sender Keys is a **simple, efficient** group messaging protocol used in WhatsApp, Signal, Matrix...
- Parties use their own symmetric key k_{ID} to encrypt. **No group key.**

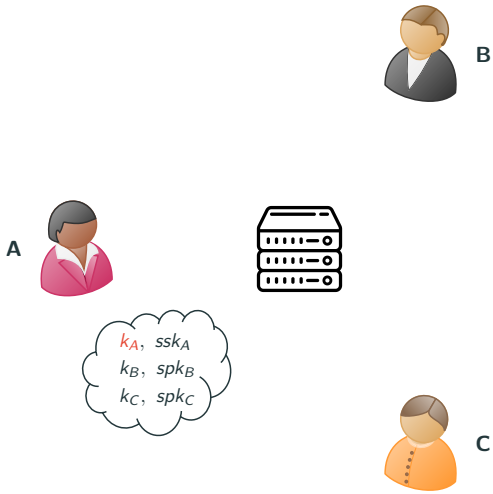
B

A

C

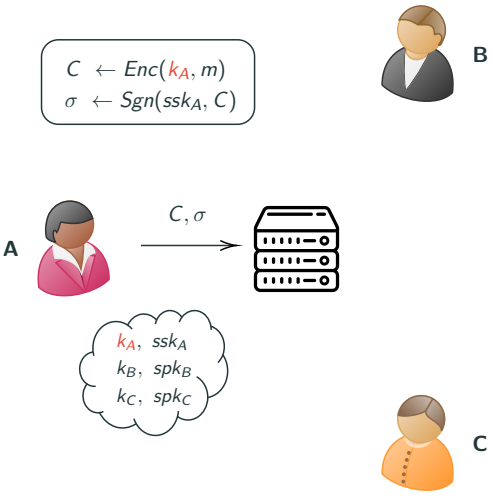
Sender Keys

- Sender Keys is a **simple, efficient** group messaging protocol used in WhatsApp, Signal, Matrix...
- Parties use their own symmetric key k_{ID} to encrypt. **No group key.**



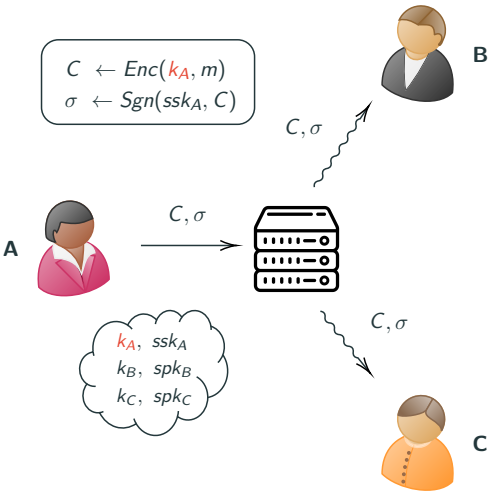
Sender Keys

- Sender Keys is a **simple, efficient** group messaging protocol used in WhatsApp, Signal, Matrix...
- Parties use their own symmetric key k_{ID} to encrypt. **No group key.**



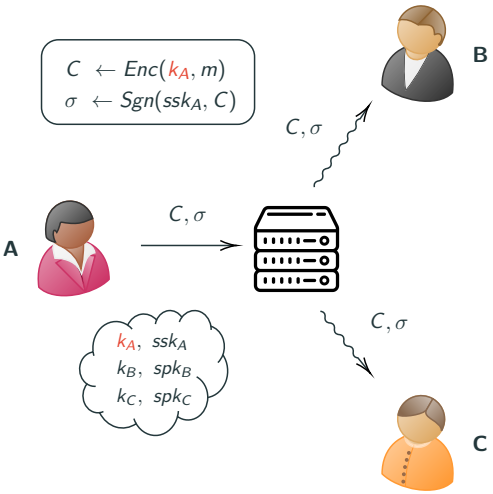
Sender Keys

- Sender Keys is a **simple, efficient** group messaging protocol used in WhatsApp, Signal, Matrix...
- Parties use their own symmetric key k_{ID} to encrypt. **No group key.**



Sender Keys

- Sender Keys is a **simple, efficient** group messaging protocol used in WhatsApp, Signal, Matrix...
- Parties use their own symmetric key k_{ID} to encrypt. **No group key.**
- Parties use *two-party messaging* to share fresh key material.



