Messaging Layer Security
The Beginning

Richard Barnes, Benjamin Beurdouche, Karthik Bhargavan, Katriel Cohn-Gordon, Cas Cremers, Jon Millican, Emad Omara, Eric Rescorla, Raphael Robert

RWC 2019, San Jose, CA
Objectives
**Context**

Lots of secure messaging apps

Some use similar protocols...

... some are quite different

... but all have similar challenges

Wildly different levels of analysis

Everyone maintaining their own libraries
Top-Level Goals

Detailed specifications for an async group messaging security protocol

Async - No two participants online at the same time
Group  - Support large, dynamic groups
Messaging security - Modern security properties (FS / PCS)

Code that is reusable in multiple contexts...

... and interoperable between different implementations

Robust, open security analysis and involvement from the academic community
**Scope (with analogy to TLS)**

XSS, Phishing

Message Content (HTTP, SMTP, SIP, ...)

Confidentiality w.r.t. Delivery Service

Security Protocol (TLS / DTLS)

Authentication (PKI)

Traffic analysis

Transport (TCP / UDP)
MLS vs. TLS

Lots of actors - 2 vs. $10^N$

Long lived sessions - seconds vs. months

Lots of mobile devices involved

Significant probability that some member is compromised at some time in the life of the session
Endpoint Compromise

Forward Security*  

Post-Compromise Security*

FS / PCS Interval

* … with regard to a participant
Prior Art

mpOTR, (n+1)sec
No PCS

S/MIME, OpenPGP
Linear scaling, difficult to achieve PCS

Client fanout
Linear scaling, but good async / PCS properties
Signal, Proteus, iMessage, et al.

Sender Keys
Linear scaling, PCS possible but very expensive
WhatsApp, FB, OMEMO, Olm, et al.

Goal: FS/PCS with sub-linear scaling as much as possible
History
Once upon an RWC...

**RWC 2015**
Millican and Barnes introduced

**2016...**
Barnes and Rescorla pondering specifications for messaging security
Millican, Cremers, Cohn-Gordon, et al. looking into tree-based schemes

**RWC 2017**
Hallway track conversations -- “Would a spec be useful?”

**July 2017**
On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees
Hey Jon! How are you?

Saw the tree-keying paper yesterday, looks like good work. Reaching out in case you're willing to answer some questions about notation 😊

Hey Richard 😊 I'm good thanks, how are you doing?

Sure I can try!

And thanks for reading it! 😊
Things Start to Come Together

- **September 2017**
  - MLS Workshop #1

- **November 2017**
  - MLS Workshop #2

- **January 2018**
  - MLS Workshop #3

- **March 2018**
  - IETF MLS BoF

- **May 2018**
  - IETF MLS WG officially formed

- **RWC 2015**
- **RWC 2017**
- **RWC 2018**
- **RWC 2019**
And Now, the Actual Work

- July 2018
  - MLS WG @ IETF 102

- September 2018
  - MLS WG interim

- November 2018
  - MLS WG @ IETF 103

- January 2019
  - MLS WG interim
Tree

Epoch Secret

Protocol Messages

Application Secret
Trees of Keys

KE state of the group comprises a left-balanced binary tree of DH key pairs

Each member of the group occupies a leaf

**Tree invariant:** The private key for an intermediate node is known to a member iff the node is an ancestor of the member’s leaf

C has private keys for H, J, K
Trees of Keys

This has a couple of nice consequences:

Intermediate nodes represent subgroups you can DH with / encrypt to

Root private key is a secret shared by the members of the group at a given time

Protocol maintains this state through group operations (Create, Add, Update, Remove)

C has private keys for H, J, K
1st Try: Asynchronous Ratchet Trees (ART)

The key pair at an intermediate node is derived from a DH operation between its children. This enables log-depth Update:

- Change the private key for a leaf
- Re-derive the nodes up the tree

Add and Remove involve “double-join”:
- A leaf private key held by two members

\[ e = g^{ab} \]
\[ f = g^{cd} \]
\[ h = g^{ef} \]

2nd Try: TreeKEM

Instead of doing DH to set intermediate nodes, when you change a leaf:

Derive from hashes up the tree  
Encrypt the hash to the other child

This one operation does two things:

Encrypt to all but the old  
Update the tree with the new
2nd Try: TreeKEM

Using encryption (vs. DH) enables blank nodes:

Add and Remove without double join

Constant-time Add

Other benefits vs. ART:

Constant time for receivers (vs. log)

More amenable to post-quantum
Protocol Messages
Update The Tree

Add:
- Add leaf to the tree
- Group hashes forward
- Encrypt secret to new joiner

Remove / Update:
- Encrypt fresh entropy to everyone but the evicted participant
Key Schedule


Sign + MAC Authentication

Members of group agree on its state, including...

- Identities and public keys of members
- The public keys in the tree used for key exchange
- The transcript of Handshake messages (as a hash chain)

Messages that change the state include...

- Signature by key corresponding to roster
- MAC over transcript and state using key derived from updated group state
Analysis
Is It Actually Secure?

MLS tries to stay close to some things that have had security analysis, ART and TLS.

ART paper has hybrid modelling: computational analysis of core and symbolic Tamarin proofs of other parts.

Work in Progress: TreeKEM, Authentication, the whole system together.

Some challenges:

- Complex threat model and security properties
- Dynamic groups of arbitrary size
Future Directions
Trade-Offs

- Log-size KE messages
- Constant-size app messages
- Avoiding Double-Join
- Constant-time Add

Shared group state

- TreeKEM + Blank nodes

- Strict message ordering
- State corruption by malicious insiders
- Linear-size state in clients
- “Warm up time” after creation
Specification and Implementation

Architecture and specification still in progress, with several TODOs, e.g.:

- Efficiency of the core protocol
- Robustness w.r.t. malicious insiders
- User-initiated add
- Recovery from state loss
- ACK / NACK messages

Help wanted:
- Reviews of the docs
- Suggestions for how to improve them
- Security analysis

Several implementations currently in progress:

- Melissa (Wire, Rust)
- mlspp (Cisco, C++)
- MLS* (Inria, F*)
- RefMLS (NYU Paris, JS)
- [REDACTED] (Google, C++)

Help wanted:
- Other stacks
- Pull requests to the above
- Suggestions for interop testing
Messaging Layer Security

Architecture:  
https://github.com/mlswg/mls-architecture  
https://protocol.messaginglayersecurity.rocks

Protocol:  
https://github.com/mlswg/mls-protocol  
https://architecture.messaginglayersecurity.rocks

Code + Interop:  
https://github.com/mlswg/mls-implementations

Discussion:  
mls@ietf.org (archives)