Oxen Session Audit

Technical Report

Reference	**20-08-Oxen-REP-v1.3**
Version	**1.3**
Date	**2021/04/20**

Quarkslab

Quarkslab SAS
13 rue Saint Ambroise
75011 Paris
France

Table of Contents

1. Project Information

Document history			
Version	**Date**	**Details**	**Authors**
1.0	28/08/2020	Android Version	Charlie Boulo
1.1	05/11/2020	iOS Version	Marwan Anastas
1.2	18/11/2020	iOS Version	Marwan Anastas
1.3	20/04/2021	Desktop Version	Charlie Boulo

Quarkslab		
Contact	**Role**	**Contact Address**
Frédéric Raynal	CEO	fraynal@quarkslab.com
Matthieu Duez	Services Manager	mduez@quarkslab.com
Charlie Boulo	R&D Engineer	cboulo@quarkslab.com
Marwan Anastas	R&D Engineer	manastas@quarkslab.com

Oxen		
Contact	**Role**	**Contact Address**
Simon Harman	CEO	simon@oxen.io
Kee Jefferys	CTO	kee@oxen.io
Chris McCabe	COO	chris@oxen.io
Niels Andriesse	Session Project Manager	niels@oxen.io
Christopher Pavlesic	Secretary, Proposal Committee Chair	secretary@oxen.io

2. Executive Summary

This report describes the results of the security evaluation made by Quarkslab of Oxen's messaging application **Session**.

As stated in their whitepaper[1], *Session is an open-source, public-key-based secure messaging application which uses a set of de-centralised storage servers and an onion routing protocol to send end-to-end encrypted messages with minimal exposure of user metadata.*

The codebase has been forked from Signal since 2018/2019 as we can see in the following github public data:

- https://api.github.com/repos/oxen-io/session-desktop "2018-08-16T03:45:21Z"
- https://api.github.com/repos/oxen-io/session-android "2019-03-05T05:10:37Z"
- https://api.github.com/repos/oxen-io/session-ios "2019-03-05T05:10:52Z"

Through this audit we reviewed three components that are part of **Session**, each evaluation was performed by one evaluator in 10 days.

Those audits were carried out sequentially in the following order:

- the Android application;
- the iOS application;
- the Desktop application.

Vendor was kept informed throughout the audit process.

Due to the large code base and multiple inclusions of third-party code, this analysis is not exhaustive. Yet, it is sufficient to highlight a few vulnerabilities that could be fixed.

[1] https://arxiv.org/pdf/2002.04609.pdf

2.1 Android application vulnerabilities, remarks and recommendations summary

Issue	Severity	Description	Recommendation	Vendor response
SESS-AND-03	High	Complete lack of TLS verification during gathering of ip and public key of Node servers	Setup a certificate pinning for SeedNodes	Lack of TLS verification was fixed in release 1.5.4 https://github.com/oxen-io/session-android/releases/tag/1.5.4 Certificate pinning was added in release 1.9.0 https://github.com/oxen-io/session-android/releases/tag/1.9.0
SESS-AND-05	Moderate	Some requests are not routed through Lokinet, resulting a potentially dangerous user tracking by ISP	Route all the possible traffic through Lokinet	This issue was fixed in Android release 1.5.4 https://github.com/oxen-io/session-android/releases/tag/1.5.4
SESS-AND-06	Moderate	User private key can be copied to clipboard	Remove this feature since the app uses a word-encoding mnemonic sentence	This is intended functionality. Managing seed words is difficult, and we want to make the process of saving seed words easier by allowing users to copy their seed. This simplifies the process of saving their seed words in a password manager or other software, but requires clipboard access. Although this can expose the seed, the alternative is that the user does not backup their seed and risks losing access to their account in the event of device failure.

Issue	Severity	Description	Recommendation	Vendor response
SESS-AND-07	Moderate	Android user private key displaying view can be screenshot	Change the default settings in application	This is intended functionality. This allows users to more easily manage their seed phrase by screenshotting the required information and storing the screenshot in a secure location.
SESS-AND-01	Low	Android notification data leak on default configuration	Change the default setting	We have added a setting inside Session which allows users to toggle the information they show on their lock screen when receiving a notification. We believe having full notification information shown by default creates the best user experience, those users who need further protection can toggle the notification setting in the app to remove identifying information.
SESS-AND-02	Low	Android view data leak on default configuration	Blur views when app goes to background	This issue was fixed in Android release 1.5.4 https://github.com/oxen-io/session-android/releases/tag/1.5.4

Issue	Severity	Description	Recommendation	Vendor response
SESS-AND-04	Low	Android key generation entropy may be weak	Use the standard entropy for key generation, as performed in Signal	Session key generation is performed as standard Ed25519 key generation, with one modification: we use a random 128-bit value for the seed rather than a 256-bit seed value. Although this reduces the number of resulting possible private key values, because Ed25519's private key generation uses a cryptographically secure hash (SHA-2/512) of the seed value, it does not introduce any correlation into the bits of the private key value which would otherwise weaken the cryptographic properties. Furthermore, the cited x25519 issue does not apply here: the SHA-2 hash is explicitly used in Ed25519 to eliminate possible correlation in the bits of the private key. This reduction was deliberately chosen to provide Session users with more usable 13-word recovery phrases as opposed to the 25 words which would be required if using a 256-bit key. Though the smaller seed does, in theory, result in a smaller brute force attack space, this is a distinction in theory only: a brute force attack against 2^{128} possible seed values is simply not practical.

2.2 iOS application vulnerabilities, remarks and recommendations summary

Issue	Severity	Description	Recommendation	Vendor response
SESS-IOS-01	Moderate	iOS message attachments are stored in plaintext	Encrypt attachments	Attachment management was inherited from Signal, we are in the process of adding a user-defined password option which will allow users to encrypt the local database with a selected password.
SESS-IOS-02	Moderate	iOS initial gathering of Lokinet Snodes ip and keys is carried out without certificate pinning	Use public key pinning	This was fixed in iOS release 1.9.4 https://github.com/oxen-io/session-ios/releases/tag/1.9.4
SESS-IOS-03	Moderate	iOS attached files are downloaded without being routed through Lokinet's nodes	Use Lokinet infrastructure for attached file download	This was fixed in iOS version 1.5.3 https://github.com/oxen-io/session-ios/releases/tag/1.5.3
SESS-IOS-05 Logging	Moderate	iOS log management may reveal valuable information	Disable logs or reduce information and send them through Lokinet anonymity infrastructure	This was fixed in iOS version 1.7.0 https://github.com/oxen-io/session-ios/releases/tag/1.7.0
SESS-IOS-06	Moderate	User private key can be copied to clipboard	Remove this feature since **Session** uses a word-encoding mnemonic sentence	This is intended functionality. Managing seed words is difficult, and we want to make the process of saving seed words easier by allowing users to copy their seed. This simplifies the process of saving their seed words in a password manager or other software, but requires clipboard access. Although this can expose the seed, the alternative is that the user does not backup their seed and risks losing access to their account in the event of device failure.
SESS-IOS-07	Moderate	Application is not protected against screen recording	Monitor screen state	This is intended functionality. It allows users to create video reviews of Session and provide more in-depth information in bug reports.

Issue	Severity	Description	Recommendation	Vendor response
SESS-IOS-04	Low	iOS key generation entropy may be weak	Use the standard entropy for key generation, as in Signal	Session key generation is performed as standard Ed25519 key generation, with one modification: we use a random 128-bit value for the seed rather than a 256-bit seed value. Although this reduces the number of resulting possible private key values, because Ed25519's private key generation uses a cryptographically secure hash (SHA-2/512) of the seed value, it does not introduce any correlation into the bits of the private key value which would otherwise weaken the cryptographic properties. Furthermore, the cited x25519 issue does not apply here: the SHA-2 hash is explicitly used in Ed25519 to eliminate possible correlation in the bits of the private key. This reduction was deliberately chosen to provide Session users with more usable 13-word recovery phrases as opposed to the 25 words which would be required if using a 256-bit key. Though the smaller seed does, in theory, result in a smaller brute force attack space, this is a distinction in theory only: a brute force attack against 2^{128} possible seed values is simply not practical.

2.3 Desktop application vulnerabilities, remarks and recommendations summary

Issue	Severity	Description	Recommendation	Vendor response
SESS-DES-01	Moderate	Onion circuit data leak in plaintext log files	Protect logfile or limit log verbosity	This has been fixed in version Session Desktop 1.5.3 by omitting sensitive information in logs https://github.com/oxen-io/session-desktop/releases/tag/v1.5.3
SESS-DES-02	Low	Update mechanism is not routed to onion network and hence may help to detect users from an ISP point of view	Route update traffic to Onion networkd	A current limitation of the onion requests protocol is that it cannot talk to arbitrary http endpoints, this means that in a few cases like fetching updates or seed node lists we need to speak clearnet regular requests, in the future we plan to integrate Lokinet allowing us to onion route these requests.

3. Context and Scope

3.1 Offer Synthesis

Quarkslab performed a security evaluation of **Session**. Each application review was performed in about 10 man-days.

The review is split between 3 main code bases that have been provided by the client: the Android application, the iOS application and the Desktop application. Below are the listed versions that have been used for the audit:

- **Android**
  - https://github.com/loki-project/session-android

    commit b5ecb8fd991c424c071ccd20f726a4c674d50b7b (tag audit)
  - https://github.com/loki-project/session-android-service

    commit 1becaa7e7051b941c3935e3203194afa948ffe87 (tag audit)
  - **APK** : v1.4.1 https://github.com/loki-project/session-android/releases/download/1.4.1/session-1.4.1-universal.apk
  - 2 phones were used to instrument application:
    - Pixel 2 running Android 10
    - Pixel 3 running Android 10

- **iOS**
  - https://github.com/loki-project/session-ios

    commit 2c18c3694d725f353cbe65c927fc37780c3c6260 (tag audit)
  - https://github.com/loki-project/session-ios-metadata-kit

    commit df787d84bb8adb23c10df669296dee8d7988e410 (tag audit)
  - https://github.com/loki-project/session-ios-pods

    commit f818a61c04eeb78662dc4626014575bff9eb879b (tag audit)
  - **App** : v1.6.0 https://github.com/loki-project/session-ios/releases/tag/1.6.0
  - 1 phone used for analysis:
    - iPhone 8 running iOS 13.4.1

- **Desktop application**
  - https://github.com/oxen-io/session-desktop/

commit 38300881bd283eaefac45b90b68d855737346983 (tag audit2)

- https://github.com/oxen-io/session-file-server/

  commit 5173163fe18ac575676020e2f8621cf7a2956df3

- Linux Debian dockers used for analysis:
  - debian:buster (Linux cddedb0c88e0 5.9.0-1-amd64 #1 SMP Debian 5.9.1-1 (2020-10-17) x86_64 GNU/Linux)

3.2 Methodology

As the audit had quite a short time limit, we engaged the target in different ways in order to cover the major aspects that could impact the security of the application. Overall, we tried to consider various attacks based on the degree of power of the attacker:

- Stolen device attacker;
- Fully remote client-side attacker;
- Network omniscient attacker;
- Compromised server attacker (or server owner);
- Side applications in the same device.

Considering the various potential attackers helped identify different kinds of vulnerabilities. Those were mostly found while reading the source code that was made available by **Session**. We interacted with the application as legitimate users, and when necessary we also performed dynamic modifications of the code thanks to the Frida framework.

4. Android Evaluation Overview

4.1 Threat hypothesis

We considered the following types of attacker:

- Stolen device attacker

  Even if just for a few minutes, this attacker has physical access to the device.

  - Locked device
  - Unlocked device

- Fully remote client-side attacker

  This attacker only owns a client and is able to connect in a more or less legitimate way to the **Session** infrastructure.

- Network omniscient attacker

  This attacker is supposed to have a global view of all the traffic between the **Session** users and the servers they connect to.

- Compromised server attacker (or server owner)

  This attacker is supposed to have full access to each server composing the Lokinet network and the **Session** infrastructure.

- Side applications in the same device

  This attacker is able to execute code from a neighboring application located in the user device.

4.2 Stolen device attacker

4.2.1 Locked device

The application properly protects itself from potential malicious backup by disallowing adb backup in the Manifest.

```
android:allowBackup="false"
```

The application protects files received in the internal database. External storage is only used when the user specifically requests export of chosen files. Sent/Saved photos containing Exif GPS data are properly sanitized, even when they are picked up from the gallery.

SESS-AND-01

Still, on the default configuration, there is no protection on the notification display feature.

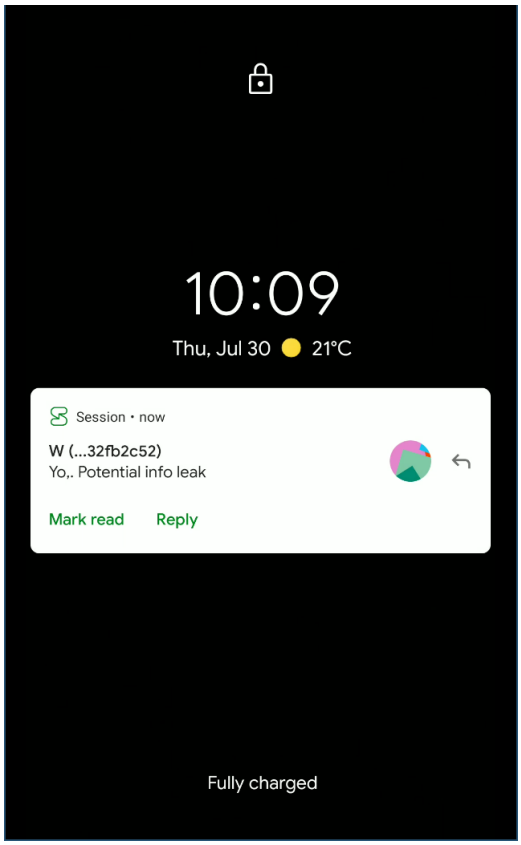


Fig. 4.1: Messages are displayed on a locked device.

This default option only induces a minor security risk of personal data disclosure, as In-Notification reply feature is only available when the device is unlocked. Hence, a temporary thief cannot answer incoming messages if the device is locked.

Session notification is by default composed of the message sender's name and content. However, this can be modified from the application settings.

4.2.2 Unlocked device

Since there are no other user authentication mechanisms for **Session** than locking the device, user impersonation and data theft are hence possible for such a threat. We don't consider this as a major concern, though.

SESS-AND-02

One could also note that, with default parameters, there is no screenshot security to blur content when app goes to background.

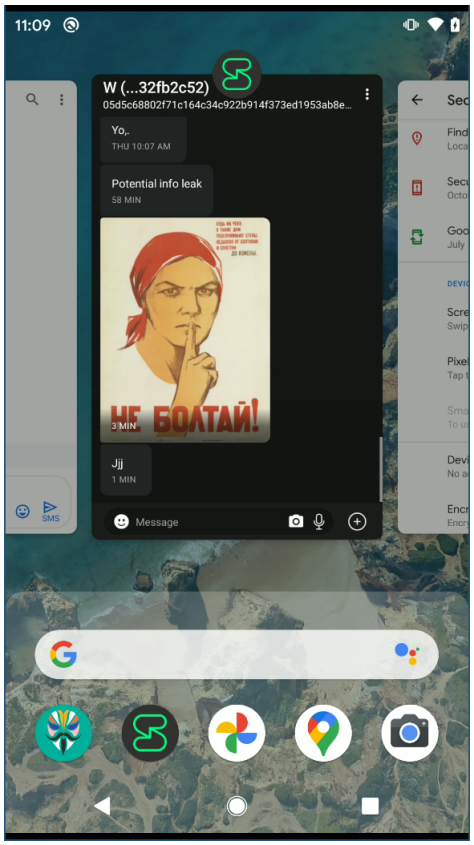


Fig. 4.2: Messages are displayed in background.

Once again, in such a scenario, this lack of security could lead to personal data leaks.

On another note, we can imagine that an attacker having temporary access to the device is able to retrieve the recovery phrase in a fairly reasonable time. This could lead a user impersonation threat, since it is possible to hijack existing session by registering a new android device with the same recovery phrase. In such a case, the attacker has to wait for new messages to come. Even though these messages cannot be deciphered at first, he will be able to discover the targeted user contacts and restore the sessions. Finally, with these pieces of information, he will be able to impersonate the original recipient completely. It is important to underline the fact that **silent eavesdropping cannot be carried out** with this approach.

This is a really good compromise between the lost device use case and the communication security.

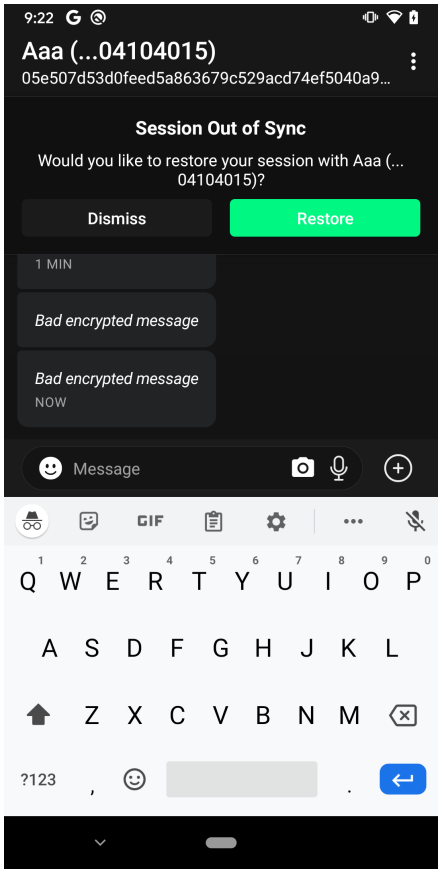


Fig. 4.3: Bad encrypted messages are received to the targeted user

Recovery phrase is hence a valuable asset that has to be protected. However it seems that no other specific mitigation has been made to prevent its leak, except for the de facto application isolation provided by Android.

```
# grep loki_seed /data/data/network.loki.messenger/shared_prefs/SecureSMS-
↪Preferences.xml
<string name="loki_seed">a506168745c65a497514cb2fb00587c0</string>
```

Python code to encode Loki seed is provided as a helper script to quickly get the recovery phrase and then register a new account.

