SOMA Cryptography Whitepaper

Draft date: Nov. 1st, 2015

Contents

Overview

SOMA Messenger is the latest product of Instanza Inc., a privately held American technology company headquartered in San Francisco, California.

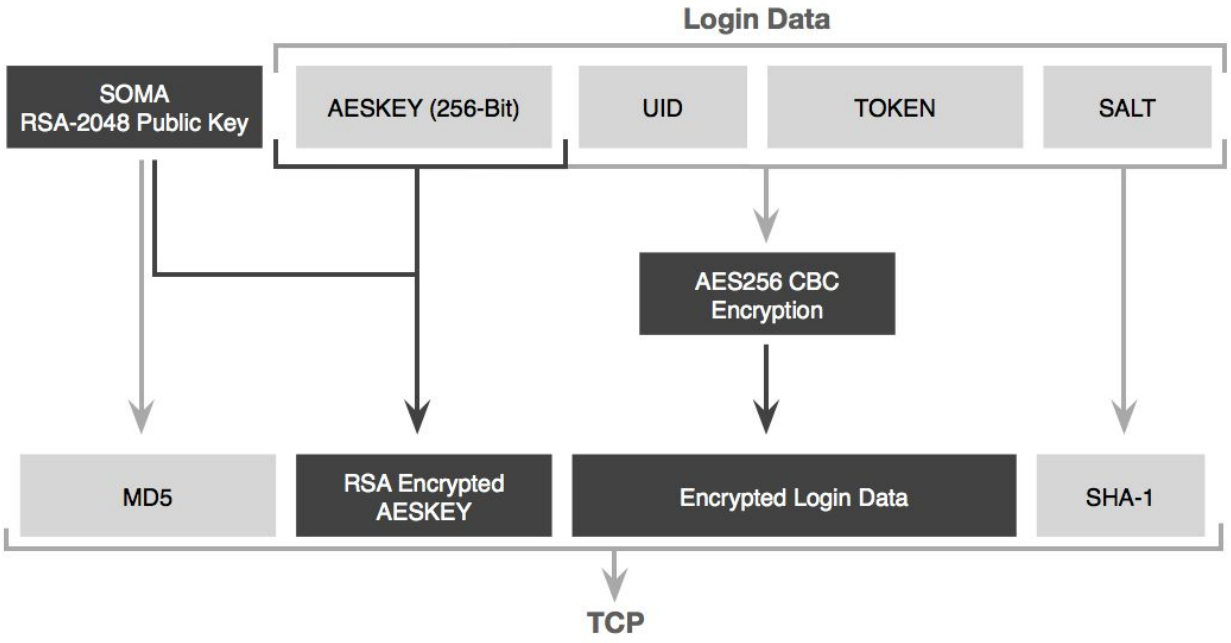
Focusing on privacy protection and security, SOMA's goal is to provide the most secure and enjoyable instant message experience.

All messages (text, photo, video, etc.) between sender and recipient use End-to-End Encryption (E2EE) by default. Under this design, even engineers at SOMA can not decrypt the messages.

In this paper, the design and algorithms behind the cryptography are explained in details.

Secure Transport Layer Protocol

SOMA use customized binary transport protocol between client and server to reduce the transport traffic, and the communication is encrypted by OpenSSL.
Each time client will generate a new AES key to encrypt message, AES key then send to the server encrypted by RSA pubkey from server, finally server verify the data by doing the SHA check.



AES256 Key Generation

When need to establish a secure transport layer, client will need to generate a 256 bits AES key, then send the key encrypted by the RSA-2048 public key delivered by SOMA (which initially saved in device through application installation). When sending encrypted AES 256 key to server, client always append a md5 hash of RSA-2048 public key to indicate server which pubkey is now using.

Login Data Verification

SOMA server uses Login Data to verify user's identity, the Login Data includes UID, User Token, AES256 key and a salt (Salt is generated by server per-connection. Client requests for random salt before sending login data to server). Client then encrypts the Login Data with 256 bits AES key generated above, summarizes the Login Data with SHA-1 algorithm, and sends all the informations to the server.

Secure Transport Layer Establishment

After server receives data above:

1. get the corresponding RSA-2048 private key through md5 of the pubkey.
2. use the private to decrypt and get user's AES256 key.
3. use AES256 key to encrypt and get user's Login Data.
4. verify user's Login Data with SHA-1 summary provided.

After transport layer established, all data in this tunnel will encrypt by the AES 256 key (using CBC mode).