What is Secure Messaging?

- Correct, authentic, confidential, and asynchronous messaging.

What is Secure Messaging?

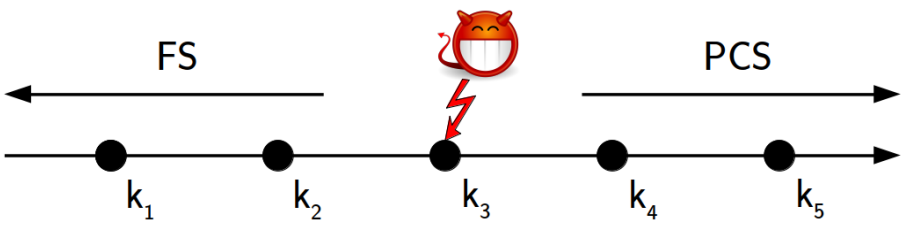
- Correct, authentic, confidential, and asynchronous messaging.
- **Secure membership.**

What is Secure Messaging?

- Correct, authentic, confidential, and asynchronous messaging.
- **Secure membership.**
- **Forward Security (FS):** *past* messages secret after compromise.

What is Secure Messaging?

- Correct, authentic, confidential, and asynchronous messaging.
- **Secure membership.**
- **Forward Security (FS):** *past* messages secret after compromise.
- **Post-Compromise Security (PCS):** *future* messages secret a key refresh.



So, WhatsUpp with Sender Keys?

So, WhatsUpp with Sender Keys?

Can we **formalise** Sender Keys in a *meaningful security model*, considering the aforementioned requirements?

So, WhatsUpp with Sender Keys?

Can we **formalise** Sender Keys in a *meaningful security model*, considering the aforementioned requirements?

What are its main **deficiencies**, and how can we address them *efficiently*?

Our Work

- **Formalization** of Sender Keys in a novel framework.

Our Work

- **Formalization** of Sender Keys in a novel framework.
- **Security model** capturing interaction between group messages and two-party channels.

Our Work

- **Formalization** of Sender Keys in a novel framework.
- **Security model** capturing interaction between group messages and two-party channels.
- **Proof of security** with some restrictions. Identified shortcomings.

Our Work

- **Formalization** of Sender Keys in a novel framework.
- **Security model** capturing interaction between group messages and two-party channels.
- **Proof of security** with some restrictions. Identified shortcomings.
- **Improvements** in *Sender Keys+*: better efficiency and security.

Our Work

- **Formalization** of Sender Keys in a novel framework.
- **Security model** capturing interaction between group messages and two-party channels.
- **Proof of security** with some restrictions. Identified shortcomings.
- **Improvements** in *Sender Keys+*: better efficiency and security.

Concurrent work [Albrecht, Dowling, Jones, S&P 2024] formalizes Matrix, similar conclusions.

Protocol and Syntax

Syntax

A **Group Messenger (GM)** includes:

Syntax

A **Group Messenger (GM)** includes:

- $C \xleftarrow{\$} \mathit{Send}(\mathrm{m}, \gamma)$

Syntax

A **Group Messenger (GM)** includes:

- $C \xleftarrow{\$} Send(\mathrm{m}, \gamma)$
- $(\mathrm{m}, ID^*, e, i) \xleftarrow{\$} Recv(C, \gamma)$

Syntax

A **Group Messenger (GM)** includes:

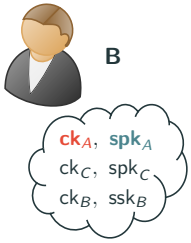
- $C \xleftarrow{\$} \mathit{Send}(\mathsf{m}, \gamma)$
- $(\mathsf{m}, \mathit{ID}^*, e, i) \xleftarrow{\$} \mathit{Recv}(C, \gamma)$
- $T \xleftarrow{\$} \mathit{Exec}(\mathtt{cmd}, \boldsymbol{\mathit{IDs}}, \gamma)$, $\mathtt{cmd} \in \{\mathtt{crt}, \mathtt{add}, \mathtt{rem}, \mathtt{upd}\}$

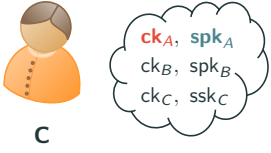
Syntax

A **Group Messenger (GM)** includes:

- $C \xleftarrow{\$} \mathit{Send}(\mathrm{m}, \gamma)$
- $(\mathrm{m}, \mathit{ID}^*, e, i) \xleftarrow{\$} \mathit{Recv}(C, \gamma)$
- $T \xleftarrow{\$} \mathit{Exec}(\mathtt{cmd}, \boldsymbol{\mathit{IDs}}, \gamma)$, $\mathtt{cmd} \in \{\mathtt{crt}, \mathtt{add}, \mathtt{rem}, \mathtt{upd}\}$
- $b \xleftarrow{\$} \mathit{Proc}(T, \gamma)$

Sender Keys: Send & Recv

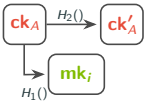




Sender Keys: Send & Recv

A

ck_B, spk_B
ck_C, spk_C
ck'_A, ssk_A



$mk_i \leftarrow H_1(ck_A)$
$ck'_A \leftarrow H_2(ck_A)$

B

ck_A, spk_A
ck_C, spk_C
ck_B, ssk_B

C

ck_A, spk_A
ck_B, spk_B
ck_C, ssk_C

Sender Keys: Send & Recv

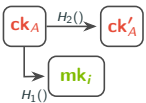
A

$(\boldsymbol{c}, i, A), \sigma$

B

$\mathsf{ck}_B,\ \mathsf{spk}_B$
$\mathsf{ck}_C,\ \mathsf{spk}_C$
$\mathsf{ck}'_A,\ \mathsf{ssk}_A$

$\mathsf{ck}_A,\ \mathsf{spk}_A$
$\mathsf{ck}_C,\ \mathsf{spk}_C$
$\mathsf{ck}_B,\ \mathsf{ssk}_B$

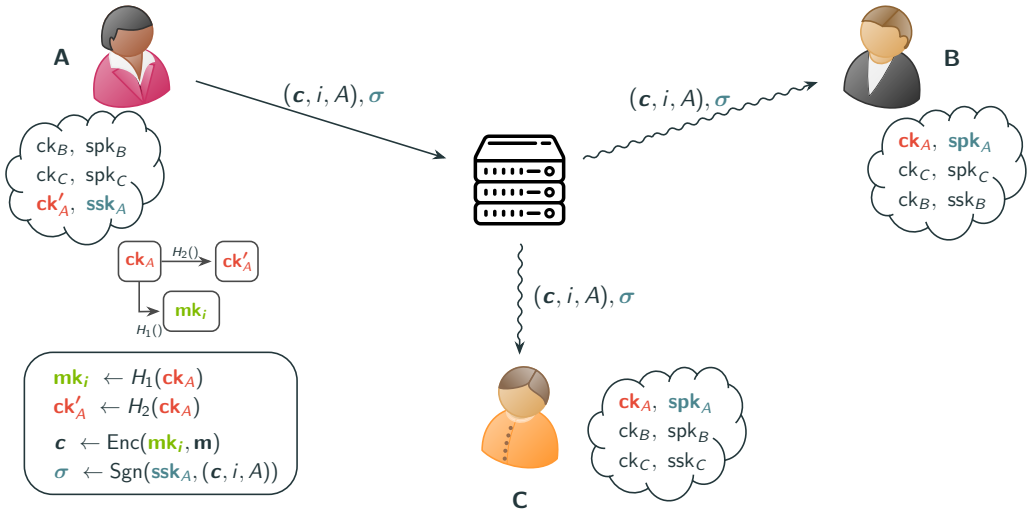


$\mathsf{mk}_i \leftarrow H_1(\mathsf{ck}_A)$
$\mathsf{ck}'_A \leftarrow H_2(\mathsf{ck}_A)$
$\boldsymbol{c} \leftarrow \mathsf{Enc}(\mathsf{mk}_i, \mathbf{m})$
$\sigma \leftarrow \mathsf{Sgn}(\mathsf{ssk}_A, (\boldsymbol{c}, i, A))$