4.2.3 Message Storage

No major changes have been made on how messages are stored. Signal application uses an SQLCipher database for which the (de)ciphering password is stored in a shared preference file:

```
/data/data/network.loki.messenger/shared_prefs# grep pref_database_
↪encrypted_secret network.loki.messenger_preferences.xml
    <string name="pref_database_encrypted_secret">{&quot;data&quot;:&quot;
↪X52RIT0DFEa3glG4+cn0rzT9jUx5rOACeG1fXdxjE1ClksZZ3PvWU5aCQwueXNMU&quot;,&
↪quot;iv&quot;:&quot;MydV5tKcu28cBFdu&quot;}</string>
```

This password is actually protected using the keystore entry *SignalSecret*, following the state-of-the-art recommendations.

Database is opened or created from DatabaseFactory

org/thoughtcrime/securesms/database/DatabaseFactory.java

```
private DatabaseFactory(@NonNull Context context) {
  SQLiteDatabase.loadLibs(context);

  DatabaseSecret       databaseSecret    = new␣
↪DatabaseSecretProvider(context).getOrCreateDatabaseSecret();
  AttachmentSecret     attachmentSecret = AttachmentSecretProvider.
↪getInstance(context).getOrCreateAttachmentSecret();

  this.databaseHelper            = new SQLCipherOpenHelper(context,␣
↪databaseSecret);
  this.sms                       = new SmsDatabase(context,␣
↪databaseHelper);
  this.mms                       = new MmsDatabase(context,␣
↪databaseHelper);
  this.attachments               = new AttachmentDatabase(context,␣
↪databaseHelper, attachmentSecret);
```

To open or create attachment and message databases, two different Secret strings are generated and stored in an Android SharedPreference file.

org/thoughtcrime/securesms/database/DatabaseFactory.java

```
public DatabaseSecret getOrCreateDatabaseSecret() {
  String unencryptedSecret = TextSecurePreferences.
↪getDatabaseUnencryptedSecret(context);
  String encryptedSecret   = TextSecurePreferences.
↪getDatabaseEncryptedSecret(context);

  if      (unencryptedSecret != null) return␣
↪getUnencryptedDatabaseSecret(context, unencryptedSecret);
  else if (encryptedSecret != null)   return␣
↪getEncryptedDatabaseSecret(encryptedSecret);
  else                                return␣
↪createAndStoreDatabaseSecret(context);
}
```

32-byte long keys are randomly generated and stored in Android Keystore, if possible.

org/thoughtcrime/securesms/crypto/AttachmentSecretProvider.java

```
public synchronized AttachmentSecret getOrCreateAttachmentSecret() {
  if (attachmentSecret != null) return attachmentSecret;

  String unencryptedSecret = TextSecurePreferences.
↪getAttachmentUnencryptedSecret(context);
  String encryptedSecret   = TextSecurePreferences.
↪getAttachmentEncryptedSecret(context);

  if      (unencryptedSecret != null) attachmentSecret =␣
↪getUnencryptedAttachmentSecret(context, unencryptedSecret);
  else if (encryptedSecret != null)   attachmentSecret =␣
↪getEncryptedAttachmentSecret(encryptedSecret);
  else                                attachmentSecret =␣
↪createAndStoreAttachmentSecret(context);

  return attachmentSecret;
```

(continues on next page)

(continued from previous page)

```
}
// [...]
private AttachmentSecret createAndStoreAttachmentSecret(@NonNull Context␣
↪context) {
  SecureRandom random = new SecureRandom();
  byte[]       secret = new byte[32];
  random.nextBytes(secret);

  AttachmentSecret attachmentSecret = new AttachmentSecret(null, null,␣
↪secret);
  storeAttachmentSecret(context, attachmentSecret);

  return attachmentSecret;
}

private void storeAttachmentSecret(@NonNull Context context, @NonNull␣
↪AttachmentSecret attachmentSecret) {
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    KeyStoreHelper.SealedData encryptedSecret = KeyStoreHelper.
↪seal(attachmentSecret.serialize().getBytes());
    TextSecurePreferences.setAttachmentEncryptedSecret(context,␣
↪encryptedSecret.serialize());
  } else {
    TextSecurePreferences.setAttachmentUnencryptedSecret(context,␣
↪attachmentSecret.serialize());
  }
}
```

org/thoughtcrime/securesms/crypto/DatabaseSecretProvider.java

```
private DatabaseSecret createAndStoreDatabaseSecret(@NonNull Context␣
↪context) {
  SecureRandom random = new SecureRandom();
  byte[]       secret = new byte[32];
  random.nextBytes(secret);

  DatabaseSecret databaseSecret = new DatabaseSecret(secret);

  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    KeyStoreHelper.SealedData encryptedSecret = KeyStoreHelper.
↪seal(databaseSecret.asBytes());
    TextSecurePreferences.setDatabaseEncryptedSecret(context,␣
↪encryptedSecret.serialize());
  } else {
    TextSecurePreferences.setDatabaseUnencryptedSecret(context,␣
↪databaseSecret.asString());
  }

  return databaseSecret;
}
```

4.3 Fully remote client-side attacker

From a completely remote client-side point of view, we identified that the following interactions are possible on Oxen infrastructure:

- send and retrieve messages;
- upload and download files (avatar pictures, stickers or message attachments);
- create and update Groups.

4.3.1 Closed Group

For the audit version of the application, we noticed differences between compiled **Session** application and the code base.

These differences rely on the implementation of Signal's Sender Key Protocol. However, the code is actually not reached since **ClosedGroupsProtocol.isSharedSenderKeysEnabled** is set to false.

The createLegacyClosedGroup is hence used for ClosedGroups creation:

org/thoughtcrime/securesms/loki/activities/CreateClosedGroupActivity.kt

```
private fun createClosedGroup() {
    if (ClosedGroupsProtocol.isSharedSenderKeysEnabled) {
        createSSKBasedClosedGroup()
    } else {
        createLegacyClosedGroup()
    }
}
```

In order to estimate the possibility of crafting a fake GroupUpdate message, altering an existing discussion thread, we looked for the group update processing:

org/thoughtcrime/securesms/jobs/PushDecryptJob.java

```
private void handleMessage(@NonNull SignalServiceEnvelope envelope,␣
↪@NonNull Optional<Long> smsMessageId, boolean isPushNotification) {
//Skipped...
    SignalServiceContent content = cipher.decrypt(envelope);
//Skipped...
if (content.getDataMessage().isPresent()) {
  SignalServiceDataMessage message        = content.getDataMessage().get();
//Skipped...
  if (message.isGroupUpdate()) {
    handleGroupMessage(content, message, smsMessageId);


private void handleGroupMessage(@NonNull SignalServiceContent content,
                                @NonNull SignalServiceDataMessage message,
                                @NonNull Optional<Long> smsMessageId)
    throws StorageFailedException
{
  GroupMessageProcessor.process(context, content, message, false);
```

The *content* parameter is parsed immediately after the decryption process:

org/whispersystems/signalservice/api/crypto/SignalServiceCipher.java

```
public SignalServiceContent decrypt(SignalServiceEnvelope envelope)
{
  try {
      Plaintext plaintext = decrypt(envelope, envelope.getContent());
      Content   message   = Content.parseFrom(plaintext.getData());
```

The parsing mechanism is made of a *com.google.protobuf.Parser* that may set an optional group in the SignalServiceContent object that is returned in PushDecryptJob.

Before acquiring the requested group info that has to be updated, the GroupMessageProcessor checks for a valid groupid in the local group database. Remote closed group intruders hence have to guess the group id in order to reach the various handlers.

org/thoughtcrime/securesms/groups/GroupMessageProcessor.java

```
public static @Nullable Long process(@NonNull Context context,
                                     @NonNull SignalServiceContent content,
                                     @NonNull SignalServiceDataMessage␣
↪message,
                                     boolean outgoing)
{
  if (!message.getGroupInfo().isPresent() || message.getGroupInfo().get().
↪getGroupId() == null) {
    Log.w(TAG, "Received group message with no id! Ignoring...");
    return null;
  }

  GroupDatabase         database = DatabaseFactory.
↪getGroupDatabase(context);
  SignalServiceGroup    group    = message.getGroupInfo().get();
  String                id       = GroupUtil.getEncodedId(group);
  Optional<GroupRecord> record   = database.getGroup(id);

  if (record.isPresent() && group.getType() == Type.UPDATE) {
    return handleGroupUpdate(context, content, group, record.get(),␣
↪outgoing);
  } else if (!record.isPresent() && group.getType() == Type.UPDATE) {
    return handleGroupCreate(context, content, group, outgoing);
  } else if (record.isPresent() && group.getType() == Type.QUIT) {
    return handleGroupLeave(context, content, group, record.get(),␣
↪outgoing);
  } else if (record.isPresent() && group.getType() == Type.REQUEST_INFO) {
    return handleGroupInfoRequest(context, content, group, record.get());
  } else {
    Log.w(TAG, "Received unknown type, ignoring...");
    return null;
  }
}
```

Session allocates this value at the group creation:

org/thoughtcrime/securesms/groups/GroupManager.java

```
public static @NonNull GroupActionResult createGroup(@NonNull  Context    ␣
↪   context,
                                                      @NonNull  Set
↪<Recipient> members,
                                                      @Nullable Bitmap     ␣
↪   avatar,
```

(continues on next page)

(continued from previous page)

```
                                                    @Nullable String
↪    name,
                                                               boolean
↪    mms,
                                                    @NonNull  Set
↪<Recipient> admins)
{
  GroupDatabase database = DatabaseFactory.getGroupDatabase(context);
  String id = GroupUtil.getEncodedId(database.allocateGroupId(), mms);
  return createGroup(id, context, members, avatar, name, mms, admins);
}
```

This id is actually rather difficult to predict as it is composed of 16 random bytes:

org/thoughtcrime/securesms/database/GroupDatabase.java

```
public byte[] allocateGroupId() {
  byte[] groupId = new byte[16];
  new SecureRandom().nextBytes(groupId);
  return groupId;
}
```

4.3.2 File Upload

File uploading only requires authentication when dealing with Open Chat Group as we can see in the following code:

org/whispersystems/signalservice/loki/api/DotNetAPI.kt

```
private fun upload(server: String, request: Request.Builder, parse: (Map<*,
↪ *>) -> UploadResult): Promise<UploadResult, Exception> {
    val promise: Promise<Map<*, *>, Exception>
    if (server == FileServerAPI.shared.server) {
        request.addHeader("Authorization", "Bearer loki")
        // Uploads to the Loki File Server shouldn't include any␣
↪personally identifiable information, so use a dummy auth token
        promise = OnionRequestAPI.sendOnionRequest(request.build(),␣
↪FileServerAPI.shared.server, FileServerAPI.fileServerPublicKey)
    } else {
        promise = FileServerAPI.shared.
↪getPublicKeyForOpenGroupServer(server).bind { openGroupServerPublicKey ->
            getAuthToken(server).bind { token ->
                request.addHeader("Authorization", "Bearer $token")
                OnionRequestAPI.sendOnionRequest(request.build(), server,␣
↪openGroupServerPublicKey)
            }
        }
    }
```

We tested (with the attached *Python code used for file server interaction*) several client-server interactions, focusing on how files were sent in a closed group rather than open chat ones.

The *DotNetAPI.uploadAttachment* method returns an UploadResult object containing the URL of the attached file alongside its sha256 digest. Therefore, we decided to hook it in order to obtain this data.

We discovered that, when files are successfully fetched by recipient users, the encrypted attachments are still lingering on the server (https://file-static.lokinet.org/f/XXXXXX) and are available without any

authentication.

However, these attachments (and profile pictures) are symmetrically ciphered with AES/CBC encryption and a 256-bit ephemeral key that is unique for each attachment. This mechanism completely prevents remote attackers from performing file attributions.

File names are composed of 6 lowercase alphanumeric random characters generated server-side. This implies that :

- collision attacks aimed at creating a valid file for other users are not possible;
- bruteforce attacks aimed at collecting various encrypted attachments for further investigation requires enumerating $(26 + 10)^6 = 2176782336 \approx 20^8$ possibilities.

We also tried to replay and alter the content of several uploaded files, by reaching directly the url "https://file.getsession.org/files" without using the SnodeAPI.

Without any credential, we succeed in pushing and fetching arbitrary content on the server **file.getsession.org**. It is worth mentioning this production server currently generates and displays useful error messages when contacted incorrectly.

The response from **file.getsession.org** provides a valid link to **file-static.lokinet.org**. Those two domains are located on the same server:

```
$ ping file-static.lokinet.org
PING file-static.lokinet.org (51.79.57.232) 56(84) bytes of data.
$ ping file.getsession.org
PING file.getsession.org (51.79.57.232) 56(84) bytes of data.
```

We replayed the upload request twice with the same uuid name but with different file content, in hope that the first attachment file would be replaced. In such case, this behavior would indicate that the target is weak against substitution attacks. Once again, the json response contained different valid links every time we contacted it, mitigating this attack vector.

4.4 Network omniscient attacker

4.4.1 Service Node Gathering

The following figure sums up the process for a **Session** client connecting Lokinet's Snodes.

This connection is necessary since message reception and sending are computed through onion requests.

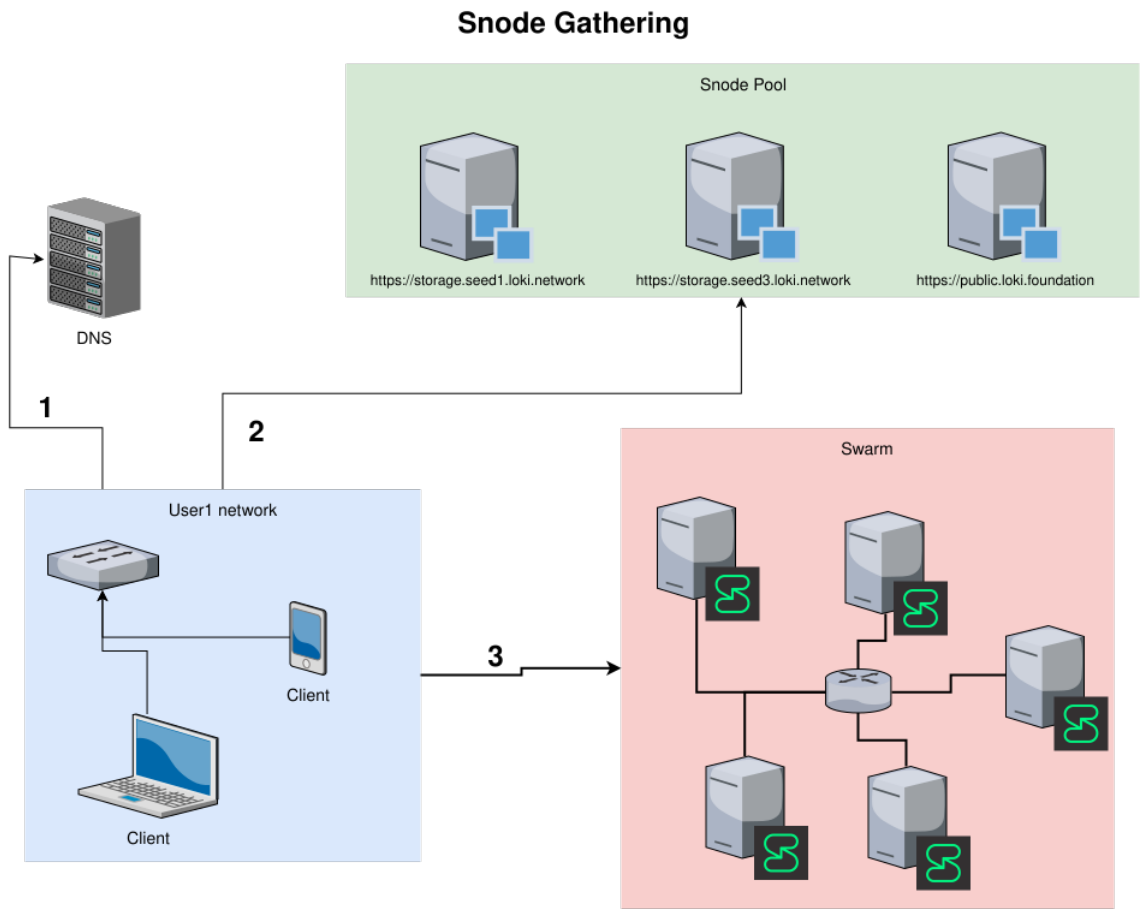


Fig. 4.4: Users resolve snode ip and cryptographic public keys

1.client gets a random server ip location from the specific pool in charge of providing service node list;

2.client contacts the specific service node through https;

3.client then contacts service nodes.

4.4.2 TLS connection consideration

Firstly, we had a look into the mechanisms used by **Session** clients to contact Swarm nodes in order to send or retrieve messages (part 3 in *Users resolve snode ip and cryptographic public keys*).

