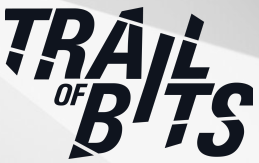

SimpleX Chat Design Review

Cryptographic Design Review (Summary Report)

August 12, 2024

Prepared for:
Evgeny Poberezkin
SimpleX Chat

Prepared by: **Filipe Casal, Markus Schiffermuller, and Joe Doyle**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
497 Carroll St., Space 71, Seventh Floor
Brooklyn, NY 11215
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

© 2024 by Trail of Bits, Inc.

All rights reserved. Trail of Bits hereby asserts its right to be identified as the creator of this report in the United Kingdom.

This report is considered by Trail of Bits to be public information; it is licensed to SimpleX Chat under the terms of the project statement of work and has been made public at SimpleX Chat's request. Material within this report may not be reproduced or distributed in part or in whole without the express written permission of Trail of Bits.

The sole canonical source for Trail of Bits publications is the Trail of Bits Publications page. Reports accessed through any source other than that page may have been modified and should not be considered authentic.

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

Anne Marie Barry, Project Manager
annemarie.barry@trailofbits.com

The following engineering director was associated with this project:

Jim Miller, Engineering Director, Cryptography
james.miller@trailofbits.com

The following consultants were associated with this project:

Filipe Casal, Consultant
filipe.casal@trailofbits.com

Markus Schiffermuller, Consultant
markus.schiffermuller@trailofbits.com

Joe Doyle, Consultant
filipe.casal@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
June 21, 2024	Pre-project kickoff call
June 28, 2024	Delivery of report draft
June 28, 2024	Report readout meeting
August 12, 2024	Delivery of summary report

Project Targets

The engagement involved a review and testing of the targets listed below.

simplex-chat/SimpleXMQ

Repository	https://github.com/simplex-chat/simplexmq/protocol
Version	`d47c099ac94eda3342d02da2da76ef5cab5793ac`
Type	Design documents

simplex-chat/simplex-chat

Repository	https://github.com/simplex-chat/simplex-chat/
Version	`fba0478e509cf56851cef44dc0543a0531bec32b`
Type	Design document

Executive Summary

Engagement Overview

SimpleX Chat engaged Trail of Bits to review the design of several protocols, the adequacy of the threat models considered, and whether the protocol design and cryptographic primitives used are secure with respect to the threat model considered. The protocols under review are the SimpleX Messaging Protocol (SMP), the SMP agent protocol, the push notification system, the file transfer protocol, the remote control protocol, and the chat protocol, along with some variants for each protocol.

A team of three consultants conducted the review from June 24 to June 28, 2024, for a total of one engineer-week of effort. With full access to the protocol documentation, we manually reviewed the documentation and performed formal verification on a simplified version of the SimpleX Messaging Protocol queue agreement.

Observations and Impact

We manually reviewed the in-scope protocols to determine whether they agree with the threat model, and whether all relevant scenarios are described in the threat model. We focused on identifying flaws in the end-to-end encryption guarantees that the protocols propose, and on determining whether there are flaws in session agreement.

Although we covered all in-scope protocols, we dedicated less time to the analysis of the agent-to-agent messaging protocol, the push notification system, and the group chat protocol.

We found that protocol specifications were often described verbosely instead of using precise and clear algebraic notation (TOB-SIMPLX-1). We recommend using a clear algebraic notation to specify each protocol, which will help developers quickly determine which keys are used, which messages are encrypted, under which keys messages are signed, and the intended sequence of steps each user takes (e.g., when the user generates new keys).

We also identified relevant threat scenarios missing from the threat model (TOB-SIMPLX-7 and TOB-SIMPLX-8), and found that attackers could perform actions that the documentation deems impossible (TOB-SIMPLX-3).

Finally, we identified a protocol-level side channel on the number of exchanged messages that could allow an attacker to correlate two users (TOB-SIMPLX-2), and that the SimpleX Remote Control Protocol (XRCP) protocol lacks forward secrecy and break-in recovery within a session (TOB-SIMPLX-5).

Recommendations

We recommend writing precise specifications for each protocol, with diagrams containing each exchanged message, and what keys were used to encrypt or sign them. A clear specification enables easier and faster understanding of the underlying protocols.

When considering each protocol, assume that some key used therein has been leaked and determine if the resulting consequences should be fixed or documented.

