MLS protocol

open questions

IETF 106 Singapore

Summary

- RTreeKEM
- Server assist
- Send from outside

RTreeKEM

Extension to TreeKEM

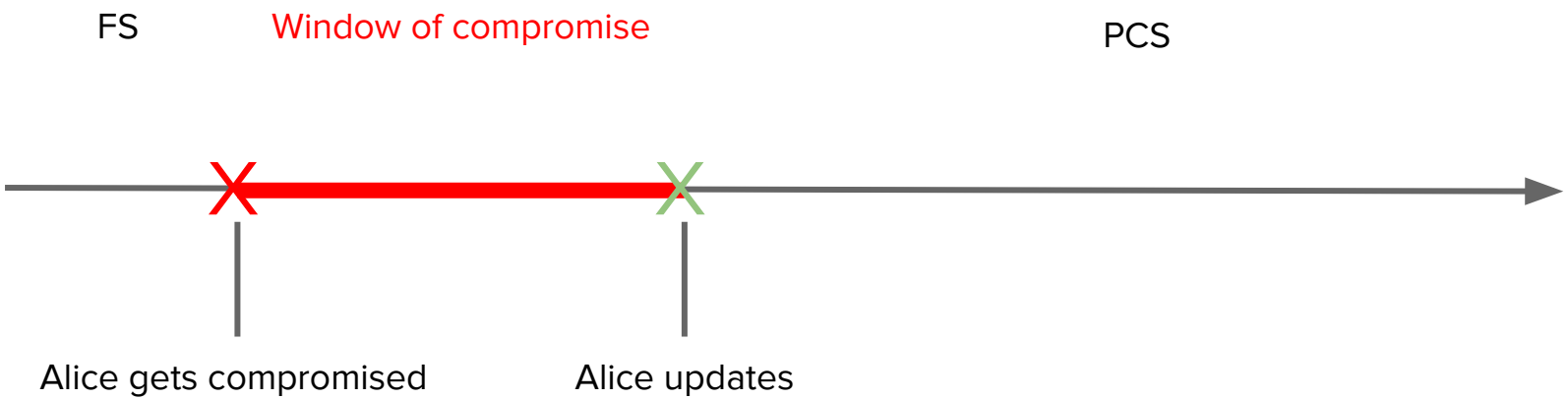
Proposal by Joël Alwen , Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis:

Re-randomized TreeKEM, aka RTreeKEM

https://eprint.iacr.org/2019/1189

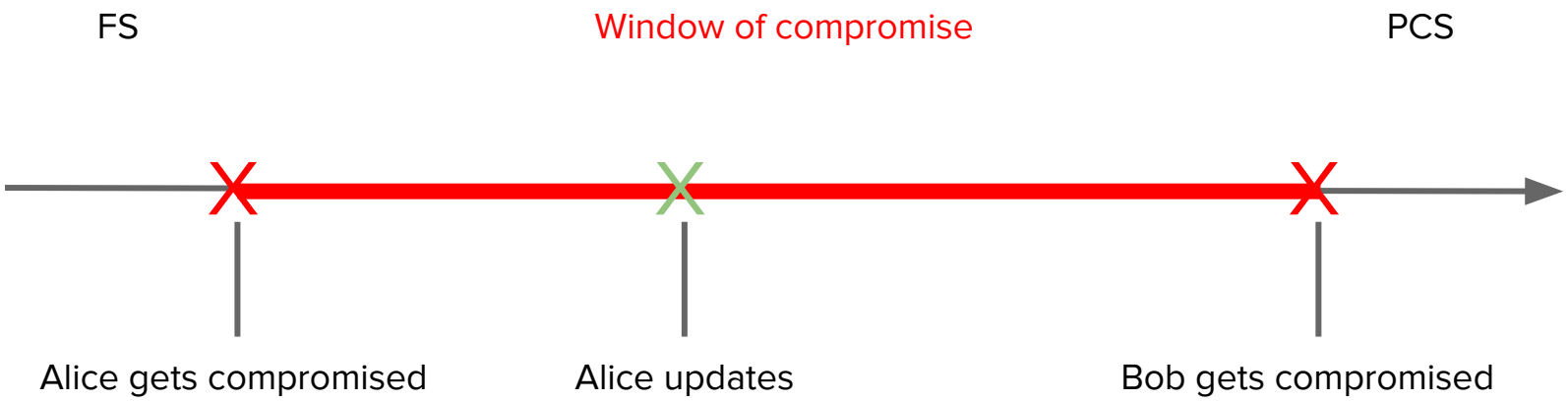
Threat model in TreeKEM

TreeKEM considers the compromise of 1 group member:



Threat model in RTreeKEM

RTreeKEM considers the compromise of 2 or more group members:



Worst case scenario

- Alice and Bob are siblings
- Alice gets compromised very on in the lifetime of a group (before members have time to update)
- Bob is a passive member, i.e. never sends updates
- Bob gets compromised towards the end of the lifetime of the group
- The confidentiality of the group is broken (is it technically FS or PCS? Let academia decide)

How does it work?

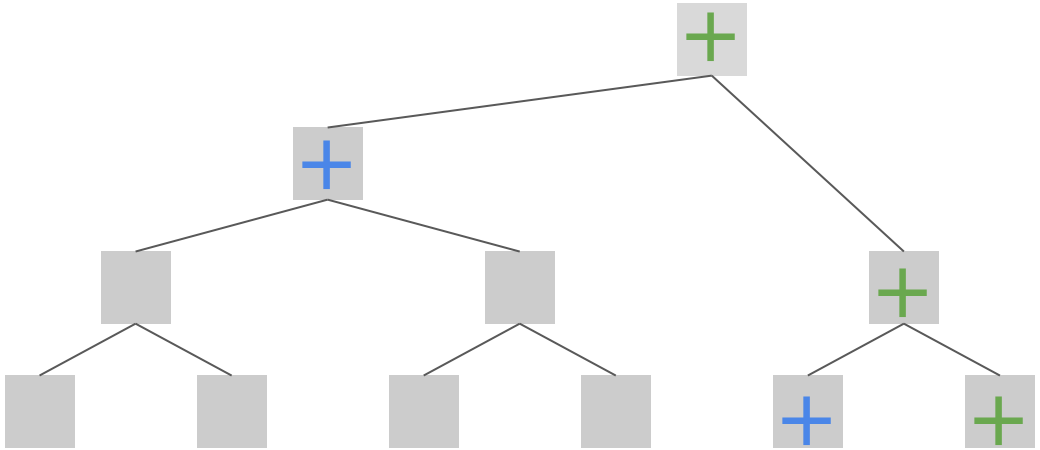
New crypto: DH with deltas (UPKE)

- A has a key pair (skA, pk(skA))
- B chooses delta and encrypts it under pkA and sends it to A
- B can compute pk(skA + delta) and advertise it without needing any response from A
- A updates its key to (skA + delta, pk(skA + delta)

How does it work?

Advantage:

If a member does an update, it can not only introduce freshness on its direct path, but also on its co-path



Pros

- Extending the threat model makes sense
- Improves things if members are passive (but provides no incentive to be passive)
- The overhead seams tolerable

Cons

- UPKE is not your usual crypto: multiplication of two private keys
- This is not well studied yet
- In the case of Curve25519: clamped private keys, what is the security level?
- None of the standard crypto libraries support this (OpenSSL, WebCrypto, libsodium, etc.)
- Increased payload for updates (commits)

Server assist

Transferring state

We currently have 3 options:

- Client-to-client: secure and private, but doesn’t scale well
- Unencrypted server assist: bad for privacy
- Encrypted server assist: better for privacy, but not yet solved

Send from outside

Send from outside

- Long-standing idea
- An external party can encrypt messages to the group
- Only group members can decrypt

Use cases: application messages

- The server can encrypt (status) messages to the group:

  That way they don’t linger unencrypted on the server until they are consumed

- External users are invited to the group, but since no client is online they are still waiting for the Welcome handshake message:

  The external users can already send messages to the group

Use cases: handshake messages

- With the Proposal-Commit scheme, external parties can now propose Adds and Removes
- Whether the proposals make it into the commit message will depend on the application policy
- Proposals can be encrypted to the group just like application messages

Proposal

```
update_secret -> HKDF-Extract = epoch_secret
                  |
                  +--> Derive-Secret(., "send from outside", GroupContext_[n])
                  |    = send_from_outside_secret
                  |
```

```
HPKEPublicKey send_from_outside_pub_key = Derive-Key-Pair(send_from_outside_secret)
```

```
struct {
     HPKEPublicKey public_key;
     HPKECiphertext message_to_group<0..2^16-1>;
} MessageToGroup;
```