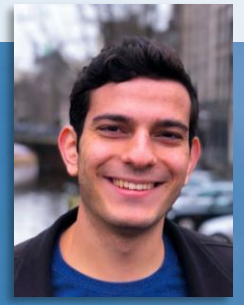
Resolving Concurrency in Group Ratcheting Protocols

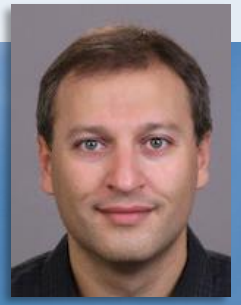
IACR RWC 2021

2021-01-07

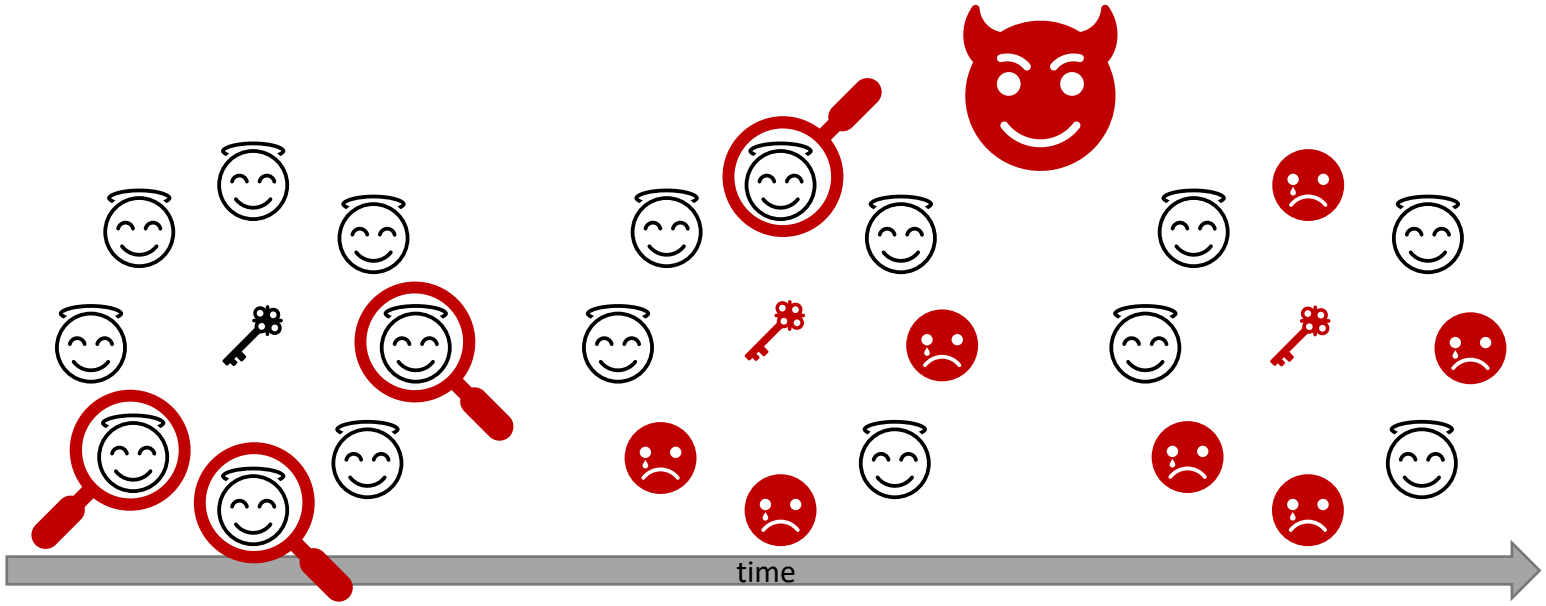
Cryptography Group
New York University
Alexander Bienstock, Yevgeniy Dodis,

