Cryptographic Administration for Secure Group Messaging

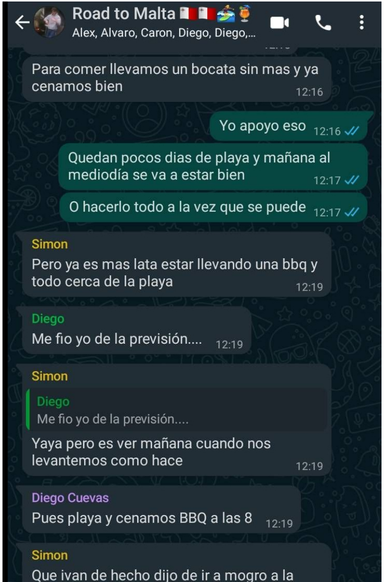
David Balbás[*†], **Daniel Collins**[‡], Serge Vaudenay[‡]

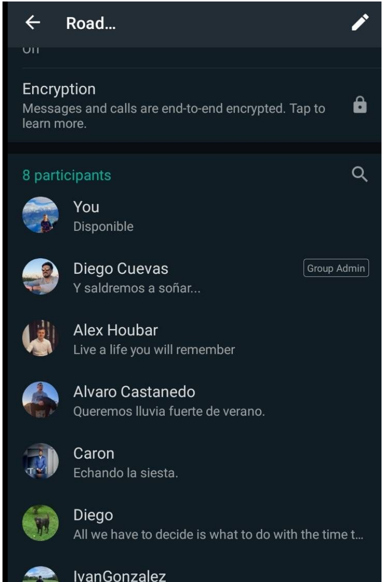
[*]IMDEA Software Institute, Madrid, Spain
[†]Universidad Politécnica de Madrid, Spain
[‡]EPFL, Lausanne, Switzerland

Swiss Crypto Day, 8th September 2023

(USENIX Security ’23... thank you David for your slides!)

Group Messaging?





thenewsminute.com

WhatsApp Group chats can be easily infiltrated, say researchers

Written by IANS

4–5 minutes

The WhatsApp attack on group chats takes advantage of a bug.

A team of German cryptographers has discovered flaws in WhatsApp's Group chats despite its end-to-end encryption, that makes it possible to infiltrate private group chats without admin permission.

According to a report in Wired.com, the cryptographers from Ruhr University Bochum in Germany announced this at the "Real World Crypto Security Conference in Zurich, Switzerland, on Wednesday.

"Anyone who controls the app's servers could insert new people into private group chats without needing admin permission," the report said, citing cryptographers.

thenewsminute.com

WhatsApp Group chats can be easily infiltrated, say researchers

Written by IANS

4–5 minutes

The WhatsApp attack on grou

A team of German cryptograph
WhatsApp's Group chats desp
makes it possible to infiltrate p
permission.

According to a report in Wired
University Bochum in German
Crypto Security Conference in

"Anyone who controls the app'
into private group chats withou
report said, citing cryptographe

ISG researchers discover vulnerabilities in Matrix protocol

Research and teaching > Departments and schools > Information Security > News

Date **28 September 2022**

A team of cryptographers – Dan Jones and Martin Albrecht (Royal Holloway), Sofía Celi (Brave) and Benjamin Dowling (University of Sheffield) has found several, practically-exploitable cryptographic vulnerabilities in the end-to-end encryption provided by the popular Matrix protocol and its flagship client implementation Element.

thenewsminute.com

WhatsApp Group chats can be easily infiltrated, say researchers

Written by IANS

4–5 minutes

The WhatsApp attack on group

A team of German cryptograph

WhatsApp's Group chats does

ISG researchers discover vulnerabilities in Matrix protocol

Research and teaching > Departments and schools > Information Security > News

Date 28 September 2022

echt
ng

end
d its

Three Lessons From Threema: Analysis of a Secure Messenger

Kenneth G. Paterson
Applied Cryptography Group, ETH Zurich

Matteo Scarlata
Applied Cryptography Group, ETH Zurich

Kien Tuong Truong
Applied Cryptography Group, ETH Zurich

Abstract

- **fine-grained perfect forward secrecy (PFS)**: compro-

Group membership?

Insecure group membership is a common design flaw in messaging.

Servers, and sometimes even users, may mount **attacks on group management**.

- Burgle into the group [RMS18]
- Censorship [BCG23]
- ...

Burgle into the Group Attack

- Reported by [RMS18] and affects WhatsApp groups.

Burgle into the Group Attack

- Reported by [RMS18] and affects WhatsApp groups.
- In messaging apps, a *server* (Meta) fowards messages between users.

Burgle into the Group Attack

- Reported by [RMS18] and affects WhatsApp groups.
- In messaging apps, a *server* (Meta) fowards messages between users.
- Suppose A is the group administrator and wants to add B to group G.

Burgle into the Group Attack

- Reported by [RMS18] and affects WhatsApp groups.
- In messaging apps, a *server* (Meta) fowards messages between users.
- Suppose A is the group administrator and wants to add B to group G.
- To do so, A sends a message $M = (A, G, name_A, ID_m, t_m, m = \{\texttt{add}, B\})$ to the server.

Burgle into the Group Attack

- Reported by [RMS18] and affects WhatsApp groups.
- In messaging apps, a *server* (Meta) fowards messages between users.
- Suppose A is the group administrator and wants to add B to group G.
- To do so, A sends a message $M = (A, G, name_A, ID_m, t_m, m = \{\texttt{add}, B\})$ to the server.
- The server then forwards it to all users.

Burgle into the Group Attack

- Reported by [RMS18] and affects WhatsApp groups.
- In messaging apps, a *server* (Meta) fowards messages between users.
- Suppose A is the group administrator and wants to add B to group G.
- To do so, A sends a message $M = (A, G, name_A, ID_m, t_m, m = \{\texttt{add}, B\})$ to the server.
- The server then forwards it to all users.
- **Problem:** M is not authenticated by A!

Burgle into the Group Attack

- Reported by [RMS18] and affects WhatsApp groups.
- In messaging apps, a *server* (Meta) fowards messages between users.
- Suppose A is the group administrator and wants to add B to group G.
- To do so, A sends a message $M = (A, G, name_A, ID_m, t_m, m = \{\texttt{add}, B\})$ to the server.
- The server then forwards it to all users.
- **Problem:** M is not authenticated by A!
- The server can trivially send $(A, ..., m = \{\texttt{add}, C\})$ instead!