We identified that TLS communication relies on self-signed certificates such as the one bellow:

```
$openssl s_client -showcerts -servername 51.83.99.236 -connect 51.83.99.
↪236:22021  < /dev/null

CONNECTED(00000003)
depth=0 C = AU, CN = localhost, O = Loki
verify error:num=18:self signed certificate
verify return:1
depth=0 C = AU, CN = localhost, O = Loki
verify return:1
---
Certificate chain
 0 s:C = AU, CN = localhost, O = Loki
```

(continues on next page)

(continued from previous page)

```
   i:C = AU, CN = localhost, O = Loki
-----BEGIN CERTIFICATE-----
# [...] SKIPPED
-----END CERTIFICATE-----
---
Server certificate
subject=C = AU, CN = localhost, O = Loki

issuer=C = AU, CN = localhost, O = Loki

---
# [...] SKIPPED
No client certificate CA names sent
SSL handshake has read 1393 bytes and written 397 bytes
Verification error: self signed certificate
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
```

The code in charge of the https connection makes use of onion routing.

org/whispersystems/signalservice/loki/api/onionrequests/OnionRequestAPI.kt

```
 /**
  * Sends an onion request to `destination`. Builds new paths as needed.
  */
 internal fun sendOnionRequest(destination: Destination, payload: Map<*, *>
↪, isJSONRequired: Boolean = true): Promise<Map<*, *>, Exception> {
     val deferred = deferred<Map<*, *>, Exception>()
     lateinit var guardSnode: Snode
     buildOnionForDestination(payload, destination).success { result ->
         guardSnode = result.guardSnode
         val url = "${guardSnode.address}:${guardSnode.port}/onion_req"
         val finalEncryptionResult = result.finalEncryptionResult
         val onion = finalEncryptionResult.ciphertext
         if (destination is Destination.Server
             && onion.count().toDouble() > 0.75 * (FileServerAPI.
↪maxFileSize.toDouble() / FileServerAPI.fileSizeORMultiplier)) {
             Log.d("Loki", "Approaching request size limit: ~${onion.
↪count()} bytes.")
         }
         @Suppress("NAME_SHADOWING") val parameters = mapOf(
             "ciphertext" to Base64.encodeBytes(onion),
             "ephemeral_key" to finalEncryptionResult.ephemeralPublicKey.
↪toHexString()
         )
         val destinationSymmetricKey = result.destinationSymmetricKey
         Thread {
             try {
                 val json = HTTP.execute(HTTP.Verb.POST, url, parameters)
```

The HTTP class uses a **connection** object:

org/whispersystems/signalservice/loki/api/utilities/HTTP.kt

```
/**
 * Sync. Don't call from the main thread.
 */
fun execute(verb: Verb, url: String, parameters: Map<String, Any>? =␣
↪null): Map<*, *> {
    val request = Request.Builder().url(url)
    when (verb) {
        Verb.GET -> request.get()
        Verb.PUT, Verb.POST -> {
            if (parameters == null) { throw Exception("Invalid JSON.") }
            val contentType = MediaType.get("application/json;␣
↪charset=utf-8")
            val body = RequestBody.create(contentType, JsonUtil.
↪toJson(parameters))
            if (verb == Verb.PUT) request.put(body) else request.
↪post(body)
        }
        Verb.DELETE -> request.delete()
    }
    lateinit var response: Response
    try {
        response = connection.newCall(request.build()).execute()
```

This object relies itself on a permissive X509TrustManager.

```
private val connection by lazy {
    // Snode to snode communication uses self-signed certificates but␣
↪clients can safely ignore this
    val trustManager = object : X509TrustManager {
        override fun checkClientTrusted(chain: Array<out X509Certificate>?,
↪ authorizationType: String?) { }
        override fun checkServerTrusted(chain: Array<out X509Certificate>?,
↪ authorizationType: String?) { }
        override fun getAcceptedIssuers(): Array<X509Certificate> {
            return arrayOf()
        }
    }
```

This fact does not represent a major risk in terms of security as underlying onion request mechanisms are properly encrypted as follows:

org/whispersystems/signalservice/loki/api/onionrequests/OnionRequestAPI.kt

```
/**
 * Builds an onion around `payload` and returns the result.
 */
private fun buildOnionForDestination(payload: Map<*, *>, destination:␣
↪Destination): Promise<OnionBuildingResult, Exception> {
// [...] skipped
    return getPath(snodeToExclude).bind(SnodeAPI.sharedContext) { path ->
        guardSnode = path.first()
        // Encrypt in reverse order, i.e. the destination first
        OnionRequestEncryption.encryptPayloadForDestination(payload,␣
↪destination).bind(SnodeAPI.sharedContext) { r ->
            destinationSymmetricKey = r.symmetricKey
            // Recursively encrypt the layers of the onion (again in␣
↪reverse order)
```

(continues on next page)

(continued from previous page)

```
            // [...] skipped
                    return OnionRequestEncryption.encryptHop(lhs, rhs,␣
↪encryptionResult).bind(SnodeAPI.sharedContext) { r ->
```

An ephemeralPublicKey is generated for the remote ending Snode replying purposes, making the request/response life cycle resistant to eavesdropping and alteration.

org/whispersystems/signalservice/loki/api/onionrequests/OnionRequestEncryption.kt

```
internal fun encryptPayloadForDestination(payload: Map<*, *>, destination:␣
↪OnionRequestAPI.Destination): Promise<EncryptionResult, Exception> {
// [...] Skipped
    val result = encryptForX25519PublicKey(plaintext,␣
↪snodeX25519PublicKey)
    deferred.resolve(result)
    return deferred.promise
}

private fun encryptForX25519PublicKey(plaintext: ByteArray,␣
↪hexEncodedX25519PublicKey: String): EncryptionResult {
    val x25519PublicKey = Hex.
↪fromStringCondensed(hexEncodedX25519PublicKey)
    val ephemeralKeyPair = Curve25519.getInstance(Curve25519.BEST).
↪generateKeyPair()
    val ephemeralSharedSecret = Curve25519.getInstance(Curve25519.BEST).
↪calculateAgreement(x25519PublicKey, ephemeralKeyPair.privateKey)
    val mac = Mac.getInstance("HmacSHA256")
    mac.init(SecretKeySpec("LOKI".toByteArray(), "HmacSHA256"))
    val symmetricKey = mac.doFinal(ephemeralSharedSecret)
    val ciphertext = encryptUsingAESGCM(plaintext, symmetricKey)
    return EncryptionResult(ciphertext, symmetricKey, ephemeralKeyPair.
↪publicKey)
 }
```

This process seems to be well designed and implemented so far.

SESS-AND-03

This is why we considered reviewing the way **Session** fetches all Lokinet's Snode public keys. This is done by asking some fixed Snodes, named *seedNode*, to get a proper list of ip and public keys as we can see in *SwarmApi*:

org.whispersystems.signalservice.loki.api.SwarmAPI

```
private val seedNodePool: Set<String> = setOf( "https://storage.seed1.loki.
↪network", "https://storage.seed3.loki.network", "https://public.loki.
↪foundation" )

private fun getPath(snodeToExclude: Snode?): Promise<Path, Exception> {
    if (pathSize < 1) { throw Exception("Can't build path of size zero.") }
    if (guardSnodes.isEmpty() && paths.count() >= pathCount) {
        guardSnodes = setOf( paths[0][0], paths[1][0] )
    }
    fun getPath(): Path {
        if (snodeToExclude != null) {
```

(continues on next page)

(continued from previous page)

```
            return paths.filter { !it.contains(snodeToExclude) }.
↪getRandomElement()
        } else {
            return paths.getRandomElement()
        }
    }
    if (paths.count() >= pathCount) {
        return Promise.of(getPath())
    } else {
        return buildPaths().map(SnodeAPI.sharedContext) { _ ->
            getPath()
        }
    }
}


/**
 * Builds and returns `pathCount` paths. The returned promise errors out␣
↪if not
 * enough (reliable) snodes are available.
 */
private fun buildPaths(): Promise<List<Path>, Exception> {
    Log.d("Loki", "Building onion request paths.")
    SnodeAPI.shared.broadcaster.broadcast("buildingPaths")
    return SwarmAPI.shared.getRandomSnode().bind(SnodeAPI.sharedContext)
↪{ // Just used to populate the snode pool
        getGuardSnodes().map(SnodeAPI.sharedContext) { guardSnodes ->
            var unusedSnodes = reliableSnodePool.minus(guardSnodes)
            val pathSnodeCount = guardSnodeCount * pathSize -␣
↪guardSnodeCount
            if (unusedSnodes.count() < pathSnodeCount) { throw␣
↪InsufficientSnodesException() }
            // Don't test path snodes as this would reveal the user's IP␣
↪to them
            guardSnodes.map { guardSnode ->
                val result = listOf( guardSnode ) + (0 until (pathSize -␣
↪1)).map {
                    val pathSnode = unusedSnodes.getRandomElement()
                    unusedSnodes = unusedSnodes.minus(pathSnode)
                    pathSnode
                }
                Log.d("Loki", "Built new onion request path: $result.")
                result
            }
        }.map { paths ->
            OnionRequestAPI.paths = paths
            SnodeAPI.shared.broadcaster.broadcast("pathsBuilt")
            paths
        }
    }
}

// region Swarm API
internal fun getRandomSnode(): Promise<Snode, Exception> {
    if (snodePool.count() < minimumSnodePoolCount) {
        val target = seedNodePool.random()
```

(continues on next page)

(continued from previous page)

```
        val url = "$target/json_rpc"
        Log.d("Loki", "Populating snode pool using: $target.")
        val parameters = mapOf(
            "method" to "get_n_service_nodes",
            "params" to mapOf(
                "active_only" to true,
                "fields" to mapOf( "public_ip" to true, "storage_port" to␣
↪true, "pubkey_x25519" to true, "pubkey_ed25519" to true )
            )
        )
        val deferred = deferred<Snode, Exception>()
        deferred<Snode, Exception>(SnodeAPI.sharedContext)
        Thread {
            try {
                val json = HTTP.execute(HTTP.Verb.POST, url, parameters)
                val intermediate = json["result"] as? Map<*, *>
                val rawSnodes = intermediate?.get("service_node_states")␣
↪as? List<*>
                if (rawSnodes != null) {
                    val snodePool = rawSnodes.mapNotNull { rawSnode ->
                        val rawSnodeAsJSON = rawSnode as? Map<*, *>
                        val address = rawSnodeAsJSON?.get("public_ip") as?
↪ String
                        val port = rawSnodeAsJSON?.get("storage_port") as?
↪ Int
                        val ed25519Key = rawSnodeAsJSON?.get("pubkey_
↪ed25519") as? String
                        val x25519Key = rawSnodeAsJSON?.get("pubkey_x25519
↪") as? String
                        if (address != null && port != null && ed25519Key␣
↪!= null && x25519Key != null && address != "0.0.0.0") {
                            Snode("https://$address", port, Snode.
↪KeySet(ed25519Key, x25519Key))
                        } else {
                            Log.d("Loki", "Failed to parse: ${rawSnode?.
↪prettifiedDescription()}.")
                            null
                        }
                    }.toMutableSet()
  // [...]
                }
            }
        }
    }
  }
```

As one can see in the previous code listing, the same HTTP class using the X509 permissive **connection** object is used to communicate with seedNodes.

This is a serious security concern, as the current object does not verify the validity of those seedNodes.

A python script provided in the annexes (*Python code used to gather service nodes*) was also developed to display public keys returned by each server. The seedNodes servers are correctly provisioned with a valid Let's encrypt signed certificate.

This design flaw could be leveraged by a malicious actor, such as a DNS or Internet Access Provider to create a rogue seedNode and Snode, creating its own Lokinet network, redirecting all the clients.

We provided *Frida code to emulate network attack for Snode bootstrap mitm* in the annexes to demonstrate this attack. Its purpose is to emulate a DNS poisoning (by hooking the application) which in turn redirects the connection to a local rogue server.

Rogue Service Node Provider is provided as a test rogue server. It should be noted that the whole protocol implementation is not complete.

The result of this kind of attack is a total control over the distributed server in charge of routing and serving messages.

Implications of this attack are furtherly discussed in *Compromised server attacker (or server owner)*, since the same access level has been granted.

4.4.3 Cryptographic consideration

Regarding the invariant private key used to identify a **Session** user, we found that the generation does not follow the proper standards.

Frida code to understand private key generation or recovering is provided to illustrate how we managed to experiment this finding.

SESS-AND-04

The private key is generated using 16 bytes of randomness instead of 32. This is defined as a security weakness[1]:

> Large deviations from uniformity can eliminate all security. For example, if the 16 bytes of the secret key *n* were instead chosen as a public constant, then a moderately large computation would deduce the remaining bytes of *n* from the public key Curve25519(n;9). This is not Curve25519's fault; the user is responsible for putting enough randomness into keys.

The code in charge of the key creation is located in *RegisterActivity* and is called by *LandingActivity* when *STATE_WELCOME_SCREEN* is returned by org/thoughtcrime/securesms/PassphraseRequiredActionBarActivity.java#getApplicationState:

org/thoughtcrime/securesms/loki/activities/LandingActivity.kt

```
class LandingActivity : BaseActionBarActivity(),␣
↪LinkDeviceSlaveModeDialogDelegate {

    override fun onCreate(savedInstanceState: Bundle?) {
        //.. skipped
        super.onCreate(savedInstanceState)
        fakeChatView.startAnimating()
        registerButton.setOnClickListener { register() }
        //.. skipped
    }

    private fun register() {
        val intent = Intent(this, RegisterActivity::class.java)
        push(intent)
    }
```

org/thoughtcrime/securesms/loki/activities/RegisterActivity.kt

[1] https://cr.yp.to/ecdh/curve25519-20060209.pdf

```
override fun onCreate(savedInstanceState: Bundle?) {
    //.. skipped
    updateKeyPair()
}

private fun updateKeyPair() {
    val seedCandidate = Curve25519.getInstance(Curve25519.BEST).
↪generateSeed(16)
    try {
        this.keyPair = Curve.generateKeyPair(seedCandidate +␣
↪seedCandidate) // Validate the seed
    } catch (exception: Exception) {
        return updateKeyPair()
    }
    seed = seedCandidate
}
```

4.4.4 Privacy consideration

We ran two tests to estimate the network footprint of the **Session** application. The first one was carried out by selecting the recommended way of fetching notifications (Google Firebase solution). The second one was carried out by selecting the "slow mode" using Oxen server to handle notification aggregation. Both of them showed that all the requests were performed using onion request API, except for the notification ones ("https://live.apns.getsession.org"). This observation is really a good thing considering privacy concerns.

Session application registers/unregisters itself depending on the user selection regarding Firebase Google Push notification provider usage:

org.thoughtcrime.securesms.ApplicationContext

```
public void registerForFCMIfNeeded(Boolean force) {
  Context context = this;
  FirebaseInstanceId.getInstance().getInstanceId().
↪addOnCompleteListener(task -> {
    if (!task.isSuccessful()) {
      Log.w(TAG, "getInstanceId failed", task.getException());
      return;
    }
    String token = task.getResult().getToken();
    String userPublicKey = TextSecurePreferences.getLocalNumber(context);
    if (userPublicKey == null) return;
    if (TextSecurePreferences.isUsingFCM(this)) {
      LokiPushNotificationManager.register(token, userPublicKey, context,␣
↪force);
    } else {
      LokiPushNotificationManager.unregister(token, context);
    }
  });
}
```

SESS-AND-05

This (un)registering process crafts a post request sent with a default OkHttpClient which is not routed through Lokinet.

org.whispersystems.signalservice.loki.api.PushNotificationAcknowledgement

```
private val connection = OkHttpClient()

private val server by lazy {
    PushNotificationAcknowledgement.shared.server
}

fun register(token: String, publicKey: String, context: Context?, force:␣
↪Boolean) {
    val oldToken = TextSecurePreferences.getFCMToken(context)
    val lastUploadDate = TextSecurePreferences.
↪getLastFCMUploadTime(context)
    if (!force && token == oldToken && System.currentTimeMillis() -␣
↪lastUploadDate < tokenExpirationInterval) { return }
    val parameters = mapOf( "token" to token, "pubKey" to publicKey )
    val url = "$server/register"
    val body = RequestBody.create(MediaType.get("application/json"),␣
↪JsonUtil.toJson(parameters))
    val request = Request.Builder().url(url).post(body).build()
    connection.newCall(request).enqueue(object : Callback {
```

A network probe, Internet Access Provider, or DNS server is hence able to log clients requests to https://live.apns.getsession.org :

org.whispersystems.signalservice.loki.api.PushNotificationAcknowledgement

```
public class PushNotificationAcknowledgement private constructor(public␣
↪val server: String) {
    private val connection = OkHttpClient()

     companion object {
         lateinit var shared: PushNotificationAcknowledgement

         fun configureIfNeeded(isDebugMode: Boolean) {
             if (::shared.isInitialized) { return; }
             val server = if (isDebugMode) "https://dev.apns.getsession.org
↪" else "https://live.apns.getsession.org"
             shared = PushNotificationAcknowledgement(server)
         }
     }
```

We suggest, if possible, to route those network messages with onion request in order to prevent an attacker to locate **Session** users with passive network eavesdropping.

Regarding the data shipped in the Push notifications messages, we tracked down the messages from the PushNotificationService to the actual message deciphering and parsing mechanism:

org.thoughtcrime.securesms.loki.api.PushNotificationService

```
override fun onMessageReceived(message: RemoteMessage) {
    val base64EncodedData = message.data["ENCRYPTED_DATA"]
    val data = base64EncodedData?.let { Base64.decode(it) }
```

(continues on next page)

(continued from previous page)

```
    if (data != null) {
        try {
            val envelope = MessageWrapper.unwrap(data)
            PushContentReceiveJob(this).
↪processEnvelope(SignalServiceEnvelope(envelope), true)
```

org.thoughtcrime.securesms.jobs.PushReceivedJob

```
public void processEnvelope(@NonNull SignalServiceEnvelope envelope,␣
↪boolean isPushNotification) {
  synchronized (RECEIVE_LOCK) {
    try {
      if (envelope.hasSource()) {
        Address source = Address.fromExternal(context, envelope.
↪getSource());
        Recipient recipient = Recipient.from(context, source, false);

        if (!isActiveNumber(recipient)) {
          DatabaseFactory.getRecipientDatabase(context).
↪setRegistered(recipient, RecipientDatabase.RegisteredState.REGISTERED);
        }
      }

      if (envelope.isReceipt()) {
        handleReceipt(envelope);
      } else if (envelope.isPreKeySignalMessage() || envelope.
↪isSignalMessage()
          || envelope.isUnidentifiedSender() || envelope.
↪isFallbackMessage() || envelope.isClosedGroupCiphertext()) {
        handleMessage(envelope, isPushNotification);
      } else {
        Log.w(TAG, "Received envelope of unknown type: " + envelope.
↪getType());
      }
    } catch (Exception e) {
      Log.d("Loki", "Failed to process envelope due to error: " + e);
    }
  }
}
```

Then we hooked processEnvelope *PushReceived ProcessEnvelope processing*, looking for leaking data parsed into SignalServiceEnvelope but none were found.