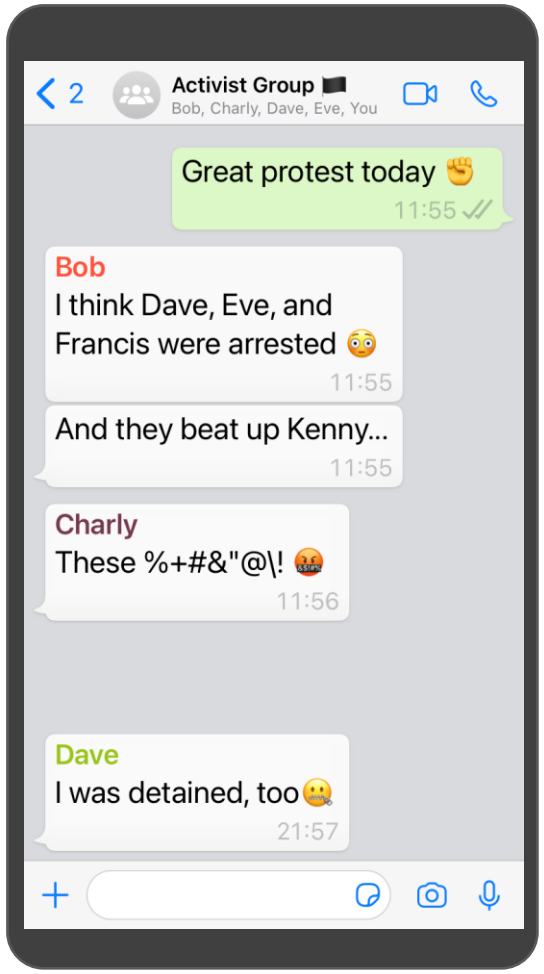
Horst Görtz Institute for IT Security
Chair for Network and Data Security
Ruhr University Bochum
Paul Rösler