Group Administration?

How meaningful is security if users can't trust/control group membership?

Group Administration?

How meaningful is security if users can't trust/control group membership?

Can we build an efficient solution for users to *administrate* groups securely?

This Work and Talk

- New **formalism** for groups (based on continuous group key agreement) with *cryptographic administrators*.

This Work and Talk

- New **formalism** for groups (based on continuous group key agreement) with *cryptographic administrators*.
- **Correctness and security notions** matching modern messaging standards (forward security, post-compromise security).

This Work and Talk

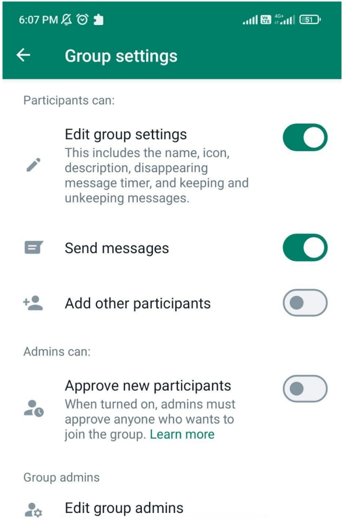
- New **formalism** for groups (based on continuous group key agreement) with *cryptographic administrators*.
- **Correctness and security notions** matching modern messaging standards (forward security, post-compromise security).
- Two modular, **provably-secure constructions**, IAS and DGS.

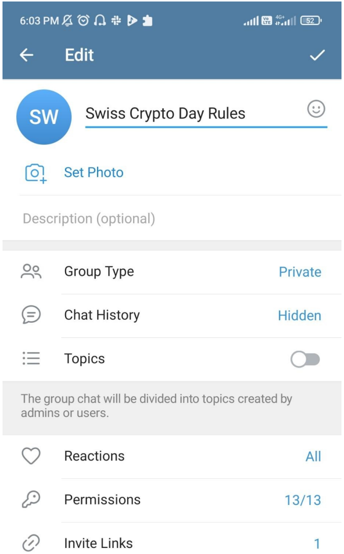
This Work and Talk

- New **formalism** for groups (based on continuous group key agreement) with *cryptographic administrators*.
- **Correctness and security notions** matching modern messaging standards (forward security, post-compromise security).
- Two modular, **provably-secure constructions**, IAS and DGS.
- Efficient **integration with MLS**, benchmarking, and admin extensions.

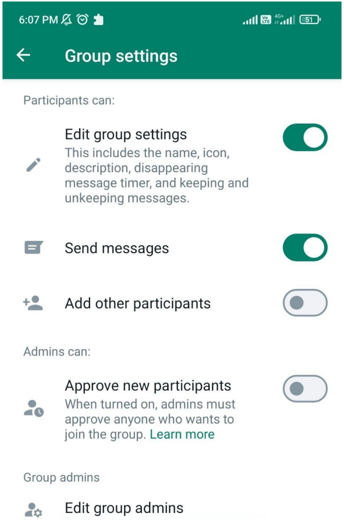
Group Messaging

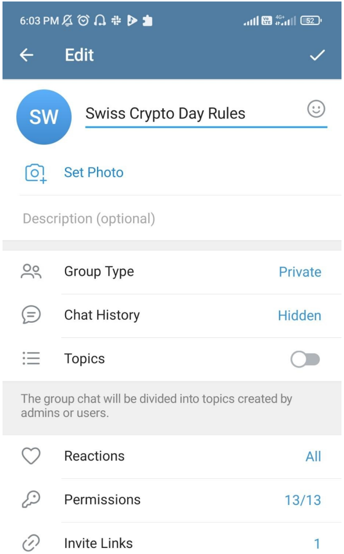
Group Administration?





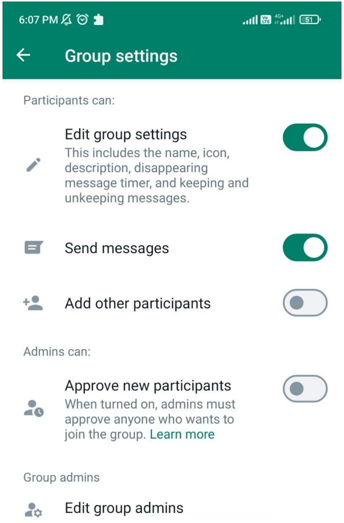
Group Administration?

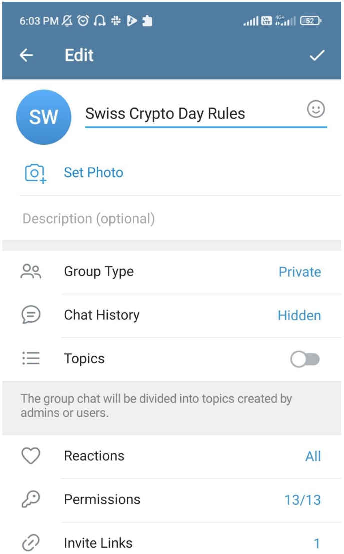




- Many features in practice!

Group Administration?





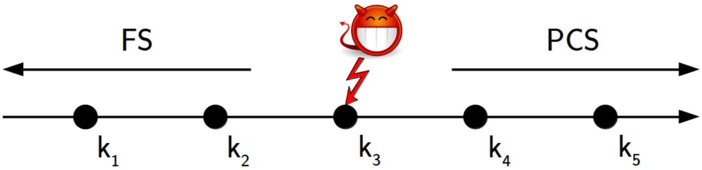
- Many features in practice!
- In this talk: only *admins* should be able to **add and remove users** (and admins).

Security of Group Messaging

- **As usual:** confidentiality, authentication, integrity.

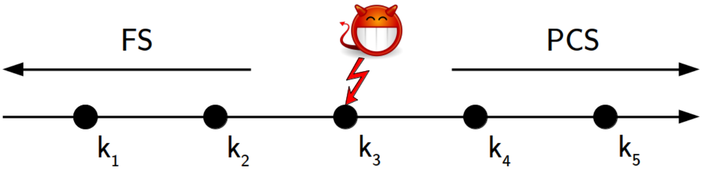
Security of Group Messaging

- **As usual:** confidentiality, authentication, integrity.
- **Forward security (FS):** past messages safe after compromise.
- **Post-compromise security (PCS):** self-healing via key updates.



