MLS BOF

Messaging Layer Security

Chair Slides:
Nick & Sean

IETF 101
This is a reminder of IETF policies in effect on various topics such as patents or code of conduct. It is only meant to point you in the right direction. Exceptions may apply. The IETF’s patent policy and the definition of an IETF "contribution" and "participation" are set forth in BCP 79; please read it carefully.

As a reminder:

- By participating in the IETF, you agree to follow IETF processes and policies.
- If you are aware that any IETF contribution is covered by patents or patent applications that are owned or controlled by you or your sponsor, you must disclose that fact, or not participate in the discussion.
- As a participant in or attendee to any IETF activity you acknowledge that written, audio, video, and photographic records of meetings may be made public.
- Personal information that you provide to IETF will be handled in accordance with the IETF Privacy Statement.
- As a participant or attendee, you agree to work respectfully with other participants; please contact the ombudsteam ([https://www.ietf.org/contact/ombudsteam/](https://www.ietf.org/contact/ombudsteam/)) if you have questions or concerns about this.

Definitive information is in the documents listed below and other IETF BCPs. For advice, please talk to WG chairs or ADs:

- [BCP 9](https://www.ietf.org/rfc/rfc7920.txt) (Internet Standards Process)
- [BCP 25](https://www.ietf.org/rfc/rfc6750.txt) (Working Group processes)
- [BCP 25](https://www.ietf.org/rfc/rfc8093.txt) (Anti-Harassment Procedures)
- [BCP 54](https://www.ietf.org/rfc/rfc6893.txt) (Code of Conduct)
- [BCP 78](https://www.ietf.org/rfc/rfc6810.txt) (Copyright)
- [BCP 79](https://www.ietf.org/rfc/rfc7990.txt) (Patents, Participation)
- [https://www.ietf.org/privacy-policy/](https://www.ietf.org/privacy-policy/) (Privacy Policy)
Requests

- Minute Taker(s)
- Jabber Scribe(s)
- Sign Blue Sheets

Info

- List: [https://www.ietf.org/mailman/listinfo/mls](https://www.ietf.org/mailman/listinfo/mls)
- Jabber: mls@jabber.ietf.org
Agenda

5min  Agenda bashing
30min Problem statement
15min Architecture
15min (draft) Protocol
15min State of formal analysis
30min Charter text discussion
10min BOF questions
Problem Statement
Messaging Layer Security

Problem Statement
Context

Lots of secure messaging apps

Some use similar protocols...

... some are quite different

Wildly different levels of analysis

Everyone maintaining their own libraries
Goals

Detailed specifications for an async group messaging security protocol

Code that is reusable in multiple contexts

Robust, open security analysis and involvement from the academic community

Non-goal: Application-level interoperability
What do we want?

**Async** - Support sessions where no two participants are online at the same time

**Group Messaging** - Support large, dynamic groups with efficient scaling

**Security Protocol** - Modern security properties
  - Forward security
  - Post-compromise security
  - Membership authentication

Non-goals: Full-time deniability, malleability
Establish intuition:
- Forward Security (FS) ~> DH
- Post-Compromise Security (PCS) ~> Keep around DH, rotate DH
Prior Art

MIKEY
- Similar options to S/MIME / OpenPGP

GDOI
- Trusted (symmetric) key server

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, but **good async / PCS properties**
  - FB Messenger, OMEMO, Olm, et al.
Key Ideas from Prior Art

“InitKeys” (or “prekeys”) for async

“Hash ratchets” forward secrecy

“DH ratchets” for PCS

Alice

FS bound

Bob

FS bound

InitKey($g^a$) -> $g^b$

$g^ab$ -> $g^c$

$g^bc$ -> $g^d$

$g^cd$

PCS w.r.t. a

PCS w.r.t. b
Scope (with analogy to TLS)

- **Transport** (TCP / UDP)
- **Message Content** (HTTP, SMTP, SIP, ...)
- **Security Protocol** (TLS / DTLS)
- **Authentication** (PKI)

Certificate [Verify]
Architecture
emadomara@google.com
@emad_omara
System Overview