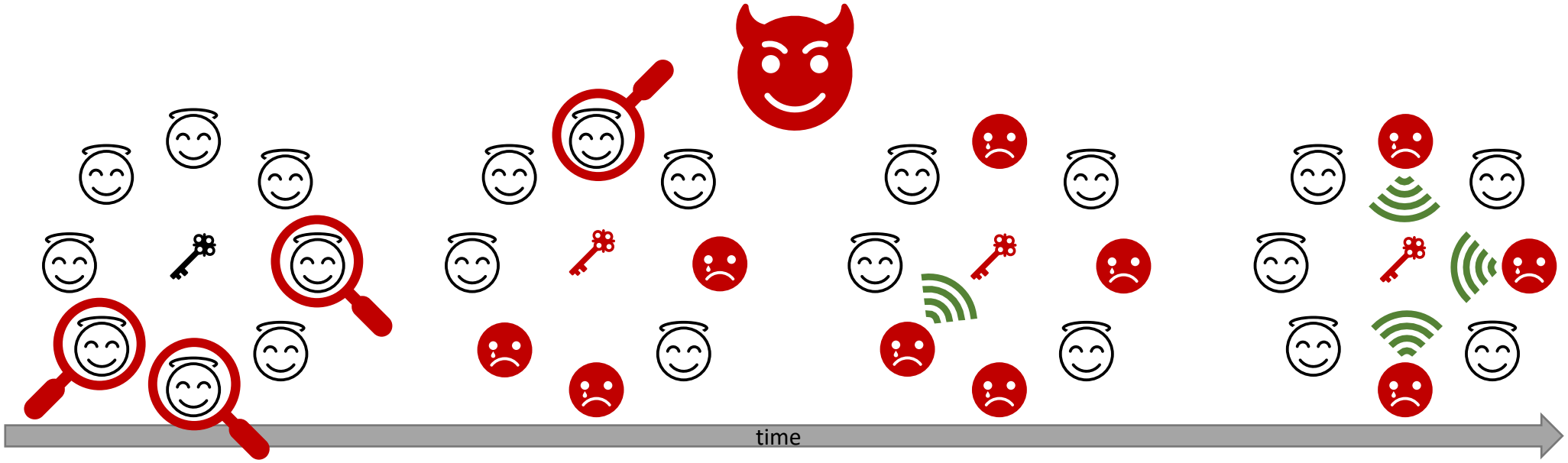


(Concurrent) Group Ratcheting

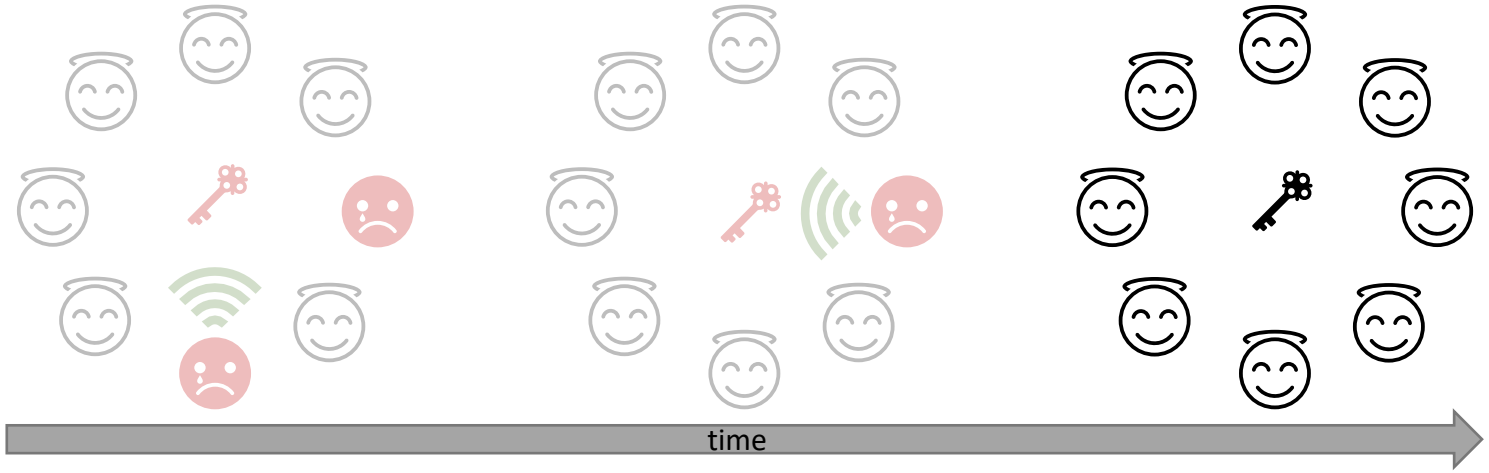


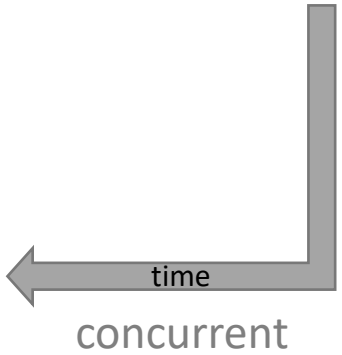


(Concurrent) Group Ratcheting



- Recovery (**PCS***):
  - After active exposure
  - Generate new secrets
  - Share public values
- Ideal protocol:
  1. Quick recovery
  2. Small shares
  3. Concurrency

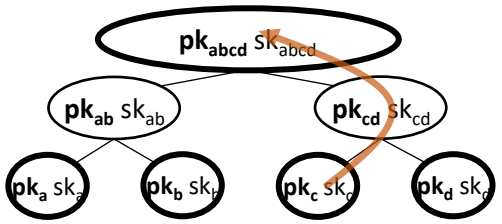


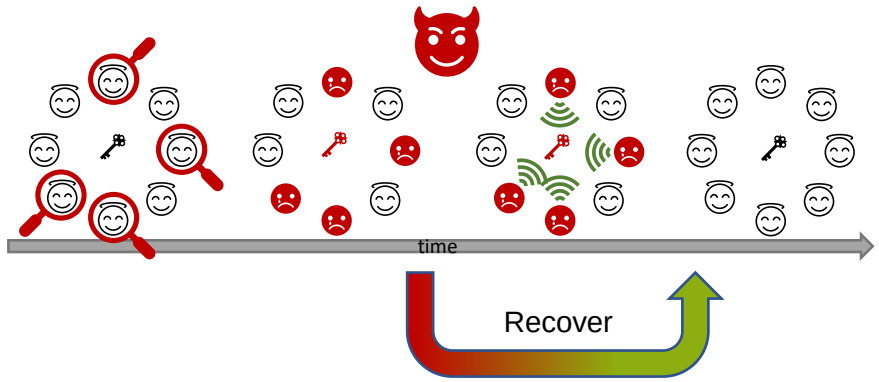


* PCS: Post-compromise security

Previous Work: What's the Problem?