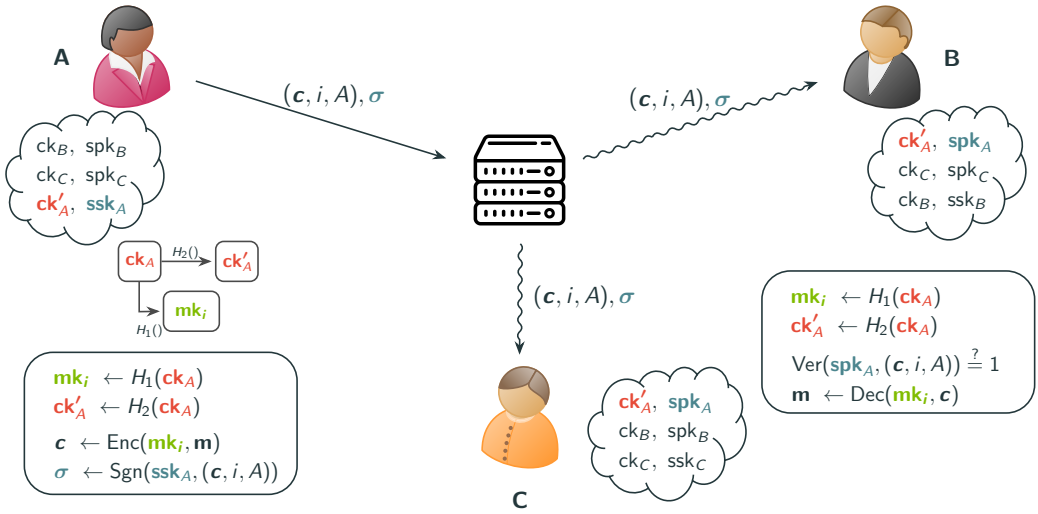
C

$\mathsf{ck}_A,\ \mathsf{spk}_A$
$\mathsf{ck}_B,\ \mathsf{spk}_B$
$\mathsf{ck}_C,\ \mathsf{ssk}_C$