Authentication Service
Delivery Service

Member A
A1
A2
A3

Member B
B1
B2

Member C
C1

Member D
D1
D2

Group (A,B,C)
System Overview

- Stores user ids to identity key mappings
- Stores initial key materials (initKeys)
- *Stores group membership

- Distributes and delivers messages and attachments
System Overview

- Register
- Send message
- Invite member
- Join group
- Add device

- Create group
- Receive message
- Remove member
- Leave group
- Remove device

Group (A,B,C)

Member A
- A1
- A2
- A3

Member B
- B1
- B2

Member C
- C1

Member D
- D1
- D2
Functional Requirements

- Scalable
  - Support group size up to 50,000 clients
- Asynchronous
  - All client operations can be performed without waiting for the other clients to be online
- Multiple devices
  - Devices are considered separate clients
  - Restoring history after joining is not allowed by the protocol, but Application can provide that.
- State recovery
  - Lost/Corrupted state must be recovered without affecting the group state.
- Metadata collection
  - AS/DS must only store data required for message delivery
- Federation
  - Multiple implementation should be able to interoperate
- Versioning
  - Support version negotiation
Security Requirements

- Message secrecy, integrity and authentication
  - Only current group member can read messages
  - Messages are only accepted if it was sent by a current group member
  - *Message padding to protect against traffic analysis
- Forward secrecy and post compromise security
- Group membership security
  - Consistent view of group members
  - Added clients can’t read messages sent before joining
  - Removed clients can’t read messages sent after leaving
- Attachments security
- Data origin authentication and *deniability
Security Considerations

- Delivery service compromise
  - Must not be able to read or inject messages
  - Modified, reordered or replayed messages must be detected by the clients
  - It can mount various DoS attacks.
- Authentication service compromise
  - Can return incorrect identities to the client
  - Can’t be defeated without transparency logging such as KT
- Client compromise
  - Can read and send messages to the group for a period of time
  - It shouldn’t be able to perform DoS attack.
  - Will be defeated once the compromised client updates their key material
Protocol Operations

• Group state at each point in time is an “asynchronous ratcheting tree”
• Each participant caches a view of the tree
• Protocol operations update the participants’ view of the tree
  • Group Creation
  • Group-initiated Add
  • User-initiated Add
  • Key Update
  • Remove
Asynchronous Ratcheting Tree

• Based on a Diffie-Hellman binary key tree.
• Updates to any leaf in logarithmic time.
• Asynchronous operation.
• Proofs of confidentiality of group keys in static groups.
• MLS defines some things that the original paper leaves out of scope:
  • More constraints on tree structure
  • Membership changes.
  • Race conditions.
DH output -> DH key pair

- Derive-Key-Pair maps random bit strings to DH key pairs
- Resulting private key known both original private key holders

\[
\text{AB} = \text{Derive-Key-Pair(DH(A,B))}
\]

\[
/ \quad \backslash
\]
\[
A \quad B
\]

e.g.:
\[
\text{Derive-Key-Pair(X)} = \text{X25519-Priv(SHA-256(X))}
\]
DH Trees

<table>
<thead>
<tr>
<th>Part</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>Group Key</td>
</tr>
<tr>
<td>Direct Path</td>
<td>Update</td>
</tr>
<tr>
<td>Copath</td>
<td>Add</td>
</tr>
<tr>
<td>Frontier</td>
<td>Add</td>
</tr>
</tbody>
</table>