Data Transportation

Now client and server share the same AESKEY.
When data needs to transfer to other side:

1. encrypt it by the AESKEY (using CBC mode)
2. append the SHA-1 summary of data

When data transferred to other side:

1. decrypt it by the AESKEY (using CBC mode)
2. verify the decrypted data by SHA-1

Message Encryption

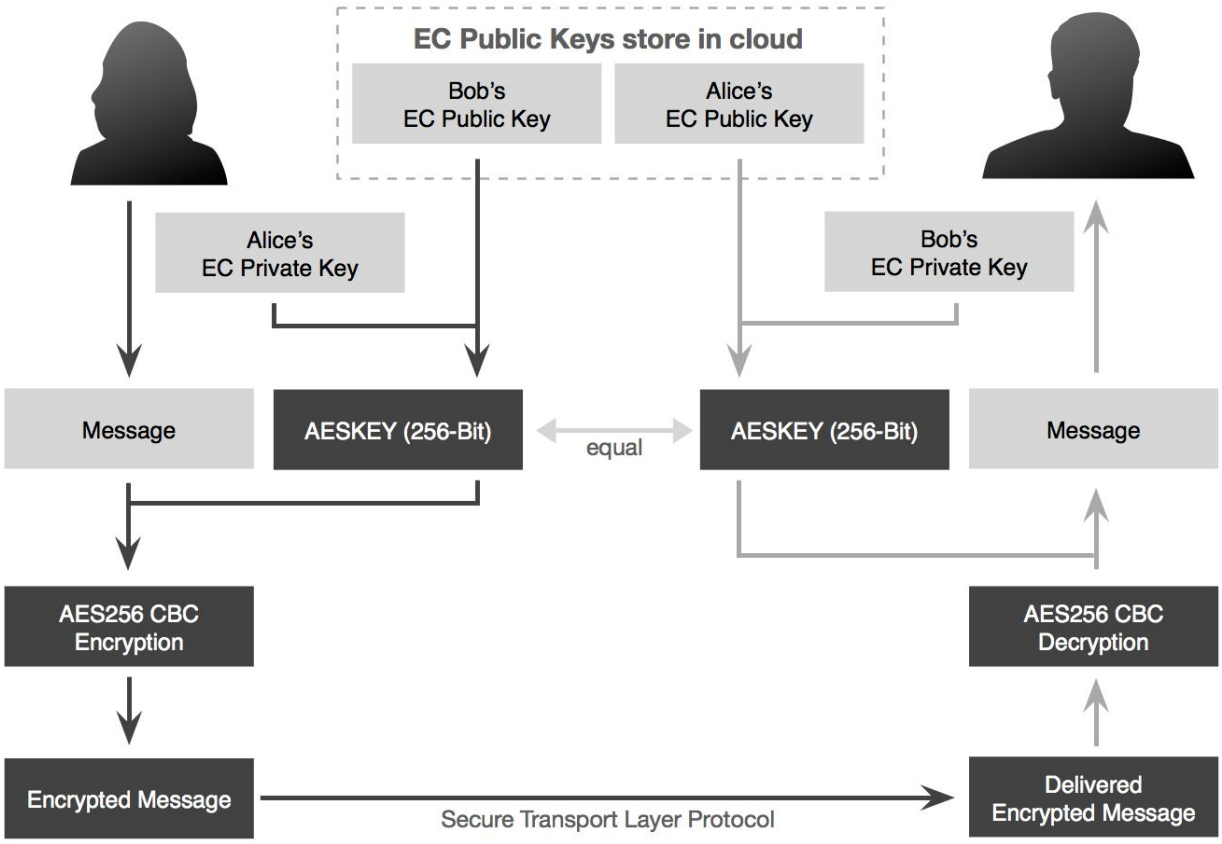
Every kinds of messages (include plain text, location info, media files such as audio, images, videos) are encrypted with combination of elliptic curve key pair and AES 256 key.

Elliptic Curve Key Pair

When user registers, client generates elliptic curve key pairs by:

1. choose a private key by random
2. store private key securely in the client
3. calculate the corresponding public key over the Elliptic Curve (Curve25519)
4. send public key to the server

User can easily query others' elliptic curve public key from server if needed.
Client re-generates the elliptic curve key pair and updates the public key to the server periodically.
The key pair will be physically deleted as soon as user deletes his SOMA account.



NOTE:
The encrypted message will send to Bob immediately. Or it'll be stored in server if Bob not online.
Server will delete the message once it's delivered or not received in 7 days.

End-to-End Encryption (E2EE)

Using end-to-end encryption, users can make sure they are communicating with exactly the one they expect. Each sending message is encrypted by a 256 bits AES key and random generated IV Key. The 256 bits AES key is generated by elliptic curve private key of sender and elliptic curve public key of recipient.
The message will send to recipient immediately, or store inside SOMA server's if recipient not online (see *Server Side Encryption & Persistence*). Server will delete the message once it's delivered or not received in 7 days.
Since EC public & private key is mathematically related, recipient can calculate the 256 bits AES key using his private key and sender's public key.