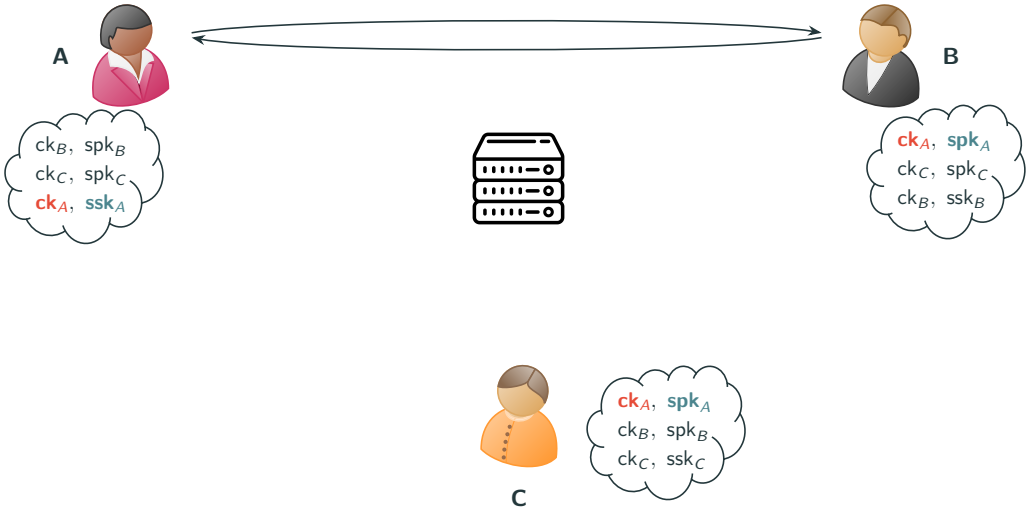
Sender Keys: Send & Recv



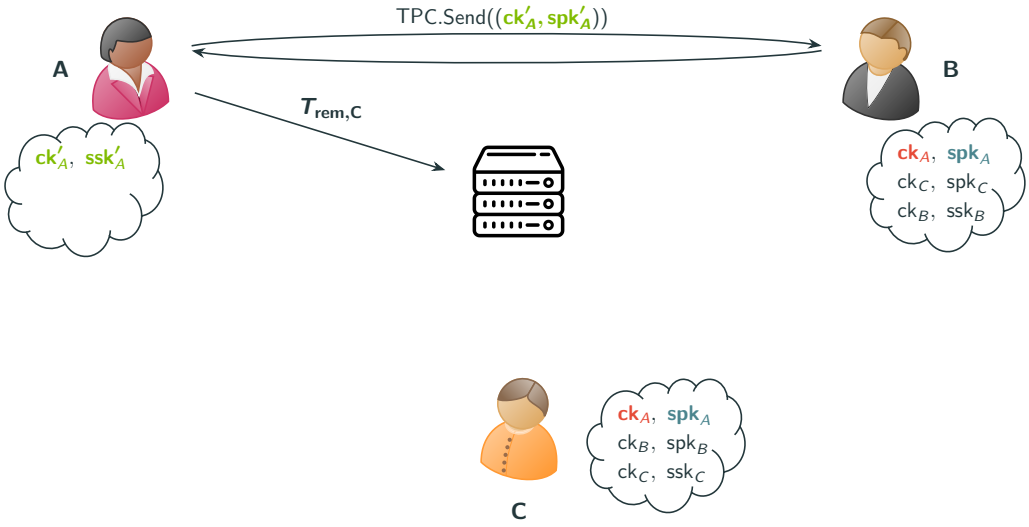
Sender Keys: Send & Recv



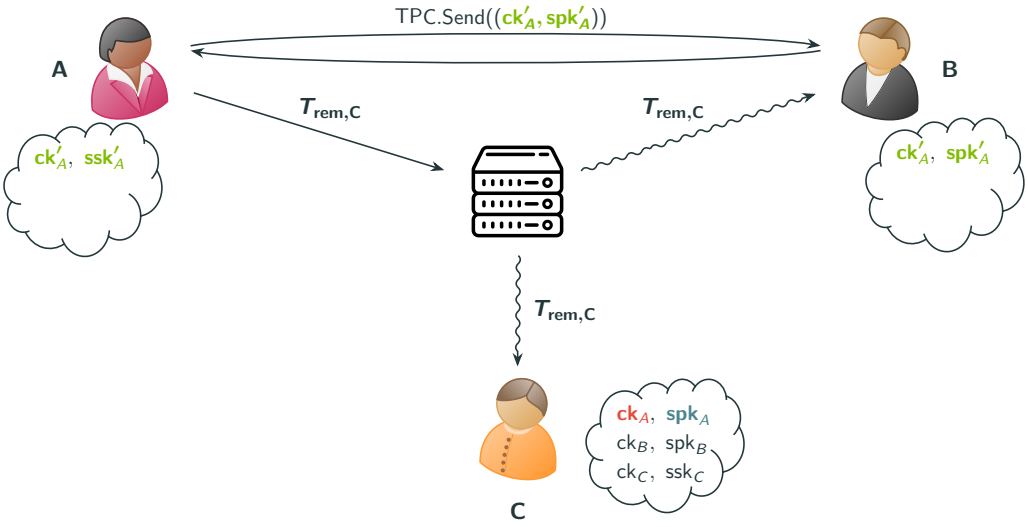
Sender Keys: Exec & Proc



Sender Keys: Exec & Proc

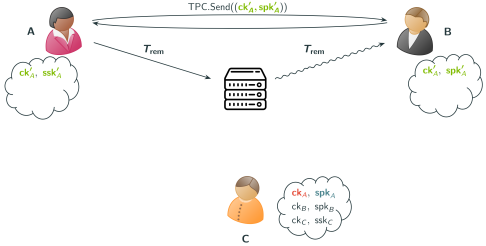


Sender Keys: Exec & Proc



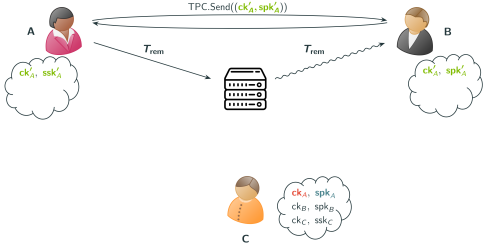
Two-Party Channels

- **If C leaves** (or someone updates):



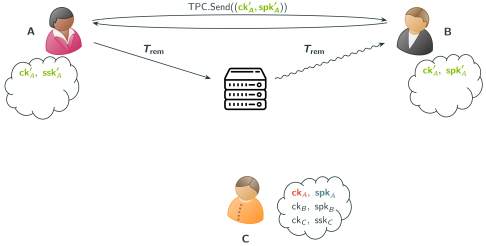
Two-Party Channels

- **If C leaves** (or someone updates):
  - A, B process removal & erase keys.
  - Fresh keys sent over secure 2PC.