leaf + copath -> root
frontier = copath(next)
Group Evolution

\[\begin{align*}
\text{Group Evolution} \\
\text{+-> Msg Secret} & \quad \text{+-> Msg Secret} & \quad \text{+-> Msg Secret} \\
\text{\quad |} & \quad \text{\quad |} & \quad \text{\quad |} \\
\text{... --+ KDF --+ Init Secret --+ KDF --+ Init Secret --+ KDF --+ Init Secret --+ ...} \\
\text{\quad ^} & \quad \text{\quad ^} & \quad \text{\quad ^} \\
\text{\quad \quad |} & \quad \text{\quad \quad |} & \quad \text{\quad \quad |} \\
\text{\quad \quad Update} & \quad \text{\quad Update} & \quad \text{\quad Update} \\
\text{\quad \quad Secret} & \quad \text{\quad Secret} & \quad \text{\quad Secret} \\
\text{\quad \quad ^} & \quad \text{\quad \quad ^} & \quad \text{\quad \quad ^} \\
\text{\quad \quad \quad Tree} & \quad \text{\quad \quad Tree} & \quad \text{\quad \quad Tree} \\
\text{\quad \quad \quad Root} & \quad \text{\quad \quad \quad Root} & \quad \text{\quad \quad \quad Root}
\end{align*}\]
Operation 0: Create group

• Can be created directly.
• Can be created by starting with an one-member group, then doing add operations.
• Current draft does the latter, so there’s no protocol for creation.
• ART paper specifies the former, but we don’t use in the draft yet.
Operation 1: Group-Initiated Add

```c
struct {
    UserInitKey init_key;
} GroupAdd;
```

// Pre-published UserInitKey for asynchronicity

// NB: Add Key has implications for removals; “double join”

// Add Key    Init
ABCD
/   \
AB   CD
/   /   
A   B   C   D
```
Operation 2: User-Initiated Add

```
struct {
    DHPublicKey add_path<1..2^16-1>;
} UserAdd;

// Pre-published frontier in
// GroupInitKey for asynchronicity
```
Operation 3: Key Update (for PCS)

```c
struct {
    DHPublicKey
    ratchetPath<1..2^16-1>
} Update;

// This approach to confidentiality
// is proved in [ART]
```

```
    ABCD
     /    \
    AB    CD
     /  \
    A    B  C  D
```
# Operation 4: Remove