Another point of concern is linked to the signal service used to handle captchas that relies on a javascript enabled webview that will not be routed through Lokinet.

org.thoughtcrime.securesms.registration.CaptchaActivity

```
@SuppressLint("SetJavaScriptEnabled")
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.captcha_activity);

  WebView webView = findViewById(R.id.registration_captcha_web_view);

  webView.getSettings().setJavaScriptEnabled(true);
```

(continues on next page)

(continued from previous page)

```
  webView.clearCache(true);

  webView.setWebViewClient(new WebViewClient() {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
      if (url != null && url.startsWith(SIGNAL_SCHEME)) {
        handleToken(url.substring(SIGNAL_SCHEME.length()));
        return true;
      }
      return false;
    }
  });

  webView.loadUrl("https://signalcaptchas.org/registration/generate.html");
}
```

This code is, however, part of signal implementation and may not be triggered (it is out of the audit scope anyway).

4.5 Compromised server attacker (or server owner)

This category of attacker is supposed to have compromised all of the Lokinet infrastructure, i.e. having remote root access in each service node and the seed node pool ("https://storage.seed1.loki.network", "https://storage.seed3.loki.network", "https://public.loki.foundation").

Implications of this seizure are:

- a complete mapping of the social relation graph of all the **Session** users;
- a selective denial of service for all or some users;
- a bad message delivery aimed to trigger undisclosed vulnerabilities;

It is really worth mentioning the fact that this could be the same for Signal servers.

It is also important to underline the fact that, unlike Signal, **Session** uses the user long-term public key as the main identifier, resulting in :

- the impossibility for a MITM/Server to impersonate a conversation by changing keys;
- the lack of phone number that could lead to an identity correlation.

4.6 Side applications on the same device

4.6.1 Clipboard

On Android 10, when a user copies the current lokiSeed (recovery passphrase) in the clipboard, remote apps are not allowed to access it[2].

[2] https://developer.android.com/about/versions/10/privacy/changes#clipboard-data

SESS-AND-06

However, this is not the case on Android < 10 and a specific attention is thus needed in order to protect the end user. Android application is currently copying the seed by using standard default api, hence further applications that will be put on the foreground could gather it.

org.thoughtcrime.securesms.loki.activities.SeedActivity

```
private fun copySeed() {
    revealSeed()
    val clipboard = getSystemService(Context.CLIPBOARD_SERVICE) as␣
↪ClipboardManager
    val clip = ClipData.newPlainText("Seed", seed)
    clipboard.primaryClip = clip
    Toast.makeText(this, R.string.copied_to_clipboard, Toast.LENGTH_SHORT).
↪show()
}
```

org.thoughtcrime.securesms.loki.dialogs.SeedDialog

```
private fun copySeed() {
    val clipboard = activity!!.getSystemService(Context.CLIPBOARD_SERVICE)␣
↪as ClipboardManager
    val clip = ClipData.newPlainText("Seed", seed)
    clipboard.primaryClip = clip
    Toast.makeText(context!!, R.string.copied_to_clipboard, Toast.LENGTH_
↪SHORT).show()
    dismiss()
}
```

4.6.2 Data blur

We checked if **org/thoughtcrime/securesms/loki/activities/SeedActivity.kt** is protected against a side application which has successfully acquired user acknowledgement[3].

This activity inherits from **org/thoughtcrime/securesms/BaseActionBarActivity.java** which have the following method to prevent screen recording:

```
private void initializeScreenshotSecurity() {
  if (TextSecurePreferences.isScreenSecurityEnabled(this)) {
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_SECURE);
  } else {
    getWindow().clearFlags(WindowManager.LayoutParams.FLAG_SECURE);
  }
}
```

[3] https://developer.android.com/reference/android/media/projection/MediaProjectionManager#createScreenCaptureIntent()

SESS-AND-07

However, as this is not the default setting, we hence suggest setting it as the default one.

Views that contain sensitive information have to check if overlays are present to prevent tapjacking attacks which, as some former studies explained it[4], are a way to gather interesting information.

For instance, Tapjacking attacks could trigger the copy button of the recovering passphrase dialog to furtherly leak it through the clipboard. A simple way to test if an overlay is present is to override the *onFilterTouchEventForSecurity*[5] from the activity and check MotionEvent's flags for MotionEvent.FLAG_WINDOW_IS_PARTIALLY_OBSCURED or MotionEvent.FLAG_WINDOW_IS_PARTIALLY_OBSCURED[6].

[4] https://www.ics.uci.edu/~alfchen/yuxuan_mobisys19.pdf

[5] https://developer.android.com/reference/android/view/View#onFilterTouchEventForSecurity(android.view.MotionEvent)

[6] https://developer.android.com/reference/android/view/MotionEvent#FLAG_WINDOW_IS_PARTIALLY_OBSCURED

5. iOS Evaluation Overview

5.1 Threat hypothesis

We considered the following types of attacker:

- Stolen device attacker;

  Even if just for a few minutes, this attacker has physical access to the device.

  - Locked device
  - Unlocked device

- Fully remote client-side attacker;
- Network omniscient attacker;
- Compromised server attacker (or server owner);
- Side applications in the same device;

5.2 Stolen device attacker

5.2.1 Locked device

When a notification is received, the name of the sender and the content of the message are not displayed if the phone is locked.

5.2.2 Unlocked device

There's no authentication required by the app, but the **Screen Lock** option can be toggled on to prevent rogue access without the ability to unlock the phone.

With the default configuration, **Session**'s screen preview is properly changed when appearing in the app switcher.

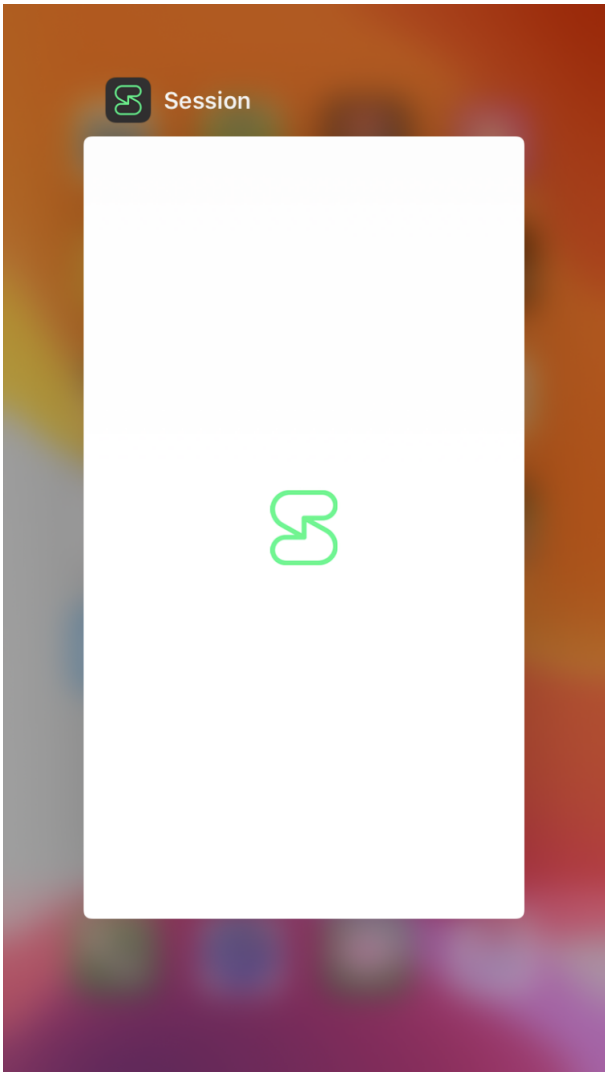


Fig. 5.1: Hidden screen when Session is in the background.

While the recovery phrase mnemonic is properly revealed by a long press the first time, it will stay like this upon closing and reopening the app. It might be preferable to stick with the first behavior at all times.

5.2.3 Data storage

Session stores sensitive user data in the same way as the Signal application. Messages and **Session** related data like the seed are stored inside an SQLCipher database. The key protecting the database is stored in the Keychain using the proper access attribute `kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly`. It prevents the key from being included in a backup and is the best choice regarding **Session** use case.

SESS-IOS-01

Attachments are stored in plaintext, however, the application does protect both the database and attachments folders from any backup:

Listing 5.1: session-ios/SignalServiceKit/src/Util/OWSFileSystem.m

```
57      NSDictionary *resourcesAttrs = @{ NSURLIsExcludedFromBackupKey :␣
↪@YES };
```

For consistency, **Session** should also encrypt the attachments on the device.

5.3 Fully remote client-side attacker

5.3.1 Closed Group

In the compiled **Session** application, `ClosedGroupsProtocol.isSharedSenderKeysEnable` is correctly set to true and uses Signal's Sender Key Protocol.

5.4 Network omniscient attacker

5.4.1 Service Node Gathering

Session properly builds its path through the onion Snode network (see the Encryption part for the random function used to pick a node).

5.4.2 TLS connection consideration

SESS-IOS-02

There's a lack of public key pinning when the client first establishes a connection to Lokinet's Snodes.

Listing 5.2: session-ios/SignalServiceKit/src/Loki/API/SnodeAPI.swift

```
60      internal static func getRandomSnode() -> Promise<Snode> {
61          if snodePool.count < minimumSnodePoolCount {
62              storage.dbReadConnection.read { transaction in
63                  snodePool = storage.getSnodePool(in: transaction)
64              }
65          }
66          if snodePool.count < minimumSnodePoolCount {
[...]
81                  HTTP.execute(.post, url, parameters: parameters,␣
↪useSeedNodeURLSession: true).map2 { json -> Snode in
```

Listing 5.3: session-ios/SignalServiceKit/src/Loki/API/Utilities/HTTP.swift

```
3  public enum HTTP {
4      private static let seedNodeURLSession = URLSession(configuration: .
↪ephemeral)
[...]
```

(continues on next page)

(continued from previous page)

```
44      public static func execute(_ verb: Verb, _ url: String,␣
↪parameters: JSON? = nil, timeout: TimeInterval = HTTP.timeout,␣
↪useSeedNodeURLSession: Bool = false) -> Promise<JSON> {
45          var request = URLRequest(url: URL(string: url)!)
46          request.httpMethod = verb.rawValue
47          if let parameters = parameters {
48              do {
49                  guard JSONSerialization.isValidJSONObject(parameters)␣
↪else { return Promise(error: Error.invalidJSON) }
50                  request.httpBody = try JSONSerialization.
↪data(withJSONObject: parameters, options: [ .fragmentsAllowed ])
51              } catch (let error) {
52                  return Promise(error: error)
53              }
54          }
55          request.timeoutInterval = timeout
56          let (promise, seal) = Promise<JSON>.pending()
57          let urlSession = useSeedNodeURLSession ? seedNodeURLSession :␣
↪defaultURLSession
```

Because this connection must be safe and trusted to prevent interaction with bad servers in case of a compromised CA or DNS, pinning must be implemented. See Apple's documentation[1] about the authentication challenge process to add certificate pinning.

5.4.3 Privacy consideration

SESS-IOS-03

Session app should pass through the Lokinet network to fetch an attachment instead of accessing the file server directly.

Listing 5.4: session-ios/SignalServiceKit/src/Messages/Attachments/OWSAttachmentDownloads.m

```
495 - (void)downloadFromLocation:(NSString *)location
496                          job:(OWSAttachmentDownloadJob *)job
497                      success:(void (^)(NSString␣
↪*encryptedDataPath))successHandler
498                      failure:(void (^)(NSURLSessionTask *_Nullable␣
↪task, NSError *error))failureHandlerParam
499 {
500     OWSAssertDebug(job);
501     TSAttachmentPointer *attachmentPointer = job.attachmentPointer;
502
503     AFHTTPSessionManager *manager = [AFHTTPSessionManager manager];
504     manager.requestSerializer = [AFHTTPRequestSerializer serializer];
505     [manager.requestSerializer␣
↪setValue:OWSMimeTypeApplicationOctetStream forHTTPHeaderField:@"Content-
↪Type"];
506     manager.responseSerializer = [AFHTTPResponseSerializer serializer];
507     manager.completionQueue = dispatch_get_global_queue(DISPATCH_QUEUE_
↪PRIORITY_DEFAULT, 0);
508
```

(continues on next page)

[1] https://developer.apple.com/documentation/foundation/url_loading_system/handling_an_authentication_challenge/performing_manual_server_trust_authentication

(continued from previous page)

```
509     // We want to avoid large downloads from a compromised or buggy service.
510     const long kMaxDownloadSize = 10 * 1024 * 1024;
511     __block BOOL hasCheckedContentLength = NO;
512
513     NSString *tempFilePath =
514         [OWSTemporaryDirectoryAccessibleAfterFirstAuth() stringByAppendingPathComponent:[NSUUID UUID].UUIDString];
515     NSURL *tempFileURL = [NSURL fileURLWithPath:tempFilePath];
516
517     __block NSURLSessionDownloadTask *task;
518     void (^failureHandler)(NSError *) = ^(NSError *error) {
519         OWSLogError(@"Failed to download attachment with error: %@", error.description);
520
521         if (![OWSFileSystem deleteFileIfExists:tempFilePath]) {
522             OWSLogError(@"Could not delete temporary file #1.");
523         }
524
525         failureHandlerParam(task, error);
526     };
527
528     NSString *method = @"GET";
529     NSError *serializationError = nil;
530     NSMutableURLRequest *request = [manager.requestSerializer requestWithMethod:method
531                                                                        URLString:location
532                                                                       parameters:nil
533                                                                            error:&serializationError];
534     if (serializationError) {
535         return failureHandler(serializationError);
536     }
537
538     task = [manager downloadTaskWithRequest:request
539         progress:^(NSProgress *progress) {
```

5.5 Cryptography

For most of the cryptographic operations[2] related to Lokinet, **Session** uses the CryptoSwift[3] library which is developed and maintained by only one open source freelance developer. This is why we advise to rely on Apple's CryptoKit[4] library instead. Indeed, the swift wrapper is also available[5] and is *suitable for use on Linux platforms.*

[2] /session-ios/SignalServiceKit/src/Loki/API/Utilities/EncryptionUtilities.swift
[3] https://github.com/krzyzanowskim/CryptoSwift
[4] https://developer.apple.com/documentation/cryptokit
[5] https://github.com/apple/swift-crypto

5.5.1 Random Data Generation

For random data generation, **Session** uses properly `SecRandomCopyBytes` by calling `getSecureRandomData`:

Listing 5.5: session-ios/SignalServiceKit/src/Loki/Utilities/Data+SecureRandom.swift

```
2  public extension Data {
3
4      /// Returns `size` bytes of random data generated using the␣
↪default secure random number generator. See
5      /// [SecRandomCopyBytes](https://developer.apple.com/documentation/
↪security/1399291-secrandomcopybytes) for more information.
6      public static func getSecureRandomData(ofSize size: UInt) -> Data? {
7          var data = Data(count: Int(size))
8          let result = data.withUnsafeMutableBytes {␣
↪SecRandomCopyBytes(kSecRandomDefault, Int(size), $0.baseAddress!) }
9          guard result == errSecSuccess else { return nil }
10          return data
11      }
12  }
```

5.5.2 Private Key Generation

The new method to generate the user private key is the following:

Listing 5.6: session-ios/SignalServiceKit/src/Loki/Utilities/Data+SecureRandom.swift

```
137  private func updateKeyPair() {
138      let padding = Data(repeating: 0, count: 16)
139      ed25519KeyPair = Sodium().sign.keyPair(seed: (seed + padding).
↪bytes)!
140      let x25519PublicKey = Sodium().sign.toX25519(ed25519PublicKey:␣
↪ed25519KeyPair.publicKey)!
141      let x25519SecretKey = Sodium().sign.toX25519(ed25519SecretKey:␣
↪ed25519KeyPair.secretKey)!
142      x25519KeyPair = ECKeyPair(publicKey: Data(x25519PublicKey),␣
↪privateKey: Data(x25519SecretKey))
143  }
```

Listing 5.7: /session-ios/Signal/src/Loki/Utilities/Sodium+Conversion.swift

```
29      public func toX25519(ed25519SecretKey: SecretKey) -> SecretKey? {
30          var x25519SecretKey = SecretKey(repeating: 0, count: 32)
31
32          // FIXME: It'd be nice to check the exit code here, but all␣
↪the properties of the object
33          // returned by the call below are internal.
34          let _ = crypto_sign_ed25519_sk_to_curve25519 (
35              &x25519SecretKey,
36              ed25519SecretKey
37          )
38
39          return x25519SecretKey
40      }
```

SESS-IOS-04

As Oxen repository shipped a compiled version of the sodium library, we found a version that actually matches the compiled code for **crypto_sign_ed25519_sk_to_curve25519** :

Listing 5.8: https://github.com/jedisct1/libsodium/blob/927dfe8e2eaa86160d3ba12a7e3258fbc322909c/src/libsodium/crypto_sign/ed25519/ref10/keypair.c#L69

```
int
crypto_sign_ed25519_sk_to_curve25519(unsigned char *curve25519_sk,
                                     const unsigned char *ed25519_sk)
{
    unsigned char h[crypto_hash_sha512_BYTES];

    crypto_hash_sha512(h, ed25519_sk, 32);
    h[0] &= 248;
    h[31] &= 127;
    h[31] |= 64;
    memcpy(curve25519_sk, h, crypto_scalarmult_curve25519_BYTES);
    sodium_memzero(h, sizeof h);

    return 0;
}
```

We still consider that the entropy of the produced key does not fit the standard recommendation. According to the reference paper[6] :

> The legitimate users are assumed to generate independent uniform random secret keys.