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	Protocols are informally described	Cryptography	Informational
2	User-correlating side channel via GET command	Data Exposure	Low
3	A compromised transport protocol allows more efficient correlation attacks	Data Exposure	Medium
4	SHA256 is used as a KDF in XRCP	Cryptography	Informational
5	The XRCP protocol does not have perfect-forward secrecy or break-in recovery within sessions	Cryptography	Informational
6	Device compromise can be hidden in some scenarios	Cryptography	Medium
7	User-correlating attack with introduced latency	Data Exposure	Medium

Detailed Findings

1. Protocols are informally described

Severity: **Informational**	Difficulty: **N/A**
Type: Cryptography	Finding ID: TOB-SIMPLX-1
Target: `protocol/xrcp.md`	

Description

The documentation describes protocols and message exchanges textually and with less precision than desired. It is often hard to understand which keys are used to encrypt messages or which parts of each message are signed.

As an example, figure 1.1 describes the keys used and what the host HELLO message for the XRCP protocol contains.

```
## Key agreement for announcement packet and for session

Initial announcement is shared out-of-band (URI with xrcp scheme), and it is not
encrypted.

This announcement contains only DH keys, as KEM key is too large to include in QR
code, which are used to agree encryption key for host HELLO block. The host HELLO
block will contain DH key in plaintext part and KEM encapsulation (public) key in
encrypted part, that will be used to determine the shared secret (using SHA256 over
concatenated DH shared secret and KEM encapsulated secret) both for controller HELLO
response (that contains KEM ciphertext in plaintext part) and subsequent session
commands and responses.
```

Figure 1.1: `protocol/xrcp.md#L238-L242`

Instead, keys should be given names, and notation should be more algebraic to enhance readability and precision.

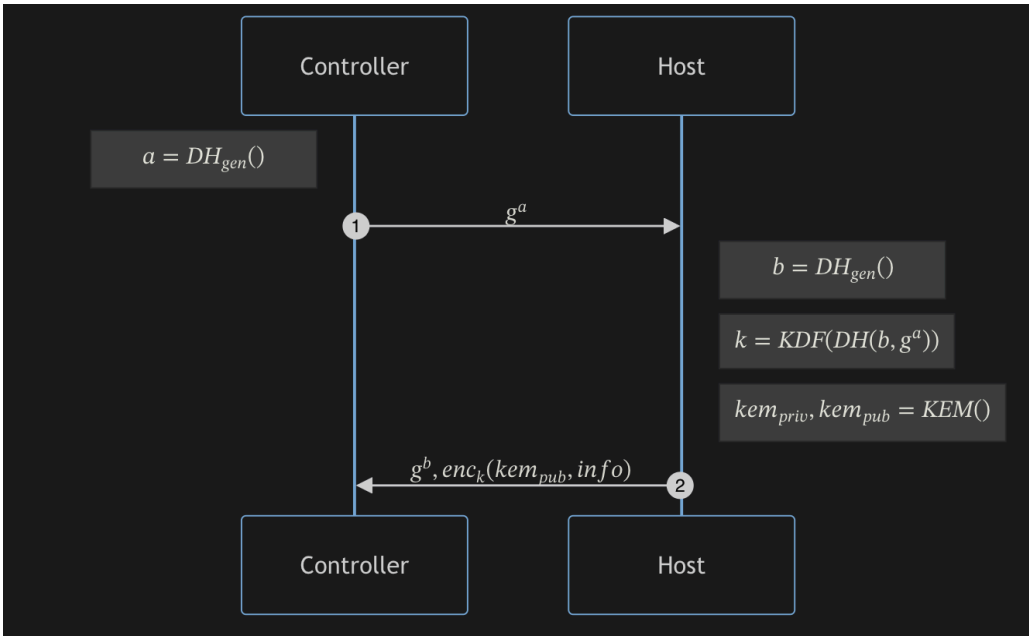

Figure 1.2: Host HELLO message specified in a diagram with algebraic notation

We also identified some protocol descriptions with inconsistencies.

```
Currently members can have one of three roles - `owner`, `admin`, `member` and
`observer`. The user that created the group is self-assigned owner role, the new
members are assigned role by the member who adds them - only `owner` and `admin`
members can add new members; only `owner` members can add members with `owner` role.
`Observer` members only receive messages and aren't allowed to send messages.
```

Figure 1.3: `simplex-chat/docs/protocol/simplex-chat.md#229`

Figure 1.4 states that the XRCP announcement includes the KEM key, but this is false; the KEM key is first sent only in the host HELLO message:

```
- Session X25519 DH key and SNTRUP761 KEM encapsulation key to agree session
encryption (both for multicast announcement and for commands and responses in TLS),
as described in https://datatracker.ietf.org/doc/draft-josefsson-ntruprime-hybrid/.
The new keys are used for each session, and if client key is already available (from
the previous session), the computed shared secret will be used to encrypt the
announcement multicast packet. The out-of-band invitation is unencrypted. DH public
key and KEM encapsulation key are sent unencrypted. NaCL crypto_box is used for
encryption.
```

Figure 1.4: `protocol/xrcp.md#70`

Recommendations

Short term, use algebraic notation for the sequence diagrams; assign names to keys and describe messages referencing those names.