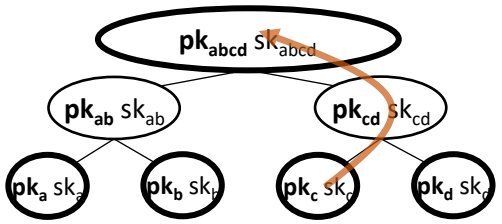
- Most promising idea: tree-based
  - Users: leaves
  - Secrets: nodes on path to root
  - Group key: root
  - Recovery: update secrets on path

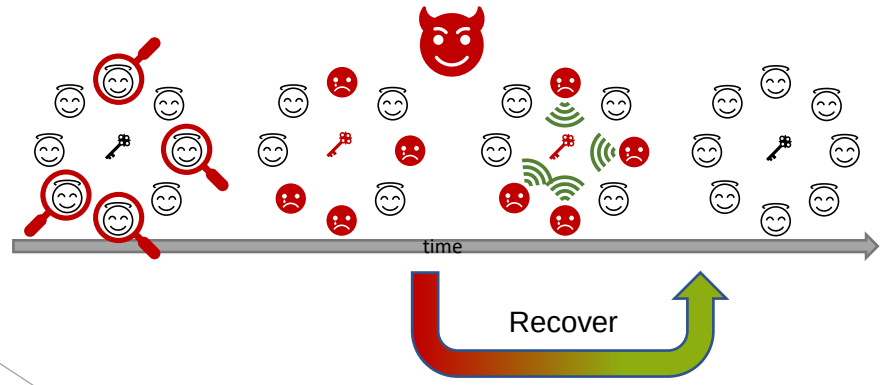


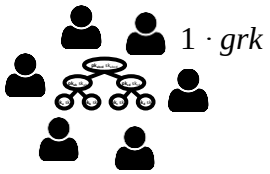


Previous Work: What's the Problem?

- Most promising idea: tree-based
  - Users: leaves
  - Secrets: nodes on path to root
  - Group key: root
  - Recovery: update secrets on path
- Tree-based protocols
  - [CCG+'18,BBR'18,ACDT'20,ACC+'19,ACJM'20,…]



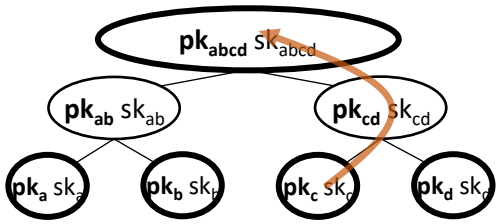


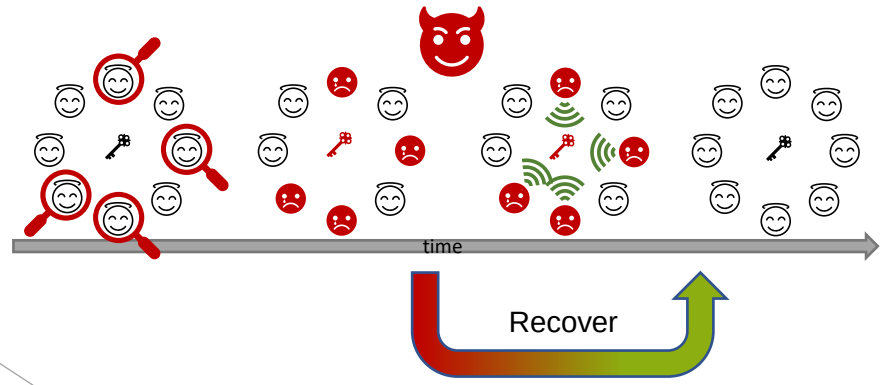


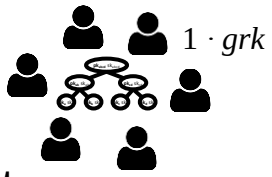
PCS	Overhead	Concurrency
✔	O(log n)	✘

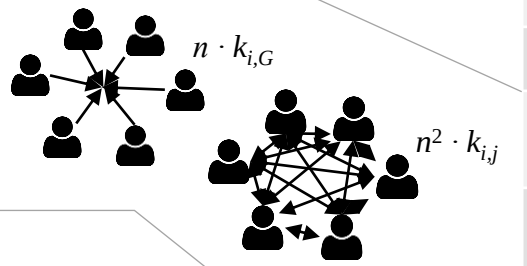
Previous Work: What's the Problem?