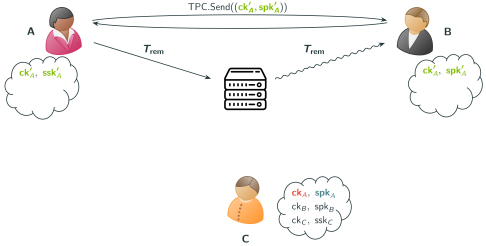
Two-Party Channels

- **If C leaves** (or someone updates):
  - A, B process removal & erase keys.
  - Fresh keys sent over secure 2PC.
- In reality, *New keys sent encrypted...* under **Double Ratchet keys!** [MP16]
- A compromise also affects 2PC keys.



Two-Party Channels

- **If C leaves** (or someone updates):
  - A, B process removal & erase keys.
  - Fresh keys sent over secure 2PC.
- In reality, *New keys sent encrypted...* under **Double Ratchet keys!** [MP16]
- A compromise also affects 2PC keys.

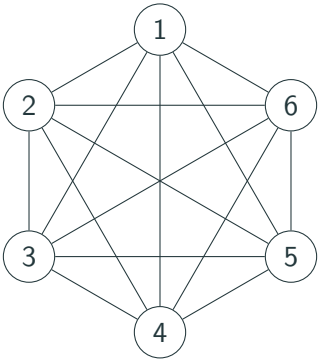


Modelling 2PC

We model *two-party channels as a primitive* 2PC, parametrized by a *PCS bound* Δ.
Δ is the number of messages needed for the channel to heal [ACD19].

Two-Party Channels

Two-party channels only refresh (i.e. achieve PCS) if users interact.



This can be problematic in practice.

Proving Security

Security Model

We introduce a *message indistinguishability* security game $\mathbf{M\text{-}IND_C}$.

Security Model

We introduce a *message indistinguishability* security game $\mathbf{M\text{-}IND_C}$.

- Adaptive $\mathcal{A}$ can *forge and inject messages*.

Security Model

We introduce a *message indistinguishability* security game $\mathbf{M\text{-}IND_C}$.

- Adaptive $\mathcal{A}$ can *forge and inject messages*.
- Users can be *exposed* at any time (capturing FS, PCS).

Security Model

We introduce a *message indistinguishability* security game $\textbf{M-IND}_{\mathbf{C}}$.

- Adaptive $\mathcal{A}$ can *forge and inject messages*.
- Users can be *exposed* at any time (capturing FS, PCS).
- **$\mathcal{A}$ wins if:**
  - breaks semantic security, or
  - forges a message
  - in a *clean*/safe execution under C.

Security Model

We introduce a *message indistinguishability* security game **M-IND$_\mathbf{C}$**.

- Adaptive $\mathcal{A}$ can *forge and inject messages*.
- Users can be *exposed* at any time (capturing FS, PCS).
- **$\mathcal{A}$ wins if:**
  - breaks semantic security, or
  - forges a message
  - in a *clean*/safe execution under C.

Oracles:

- $\texttt{Create}(ID, \boldsymbol{IDs})$
- $\texttt{Challenge}(ID, \mathrm{m}_0, \mathrm{m}_1)$
- $\texttt{Send}(ID, \mathrm{m})$
- $\texttt{Receive}(ID, C)$
- $(\texttt{Add}/\texttt{Remove})(ID, ID')$
- $\texttt{Update}(ID)$
- $\texttt{Deliver}(ID, T)$
- $\texttt{Expose}(ID)$

Security

Security of Sender Keys (informal)

Let

- SymEnc a IND-CPA symmetric encryption scheme
- Sig a SUF-CMA signature scheme
- H a PRG.
- 2PC a 2PC-IND$_\Delta$ two-party channels scheme for PCS bound $\Delta > 0$.

Then, with **some adversarial restrictions**, Sender Keys is M-IND$_{C(\Delta)}$ secure.

Security

Security of Sender Keys (informal)

Let

- SymEnc a IND-CPA symmetric encryption scheme
- Sig a SUF-CMA signature scheme
- H a PRG.
- 2PC a 2PC-IND$_\Delta$ two-party channels scheme for PCS bound $\Delta > 0$.

Then, with **some adversarial restrictions**, Sender Keys is M-IND$_{C(\Delta)}$ secure.

Conclusion: The core of the protocol has *no fundamental flaws*. But it still presents some drawbacks.

Limitations

- **Slow healing** due to two-party channels, inefficient updates.