Security of Group Messaging

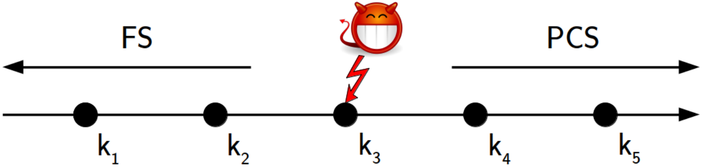
- **As usual:** confidentiality, authentication, integrity.
- **Forward security (FS):** past messages safe after compromise.
- **Post-compromise security (PCS):** self-healing via key updates.



- **Security game:** $\mathcal{A}$ controls network, can expose users [ACDT20, KPWK+21].

Security of Group Messaging

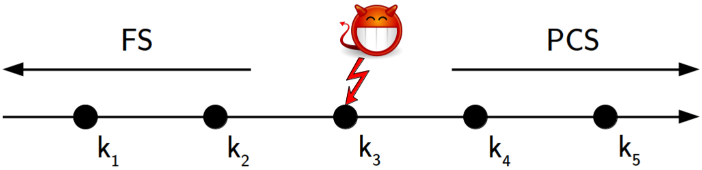
- **As usual:** confidentiality, authentication, integrity.
- **Forward security (FS):** past messages safe after compromise.
- **Post-compromise security (PCS):** self-healing via key updates.



- **Security game:** $\mathcal{A}$ controls network, can expose users [ACDT20, KPWK+21].
- **Group dynamics:** cryptographic adds and removes from group G.

Security of Group Messaging

- **As usual:** confidentiality, authentication, integrity.
- **Forward security (FS):** past messages safe after compromise.
- **Post-compromise security (PCS):** self-healing via key updates.



- **Security game:** $\mathcal{A}$ controls network, can expose users [ACDT20, KPWK+21].
- **Group dynamics:** cryptographic adds and removes from group G.
- ***Administration:*** only admins $G^* \subseteq G$ can make group changes.

Towards MLS (Message Layer Security)

Existing group messaging protocols:

- Pairwise two-party channels: $O(n)$ sender communication.

Towards MLS (Message Layer Security)

Existing group messaging protocols:

- Pairwise two-party channels: $O(n)$ sender communication.
- Sender Keys (WhatsApp, Signal, ...): $O(1)$ sender communication.

Towards MLS (Message Layer Security)

Existing group messaging protocols:

- Pairwise two-party channels: $O(n)$ sender communication.
- Sender Keys (WhatsApp, Signal, ...): $O(1)$ sender communication.
- Both have $O(n^2)$ *communication* for full key refreshes!

Towards MLS (Message Layer Security)

Existing group messaging protocols:

- Pairwise two-party channels: $O(n)$ sender communication.
- Sender Keys (WhatsApp, Signal, ...): $O(1)$ sender communication.
- Both have $O(n^2)$ *communication* for full key refreshes!
- *Cannot* easily scale to 1000s of users.

Towards MLS (Message Layer Security)

Existing group messaging protocols:

- Pairwise two-party channels: $O(n)$ sender communication.
- Sender Keys (WhatsApp, Signal, ...): $O(1)$ sender communication.
- Both have $O(n^2)$ *communication* for full key refreshes!
- *Cannot* easily scale to 1000s of users.

MLS:

- $O(\log n)$ fair-weather sender communication for key updates.

Towards MLS (Message Layer Security)

Existing group messaging protocols:

- Pairwise two-party channels: $O(n)$ sender communication.
- Sender Keys (WhatsApp, Signal, ...): $O(1)$ sender communication.
- Both have $O(n^2)$ *communication* for full key refreshes!
- *Cannot* easily scale to 1000s of users.

MLS:

- $O(\log n)$ fair-weather sender communication for key updates.
- Recently became IETF RFC 9420.

Towards MLS (Message Layer Security)

Existing group messaging protocols:

- Pairwise two-party channels: $O(n)$ sender communication.
- Sender Keys (WhatsApp, Signal, ...): $O(1)$ sender communication.
- Both have $O(n^2)$ *communication* for full key refreshes!
- *Cannot* easily scale to 1000s of users.

MLS:

- $O(\log n)$ fair-weather sender communication for key updates.
- Recently became IETF RFC 9420.
- Many features; a complex standard.

Towards MLS (Message Layer Security)

Existing group messaging protocols:

- Pairwise two-party channels: $O(n)$ sender communication.
- Sender Keys (WhatsApp, Signal, ...): $O(1)$ sender communication.
- Both have $O(n^2)$ *communication* for full key refreshes!
- *Cannot* easily scale to 1000s of users.

MLS:

- $O(\log n)$ fair-weather sender communication for key updates.
- Recently became IETF RFC 9420.
- Many features; a complex standard.
- Interest from academia and industry.

Key Agreement: (A-)CGKA

Recently popular formalism: **Continuous Group Key Agreement** (CGKA) [ACDT20].

Forms the basis of MLS (as TreeKEM).

Key Agreement: (A-)CGKA

Recently popular formalism: **Continuous Group Key Agreement** (CGKA) [ACDT20].
Forms the basis of MLS (as TreeKEM).

- Dynamic *secret* I known to members.

Key Agreement: (A-)CGKA

Recently popular formalism: **Continuous Group Key Agreement** (CGKA) [ACDT20].
Forms the basis of MLS (as TreeKEM).

- Dynamic *secret* I known to members.
- Members ID *propose* adds, removals, and key updates [AJM20, RFC9420].

Key Agreement: (A-)CGKA

Recently popular formalism: **Continuous Group Key Agreement** (CGKA) [ACDT20]. Forms the basis of MLS (as TreeKEM).

- Dynamic *secret* I known to members.
- Members ID *propose* adds, removals, and key updates [AJM20, RFC9420].
- Later, ID' *commits* several proposals and users then *process* the commit.

Key Agreement: (A-)CGKA

Recently popular formalism: **Continuous Group Key Agreement** (CGKA) [ACDT20]. Forms the basis of MLS (as TreeKEM).

- Dynamic *secret* I known to members.
- Members ID *propose* adds, removals, and key updates [AJM20, RFC9420].
- Later, ID' *commits* several proposals and users then *process* the commit.
- Can build *group messaging* using I (KEM/DEM-style with signatures for example).