However, 16 bytes of entropy passing through a hash function could be considered as a valid random secret key since SHA-512 can be considered as a pseudorandom function and Curve25519 offer 128 bits of security.

Also, it might be preferable to use a better domain separation tag instead of null bytes for padding the seed.

5.6 SESS-IOS-05 Logging

Logs are stored by **Session** application under `Library/Logs/`.

They contain sensitive data about conversations like the timestamps of the messages and the identity of the sender and receiver. Moreover, in the main configuration file of the app (`Info.plist`), the `LOGS_URL` and `LOGS_EMAIL` keys are still set to `https://github.com/WhisperSystems/Signal-iOS/issues` and `support@whispersystems.org`. If logs are not needed, Oxen should at least disable them (or better, completely remove it from production builds):

Listing 5.9: session-ios/Signal/src/ViewControllers/AppSettings/AdvancedSettingsTableViewController.m

```
283          OWSLogInfo(@"disabling logging.");
284          [[DebugLogger sharedLogger] wipeLogs];
285          [[DebugLogger sharedLogger] disableFileLogging];
```

Otherwise, data like users identity and timestamps should not be written. Also, when sending logs,

[6] https://cr.yp.to/ecdh/curve25519-20060209.pdf

it must pass through the Lokinet anonymization network and no information that uniquely identifies a device should be appended.

5.7 Side applications on the same device

5.7.1 Clipboard

The recovery phrase mnemonic derived from the *seed* can be copied to the pasteboard:

Listing 5.10: session-ios/Signal/src/Loki/View Controllers/SeedVC.swift

```
180          UIPasteboard.general.string = mnemonic
```

SESS-IOS-06

Using the systemwide general pasteboard should be avoided when dealing with sensitive data. Otherwise, an app monitoring the board could retrieve the **seed** if brought to foreground.

5.7.2 Screen recording

SESS-IOS-07

The app is not protected against screen recording. It can be done by checking the `UIScreen` instance property `isCaptured` and monitoring the change of state with `capturedDidChangeNotification`.

6. Desktop Evaluation Overview

6.1 Threat hypothesis

We considered the following types of attacker:

- Stolen/Compromised computer attacker

  Even if just for a few minutes, this attacker has either a physical or a remote access to the computer running Session.

  - Session is running during the attacker access
  - Session is not running during the attacker access

- Fully remote client-side attacker

  This attacker only owns a client and is able to connect in a more or less legitimate way to the **Session** infrastructure.

- Network omniscient attacker

  This attacker is supposed to have a global view of all the traffic between the Session users and the servers they connect to.

6.2 Stolen/Compromised computer attacker

6.2.1 Session is not running during the attacker access

Messages are kept in a sqlcipher database[1] located in **${HOME}/.config/Session/sql/db.sqlite**. Database is handled by a Node.js binding of SQLcipher[2].

By default, the hex key to open this database is stored in the config.json file :

```
cat ${HOME}/.config/Session/config.json | jq
{
  "key": "e4cd3072fa25eb571cbd071c135bea88dabd95497b3dfddb6a770a4988c82ea9
↪",
  "dbHasPassword": false
}
```

```
sqlite> PRAGMA key = "x
↪'e4cd3072fa25eb571cbd071c135bea88dabd95497b3dfddb6a770a4988c82ea9'";
ok
sqlite> PRAGMA cipher_migrate;
0
sqlite> select * from identityKeys;
05d3a267c6c60b338c7e98e1c26d041db0bdad24dd9bab8ccb965191d3b17dd748|{"id":
↪"05d3a267c6c60b338c7e98e1c26d041db0bdad24dd9bab8ccb965191d3b17dd748",
↪"publicKey":"BdOiZ8bGCzOMfpjhwm0EHbC9rSTdm6uMy5ZRkdOxfddI","firstUse
↪":true,"timestamp":1616596496902,"nonblockingApproval":true}
```

(continues on next page)

[1] https://www.zetetic.net/sqlcipher/sqlcipher-api/#cipher_salt
[2] https://github.com/sqlcipher/sqlcipher/blob/0663d8500204e14bd2bb0ca25162d91e4555528d/src/crypto_impl.c#L757

(continued from previous page)

```
sqlite> select * from messages;
f1776cdb-63f6-4092-bb3c-bc4976e45aa4|{/*skipped*/}|||0|1616596684045|10|/
↪*skipped*/|1616596690767|/*skipped*/|1|0|||||incoming|Hi Alice||
```

This default behavior can be switched to a more secure mode if the user set up a custom password in the Settings => Privacy => Change Account Password.

Nonetheless, when the user password is set, it is used as follows:

/session-desktop/main.js

```
// Password screen related IPC calls
ipc.on('password-window-login', async (event, passPhrase) => {
  const sendResponse = e =>
    event.sender.send('password-window-login-response', e);

  try {
    const passwordAttempt = true;
    await showMainWindow(passPhrase, passwordAttempt);
```

/session-desktop/main.js

```
async function showMainWindow(sqlKey, passwordAttempt = false) {
  const userDataPath = await getRealPath(app.getPath('userData'));

  await sql.initialize({
    configDir: userDataPath,
    key: sqlKey,
    messages: locale.messages,
    passwordAttempt,
  });
```

/session-desktop/app/sql.js

```
async function initialize({ configDir, key, messages, passwordAttempt }) {
  try {
    const sqlInstance = await openDatabase(filePath);
    const promisified = promisify(sqlInstance);

    // promisified.on('trace', async statement => {
    //   if (!db || statement.startsWith('--')) {
    //     console._log(statement);
    //     return;
    //   }
    //   const data = await db.get(`EXPLAIN QUERY PLAN ${statement}`);
    //   console._log(`EXPLAIN QUERY PLAN ${statement}\n`, data && data.
↪detail);
    // });

    try {
      await setupSQLCipher(promisified, { key });
```

While the user interface constraints the password to be between 6 and 64-character long, the latter was first thought to be suffering from several dictionary attacks.

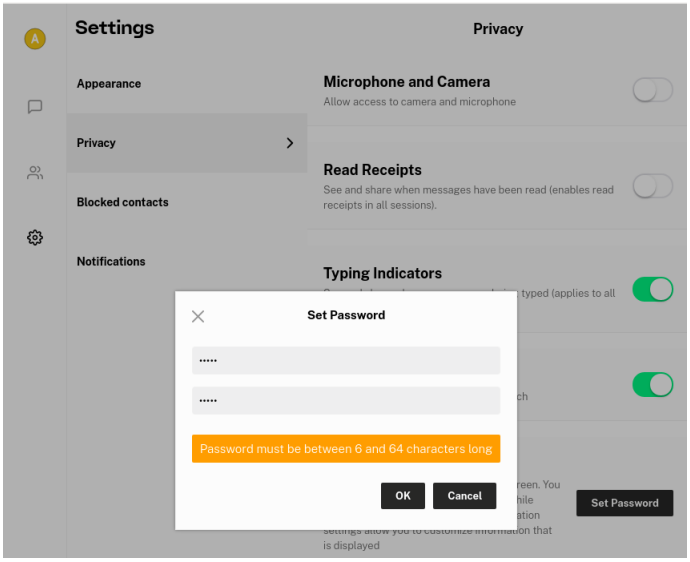


Fig. 6.1: User provided password have a mandatory policy

After further discussion, Oxen outlined the fact that sqlcipher uses an randomly generated salt[3]. We then checked that the salt was effectively randomly generated by default when the database was initially generated[4]. This salt is used even with the default hex key that is employed when the user has not provided his own password yet.

6.2.2 Privacy concerns

Session generates various logs in the user home directory from which a potential threat actor could retrieve onion circuits IP and access timestamps:

```
cat ${HOME}/.config/Session/logs/logs.logs | jq
{
  "name": "log",
  "hostname": "cdeaa5abf5ad",
  "pid": 220,
  "level": 30,
  "msg": "Built 3 onion paths [
        {\"path\":[{
            \"ip\":\"161.35.238.132\",\"port\":22021,\"pubkey_x25519\":\
↪"4c2a8c5760b7232db2286d5cb3b4478951594df1fe6c4f0a33e9caaa4a53ff7c\",\
↪"pubkey_ed25519\":\
↪"2be3bdf306fa8fed22d4e639cf578723e0fff99ba25a7a9ff2aef74bee6d673f\",\
↪"version\":\"\"},
            {\"ip\":\"46.188.92.16\",\"port\":22021,\"pubkey_x25519\":\
↪"41c712e48ae53a45ddaa41f5d2a92fe9dbbedbd35232536779f3c5511778546b\",\
↪"pubkey_ed25519\":\
↪"6b2ef245c9d2fe5363571a1f9a2f9151147626af9b32bb90a9e1ee56c96fd8ac\",\
↪"version\":\"\"},
            {\"ip\":\"116.203.136.1\",\"port\":22021,\"pubkey_x25519\":\
↪"4975df59ec5c0015368a32900e614fbf21f02c4331855836dc981dc546b3082c\",\
↪"pubkey_ed25519\":\
↪"492edaa8001897e45cc57fadca56a40b9c92af72397efb3fcd3a741775e19bba\",\
↪"version\":\"\"}
```

(continues on next page)

[3] https://github.com/journeyapps/node-sqlcipher
[4] https://github.com/sqlcipher/sqlcipher

(continued from previous page)


```
            ],\"bad\":false},]",
  "time": "2021-03-26T12:43:37.225Z",
  "v": 0
}
{
  "name": "log",
  "hostname": "cdeaa5abf5ad",
  "pid": 220,
  "level": 50,
  "msg": "[path] Error on the path: 163.172.149.9:22021, 199.195.251.
↪67:22021, 88.99.13.182:19813 to 66.42.41.66:22021",
  "time": "2021-03-26T14:04:40.083Z",
  "v": 0
}
{
  "name": "log",
  "hostname": "cdeaa5abf5ad",
  "pid": 220,
  "level": 40,
  "msg": "loki_message:::_retrieveNextMessages - lokiRpc could not talk to␣
↪66.42.41.66:22021",
  "time": "2021-03-26T14:04:40.084Z",
  "v": 0
}
```


SESS-DES-01

To prevent this leak of information, we highly recommend removing those logs or cipher them with the current password.

6.2.3 Session is running during the attacker access

When a user decides to switch to a password-protected database, this process happens immediately. This prevents potential theft when migrations occur or in the event the application is kept in a running state inside a virtual machine. Settings are also password protected, preventing threats from altering the password for message recovery purposes.

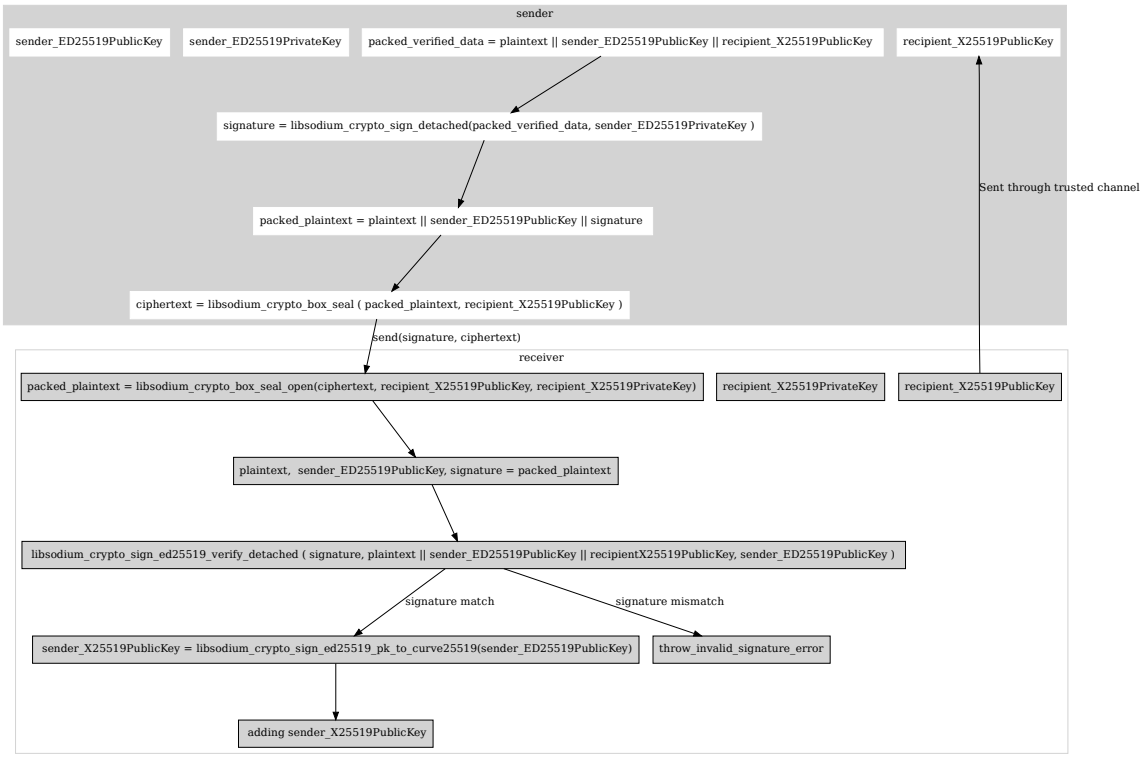
6.3 Fully remote client-side attacker

From a completely remote client-side point of view, we identified that the following interactions are possible on **Session** infrastructure:

- send and retrieve messages;
- upload and download files (avatar pictures, stickers or message attachments);
- create and update Groups.

6.3.1 Private Conversations

A new protocol has been implemented for this **Session** version to replace Signal protocol. The following figure illustrates our understanding of this protocol :



Private conversation protocol : unidentified sender

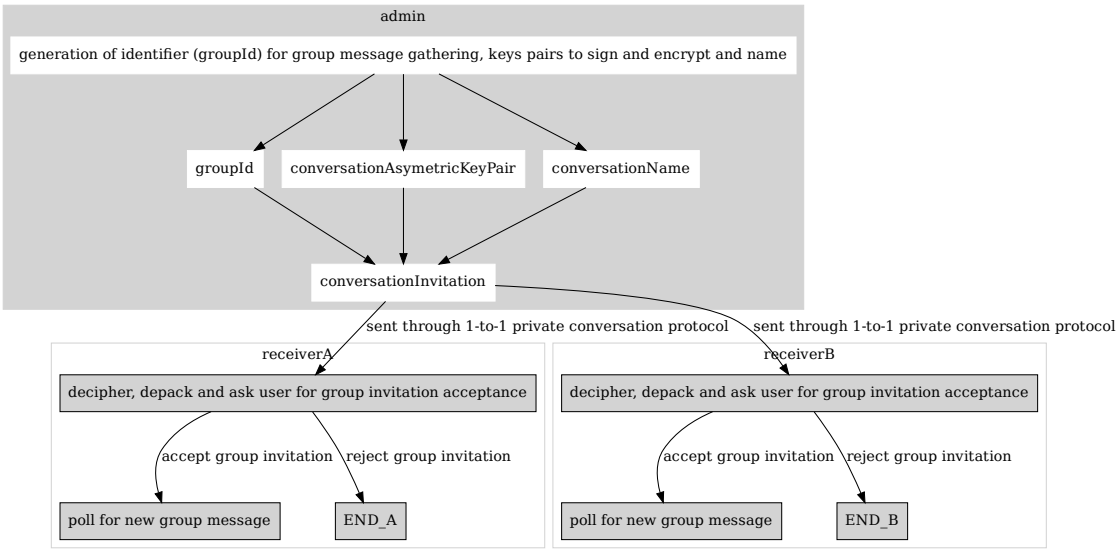
Even though there is no ratchet mechanism as in Signal, no correlation exists between ciphering keys over time. This observation is made on the basis that crypto_box_seal[5] creates *a new key pair for each message, and attaches the public key to the ciphertext*. **crypto_box_seal** creates an ephemeral keypair and uses the secret part with the recipient public key to craft a symmetric key in charge of ciphering messages. The recipient will extract the ephemeral public key from the ciphered message and will use their private key to regenerate the ephemeral symmetric key for this message.

Sender message authentication is correctly provided by crypto_sign_detached[6].