2. User-correlating side channel via GET command	
Severity: **Low**	Difficulty: **High**
Type: Data Exposure	Finding ID: TOB-SIMPLX-2
Target: `protocol/simplex-messaging.md`	

Description

A side channel in the SimpleX Messaging Protocol (SMP) allows an attacker to determine if a user with push notifications enabled has a message in the server to be received.

When the message receiver calls the `GET` command, one of the following scenarios occurs:

- There is one message in the server (so the exchange has a total of four messages):
  - The server sends the message to the client,
  - the client `ACK`s that it received the message, and
  - the server returns `OK`; or,
- If there is no message in the server, the exchange has only two messages:
  - The server simply responds with `OK`, and the interaction stops.

For an attacker trying to determine if two users are talking to each other via SimpleX Chat, this side channel allows correlating packets leaving the sender, with the number of messages exchanged by the receiver and the queue server.

Exploit Scenario

An attacker wants to determine if two users are talking to each other. They use the described side channel and count how many messages leave the sender, and how many messages are exchanged between the queue server and the receiver. This evidence allows the attacker to determine whether these users are talking to each other.

Recommendations

Short term, make the number of messages exchanged when fetching a message on the server independent of whether there is a message on the server or not. Note that if multiple messages are on the server, this solution does not fully resolve the side-channel issue, as the server will continue to deliver messages until none are available on the queue.

Long term, add the ability to introduce large delivery latency for push notifications, hardening the ability to correctly correlate messages between two users.

3. A compromised transport protocol allows more efficient correlation attacks

Severity: **Medium**	Difficulty: **High**
Type: Data Exposure	Finding ID: TOB-SIMPLX-3
Target: `protocol/xftp.md`	

Description

The threat model for the SMP and the XFTP protocol states that an attacker who compromises the transport protocol cannot perform more efficient correlation attacks than a non-compromised protocol. However, because the attacker can see which commands are being sent, correlation attacks will be strictly more efficient.

```
- perform traffic correlation attacks with any increase in efficiency over a
non-compromised transport protocol
```

Figure 3.1: protocol/xftp.md#L566-L566

Without compromising the transport protocol, this correlation would be worse, since other commands could add noise.

Exploit Scenario

An attacker counts how many FPUT commands were issued by the sender and correlates that number with the number of FGET commands issued by a receiver.

Recommendations

Short term, correct the threat model and state that attackers who compromise the transport protocol can correlate sets of users more efficiently than in the case of a non-compromised transport protocol.

Long term, consider encrypting the command identifiers to hide the commands issued by a user from an attacker who compromises the transport protocol.

4. SHA256 is used as a KDF in XRCP

Severity: **Informational**	Difficulty: **N/A**
Type: Cryptography	Finding ID: TOB-SIMPLX-4
Target: `protocol/xrcp.md`	

Description

The XRCP uses SHA-256 to derive session keys from Diffie-Helman and KEM keys. However, SHA-256 is not designed to meet all of the requirements of a proper KDF. Since the secret is derived from other high-entropy secrets, one solution is to use HKDF.

```
## Key agreement for announcement packet and for session

Initial announcement is shared out-of-band (URI with xrcp scheme), and it is not
encrypted.

This announcement contains only DH keys, as KEM key is too large to include in QR
code, which are used to agree encryption key for host HELLO block. The host HELLO
block will contain DH key in plaintext part and KEM encapsulation (public) key in
encrypted part, that will be used to determine the shared secret (using SHA256 over
concatenated DH shared secret and KEM encapsulated secret) both for controller HELLO
response (that contains KEM ciphertext in plaintext part) and subsequent session
commands and responses.

During the next session the announcement is sent via encrypted multicast block. The
shared key for this announcement and for host HELLO block is determined using the
KEM shared secret from the previous session and DH shared secret computed using the
host DH key from the previous session and the new controller DH key from the
announcement.

For the session, the shared secret is computed again using the KEM shared secret
encapsulated by the controller using the new KEM key from the host HELLO block and
DH shared secret computed using the host DH key from HELLO block and the new
controller DH key from the announcement.

In pseudo-code:

```
// session 1
hostHelloSecret(1) = dhSecret(1)
sessionSecret(1) = sha256(dhSecret(1) || kemSecret(1)) // to encrypt session 1 data,
incl. controller hello
```

Figure 4.1: protocol/xrcp.md#L238-L253
```

In particular, define `key-intermediate = HKDF.Extract(key-kem, DH-KDF-secret) = HMAC(key-kem, DH-KDF-secret)`, where `DH-KDF-secret` is the output of HKDF on the DH shared secret, `HKDF.Extract(salt=0,DH-shared-secret)`. Define the final key as `HKDF.Expand(key-intermediate, info=Hash(transcript))`. The hash of the transcript (all the DH shares, KEM ciphertext) is generally recommended for binding security.