Key Agreement: (A-)CGKA

Recently popular formalism: **Continuous Group Key Agreement** (CGKA) [ACDT20]. Forms the basis of MLS (as TreeKEM).

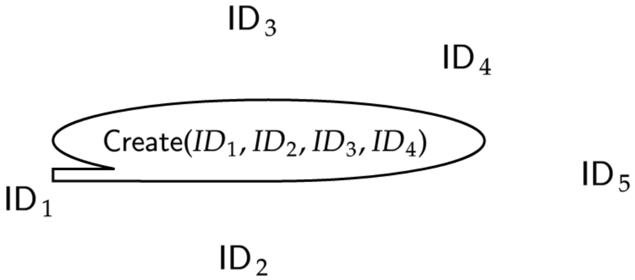
- Dynamic *secret* I known to members.
- Members ID *propose* adds, removals, and key updates [AJM20, RFC9420].
- Later, ID' *commits* several proposals and users then *process* the commit.
- Can build *group messaging* using I (KEM/DEM-style with signatures for example).

CGKA (simplified):

- $\mathsf{Init}(1^\lambda, ID)$
- $\mathsf{Create}(G) \rightarrow T$
- $\mathsf{Prop}(ID, \mathsf{type}) \rightarrow P$
- $\mathsf{Commit}(\vec{P}) \rightarrow T$
- $\mathsf{Proc}(T) \rightarrow I'$

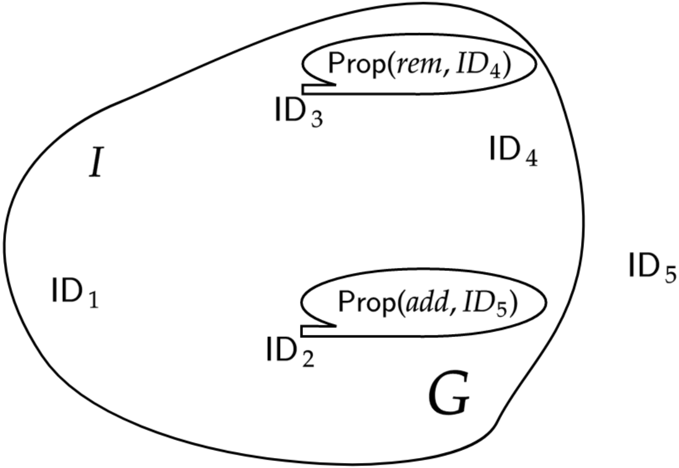
CGKA: Create

- ID_1 creates a group $G = \{ID_1, ID_2, ID_3, ID_4\}$.



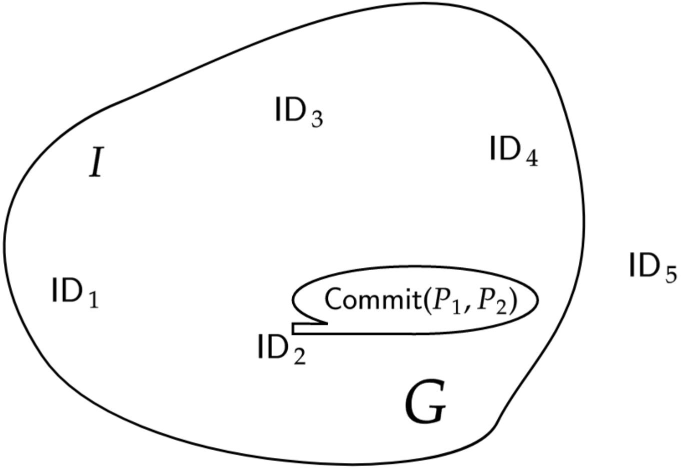
CGKA: Proposals

- ID_2 and ID_3 propose changes.



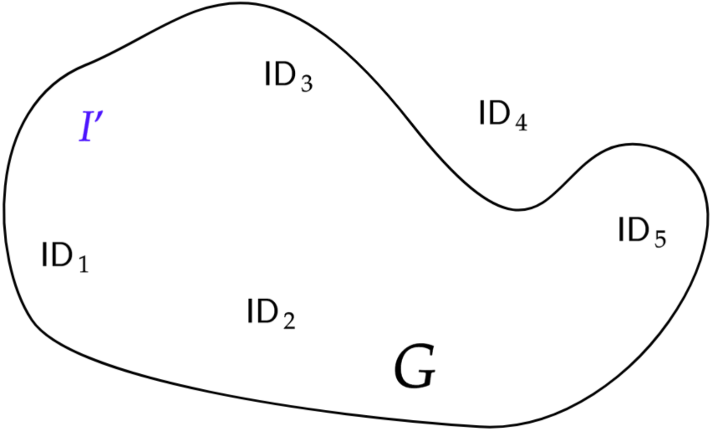
CGKA: Commit

- ID_2 commits both proposals.



CGKA: Process Changes

- The group evolves to a new *epoch* and I' is refreshed.



Key Agreement: (A-)CGKA

Administrated Continuous Group Key Agreement (A-CGKA).

- Dynamic *secret* I known to members.
- Members ID *propose* adds, removals, and key updates [AJM20, RFC9420].

  A-CGKA includes new proposal types: add/remove/update admin.
- Later, ID' *commits* several proposals and users then *process* the commit.

Key Agreement: (A-)CGKA

Administrated Continuous Group Key Agreement (A-CGKA).

- Dynamic *secret* I known to members.
- Members ID *propose* adds, removals, and key updates [AJM20, RFC9420].

  A-CGKA includes new proposal types: add/remove/update admin.
- Later, ID' *commits* several proposals and users then *process* the commit.

A-CGKA (simplified):

- $\mathsf{Init}(1^\lambda, ID)$
- $\mathsf{Create}(G, G^*) \rightarrow T$
- $\mathsf{Prop}(ID, \mathsf{type}) \rightarrow P$
- $\mathsf{Commit}(\vec{P}, \mathsf{com\text{-}type}) \rightarrow T$
- $\mathsf{Proc}(T) \rightarrow I'$

Key Agreement: (A-)CGKA

Administrated Continuous Group Key Agreement (A-CGKA).

- Dynamic *secret* I known to members.
- Members ID *propose* adds, removals, and key updates [AJM20, RFC9420].