```c
struct {
    uint32 deleted;
    DHPublicKey path<1..2^16-1>;
} Delete;
```

// To lock out, update to a key the deleted node doesn’t know

// “Double join” issues similar to GroupAdd
Open Issues

- Tuning up, proving FS and PCS properties of the operations
  - … especially Add, Remove
- Logistical details, especially around Remove
- Message sequencing
- Message protection, transcript integrity
- Authentication
  - Current draft has a very basic scheme, needs elaboration
  - Deniable authentication?
- *Attachments
Summary

- Group keys derived from an Asynchronous Ratcheting Tree
- Group operations update the tree
  - Creation
  - Group-Initiated Add
  - User-Initiated Add
  - Update
  - Remove
- Several open issues to address in the WG
the ART of analysing MLS

Katriel Cohn-Gordon
University of Oxford
people involved

- Karthik (HACL*, miTLS, F*)
- Benjamin (F*, NSS)
- Cas (TAMARIN, TLS 1.3)
- Katriel (Signal, PCS)
- ...

similar projects

- F*
- miTLS
- TLS 1.3: the swamp
- 5G-ENSUUUUURRE
- ...

...
On Ends-to-Ends Encryption
Asynchronous Group Messaging with Strong Security Guarantees

Abstract—In the past five years, messaging has become_mainstream, with billions of users of end-to-end encryption protocols through apps such as WhatsApp, Signal, Facebook Messenger, Gizmo沟, Alia, Viber, and many more. While these most popular communication apps enjoy very strong security guarantees, almost all are “incompletely implemented,” even if users are notified, they have no way to implement an alternative if they choose to do so. This is because the security guarantees are often obscure, and the app developers are not required to follow any standard or best practices. The result is that while end-to-end encryption is widely adopted, it is often not as secure as it should be.

I. INTRODUCTION

The level of security offered by modern messaging systems has improved substantially over recent years, for example. We have seen significant advances in security for end-to-end messaging systems, the first of which was the development of Signal, a messaging system that uses strong encryption and is available on both Android and iOS. However, the level of security offered by these systems is still not sufficient to protect against all potential threats.

This is because modern messaging systems are often not designed with security in mind, and as a result, they are vulnerable to a wide range of attacks. These attacks can include man-in-the-middle attacks, where the attacker intercepts and modifies the messages being sent between users, and side-channel attacks, where the attacker tries to learn information about the users or the messages being sent by monitoring the communications.

ART construction is new but has some early formal analysis

On Ends-to-Ends Encryption
https://ia.cr/2017/666
Katriel, Cas, Luke, Kevin, Jon

L. CORRUGATION

No full group protocol: more to do!
a bit more on the formal analysis

**Theorem VI.1.** Let $n_P$, $n_S$ and $n_T$ denote bounds on the number of parties, sessions and stages in the security experiment respectively. Under the decisional DH assumption, where $\iota$ is instantiated as a random oracle, the success probability of any ppt adversary against the key indistinguishability game of our protocol is bounded above by

$$
\frac{1}{2} + \frac{(n_P n_S n_T \gamma)}{2q} + \gamma (n_P n_S n_T^2)^\gamma (\epsilon_{DDH} + 1/q) + \text{negl}(\lambda)
$$

where $\epsilon_{DDH}$ bounds the advantage of a PPT adversary against the decisional DH game.
going forward

- precise definitions of the properties we would like
  - interactions with “practical” constraints such as recovery from lost devices
  - general enough to cover different use cases

- proofs for the whole system
  - authentication
  - malicious insiders
  - adding and removing people

- verified implementations in F∗?
tl;dr

no proofs yet, but early work on ART and we’re still going :)

Several Internet applications have a need for group key establishment and message protection protocols with the following properties:

- **Asynchronicity** - Keys can be established without any two participants being online at the same time
- **Forward secrecy** - Full compromise of a node at a point in time does not reveal past group keys
- **Post-compromise security** - Full compromise of a node at a point in time does not reveal future group keys
- **Membership Authentication** - Each participant can verify the set of members in the group
- **Message Authentication** - Each message has an authenticated sender
- **Scalability** - Resource requirements that have good scaling in the size of the group (preferably sub-linear)
Several widely-deployed applications have developed their own protocols to meet these needs. While these protocols are similar, no two are close enough to interoperate. As a result, each application vendor has had to maintain their own protocol stack and independently build trust in the quality of the protocol. The primary goal of this working group is to develop a standard messaging security protocol so that applications can share code, and so that there can be shared validation of the protocol (as there has been with TLS 1.3).

It is not a goal of this group to enable interoperability between messaging applications beyond the key establishment, authentication, and confidentiality services. Full interoperability would require alignment at many different layers beyond security, e.g., standard message transport and application semantics. The focus of this work is to develop a messaging security layer that different applications can adapt to their own needs.
In developing this protocol, we will draw on lessons learned from several prior message-oriented security protocols, in addition to the proprietary messaging security protocols deployed within existing applications:

- S/MIME | OpenPGP | Off the Record | Signal

The intent of this working group is to follow the pattern of TLS 1.3, with specification, implementation, and verification proceeding in parallel. By the time we arrive at RFC, we hope to have several interoperable implementations as well as a thorough security analysis.

The specifications developed by this working group will be based on pre-standardization implementation and deployment experience, and generalizing the design described in:

- draft-omara-mls-architecture
- draft-barnes-mls-protocol
Note that consensus is required both for changes to the current protocol mechanisms and retention of current mechanisms. In particular, because something is in the initial document set does not imply that there is consensus around the feature or around how it is specified.

Milestones:
May 2018 Initial working group documents for architecture and key management
Sept 2018 Initial working group document adopted for message protection
Jan 2019 Submit architecture document to IESG as Informational
Jun 2019 Submit key management protocol to IESG as Proposed Standard
Sept 2019 Submit message protection protocol to IESG as Proposed Standard
Scoping Questions
Should the IETF do the work?

Does the scope sound reasonable?

Are the boundaries presented suitable for a security analysis?

Do we agree that the application layer interface is the correct place to enable visibility requirements should they exist?

Do the documents presented represent a good starting point?

Is this proposal flexible enough for the common use cases of secure messaging applications?
BOF Questions
Successful BOF Questions

Is the problem sufficiently understood?

Is the problem tractable?

Is this the right place to address “the problem”?

Who is willing to author specs?

Who is willing to review specs?