- Most promising idea: tree-based
  - Users: leaves
  - Secrets: nodes on path to root
  - Group key: root
  - Recovery: update secrets on path
- Tree-based protocols
  - [CCG+'18,BBR'18,ACDT'20,ACC+'19,ACJM'20,…]
  - Merging paths without PCS [Weidner'19]
  - IETF Messaging Layer Security (MLS) propose-then-commit
- Alternatives
  - Forward-secure hash chain [WhatsApp]
  - Parallel pair-wise communication [Signal,WKHB'20]





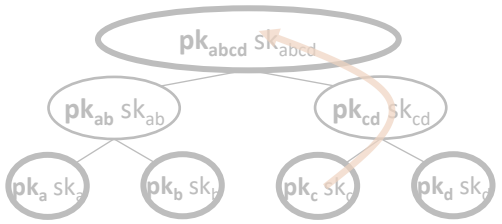


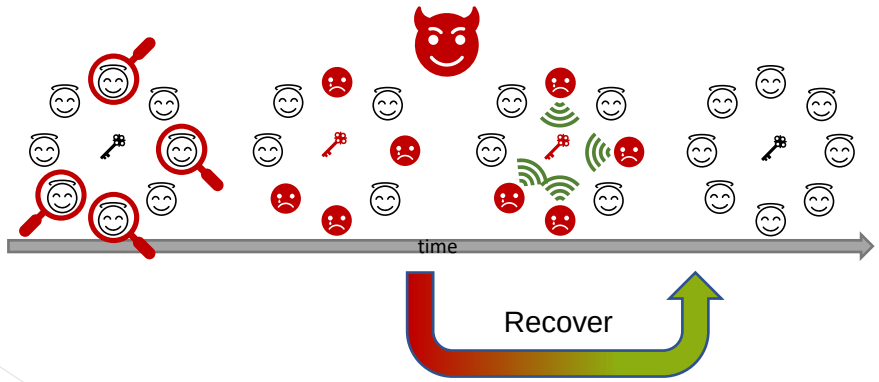


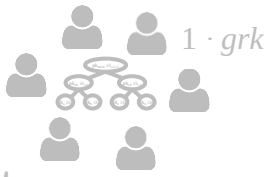
PCS	Overhead	Concurrency
✔	O(log n)	✘
(✘)	O(log n)	✔
(✔)	**O(n)**	(✔)
✘	O(1)	✔
✔	**O(n)**	✔

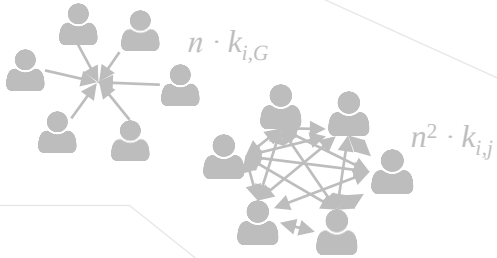
Previous Work: What's the Problem?

- Most promising idea: tree-based
  - Users: leaves
  - Secrets: nodes on path to root
  - Group key: root
  - Recovery: update secrets on path
- Tree-based protocols
  - [CCG+'18,BBR'18,ACDT'20,ACC+'19,ACJM'20,…]
  - Merging paths without PCS [Weidner'19]
  - IETF Messaging Layer Security (MLS) propose-then-commit
- Alternatives
  - Forward-secure hash chain [WhatsApp]
  - Parallel pair-wise communication [Signal,WKHB'20]
- **Is this trade-off inherent?**
  - Minimal communication overhead when **t** users recover (PCS) concurrently?
  - Group ratcheting protocol that reaches this minimum?