  A-CGKA includes new proposal types: add/remove/update admin.
- Later, ID' *commits* several proposals and users then *process* the commit.

A-CGKA (simplified):

- $\mathsf{Init}(1^\lambda, ID)$
- $\mathsf{Create}(G, G^*) \rightarrow T$
- $\mathsf{Prop}(ID, \mathsf{type}) \rightarrow P$
- $\mathsf{Commit}(\vec{P}, \mathsf{com\text{-}type}) \rightarrow T$
- $\mathsf{Proc}(T) \rightarrow I'$

Administration security: Non-admins cannot commit (except updates and self-removes).

A-CGKA Correctness

$$\mathsf{CORR}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{corr}}}(1^{\lambda})$$

A-CGKA Correctness

$$\mathsf{CORR}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C_{corr}}}(1^\lambda)$$

- All users who transition to the same epoch have the same view of the group and key.

A-CGKA Correctness

$$\mathsf{CORR}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{corr}}}(1^{\lambda})$$

- All users who transition to the same epoch have the same view of the group and key.
- Only Proc modifies the group and key.

A-CGKA Correctness

$$\mathsf{CORR}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{corr}}}(1^{\lambda})$$

- All users who transition to the same epoch have the same view of the group and key.
- Only Proc modifies the group and key.
- Proc has its intended effect.

A-CGKA Correctness

$$\mathsf{CORR}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{corr}}}(1^\lambda)$$

- All users who transition to the same epoch have the same view of the group and key.
- Only Proc modifies the group and key.
- Proc has its intended effect.
- Admins always form a non-empty subset of the group.

A-CGKA Correctness

$$\mathsf{CORR}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{corr}}}(1^{\lambda})$$

- All users who transition to the same epoch have the same view of the group and key.
- Only Proc modifies the group and key.
- Proc has its intended effect.
- Admins always form a non-empty subset of the group.

$\mathsf{CheckSameGroupState}(\gamma_1, \gamma_2, \mathsf{gid})$

1 : **reward** $\gamma_1[\mathsf{gid}].\mathsf{k} \neq \gamma_2[\mathsf{gid}].\mathsf{k}$

2 : **reward** $\gamma_1[\mathsf{gid}].G \neq \gamma_2[\mathsf{gid}].G$

3 : **reward** $\gamma_1[\mathsf{gid}].G^* \neq \gamma_2[\mathsf{gid}].G^*$

A-CGKA Correctness

$$\mathsf{CORR}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{corr}}}(1^\lambda)$$

- All users who transition to the same epoch have the same view of the group and key.
- Only Proc modifies the group and key.
- Proc has its intended effect.
- Admins always form a non-empty subset of the group.

$\mathsf{CheckSameGroupState}(\gamma_1, \gamma_2, \mathsf{gid})$

1 : **reward** $\gamma_1[\mathsf{gid}].\mathsf{k} \neq \gamma_2[\mathsf{gid}].\mathsf{k}$

2 : **reward** $\gamma_1[\mathsf{gid}].G \neq \gamma_2[\mathsf{gid}].G$

3 : **reward** $\gamma_1[\mathsf{gid}].G^* \neq \gamma_2[\mathsf{gid}].G^*$

reward $\mathsf{props}(\mathsf{ST}[\mathsf{ID}], T) \neq \mathsf{T}[\mathsf{gid}, (t, c), \texttt{'vec'}, c']$

A-CGKA Correctness

$$\mathsf{CORR}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C_{corr}}}(1^{\lambda})$$

- All users who transition to the same epoch have the same view of the group and key.
- Only Proc modifies the group and key.
- Proc has its intended effect.
- Admins always form a non-empty subset of the group.

$\mathsf{CheckSameGroupState}(\gamma_1, \gamma_2, \mathsf{gid})$

1 : **reward** $\gamma_1[\mathsf{gid}].\mathsf{k} \neq \gamma_2[\mathsf{gid}].\mathsf{k}$
2 : **reward** $\gamma_1[\mathsf{gid}].G \neq \gamma_2[\mathsf{gid}].G$
3 : **reward** $\gamma_1[\mathsf{gid}].G^* \neq \gamma_2[\mathsf{gid}].G^*$

reward $\mathsf{props}(\mathsf{ST}[\mathsf{ID}], T) \neq \mathsf{T}[\mathsf{gid}, (t, c), \text{'}\mathtt{vec}\text{'}, c']$

reward $\neg(\emptyset \neq \gamma[\mathsf{gid}].G^* \subseteq \gamma[\mathsf{gid}].G)$

A-CGKA Security

$$\mathsf{KIND}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{cgka}},\mathsf{C}_{\mathsf{adm}},\mathsf{C}_{\mathsf{forgery}}}(1^\lambda)$$

A-CGKA Security

$$\mathsf{KIND}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{cgka}},\mathsf{C}_{\mathsf{adm}},\mathsf{C}_{\mathsf{forgery}}}(1^{\lambda})$$

- A key indistinguishability game.

A-CGKA Security

$$\mathsf{KIND}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{cgka}},\mathsf{C}_{\mathsf{adm}},\mathsf{C}_{\mathsf{forgery}}}(1^{\lambda})$$

- A key indistinguishability game.
- Formally captured in *cleanness predicates*.

A-CGKA Security

$$\mathsf{KIND}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C_{cgka}},\mathsf{C_{adm}},\mathsf{C_{forgery}}}(1^{\lambda})$$

- A key indistinguishability game.
- Formally captured in *cleanness predicates*.
- Cannot inject commits **even if non-admins are corrupted** (except for non-admin updates and self-removes).

A-CGKA Security

$$\mathsf{KIND}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{cgka}},\mathsf{C}_{\mathsf{adm}},\mathsf{C}_{\mathsf{forgery}}}(1^{\lambda})$$

- A key indistinguishability game.
- Formally captured in *cleanness predicates*.
- Cannot inject commits **even if non-admins are corrupted** (except for non-admin updates and self-removes).
- Security is restored after compromised users (and admins) *update* or are *removed* (PCS).

A-CGKA Security

$$\mathsf{KIND}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{cgka}},\mathsf{C}_{\mathsf{adm}},\mathsf{C}_{\mathsf{forgery}}}(1^\lambda)$$

$\mathsf{C}_{\mathsf{cgka\text{-}opt}}$ $\mathsf{C}_{\mathsf{adm\text{-}opt}}$ $\mathsf{C}_{\mathsf{forgery}}$

- A key indistinguishability game.
- Formally captured in *cleanness predicates*.
- Cannot inject commits **even if non-admins are corrupted** (except for non-admin updates and self-removes).
- Security is restored after compromised users (and admins) *update* or are *removed* (PCS).