[5] https://libsodium.gitbook.io/doc/public-key_cryptography/sealed_boxes#usage
[6] https://libsodium.gitbook.io/doc/public-key_cryptography/public-key_signatures#detached-mode

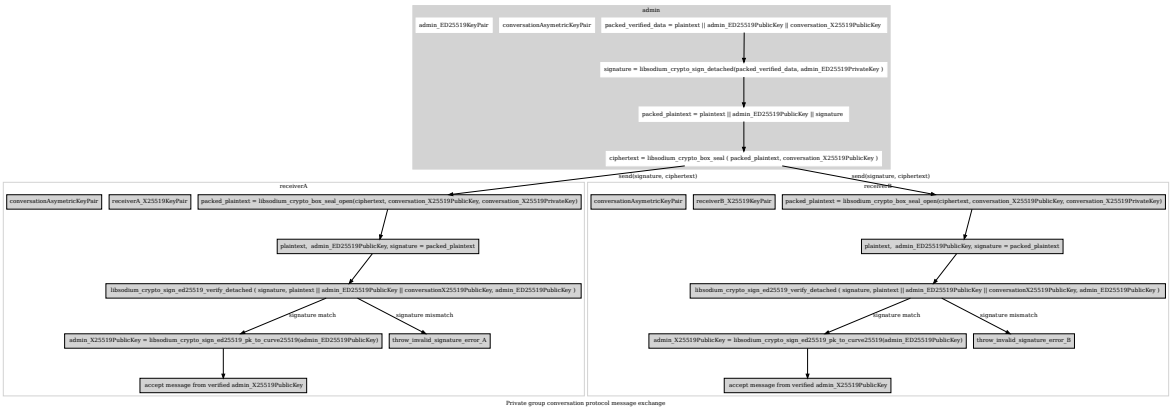
6.3.2 Private Group Conversations

Private groups take advantage of 1-to-1 conversation protocol, enabling the group creator (called admin) to initially add existing contact members to the group. The following figure summarizes our understanding of how this protocol works:



Private group conversation protocol invitation

The following figure shows a message sent by the admin for the whole group. The same mechanism would be employed whether the sender be A or B:



Private group conversation protocol message exchange

6.3.3 Sandboxing and Electron security

We tried to determine if the Electron framework security criteria have been respected in the development process. To quickly verify this aspect, we used the electronegativity[7] tool by DoyenSec. A summary of these results is presented in the appendices *Electronegativity Report*.

All those results were examined but none of them seems to lead to a real problem. Specific attention has been paid to the isolation context since if the contextIsolation is not enabled then malicious JS code execution of the Node's APIs will be allowed.

/session-desktop/main.js

[7] https://github.com/doyensec/electronegativity

```
async function createWindow() {
  const { screen } = electron;
  const { minWidth, minHeight, width, height } = getWindowSize();
  const windowOptions = Object.assign(
    {
      show: !startInTray, // allow to start minimised in tray
      width,
      height,
      minWidth,
      minHeight,
      autoHideMenuBar: false,
      backgroundColor: '#fff',
      webPreferences: {
        nodeIntegration: false,
        nodeIntegrationInWorker: false,
        contextIsolation: false,
        preload: path.join(__dirname, 'preload.js'),
        nativeWindowOpen: true,
        spellcheck: await getSpellCheckSetting(),
      },
      icon: path.join(__dirname, 'images', 'session', 'session_icon_64.png
↪'),
    },
```

contextIsolation aims to provide a different execution context for:

- loaded pages;
- electron's internal code and **Session** preloaded page.

Our concerns are gone since no external webcontent seems to be loaded from non-isolated BrowserWindows.

6.4 Network omniscient attacker

6.4.1 Update mechanisms

We also paid attention to the update mechanism. This feature relies on electron build system[8].

/session-desktop/ts/updater/updater.ts

```
import { autoUpdater, UpdateInfo } from 'electron-updater';

export async function start(
  getMainWindow: () => BrowserWindow,
  messages: MessagesType,
  logger: LoggerType
) {
  if (interval) {
    logger.info('auto-update: Already running');

    return;
  }
```

(continues on next page)

[8] https://www.electronjs.org/docs/tutorial/context-isolation

(continued from previous page)

```
  logger.info('auto-update: starting checks...');
// [Code skipped...]
await checkForUpdates(getMainWindow, messages, logger);
// [Code skipped...]
}

async function checkForUpdates(
  getMainWindow: () => BrowserWindow,
  messages: MessagesType,
  logger: LoggerType
) {
  if (stopped || isUpdating || downloadIgnored) {
    return;
  }

//[Code skipped...]
  // Get the update using electron-updater
  const result = await autoUpdater.checkForUpdates();
//[Code skipped...]
  const hasUpdate = isUpdateAvailable(result.updateInfo);
  if (!hasUpdate) {
    logger.info('auto-update: no update available');

    return;
  }

  logger.info('auto-update: showing download dialog...');
  const shouldDownload = await showDownloadUpdateDialog(
   getMainWindow(),
    messages
  );
  if (!shouldDownload) {
    downloadIgnored = true;

    return;
  }

  await autoUpdater.downloadUpdate();

// [Code skipped...]

  // Update downloaded successfully, we should ask the user to update
  logger.info('auto-update: showing update dialog...');
  const shouldUpdate = await showUpdateDialog(getMainWindow(), messages);
  if (!shouldUpdate) {
    return;
  }

  logger.info('auto-update: calling quitAndInstall...');
  markShouldQuit();
  autoUpdater.quitAndInstall();
```

This is a standard way of handling updates in an electron-based application. Updated URLs are located in *app-update.yml* which is generated from *package.json*.

/session-desktop/ts/updater/updater.ts

```
/*
  Check if we have the required files to auto update.
  These files won't exist inside certain formats such as a linux deb file.
*/
async function canAutoUpdate(): Promise<boolean> {
  const isPackaged = app.isPackaged;

  // On a production app, we need to use resources path to check for the␣
↪file
  if (isPackaged && !process.resourcesPath) {
    return false;
  }

  // Taken from: https://github.com/electron-userland/electron-builder/
↪blob/d4feb6d3c8b008f8b455c761d654c8088f90d8fa/packages/electron-updater/
↪src/ElectronAppAdapter.ts#L25
  const updateFile = isPackaged ? 'app-update.yml' : 'dev-app-update.yml';
```

/session-desktop/package.json

```
"repository": {
  "type": "git",
  "url": "https://github.com/loki-project/session-desktop.git"
},
```

/nodes-modules/electron-updater/src/providers/GitHubProvider.ts

```
private async getLatestVersionString(cancellationToken:␣
↪CancellationToken): Promise<string | null> {
  const options = this.options
  // do not use API for GitHub to avoid limit, only for custom host or␣
↪GitHub Enterprise
  const url = (options.host == null || options.host === "github.com") ?
    newUrlFromBase(`${this.basePath}/latest`, this.baseUrl) :
    new URL(`${this.computeGithubBasePath(`/repos/${options.owner}/$
↪{options.repo}/releases`)}/latest`, this.baseApiUrl)
  try {
    const rawData = await this.httpRequest(url, {Accept: "application/json
↪"}, cancellationToken)
    if (rawData == null) {
      return null
    }

    const releaseInfo: GithubReleaseInfo = JSON.parse(rawData)
    return (releaseInfo.tag_name.startsWith("v")) ? releaseInfo.tag_name.
↪substring(1) : releaseInfo.tag_name
  }
  catch (e) {
    throw newError(`Unable to find latest version on GitHub (${url}),␣
↪please ensure a production release exists: ${e.stack || e.message}`,
↪"ERR_UPDATER_LATEST_VERSION_NOT_FOUND")
  }
}

private get basePath() {
  return `/${this.options.owner}/${this.options.repo}/releases`
}
```

SESS-DES-02

The computed URL enabling to check for updates is **https://github.com/loki-project/session-desktop/releases/latest**. Latest releases are fetched with this.httpRequest method that does not use the anonymous routing infrastructure. This could lead to a passive network discovery and tracking of **Session** users.

This was confirmed with a network capture carried out during the app launch where 3 DNS queries have been discovered for:

- github.com;
- github-releases.githubusercontent.com;
- public.loki.foundation.

7. Conclusion

Oxen **Session** really improves Signal privacy and resilience by using an overlay network to the existent end-to-end encryption instant messaging solution. The onion-routing mechanisms make use of Oxen's Snodes to store and exchange messages, however, there are some other centralized standard web services that are still used through the overlay network (for the push service and to deliver attachments files).

All major concerns have quickly been fixed.

The overall security level of this application is good. With slight exceptions, all the good practices have been kept in mind when developing this product in each platform specificities as well as in the global architecture of **Session**.

8. Annex

8.1 Python code used to gather service nodes

```
import random
import requests
import json
url = "{}/json_rpc".format( random.choice(["https://storage.seed1.loki.
↪network", "https://storage.seed3.loki.network", "https://public.loki.
↪foundation"]))
headers = {'Content-Type': 'application/json; charset=utf-8'}

s = requests.Session()
param = {'method': 'get_n_service_nodes', 'params': {'active_only': True,
↪'fields': {'public_ip': True, 'storage_port': True, 'pubkey_x25519':␣
↪True, 'pubkey_ed25519': True}}, 'jsonrpc': '2.0'}
r = s.post(url,json=param, headers = headers)

assert(200 == r.status_code)

snodes_resp = json.loads(r.content)

assert("OK" == snodes_resp["result"]["status"])

snode_list = snodes_resp["result"]["service_node_states"]

#json list of
#{
#    'pubkey_ed25519' :
↪'a43a1a03c42144d8f875f909ca4bee96f07078577d5adf9305b307c90986974b',
#    'pubkey_x25519'  :
↪'f1f2e34ad45156b02091bf7870d80eeb3a052439cadb35a3f1aab7c77d1a4a7f',
#    'public_ip'      : '51.68.197.123',
#    'storage_port'   : 22021
#}
```

8.2 Python code to encode Loki seed

This code can be used with different word list available in the **Session** apk :

```
unzip -l session-1.3.2-universal.apk | grep mnemonic
11467  1980-00-00 00:00   assets/mnemonic/english.txt
19428  1980-00-00 00:00   assets/mnemonic/japanese.txt
13139  1980-00-00 00:00   assets/mnemonic/portuguese.txt
10898  1980-00-00 00:00   assets/mnemonic/spanish.txt
```

```
import zlib
english_wordlist = open("assets/mnemonic/english.txt", 'r').read().split(",
↪")
# https://github.com/loki-project/session-android-service/blob/master/java/
↪src/main/java/org/whispersystems/signalservice/loki/crypto/MnemonicCodec.
↪kt
english_prefix_length = 3
```

(continues on next page)

(continued from previous page)

```
japanese_prefix_length  = 3
portuguese_prefix_length  = 4
spanish_prefix_length  = 4

def getCheckSumIndex(words, prefix_length):
     b = "".join([w[:prefix_length] for w in words])
     return zlib.crc32(b.encode("utf8")) % len(words)

def encode(hexstr, wordlist, prefix_length):
  result_list=[]
  wordlist_len = len(wordlist)

  for i in range(0,len(hexstr),8):
    swappedstr = hexstr[i+6]
    swappedstr += hexstr[i+7]
    swappedstr += hexstr[i+4]
    swappedstr += hexstr[i+5]
    swappedstr += hexstr[i+2]
    swappedstr += hexstr[i+3]
    swappedstr += hexstr[i+0]
    swappedstr += hexstr[i+1]
    hexstr = hexstr[:i] + swappedstr + hexstr[i+8:]

  for i in range(0,len(hexstr),8):
    swapped_to_int = int(hexstr[i:i+8],16)
    w1 = swapped_to_int % wordlist_len
    w2 = (int(swapped_to_int/wordlist_len)  + w1) % wordlist_len
    w3 = (int( int(swapped_to_int / wordlist_len)  / wordlist_len) + w2) %␣
↪wordlist_len
    result_list.extend([ english_wordlist[int(w)] for w in [w1,w2,w3]])
  result_list.append(result_list[getCheckSumIndex(result_list, prefix_
↪length)])
  return " ".join(result_list)

loki_seed = 'a506168745c65a497514cb2fb00587c0'

print(encode(loki_seed,english_wordlist , english_prefix_length ))
# amply equip slug fuming soprano azure irate rebel umpire mural exit␣
↪agenda equip
```

8.3 Python code used for file server interaction

```
import requests
import uuid
import base64
import json
from cryptography.hazmat.primitives.asymmetric.x25519 import␣
↪X25519PrivateKey
from cryptography.hazmat.primitives.asymmetric.x25519 import␣
↪X25519PublicKey
from cryptography.hazmat.primitives.ciphers.algorithms import AES
from cryptography.hazmat.primitives.ciphers.modes import CBC
from cryptography.hazmat.primitives.ciphers import Cipher
```

(continues on next page)

(continued from previous page)

```
from cryptography.hazmat.backends import default_backend

submitFileServerPublicKey =
↪"62509D59BDEEC404DD0D489C1E15BA8F94FD3D619B01C1BF48A9922BFCB7311C"

# walleye:/data/data/network.loki.messenger/shared_prefs # grep pref_
↪identity_private_v3 *
# SecureSMS-Preferences.xml:    <string name="pref_identity_private_v3">
↪yKAMUCNJtkNL3eJchtzOic+gDFAjSbZDS93iXIbczkk=</string>
# walleye:/data/data/network.loki.messenger/shared_prefs # grep pref_
↪identity_public_v3 *
# SecureSMS-Preferences.xml:    <string name="pref_identity_public_v3">
↪BW8buG3Z6ekTkJxNV3plnfLhEz1kp+Rm1TH2Ux343KsJ</string>

pubKey  =
↪"056f1bb86dd9e9e913909c4d577a659df2e1133d64a7e466d531f6531df8dcab09"
privKey = "c8a00c502349b6434bdde25c86dcce89cfa00c502349b6434bdde25c86dcce49
↪"
token = ""

def pkcs5_unpad(p):
    return p[:len(p)-p[-1]]

def ECDH_decrypt(cipheredToken, serverPubKey, privKey ):
    pub = X25519PublicKey.from_public_bytes(serverPubKey)
    priv = X25519PrivateKey.from_private_bytes(privKey)
    shared_key = priv.exchange(pub)
    iv = cipheredToken[:16]
    cipheredText = cipheredToken[16:]
    c = Cipher( AES(shared_key), CBC(iv), backend = default_backend())
    d = c.decryptor()
    return pkcs5_unpad(d.update(cipheredText) + d.finalize())

# Get Auth Token
charl_url = "https://file.getsession.org/loki/v1/get_challenge"
submit_charl_url = "https://file.getsession.org/loki/v1/submit_challenge"
submit_files = "https://file.getsession.org/files"
request_real_token_for_open_groups = False
token = "loki"

s = requests.Session()
if request_real_token_for_open_groups :
    r = s.get(charl_url, params = {'pubKey':pubKey  })
    chal = json.loads(r.content)
    cipheredToken = base64.b64decode(chal["cipherText64"])
    serverPubKey = base64.b64decode(chal["serverPubKey64"])
    if len(serverPubKey) == 33 :
        serverPubKey = serverPubKey[1:]
    pretoken = ECDH_decrypt(cipheredToken, serverPubKey, bytes.
↪fromhex(privKey))
    token = str(pretoken, 'utf-8')
    #submit may be carried out through onion request, meaning that the␣
↪onion exit node could know the auth token
    r = s.post(submit_charl_url, data =  {"pubKey" : pubKey, 'token' :␣
↪token } )
    if r.status_code==200:
```

(continues on next page)

(continued from previous page)

```
        print( "Auth granted {}".format(s.headers["Authorization"]) )

s.headers["Authorization"] = "Bearer {}".format(token)

payload = b"\x4b"*20
payload_content_type = "application/octet-stream"
payload_name = str(uuid.uuid1())

def post_file(s,payload, payload_content_type, payload_name):
    return  s.post(url=submit_files,
    files=(
      ("type", (None,"network.loki")),
      ("Content-Type",(None,payload_content_type )),
     ("content", ( payload_name , payload ))
    )
    )

r = post_file(s,b"\x4b"*20, "application/octet-stream", payload_name )
content = json.loads(r.content)
print(json.dumps(content["data"], indent=3, sort_keys=True))

r = post_file(s,b"\x4b"*20, "application/octet-stream", payload_name )
content = json.loads(r.content)
print(json.dumps(content["data"], indent=3, sort_keys=True))

# This cause a 500 error and a crash
#r = s.post(url=submit_files,
#    files={"type":"network.loki", "Content-Type":"application/octet-stream
↪", "content": "zoeife" } )
# r.content ==
#b'\n\n\n\n\n\n\n
#MulterError: Unexpected field
#    at wrappedFileFilter (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/multer/index.js:40:19)
#    at Busboy.<anonymous> (/root/production/loki-file-server-refactor/
↪loki/server/node_modules/multer/lib/make-middleware.js:114:7)
#    at Busboy.emit (events.js:209:13)
#    at Busboy.emit (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/busboy/lib/main.js:38:33)
#    at PartStream.<anonymous> (/root/production/loki-file-server-refactor/
↪loki/server/node_modules/busboy/lib/types/multipart.js:213:13)
#    at PartStream.emit (events.js:209:13)
#    at HeaderParser.<anonymous> (/root/production/loki-file-server-
↪refactor/loki/server/node_modules/dicer/lib/Dicer.js:51:16)
#    at HeaderParser.emit (events.js:209:13)
#    at HeaderParser._finish (/root/production/loki-file-server-refactor/
↪loki/server/node_modules/dicer/lib/HeaderParser.js:68:8)
#    at SBMH.<anonymous> (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/dicer/lib/HeaderParser.js:40:12)
#    at SBMH.emit (events.js:209:13)
#    at SBMH._sbmh_feed (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/streamsearch/lib/sbmh.js:159:14)
#    at SBMH.push (/root/production/loki-file-server-refactor/loki/server/
↪node_modules/streamsearch/lib/sbmh.js:56:14)
#    at HeaderParser.push (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/dicer/lib/HeaderParser.js:46:19)
```

(continues on next page)

(continued from previous page)

```
#    at Dicer._oninfo (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/dicer/lib/Dicer.js:197:25)
#    at SBMH.<anonymous> (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/dicer/lib/Dicer.js:127:10)
#    at SBMH.emit (events.js:209:13)
#    at SBMH._sbmh_feed (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/streamsearch/lib/sbmh.js:159:14)
#    at SBMH.push (/root/production/loki-file-server-refactor/loki/server/
↪node_modules/streamsearch/lib/sbmh.js:56:14)
#    at Dicer._write (/root/production/loki-file-server-refactor/loki/
↪server/node_modules/dicer/lib/Dicer.js:109:17)
#    at doWrite (_stream_writable.js:428:12)
#    at writeOrBuffer (_stream_writable.js:412:5)
#\n\n\n'
```