Limitations

- **Slow healing** due to two-party channels, inefficient updates.
- **Total ordering** of control messages required.
  Security against network adversaries unclear.

Limitations

- **Slow healing** due to two-party channels, inefficient updates.
- **Total ordering** of control messages required.
  Security against network adversaries unclear.
- No **authentication** for control messages:
  - Burgle into the group [Rösler et. al, 2018]
  - Censorship
  - Insecure administration

Limitations

- **Slow healing** due to two-party channels, inefficient updates.
- **Total ordering** of control messages required.
  Security against network adversaries unclear.
- No **authentication** for control messages:
  - Burgle into the group [Rösler et. al, 2018]
  - Censorship
  - Insecure administration
- **Weak forward security** for authentication.

Limitations

- **Slow healing** due to two-party channels, inefficient updates.
- **Total ordering** of control messages required.
  Security against network adversaries unclear.
- No **authentication** for control messages:
  - Burgle into the group [Rösler et. al, 2018]
  - Censorship
  - Insecure administration
- **Weak forward security** for authentication.

Sender Keys+

- **Slow healing** due to two-party channels, *inefficient updates.*
- **Total ordering** of control messages required.
  Security against network adversaries unclear.
- No **authentication** *for control messages*:
  - Burgle into the group [Rösler et. al, 2018]
  - Censorship
  - Insecure administration
- **Weak forward security** *for authentication*.

We propose and formalize **Sender Keys+** *as a practical, improved alternative*!

Efficient Key Updates

- **Naive approach:** Alice sends a fresh pair $(\mathsf{ck}_A, \mathsf{spk}_A)$ to everyone.
  - Heals only Alice, $\mathcal{O}(n^2)$ messages for full group.

Efficient Key Updates

- **Naive approach:** Alice sends a fresh pair $(\mathsf{ck}_A, \mathsf{spk}_A)$ to everyone.
  - Heals only Alice, $\mathcal{O}(n^2)$ messages for full group.
- **Another approach:** Alice sends $r \xleftarrow{\$} \{0,1\}^\lambda$ to all users; $H(\mathsf{ck}_{ID}, r)$ is computed for all ck_{ID}.
  - Heals all users, $\mathcal{O}(n)$ communication.
  - Problem: concurrency; when to delete keys?

Efficient Key Updates

- **Naive approach:** Alice sends a fresh pair $(\mathsf{ck}_A, \mathsf{spk}_A)$ to everyone.
  - Heals only Alice, $\mathcal{O}(n^2)$ messages for full group.
- **Another approach:** Alice sends $r \xleftarrow{\$} \{0,1\}^\lambda$ to all users; $H(\mathsf{ck}_{ID}, r)$ is computed for all ck_{ID}.
  - Heals all users, $\mathcal{O}(n)$ communication.
  - Problem: concurrency; when to delete keys?
- **Solution:** All *keys hashed forward* N times before r is applied + keep track of indices.
  - Can *remove parties efficiently* – $O(n)$ communication.
  - Drawback: signature keys not updated.

Final Remarks

Takeaways

- *New modelling* for group messaging deviating from CGKA.

Takeaways

- *New modelling* for group messaging deviating from CGKA.
- Sender Keys is *provably secure* but in a *weak model*.

Takeaways

- *New modelling* for group messaging deviating from CGKA.
- Sender Keys is *provably secure* but in a *weak model*.
- *Sender Keys+*: improved security and efficiency.

Takeaways

- *New modelling* for group messaging deviating from CGKA.
- Sender Keys is *provably secure* but in a *weak model*.
- *Sender Keys+*: improved security and efficiency.
- Careful with not *refreshing 2pc!* Stale channels are unsafe.

Takeaways

- *New modelling* for group messaging deviating from CGKA.
- Sender Keys is *provably secure* but in a *weak model*.
- *Sender Keys+*: improved security and efficiency.
- Careful with not *refreshing 2pc!* Stale channels are unsafe.

Future work: what if *total ordering* is violated?

Takeaways

- *New modelling* for group messaging deviating from CGKA.
- Sender Keys is *provably secure* but in a *weak model*.
- *Sender Keys+*: improved security and efficiency.
- Careful with not *refreshing 2pc!* Stale channels are unsafe.

Future work: what if *total ordering* is violated?

Thank you!

ia.cr/2023/1385

david.balbas@imdea.org
daniel.collins@epfl.ch
phillip.gajland@mpi-sp.org