A-CGKA Security

$$\mathsf{KIND}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{cgka}},\mathsf{C}_{\mathsf{adm}},\mathsf{C}_{\mathsf{forgery}}}(1^\lambda)$$

- A key indistinguishability game.
- Formally captured in *cleanness predicates*.
- Cannot inject commits **even if non-admins are corrupted** (except for non-admin updates and self-removes).
- Security is restored after compromised users (and admins) *update* or are *removed* (PCS).

$\mathsf{C}_{\mathsf{cgka\text{-}opt}}$ $\mathsf{C}_{\mathsf{adm\text{-}opt}}$ $\mathsf{C}_{\mathsf{forgery}}$

$\mathcal{O}^{\mathsf{Inject}}(\mathsf{ID}, m, t_a)$

1 : **require** $\mathsf{C}_{\mathsf{adm}} \wedge (\mathsf{ep}[\mathsf{ID}] = (\cdot, t_a)) \wedge (t_a \neq -1)$
2 : **require** $(m, \cdot) \notin \mathsf{T}$ // external forgery
3 : $(\gamma, \perp) \leftarrow \mathsf{proc}(\mathsf{ST}[\mathsf{ID}], m)$
4 : **if** $\mathsf{C}_{\mathsf{forgery}}$
5 : $\quad$ $\mathsf{forged} \leftarrow \mathsf{true}$ // successful forgery
6 : $\quad$ **return** b // adversary wins
7 : **else return** $\perp$

A-CGKA Security

$$\mathsf{KIND}^{\mathcal{A}}_{\mathsf{(A)\text{-}CGKA},\mathsf{C}_{\mathsf{cgka}},\mathsf{C}_{\mathsf{adm}},\mathsf{C}_{\mathsf{forgery}}}(1^\lambda)$$

- A key indistinguishability game.
- Formally captured in *cleanness predicates*.
- Cannot inject commits **even if non-admins are corrupted** (except for non-admin updates and self-removes).
- Security is restored after compromised users (and admins) *update* or are *removed* (PCS).

$\mathsf{C}_{\mathsf{cgka\text{-}opt}}$ $\mathsf{C}_{\mathsf{adm\text{-}opt}}$ $\mathsf{C}_{\mathsf{forgery}}$

$\mathcal{O}^{\mathsf{Inject}}(\mathsf{ID}, m, t_a)$

1 : **require** $\mathsf{C}_{\mathsf{adm}} \wedge (\mathsf{ep}[\mathsf{ID}] = (\cdot, t_a)) \wedge (t_a \neq -1)$
2 : **require** $(m, \cdot) \notin \mathsf{T}$ // external forgery
3 : $(\gamma, \perp) \leftarrow \mathsf{proc}(\mathsf{ST}[\mathsf{ID}], m)$
4 : **if** $\mathsf{C}_{\mathsf{forgery}}$
5 : $\mathsf{forged} \leftarrow \mathsf{true}$ // successful forgery
6 : **return** b // adversary wins
7 : **else return** $\perp$

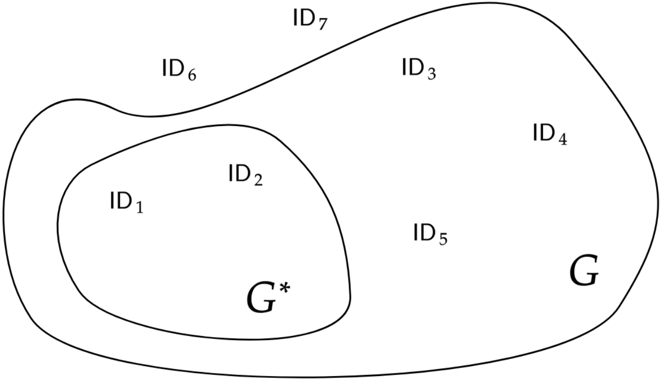
$\mathsf{hasUpd}_{\mathsf{std}}$ $\mathsf{hasUpd}_{\mathsf{adm}}$

Protocols for Secure Administration

We introduce **IAS** (*Individual Admin Signatures*) and **DGS** (*Dynamic Group Signature*).

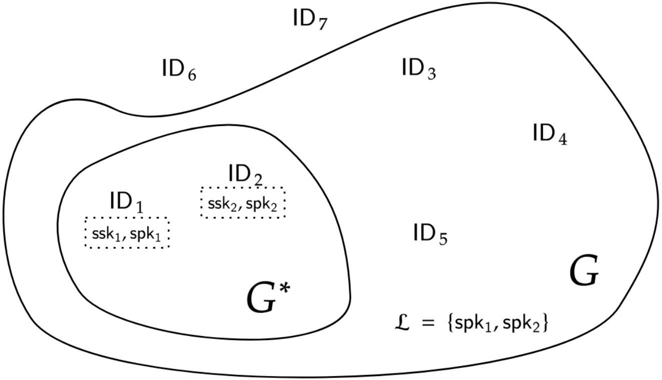
- Modular.
- Authenticate administrators (with different efficiency trade-offs).
- Allow for admin key refresh for PCS and FS.

Individual Admin Signatures (IAS)



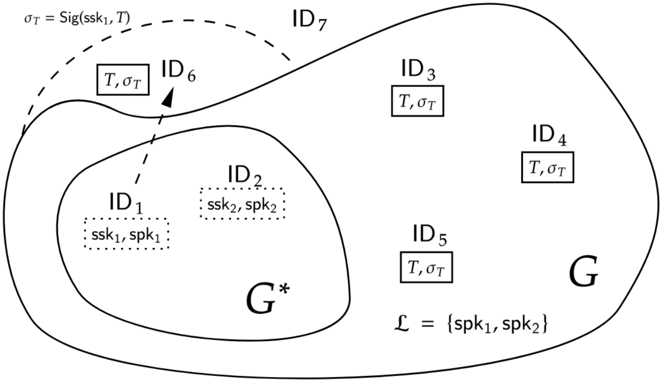
- We construct A-CGKA on top of any CGKA.
- Based on signatures.