8.4 Frida code to understand private key generation or recovering

```
function javaByteArrayToString(barr){
   var ret = "";
   for(var i=0; i<barr.length;i++){
      ret += (barr[i] & 0xff).toString(16);
   }
  return ret;
}

function understandRecoveryPhraseCrypto(){
  var Curve = Java.use("org.whispersystems.libsignal.ecc.Curve");
  Curve.generateKeyPair.overload("[B").implementation = function(barr){
    console.log("Curve.generateKeyPair with this random :
↪"+javaByteArrayToString(barr))
    printBacktrace();
    return this.generateKeyPair(barr)
  }
  var JavaCurveProvider = Java.use("org.whispersystems.curve25519.
↪JavaCurve25519Provider");
  JavaCurveProvider.generatePrivateKey.overload().implementation =␣
↪function(){
    console.log("generatePrivateKey( );");
    printBacktrace();
    return this.generatePrivateKey();
  }
  JavaCurveProvider.generatePrivateKey.overload("[B").implementation =␣
↪function(barr){
    console.log("generatePrivateKey( "+javaByteArrayToString(barr)+");");
    printBacktrace();
    return this.generatePrivateKey(barr);
  }
}

Java.perform(understandRecoveryPhraseCrypto);
```

Hooks output for a fresh install of the application :

```
$ frida -O frida_option_launch_loki
     _____
    / _  |   Frida 12.10.4 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Spawned `network.loki.messenger`. Resuming main thread!
#
# Skipped some traces
#
Curve.generateKeyPair with this random : 
↪0d40b60550e006f05c0c03f02702b02e080020240810aa 
↪0d40b60550e006f05c0c03f02702b02e080020240810aa
java.lang.Exception
     at org.whispersystems.libsignal.ecc.Curve.generateKeyPair(Native 
↪Method)
     at org.thoughtcrime.securesms.loki.activities.RegisterActivity.
↪updateKeyPair(RegisterActivity.kt:2)
     at org.thoughtcrime.securesms.loki.activities.RegisterActivity.
↪onCreate(RegisterActivity.kt:14)
     at android.app.Activity.performCreate(Activity.java:7825)
     at android.app.Activity.performCreate(Activity.java:7814)
     at android.app.Instrumentation.callActivityOnCreate(Instrumentation.
↪java:1306)
     at android.app.ActivityThread.performLaunchActivity(ActivityThread.
↪java:3245)
     at android.app.ActivityThread.handleLaunchActivity(ActivityThread.
↪java:3409)
     at android.app.servertransaction.LaunchActivityItem.
↪execute(LaunchActivityItem.java:83)
     at android.app.servertransaction.TransactionExecutor.
↪executeCallbacks(TransactionExecutor.java:135)
     at android.app.servertransaction.TransactionExecutor.
↪execute(TransactionExecutor.java:95)
     at android.app.ActivityThread$H.handleMessage(ActivityThread.
↪java:2016)
     at android.os.Handler.dispatchMessage(Handler.java:107)
     at android.os.Looper.loop(Looper.java:214)
     at android.app.ActivityThread.main(ActivityThread.java:7356)
     at java.lang.reflect.Method.invoke(Native Method)
     at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.
↪run(RuntimeInit.java:492)
     at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:930)

generatePrivateKey( 0d40b60550e006f05c0c03f02702b02e080020240810aa 
↪0d40b60550e006f05c0c03f02702b02e080020240810aa);
java.lang.Exception
     at org.whispersystems.curve25519.JavaCurve25519Provider.
↪generatePrivateKey(Native Method)
     at org.whispersystems.curve25519.OpportunisticCurve25519Provider.
↪generatePrivateKey(OpportunisticCurve25519Provider.java:2)
     at org.whispersystems.curve25519.Curve25519.
↪generateKeyPair(Curve25519.java:1)
```

(continues on next page)

(continued from previous page)

```
    at org.whispersystems.libsignal.ecc.Curve.generateKeyPair(Curve.
↪java:1)
    at org.whispersystems.libsignal.ecc.Curve.generateKeyPair(Native␣
↪Method)
    at org.thoughtcrime.securesms.loki.activities.RegisterActivity.
↪updateKeyPair(RegisterActivity.kt:2)
    at org.thoughtcrime.securesms.loki.activities.RegisterActivity.
↪onCreate(RegisterActivity.kt:14)
    at android.app.Activity.performCreate(Activity.java:7825)
    at android.app.Activity.performCreate(Activity.java:7814)
    at android.app.Instrumentation.callActivityOnCreate(Instrumentation.
↪java:1306)
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.
↪java:3245)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.
↪java:3409)
    at android.app.servertransaction.LaunchActivityItem.
↪execute(LaunchActivityItem.java:83)
    at android.app.servertransaction.TransactionExecutor.
↪executeCallbacks(TransactionExecutor.java:135)
    at android.app.servertransaction.TransactionExecutor.
↪execute(TransactionExecutor.java:95)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.
↪java:2016)
    at android.os.Handler.dispatchMessage(Handler.java:107)
    at android.os.Looper.loop(Looper.java:214)
    at android.app.ActivityThread.main(ActivityThread.java:7356)
    at java.lang.reflect.Method.invoke(Native Method)
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.
↪run(RuntimeInit.java:492)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:930)
```

8.5 Frida code to emulate network attack for Snode bootstrap mitm

```
// Install loki-session in a android rooted phone with frida running
// frida -U -l network_attack_snode_fetching.js -f network.loki.messenger -
↪-no-pause

var external_address_reacheable_by_application = "192.168.0.16"


function isInStrArray(arr, str){
  for(var i = 0; i<arr.length; i++){
     if(arr[i].includes(str)){
        return true;
     }
  }
  return false;
}

var printBacktrace = function () {
    Java.perform(function() {
        var android_util_Log = Java.use('android.util.Log'), java_lang_
↪Exception = Java.use('java.lang.Exception');
```

(continues on next page)

(continued from previous page)

```
        // getting stacktrace by throwing an exception
            console.log(android_util_Log.getStackTraceString(java_lang_
↪Exception.$new()));
    });
};

function attack_dns_redirect(){

            var HTTP_class = Java.use("org.whispersystems.signalservice.
↪loki.api.utilities.HTTP");
            HTTP_class.execute.implementation = function(verb, url, map){
                    var tag = "HTTP.execute";
                    if( isInStrArray( ["https://storage.seed1.loki.
↪network/json_rpc", "https://storage.seed3.loki.network/json_rpc",
↪"https://public.loki.foundation/json_rpc"], url ) ) {
                            console.log(tag + " emulate dns poisonin␣
↪redirection !!!! "+ url);
        url = "https://" + external_address_reacheable_by_application + "/
↪json_rpc"
                    }
                    return this.execute(verb, url, map);
            }
}

Java.perform(attack_dns_redirect);
```

8.6 PushReceived ProcessEnvelope processing

```
function signalEnveloppeToString(se){
  var ret = "";
  if (null == ret ){
     return ret;
  }
  var SSEclass = Java.use("org.whispersystems.signalservice.api.messages.
↪SignalServiceEnvelope");
  se = Java.cast(se, SSEclass );
  ret += 'timestamp = '+se.getServerTimestamp()+" ";
  ret += 'type = '+se.getType()+" ";
  if ( se.getType()==6 ) {
    ret+= "( CLOSED_GROUP_CIPHERTEXT ) "
  }
  ret += 'source = '+se.getSource()+" ";
  ret += "sourceDevice = "+se.getSourceDevice()+" ";
  ret += "uuid = "+se.getUuid()+" ";
  ret += "isReceipt = "+se.isReceipt()+" ";
  ret += "sourceAddress = "+se.getSourceAddress().getNumber()+" "+se.
↪getSourceAddress().getRelay()+" ";
  ret += "sourceDevice = "+se.getSourceDevice()+" ";
  return ret;
}

var PRJ = Java.use("org.thoughtcrime.securesms.jobs.PushReceivedJob");
```

(continues on next page)

(continued from previous page)

```
PRJ.processEnvelope.implementation = function( signalEnvelope,␣
↪isPushNotification) {
 console.log("org.thoughtcrime.securesms.jobs.PushReceivedJob.
↪processEnvelope( "+signalEnveloppeToString(signalEnvelope)+",");
 console.log(isPushNotification ? "isPushNotification":"NotAPushNotif" +" )
↪");
 return this.processEnvelope(signalEnvelope, isPushNotification);
}
```

Hooks output reveals only *CLOSED_GROUP_CIPHERTEXT* envelopes, with encrypted metadata.

```
[Pixel 3::network.loki.messenger]-> org.thoughtcrime.securesms.jobs.
↪PushReceivedJob.processEnvelope( timestamp = 0 type = 6 ( CLOSED_GROUP_
↪CIPHERTEXT ) source =  sourceDevice = 0 uuid =  isReceipt = false␣
↪sourceAddress =  Optional.absent() sourceDevice = 0 ,
NotAPushNotif )
org.thoughtcrime.securesms.jobs.PushReceivedJob.processEnvelope( timestamp␣
↪= 0 type = 6 ( CLOSED_GROUP_CIPHERTEXT ) source =  sourceDevice = 0 uuid␣
↪=  isReceipt = false sourceAddress =  Optional.absent() sourceDevice = 0␣
↪,
NotAPushNotif )
org.thoughtcrime.securesms.jobs.PushReceivedJob.processEnvelope( timestamp␣
↪= 0 type = 6 ( CLOSED_GROUP_CIPHERTEXT ) source =  sourceDevice = 0 uuid␣
↪=  isReceipt = false sourceAddress =  Optional.absent() sourceDevice = 0␣
↪,
NotAPushNotif )
```

8.7 Rogue Service Node Provider

```
package main

import (
    "flag"
    "fmt"
    "encoding/json"
    "errors"
    "log"
    "net"
    "net/http"
)

func externalIP() (string, error) {
    ifaces, err := net.Interfaces()
    if err != nil {
        return "", err
    }
    for _, iface := range ifaces {
        if iface.Flags&net.FlagUp == 0 {
            continue // interface down
        }
        if iface.Flags&net.FlagLoopback != 0 {
            continue // loopback interface
        }
```

(continues on next page)

(continued from previous page)

```
        addrs, err := iface.Addrs()
        if err != nil {
            return "", err
        }
        for _, addr := range addrs {
            var ip net.IP
            switch v := addr.(type) {
            case *net.IPNet:
                ip = v.IP
            case *net.IPAddr:
                ip = v.IP
            }
            if ip == nil || ip.IsLoopback() {
                continue
            }
            ip = ip.To4()
            if ip == nil {
                continue // not an ipv4 address
            }
            return ip.String(), nil
        }
    }
    return "", errors.New("are you connected to the network?")
}

//pubkey_ed25519':
↪'a0c6a10a5789287f4a79f43dceba06770fd843b51f73862377dbf89f9a0b88da',
↪'pubkey_x25519':
↪'059624b15c583affcb73c2325678b0b10be45df6871e6476fde6df3a23143359',
↪'public_ip': '95.216.212.3', 'storage_port': 22021
type Snode struct {
    Pubkey_ed25519 string `json:"pubkey_ed25519"`
    Pubkey_x25519  string `json:"pubkey_x25519"`
    Public_ip      string `json:"public_ip"`
    Storage_port   int    `json:"storage_port"`
}

type ResultObj struct {
    Service_node_states []Snode `json:"service_node_states"`
    Status              string  `json:"status"`
}

type JsonResponse struct {
    Id      int       `json:"id"`
    Jsonrpc string    `json:"jsonrpc"`
    Result  ResultObj `json:"result"`
}

func loggingMiddleware(handler http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        log.Printf("Got a %s request for: %v%v from %v\n", r.Method, r.
↪Host, r.RequestURI, r.RemoteAddr)
        handler.ServeHTTP(w, r)
        log.Printf("Handler finished processing request")
    })
}
```

(continues on next page)

(continued from previous page)

```
func settingFakeNginxHeadersForJsonRpc(w http.ResponseWriter) {
    w.Header().Set("Server", "nginx/1.14.0 (Ubuntu)")
    w.Header().Set("Content-Type", "application/json")
    w.Header().Set("Accept-Ranges", "bytes")
}

func GetStatsServer(w http.ResponseWriter, req *http.Request) {
    settingFakeNginxHeadersForJsonRpc(w)

    w.Write([]byte(`
{
  "connections_in": 4711,
  "height": 602076,
  "http_connections_out": 0,
  "https_connections_out": 10,
  "peers": {
    "4h6jnmxpphohy1xqqrsepd69yzzxm5bi8b5h4zsex4w6ina84s9o": {
      "blockchain_tests": [],
      "pushes_failed": 0,
      "requests_failed": 0,
      "storage_tests": []
    },
    "7snxxcsipdzsxsrerdrk7rk3bc38qdsjpgia5drykyrwifo6yoto": {
      "blockchain_tests": [
        {
          "result": "OK",
          "timestamp": 1597758188
        },
        {
          "result": "OK",
          "timestamp": 1597762503
        }
      ],
      "pushes_failed": 0,
      "requests_failed": 0,
      "storage_tests": [
        {
          "result": "OK",
          "timestamp": 1597758172
        },
        {
          "result": "OK",
          "timestamp": 1597762491
        }
      ]
    },
    "7wbp7zidmpbby4djjb61d76z1ms3ijjyt4kb9trxxppfukisksmo": {
      "blockchain_tests": [],
      "pushes_failed": 0,
      "requests_failed": 0,
      "storage_tests": []
    },
    "85f7tzb9qfygyh1hc4m6cuozfaswhm5bbxsh847f8jxnki6bpj8o": {
      "blockchain_tests": [
        {
```

(continues on next page)

(continued from previous page)


```
          "result": "OK",
          "timestamp": 1597760588
        },
        {
          "result": "OK",
          "timestamp": 1597762942
        }
      ],
      "pushes_failed": 0,
      "requests_failed": 0,
      "storage_tests": [
        {
          "result": "OTHER",
          "timestamp": 1597760611
        },
        {
          "result": "OTHER",
          "timestamp": 1597762966
        }
      ]
    },
    "c8gjpueqoyr75pddadzpb3qwx4g9xb3kgp6zjo339m6ehua8depo": {
      "blockchain_tests": [],
      "pushes_failed": 0,
      "requests_failed": 0,
      "storage_tests": []
    },
    "dhnwnsaeyi6p5f6n8hwtct94koqya44yebf78ruubojey6nwrw4o": {
      "blockchain_tests": [
        {
          "result": "OK",
          "timestamp": 1597757265
        },
        {
          "result": "OK",
          "timestamp": 1597760694
        },
        {
          "result": "OK",
          "timestamp": 1597763165
        }
      ],
      "pushes_failed": 0,
      "requests_failed": 0,
      "storage_tests": [
        {
          "result": "OK",
          "timestamp": 1597757262
        },
        {
          "result": "OK",
          "timestamp": 1597760690
        },
        {
          "result": "OK",
          "timestamp": 1597763161
```


(continues on next page)

(continued from previous page)


```
      }
    ]
  },
  "frjste5rw4bjdbxxk1s9ez4upr3kjoorbqdchcnhh7i5t8oy93to": {
    "blockchain_tests": [],
    "pushes_failed": 0,
    "requests_failed": 0,
    "storage_tests": []
  },
  "khyhnmh7uietnowhyap79s1i86aumig6ydet9jydrajob7hxrbby": {
    "blockchain_tests": [
      {
        "result": "OK",
        "timestamp": 1597756773
      },
      {
        "result": "OK",
        "timestamp": 1597759905
      },
      {
        "result": "OK",
        "timestamp": 1597761231
      }
    ],
    "pushes_failed": 0,
    "requests_failed": 0,
    "storage_tests": [
      {
        "result": "OK",
        "timestamp": 1597756768
      },
      {
        "result": "OK",
        "timestamp": 1597759899
      },
      {
        "result": "OK",
        "timestamp": 1597761226
      }
    ]
  },
  "ng5nufu5f1g836zai4mz3c6tizybw77t4e11ukrgzoxcwg5fzh6y": {
    "blockchain_tests": [],
    "pushes_failed": 0,
    "requests_failed": 0,
    "storage_tests": []
  },
  "oyz3sfae64g44agugya4dgf1utbza71rps8rjzysgkijh6c9ctpo": {
    "blockchain_tests": [
      {
        "result": "OK",
        "timestamp": 1597759068
      },
      {
        "result": "OK",
        "timestamp": 1597762767
```


(continues on next page)

(continued from previous page)

```
        }
      ],
      "pushes_failed": 0,
      "requests_failed": 0,
      "storage_tests": [
        {
          "result": "OK",
          "timestamp": 1597759062
        },
        {
          "result": "OK",
          "timestamp": 1597762761
        }
      ]
    },
    "tob7jopyiy3xoaerdsbphy6x4znwp4gwd95txz1xtqn5smqu3mhy": {
      "blockchain_tests": [],
      "pushes_failed": 0,
      "requests_failed": 0,
      "storage_tests": []
    }
  },
  "previous_period_retrieve_requests": 2669,
  "previous_period_store_requests": 0,
  "recent_store_requests": 0,
  "reset_time": 3009345,
  "target_height": 551808,
  "total_retrieve_requests": 2499894,
  "total_store_requests": 8405,
  "total_stored": 459,
  "version": "2.0.6"
}

`))
}

func FakeJsonRpcServer(w http.ResponseWriter, req *http.Request) {
    settingFakeNginxHeadersForJsonRpc(w)

    log.Printf("Sending Fake NODES ")

    ip, err := externalIP()
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
    snodes := JsonResponse{0,
        "2.0",
        ResultObj{[]Snode{
            Snode{
"c7a93b461c0717f82bf341e67276a22d488e6a0de424beca5186e13339fd5f8e",
"bf6aa79f3fb36532ef83cef3fcfa91146d3c8e62bcb66e909c233f0e0bbf7462", ip,␣
22021},
            Snode{
"a43a1a03c42144d8f875f909ca4bee96f07078577d5adf9305b307c90986974b",
"f1f2e34ad45156b02091bf7870d80eeb3a052439cadb35a3f1aab7c77d1a4a7f", ip,␣
22021},
```