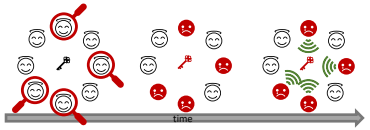




PCS	Overhead	Concurrency
✔	O(log n)	✘
(✘)	O(log n)	✔
(✔)	**O(n)**	(✔)
✘	O(1)	✔
✔	**O(n)**	✔
✔	**?**	✔

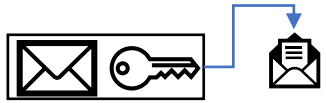
Our Results*: t-Concurrency

Theoretically Minimal Overhead: Ω(t)

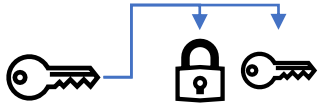


Round-based, Symbolic model

- Full asynchrony ⇒ worse overhead
- Fixed computation rules, no bit representation



- Models building blocks covering *all* tools used so far
  → Dual PRFs, Broadcast encryption, HIBE, …
- Excludes *exotic* tools (e.g., multi-party NIKE)



…

We prove for any protocol in this model …

- … combining these building blocks arbitrarily …
- … to handle t concurrent recoveries …
- … every recovery message contains Ω(t) distinct shares
- Proof idea: Each user must reply individually to last round's shares
  → Preparation useless

…

Protocol: O(t+t·log(n/t))

Our Results*: t-Concurrency

Theoretically Minimal Overhead: Ω(t)

Round-based, Symbolic model

- Full asynchrony ⇒ worse overhead
- Fixed computation rules, no bit representation

- Models building blocks covering *all* tools used so far
  → Dual PRFs, Broadcast encryption, HIBE, …
- Excludes *exotic* tools (e.g., multi-party NIKE)

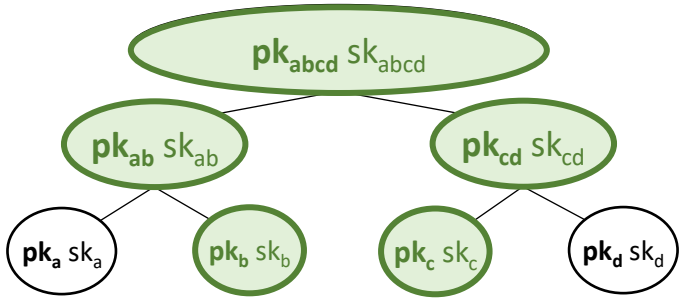
We prove for any protocol in this model …

- … combining these building blocks arbitrarily …
- … to handle t concurrent recoveries …
- … every recovery message contains Ω(t) distinct shares
- Proof idea: Each user must reply individually to last round's shares
  → Preparation useless

Protocol: O(t+t·log(n/t))

1. Users only heal their leaf secret when sending (not full path)
   → O(t)
2. … and help previous senders by healing remaining path secrets
   → O(log(n/t)) per sender in last round

Concurrent recovery ✔



Yes, this trade-off is inherent (lower bound), but not as bad as it seemed (upper bound).

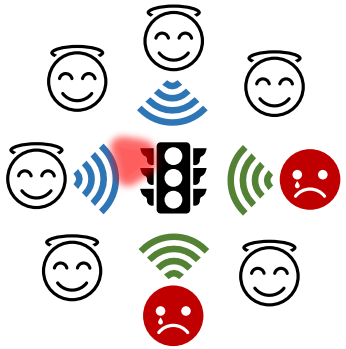
MLS: Success and Open Problems

- Great support from industry
- Pushed academic research
- Scalable performance with reasonable security

R. Barnes
Cisco
B. Beurdouche
Inria
J. Millican
Facebook
E. Omara
Google
K. Cohn-Gordon
University of Oxford
R. Robert
Wire
December 22, 2020

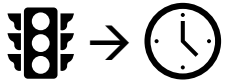
Propose-then-commit recovery:

- Reveals meta data
  - → Type and context of messages (direct vs. group; which group)
  - → Coordination (propose vs. commit for recovery)
- Server coordinates recovery
  - → Federated settings



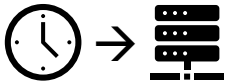
Our solutions:

- Hides meta data
  - → All messages look identical (except for size → use padding)
  - → No active coordination (only round scheduling)
- Steps towards avoiding central services
  - → Active coordination to round scheduling



Remaining questions:

- Equip our protocol with practical features (dynamic groups, malicious insiders, etc.)
- Only rely on message delivery (= no central service needed)



- Determine further fundamental limits of group ratcheting
- …

PCS	Overhead (t-concurrency)	Concurrency
✔	O(log n)	✘
✔	**O(n)**	✔
✔	**Ω(t) < x < O(t+t·log(n/t))**	✔

@roeslpa

Full details & formal proofs: ia.cr/2020/1171