IAS



- Admins have individual signature key pairs $(\mathsf{ssk}, \mathsf{spk})$.
- Users keep an admin list $\mathcal{L}$.

IAS: Add Participant



- Admin signs commit T with $\mathsf{ssk}_1 \longrightarrow \sigma_T$.
- Users verify σ_T with spk_1 from $\mathcal{L}$.

On IAS

- IAS is simple and efficient!

On IAS

- IAS is simple and efficient!
- We prove IAS secure in our model.

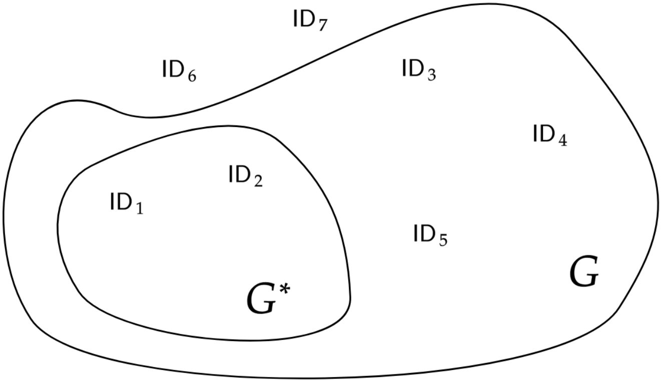
On IAS

- IAS is simple and efficient!
- We prove IAS secure in our model.
- (Informal:) For an adversary that makes at most q oracle queries, IAS is $(q \cdot \epsilon_F + \epsilon_{CGKA} + q^2 \cdot \epsilon_{Sig})$-secure for PRF F, CGKA $CGKA$ and SUF-CMA signature scheme Sig.

On IAS

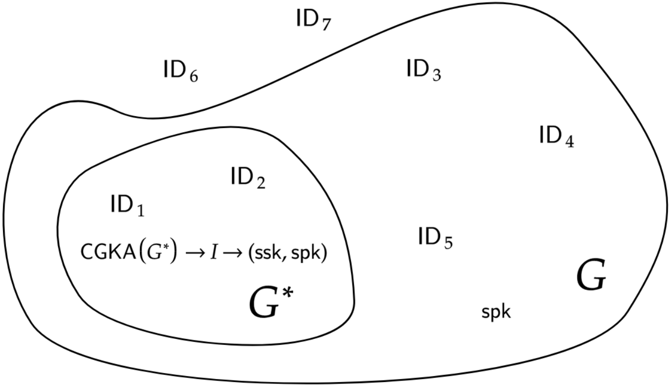
- IAS is simple and efficient!
- We prove IAS secure in our model.
- (Informal:) For an adversary that makes at most q oracle queries, IAS is $(q \cdot \epsilon_F + \epsilon_{CGKA} + q^2 \cdot \epsilon_{Sig})$-secure for PRF F, CGKA $CGKA$ and SUF-CMA signature scheme Sig.
- Can use forward-secure signatures for better (optimal) forward security.

Dynamic Group Signature (DGS)



- In DGS, all admins in G^* use the *same* signature key pair.
- Built from two CGKAs: the *core* CGKA $CGKA$ and the *admin* CGKA $CGKA^*$.

DGS



- Admin operations are managed through G^*.
- New admin public keys *spk* are signed under the old key.

On DGS

- (Conceptually) simple.

On DGS

- (Conceptually) simple.
- Can use different core and admin CGKAs.

On DGS

- (Conceptually) simple.
- Can use different core and admin CGKAs.
    - Could be useful for avoiding MLS's robustness issues in the admin CGKA.

On DGS

- (Conceptually) simple.
- Can use different core and admin CGKAs.
  - Could be useful for avoiding MLS's robustness issues in the admin CGKA.
- The admins can be *private* depending on the admin CGKA.

On DGS

- (Conceptually) simple.
- Can use different core and admin CGKAs.
  - Could be useful for avoiding MLS's robustness issues in the admin CGKA.
- The admins can be *private* depending on the admin CGKA.
- We prove security assuming the underlying CGKAs are secure in the ROM.

On DGS

- (Conceptually) simple.
- Can use different core and admin CGKAs.
  - Could be useful for avoiding MLS's robustness issues in the admin CGKA.
- The admins can be *private* depending on the admin CGKA.
- We prove security assuming the underlying CGKAs are secure in the ROM.
- (Informal:) For an adversary that makes at most q/q_{RO} oracle/RO queries, DGS is $(q \cdot \epsilon_F + \epsilon_{CGKA} + q \cdot \epsilon_{Sig} + q \cdot q_{RO} \cdot \epsilon_{cgka^*} + q \cdot 2^{-\lambda})$-secure for PRF F, RO H CGKA $CGKA$ and SUF-CMA signature scheme Sig.

Practical Administration for MLS

We also **integrate an IAS-based solution into MLS:**

Practical Administration for MLS

We also **integrate an IAS-based solution into MLS:**

- Leverages MLS' key credentials.

Practical Administration for MLS

We also **integrate an IAS-based solution into MLS:**

- Leverages MLS' key credentials.
- Extended proposal types.

Practical Administration for MLS

We also **integrate an IAS-based solution into MLS:**

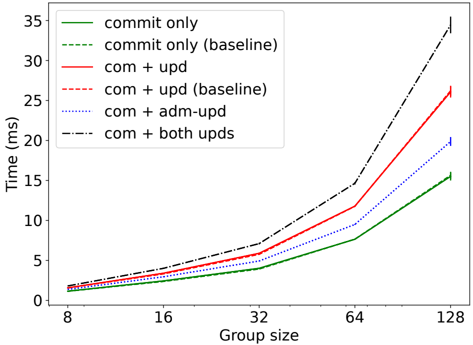
- Leverages MLS' key credentials.
- Extended proposal types.
- Could be integrated as an *MLS extension*.

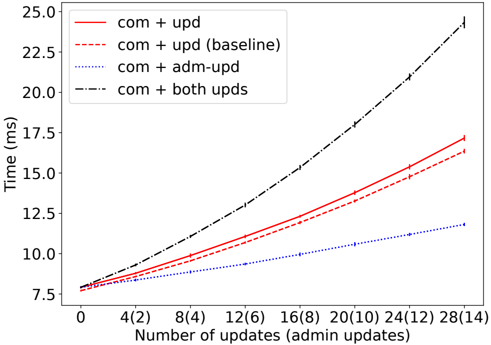
Practical Administration for MLS

We also **integrate an IAS-based solution into MLS:**

- Leverages MLS' key credentials.
- Extended proposal types.
- Could be integrated as an *MLS extension*.
- **Minimal overhead** (from benchmarking):
  - We forked CISCO's golang MLS implementation.
  - Benchmarking setup: 11th Gen Intel i5-1135G7, 16GB RAM.
  - Operations are executed by a single party.