Recommendations

Short term, replace the use of SHA-256 with HKDF as described above.

Long term, investigate other locations where SHA-256 is used to derive keys.

5. The XRCP protocol does not have perfect-forward secrecy or break-in recovery within sessions

Severity: **Informational**	Difficulty: **High**
Type: Cryptography	Finding ID: TOB-SIMPLX-5
Target: `protocol/xrcp.md`	

Description
The session key stays constant during an XRCP session. This means that if this key is compromised after the session ends, all messages exchanged during the session will be visible to an attacker who saves the encrypted session transcript.

On the other hand, if the key is leaked during the session, the key is rotated only after a new session is announced, allowing an attacker to decrypt the session as it is occurring.

Exploit Scenario
An attacker saves the transcript of an XRCP session. A few years later, they obtain the session key used to encrypt the messages and decrypt the XRCP session.

Recommendations
Short term, consider adding ratchets to rotate keys. These will provide the XRCP sessions with perfect-forward secrecy and break-in recovery. Alternatively, consider using the SimpleX messaging protocol as a primitive to build XRCP, instead of defining a new protocol with different properties.

Long term, clearly document that this protocol does not have such properties, which are generally expected from an end-to-end encrypted protocol. Document which keys an attacker would need to break protocol properties.

6. Device compromise can be hidden in some scenarios

Severity: **Medium**	Difficulty: **High**
Type: Cryptography	Finding ID: TOB-SIMPLX-7
Target: `protocol/overview-tjr.md`	

Description

The threat model states that an attacker who compromises Alice's decrypted database can "send messages from the user to their contacts," but additionally states that "recipients will detect it as soon as the user sends the next message, because the previous message hash won't match (and potentially won't be able to decrypt them in case they don't keep the previous ratchet keys)."

The second part of this statement implicitly relies on some assumptions that are not true in all cases, such as that device compromises are read-only and that no queue rotation occurs.

To illustrate the first assumption, suppose Mallory is able to not just copy Alice's local state, but also to modify it. Mallory could take Alice's keys for the chat with Bob, then replace the on-device state for the chat with a chat with herself, and surreptitiously perform an agent-in-the-middle attack between Alice and Bob. If Alice does not detect that the chat's local state has changed, neither party would notice the ongoing compromise. Although in most cases this type of attack is unlikely, SimpleX could enable high-risk users to detect this situation. By using an off-device "state fingerprint" that a user can save and check against on a regular basis – e.g., a digest of all safety numbers active at a certain point in time – Alice would be able to detect this attack, and cease using SimpleX until she can re-establish secure communication through out-of-band channels.

To illustrate the second assumption, suppose that Alice and Bob are communicating via the SMP agent protocol. In this protocol, each party acts as the recipient of a different queue, and it is possible for each recipient to "rotate" their queue to a different queue. If Mallory has compromised Alice's local state, she can rotate to a different receive queue, causing all of Bob's messages to exclusively go to Mallory. Although Alice will still be able to send messages that will be delivered to Bob, Bob may not be able to determine whether Alice or the server had been compromised. Additionally, if Bob eventually rotates all of his own receive queues, Alice will not be able to send any messages to Bob, and Mallory will be able to assume Alice's identity until she performs additional out-of-band communication with Bob.

Recommendations

Short term, modify the threat model to explicitly describe what the adversary is and is not allowed to do during device compromise, and which specific protocol configurations this threat model applies to.

Long term, ensure that security documentation such as the threat model is always as precise as possible.

7. User-correlating attack with introduced latency	
Severity: **Medium**	Difficulty: **High**
Type: Data Exposure	Finding ID: TOB-SIMPLX-8
Target: `protocol/simplex-messaging.md`	

Description
An attacker who controls network latency, or can hold the sender's packets for some time, can correlate two users as follows:

- If the attacker holds the sender's packets but the receiver is still exchanging several messages with the queue server, it means the receiver is probably talking to someone else.
- If the receiver starts exchanging messages with the server as soon as the sender's packets are relayed, it is likely that these two users are talking to each other.

Exploit Scenario
An attacker wants to determine if two users are talking to each other. By controlling the network latency, they can determine if that is the case.

Recommendations
Short term, make this scenario explicit in the threat model, so that users are aware of this possibility.

Long term, add the ability to introduce random and large delays in message delivery, diminishing the effects of an attacker controlling the network latency.

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.