(continues on next page)

(continued from previous page)

```
            Snode{
↪"6db2e5416f980dae4b89e9e3fddbb374633e7a6e68452e285367704967e45243",
↪"72e74bf91706fe26b0cd5d96ab67241c1008fbf6b2daa22ca99329e0663da771", ip,␣
↪22104},
        },
            "OK"}}
    js, err := json.Marshal(snodes)
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
    w.Write(js)
}
func JsonRpcServer(w http.ResponseWriter, req *http.Request) {
    settingFakeNginxHeadersForJsonRpc(w)

    log.Printf("Sending Real NODES ")

    snodes := JsonResponse{0,
        "2.0",
        ResultObj{[]Snode{
            Snode{
↪"c7a93b461c0717f82bf341e67276a22d488e6a0de424beca5186e13339fd5f8e",
↪"bf6aa79f3fb36532ef83cef3fcfa91146d3c8e62bcb66e909c233f0e0bbf7462", "51.
↪68.172.14", 22021},
            Snode{
↪"a43a1a03c42144d8f875f909ca4bee96f07078577d5adf9305b307c90986974b",
↪"f1f2e34ad45156b02091bf7870d80eeb3a052439cadb35a3f1aab7c77d1a4a7f", "51.
↪68.197.123", 22021},
            Snode{
↪"6db2e5416f980dae4b89e9e3fddbb374633e7a6e68452e285367704967e45243",
↪"72e74bf91706fe26b0cd5d96ab67241c1008fbf6b2daa22ca99329e0663da771", "159.
↪69.113.244", 22104},
            Snode{
↪"c6fb4f6aa566fe14c8fd2df16e2d3f6e6b3021a22bf55679336343aa1dfd540e",
↪"f2334e574a5447187a8b08520f18925db14e951daadffc2168d7ec3a67a4b76e", "198.
↪98.55.115", 22021},
            Snode{
↪"8d67fd1815c13d34bd9e679d4fe5f96362abbc3fc4f84da474e6f0041295364e",
↪"4d5a4af91242ea2e52cba57416bd3f645751c2e7c203385a6736a2581659bd01", "198.
↪98.51.229", 22021},
            Snode{
↪"175b3aafe37a18df05f50f9ba06c21a67152470d27737ed1b6da919a46e74c15",
↪"131911e3453892bdf3b852da3cc9caee1a13cd42077be8a138085b9c22a1263d", "198.
↪98.61.235", 22021},
            Snode{
↪"760f8dcc675f3fdda20b08c477a696510d6b90ea496a6ac127460de150b41b4f",
↪"ae5b29371929950e7e39714dc2300e892a84d5f12a535c4451a75f303842ef08", "198.
↪98.59.220", 22021},
            Snode{
↪"9c836eeee252fd38f4c3b299e93545f5b56372d0ff29b535b23cde4cb5750abb",
↪"5ba1cf65c8ac6c4eb68b11c12dcc5e1602d8a25ada4cd65e03a7cccc081ac222", "198.
↪98.62.177", 22021},
            Snode{
↪"aec0849423f1f4c4f91740ffcaf9f0a3cb1d9f50098d120383b9587a1169880a",
↪"6d0b6ff2a3db739099ed61fe4412eb8f817eaeab2980d08cd2d4d6e78dd45c14", "199.
↪195.254.39", 22021},
```

(continues on next page)

(continued from previous page)

```
            Snode{
↪"4f98a95e986938175ed8185d064151d469b16f796a96f0d2881448c84a9cddf0",
↪"ceff90939c45df9573069cf2ac254c536ba23b8b67c4c343258958a45bf67055", "199.
↪195.254.31", 22021},
            Snode{
↪"3fa73c052279bfe7500b8fc7385cb59ba29f8658cac420bd32a5120e2903583a",
↪"4b1bb64204ae4c4d2008f44e025c7216a53e0f33a22aa2c7dbc4678c2ac4846c", "199.
↪195.251.67", 22021},
            Snode{
↪"d111cf47dbf71f86cb3ed715be888cada56e10dcb3a2615e07068702c43ca4c4",
↪"63367a7bf6264a60a6eb5cf0c2cb3a4cb80f422435d037e9649e52c1aa27bc13", "198.
↪98.56.8", 22021},
            Snode{
↪"0a429da4bf0076db86acf1dd40446f1f2171b1917a1e019f84cc0751279c3c16",
↪"fc8a7e954b04d5bb3e08417b19fdd7c22389807e4f45363396aec419e5c12d69", "199.
↪19.225.94", 22021},
            Snode{
↪"159604cd29c1915956f8da941d409866dce279f22d634da4dabcb5f57b840b63",
↪"70d1ec83b5325ab09c7b7132c2b4d3ca0bc86b3428a0e4866a73b93c36cb1863", "209.
↪141.57.186", 22021},
            Snode{
↪"ca80b7d64c75f63d216c308e082b0172b843a1df789da05f0b0064c1d4840d11",
↪"404fa9af3889236d1fc1b98327599be7cbbf1b6877f7f67b407d0b689b5a1f0e", "104.
↪244.78.250", 22021},
            Snode{
↪"87e2ea1e830a97b482907ce50e7fe242e666466301307019cc3065eb0f578bc4",
↪"3eabd0d4ed1bdd9d6493bbbe83d0498155bf2f86220a544cde7d224060b45101", "205.
↪185.115.143", 22021},
            Snode{
↪"e9b8b1234de815f57e74e12b99d4172b67a5ab616ded337b42306019d2c4740e",
↪"bed18198aa834bca08860ba67556c3f31ff45b89a7e2e90a215b86d11508d87d", "209.
↪141.53.153", 22021},
            Snode{
↪"5290595d75ea11f8c0659a16ada50df4ec60a3d879a5ad6b54b68cb668cb6e80",
↪"bd6f4d36f707e2efce0903ee7961088b0ca4f17dab29761a17e5993b0b57861d", "104.
↪244.77.125", 22021},
            Snode{
↪"8b90840c32fa3117aa4823e663f2215e6c8fdaa38ca754a8f45dbbd37b55b9b7",
↪"244e49968e4f7fd82307444daeb4e853a44042304f9e4212f6d195e6371b1c20", "104.
↪244.74.126", 22021},
            Snode{
↪"597c8983c07aad70c00d7027816af8f357c3e896ea529f2f04426ffde968042d",
↪"053cf73e946a85c373c1bf42a3dab2e578c6c1122cfc28e3518da4519ce92104", "104.
↪244.79.152", 22021},
            Snode{
↪"d2ee24a46605daf3ba7674c431ac4559032956c270609c1feb81bc7e4dea6338",
↪"4a14a902fbf68e95ae245a86e56689643b739df5b9ee239d2eb98e44c1f04641", "209.
↪141.36.242", 22021},
            Snode{
↪"22be874786ff3c47a0ca0fa4b15278055557646cb45c1979236362402cd91937",
↪"148fdf6ab35679a32a22f52eaa6eba1a4a4c063bedebb8cefa14142c60b7d572", "209.
↪141.59.172", 22021},
            Snode{
↪"5cc781055bd4d2a580f334a5d8de0633ddf6339da1392e8205c25d4599888601",
↪"0956a0932d46afbaf02e36e68c2698c322c0e126349a5f402e559cefb97f5002", "199.
↪19.225.218", 22021},
```

(continues on next page)

(continued from previous page)

```
            Snode{
↪"f8876f762bd87b4a87c0063ab7d5fcde9eea8ac8a9df34f22839302d46f09a59",
↪"5c0f76a3e8f0b0dc851364d2bf30d033349311e89600793bf10bd34eabc10139", "104.
↪244.74.251", 22021},
            Snode{
↪"6f9501ade5ca2edf17b5596c709e84001981d8784be9286bcba2a4ae95d63b50",
↪"1e55c3da36c33e8353274ac35ced3e36c1232619086225ca5e57b42884f7ea58", "205.
↪185.115.83", 22021},
            Snode{
↪"916cdaae77d1ed4d78e4a3a3675f07588f2ca693c7389b546b73bd53b9e5ec4e",
↪"390a654c42ef069ba3dd998b263539322f25135df0cc235ccff6858fd21c0a7c", "104.
↪244.76.52", 22021},
            Snode{
↪"b96f769e937bbc868e487279a24fa1dd904d1ef7dee036a14e12d95b5fde2a19",
↪"99d7a4da3db0f2b11ad6ca08dcdc12f1a651f39206c84e36fd7aa250903c4961", "104.
↪244.76.71", 22021},
            Snode{
↪"4e9074b91ab5be5bebd12a31ce810272e2982e16f3812c748586bdb463c78832",
↪"442a7ae67422af24e31c78d5017c9be9e56677fa84323997a07cbade9d522203", "209.
↪141.55.83", 22021},
        },
            "OK"}}
    js, err := json.Marshal(snodes)
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
    w.Write(js)
}

//
func main() {
    var listeningAddrForSnodeProvider string
    var listeningAddrForSnode string
    var provideFakeSnode bool = false

    ip, err := externalIP()
    defaultListeningAddrForSnodeProvider := "127.0.0.1:443"
    defaultListeningAddrForSnode := "127.0.0.1:22021"
    if err != nil {
        defaultListeningAddrForSnodeProvider = fmt.Sprintf("%v:%v", ip,␣
↪443)
        defaultListeningAddrForSnode = fmt.Sprintf("%v:%v", ip, 22021)
    }

    flag.StringVar(&listeningAddrForSnodeProvider, "ip",␣
↪defaultListeningAddrForSnodeProvider, "Listening address for snode␣
↪provider, default is external_addr:443 so do not forget to /sbin/setcap
↪'cap_net_bind_service=+ep' current binary")
    flag.BoolVar(&provideFakeSnode, "f", false, "provide fake snode with␣
↪current external address")
    flag.StringVar(&listeningAddrForSnode, "is",␣
↪defaultListeningAddrForSnode, "Listening address for snode , default is␣
↪external_addr:22021")
    flag.Parse()
```

(continues on next page)

(continued from previous page)

```
    snodeProviderMux := http.NewServeMux()
    snodeMux := http.NewServeMux()

    if provideFakeSnode {
        snodeProviderMux.HandleFunc("/json_rpc", FakeJsonRpcServer)
    } else {
        snodeProviderMux.HandleFunc("/json_rpc", JsonRpcServer)
    }

    snodeMux.HandleFunc("/get_stats/v1", GetStatsServer)

    go http.ListenAndServeTLS(listeningAddrForSnode, "server.crt", "server.
↪key", loggingMiddleware(snodeMux))

    err = http.ListenAndServeTLS(listeningAddrForSnodeProvider, "server.crt
↪", "server.key", loggingMiddleware(snodeProviderMux))

    if err != nil {
        log.Fatal("ListenAndServe: ", err)
    }
}
```

8.8 Electronegativity Report

Table 8.1: Table Title

issue	severity	filename	location	sample
https://github.com/doyensec/electronegativity/wiki/LIMIT_NAVIGATION_JS_CHECK	HIGH	/app/main.js	205:2	window.webContents.on('will-navigate', handleUrl);
https://github.com/doyensec/electronegativity/wiki/LIMIT_NAVIGATION_JS_CHECK	HIGH	/app/main.js	206:2	window.webContents.on('new-window', handleUrl);
https://github.com/doyensec/electronegativity/wiki/CONTEXT_ISOLATION_JS_CHECK	HIGH	/app/main.js	324:15	mainWindow = new BrowserWindow(windowOptions)

continues on next page

Table 8.1 – continued from previous page

issue	severity	filename	location	sample
https://github.com/doyensec/electronegativity/wiki/NODE_INTEGRATION_JS_CHECK	HIGH	/app/main.js	324:15	mainWindow = new BrowserWindow(windowOptions)
https://github.com/doyensec/electronegativity/wiki/CONTEXT_ISOLATION_JS_CHECK	HIGH	/app/main.js	540:19	passwordWindow = new BrowserWindow(windowOptions)
https://github.com/doyensec/electronegativity/wiki/CONTEXT_ISOLATION_JS_CHECK	HIGH	/app/main.js	605:6	contextIsolation: false,
https://github.com/doyensec/electronegativity/wiki/CONTEXT_ISOLATION_JS_CHECK	HIGH	/app/main.js	648:6	contextIsolation: false,
https://github.com/doyensec/electronegativity/wiki/LIMIT_NAVIGATION_JS_CHECK	HIGH	/app/main.js	904:2	contents.on('new-window', newEvent => {
https://github.com/doyensec/electronegativity/wiki/OPEN_EXTERNAL_JS_CHECK	MEDIUM	/app/main.js	200:4	shell.openExternal(target)
https://github.com/doyensec/electronegativity/wiki/AUXCLICK_JS_CHECK	MEDIUM	/app/main.js	324:15	mainWindow = new BrowserWindow(windowOptions)

continues on next page

Table 8.1 – continued from previous page

issue	severity	filename	location	sample
https://github.com/doyensec/electronegativity/wiki/REMOTE_MODULE_JS_CHECK	MEDIUM	/app/main.js	324:15	mainWindow = new BrowserWindow(windowOptions)
https://github.com/doyensec/electronegativity/wiki/SANDBOX_JS_CHECK	MEDIUM	/app/main.js	324:15	mainWindow = new BrowserWindow(windowOptions)
https://github.com/doyensec/electronegativity/wiki/OPEN_EXTERNAL_JS_CHECK	MEDIUM	/app/main.js	493:2	shell.openExternal(
https://github.com/doyensec/electronegativity/wiki/AUXCLICK_JS_CHECK	MEDIUM	/app/main.js	540:19	passwordWindow = new BrowserWindow(windowOptions)
https://github.com/doyensec/electronegativity/wiki/REMOTE_MODULE_JS_CHECK	MEDIUM	/app/main.js	540:19	passwordWindow = new BrowserWindow(windowOptions)
https://github.com/doyensec/electronegativity/wiki/SANDBOX_JS_CHECK	MEDIUM	/app/main.js	540:19	passwordWindow = new BrowserWindow(windowOptions)
https://github.com/doyensec/electronegativity/wiki/PRELOAD_JS_CHECK	MEDIUM	/app/main.js	534:6	preload: path.join(__dirname, 'password_preload.js'),
https://github.com/doyensec/electronegativity/wiki/AUXCLICK_JS_CHECK	MEDIUM	/app/main.js	612:16	aboutWindow = new BrowserWindow(options)

continues on next page

Table 8.1 – continued from previous page

issue	severity	filename	location	sample
https://github.com/doyensec/electronegativity/wiki/REMOTE_MODULE_JS_CHECK	MEDIUM	/app/main.js	612:16	aboutWindow = new BrowserWindow(options)
https://github.com/doyensec/electronegativity/wiki/SANDBOX_JS_CHECK	MEDIUM	/app/main.js	612:16	aboutWindow = new BrowserWindow(options)
https://github.com/doyensec/electronegativity/wiki/PRELOAD_JS_CHECK	MEDIUM	/app/main.js	606:6	preload: path.join(__dirname, 'about_preload.js'),
https://github.com/doyensec/electronegativity/wiki/AUXCLICK_JS_CHECK	MEDIUM	/app/main.js	655:19	debugLogWindow = new BrowserWindow(options)
https://github.com/doyensec/electronegativity/wiki/REMOTE_MODULE_JS_CHECK	MEDIUM	/app/main.js	655:19	debugLogWindow = new BrowserWindow(options)
https://github.com/doyensec/electronegativity/wiki/SANDBOX_JS_CHECK	MEDIUM	/app/main.js	655:19	debugLogWindow = new BrowserWindow(options)
https://github.com/doyensec/electronegativity/wiki/PRELOAD_JS_CHECK	MEDIUM	/app/main.js	649:6	preload: path.join(__dirname, 'debug_log_preload.js'),

continues on next page

Table 8.1 – continued from previous page

issue	severity	filename	location	sample
https://github.com/doyensec/electronegativity/wiki/PERMISSION_REQUEST_HANDLER_JS_CHECK	MEDIUM	/app/app/permissions.js	40:2	session.defaultSession .setPermissionRequestHandler(null)
https://github.com/doyensec/electronegativity/wiki/PERMISSION_REQUEST_HANDLER_JS_CHECK	MEDIUM	/app/app/permissions.js	42:2	session.defaultSession .setPermissionRequestHandler(
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/libsignal-protocol-worker.js	300:20	var evalled = eval(‘. . .
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/libsignal-protocol-worker.js	442:13	func = eval(‘_’ + ident)
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/libsignal-protocol-worker.js	560:11	return eval(funcstr)
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/ libtextsecure.js	525:22	var evalled = eval(‘. . .

continues on next page

Table 8.1 – continued from previous page

issue	severity	filename	location	sample
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/ libtextsecure.js	667:15	func = eval('_' + ident)
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/ libtextsecure.js	785:13	return eval(funcstr)
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/curve/curve25519_compiled.js	319:20	var evalled = eval(
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/curve/curve25519_compiled.js	522:13	func = eval('_' + ident)
https://github.com/doyensec/electronegativity/wiki/DANGEROUS_FUNCTIONS_JS_CHECK	MEDIUM	/app/js/curve/curve25519_compiled.js	658:11	return eval(funcstr)
https://github.com/doyensec/electronegativity/wiki/CSP_GLOBAL_CHECK	LOW	/app/about.html	3:0	**default-src 'none';** child-src 'self'; connect-src 'self' https: wss:; font-src 'self'; form-action 'self'; frame-src 'none'; img-src 'self' blob: data:; media-src 'self' blob:; object-src 'none'; script-src 'self'; style-src 'self' 'unsafe-inline';