Benchmarking (commits)





- upd: $|G|/4$ updates; adm-upd: $|G|/8$ admin updates.
- *Less than 20%* overhead when $|G|/8$ admins update simultaneously.
- *Additional communication* $< 3\%$ for $|G| = 128$ members.
- Overhead comes from admins performing CGKA updates.

Remarks on Performance

- IAS admin overhead:
  - Admin proposal: key pair generation and signing.
  - Commit and process: verifying $\leq |G^*|$ signatures.
  - In MLS messages are signed anyway!

Remarks on Performance

- IAS admin overhead:
  - Admin proposal: key pair generation and signing.
  - Commit and process: verifying $\leq |G^*|$ signatures.
  - In MLS messages are signed anyway!
- DGS admin overhead:
  - Depends on the admin CGKA; can be up to linear in $|G^*|$.
  - Generally efficient for standard users: admin-only commits could be just a new public key and signature verification if commits are splittable.

Remarks on Performance

- IAS admin overhead:
  - Admin proposal: key pair generation and signing.
  - Commit and process: verifying $\leq |G^*|$ signatures.
  - In MLS messages are signed anyway!
- DGS admin overhead:
  - Depends on the admin CGKA; can be up to linear in $|G^*|$.
  - Generally efficient for standard users: admin-only commits could be just a new public key and signature verification if commits are splittable.
- Admin operations may be less frequent than regular ones.

Remarks on Performance

- IAS admin overhead:
  - Admin proposal: key pair generation and signing.
  - Commit and process: verifying $\leq |G^*|$ signatures.
  - In MLS messages are signed anyway!
- DGS admin overhead:
  - Depends on the admin CGKA; can be up to linear in $|G^*|$.
  - Generally efficient for standard users: admin-only commits could be just a new public key and signature verification if commits are splittable.
- Admin operations may be less frequent than regular ones.
- Forward-secure signatures: constant asymptotic overhead but non-standard.

Admin Extensions and Future Work

Our work serves to initiate the study of provably-secure administration:

- Admins beyond CGKA.
  - Signal private group system [CPZ20]

Admin Extensions and Future Work

Our work serves to initiate the study of provably-secure administration:

- Admins beyond CGKA.
  - Signal private group system [CPZ20]
- Private admins.

Admin Extensions and Future Work

Our work serves to initiate the study of provably-secure administration:

- Admins beyond CGKA.
  - Signal private group system [CPZ20]
- Private admins.
- External admins.

Admin Extensions and Future Work

Our work serves to initiate the study of provably-secure administration:

- Admins beyond CGKA.
  - Signal private group system [CPZ20]
- Private admins.
- External admins.
- Threshold admins.

Admin Extensions and Future Work

Our work serves to initiate the study of provably-secure administration:

- Admins beyond CGKA.
  - Signal private group system [CPZ20]
- Private admins.
- External admins.
- Threshold admins.
- Advanced admins:

Admin Extensions and Future Work

Our work serves to initiate the study of provably-secure administration:

- Admins beyond CGKA.
  - Signal private group system [CPZ20]
- Private admins.
- External admins.
- Threshold admins.
- Advanced admins:
  - Muting admins.

Admin Extensions and Future Work

Our work serves to initiate the study of provably-secure administration:

- Admins beyond CGKA.
  - Signal private group system [CPZ20]
- Private admins.
- External admins.
- Threshold admins.
- Advanced admins:
  - Muting admins.
  - Hierarchical admins.

Admin Extensions and Future Work

Our work serves to initiate the study of provably-secure administration:

- Admins beyond CGKA.
  - Signal private group system [CPZ20]
- Private admins.
- External admins.
- Threshold admins.
- Advanced admins:
  - Muting admins.
  - Hierarchical admins.
- Preventing insider attacks with trusted admins.

Conclusion

- Securing *membership* is essential in group messaging security.
- We treat cryptographic *administration* as a first-class (provable) security property.
- Can be implemented with small overhead.
- Modular solutions *readily compatible* with CGKAs and MLS.

Conclusion

- Securing *membership* is essential in group messaging security.
- We treat cryptographic *administration* as a first-class (provable) security property.
- Can be implemented with small overhead.
- Modular solutions *readily compatible* with CGKAs and MLS.

Thank you!

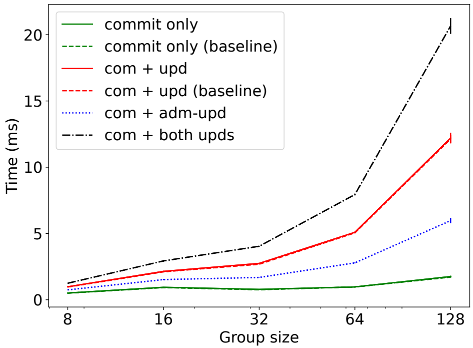
ia.cr/2022/1411

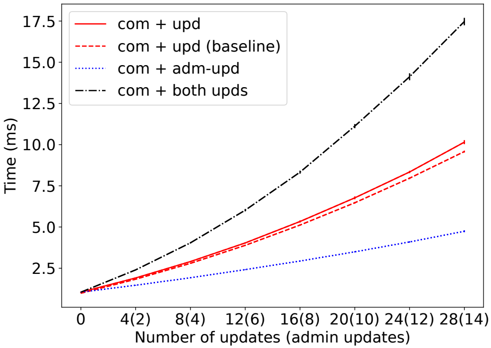
david.balbas@imdea.org
daniel.collins@epfl.ch

Backup Slides

Some additional slides follow.

Benchmarking (process)





- Comparable behaviour to commits.

On PKI

- We assume an incorruptible PKI.
- This follows previous work, except [AJM22] and [ACDT21] that allow malicious key uploads.
- Naturally, no security guarantees can be provided for users associated with these keys.
- All users always are assumed to share the same view of the PKI in all works we are aware of.