Media Data

To encrypt a media file, one 256 bits AES key is chosen by random. Then the encrypted media file is uploaded to server through HTTPS. Server will return a media file url when upload finished. After that, sender sends the 256 bits AES key and the url in the same way they send E2EE message. When recipients receive the 256 bits key and the url, they:

1. download the encrypted media file from url through HTTPS
2. decrypt the media file by 256 bits AES key
3. store media file securely in the client
4. send a request to delete the uploaded media file in server side

Server Side Encryption & Persistence

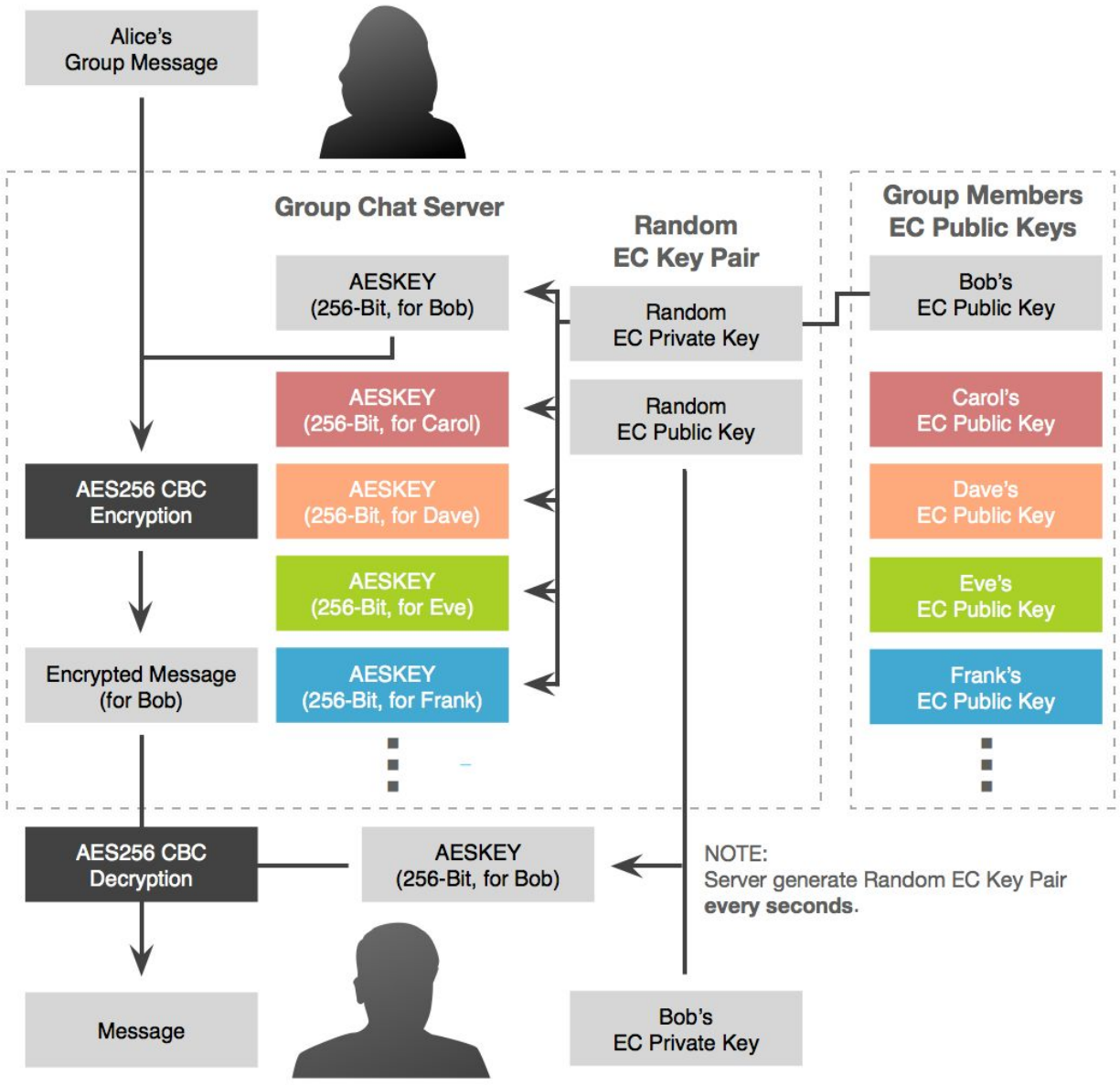
Server generates random EC key pair every seconds, and use the E2EE mechanism to store the data in secure (server only keeps elliptic curve public key and encrypted data, which can only decrypted together with recipient's elliptic curve private key). Hackers can do nothing to the encrypted message even if they invade the databases.
Media Messages are encrypted by 256 bits AES key from sender, whose key is never stored in the server side, to ensure the media file can not be decrypted even hackers get it.

VOIP

Normally, a voip communication between users in same country establishes over P2P (peer-to-peer), data will not transfer through the relay servers. For better quality, when long-distance transnational communication happens, VOIP data will pass by our relay servers. All VOIP data is encrypted by AES, and the key for encryption is generated temporarily each VOIP call.

Group Message

SOMA uses secure transport layer to transfer group message to server instead of E2EE. After message is transmitted to server, it is encrypted by a 256 bits AES key and random generated IV Key. The 256 bits AES key is generated by EC private key in server memory and EC public key of each recipient. Then the encrypted message together with EC public key inside memory again encrypted by the AES 256 key which is exchanged during previous transport layer establishment.



The random EC key pair generated every second stays in server memory, which won't save to persistent storage.

Why we don't use E2EE when sending a group message
Mainly on account of the traffic problem. Let's suppose a 300 bytes group message, and a 500-person group (we will support larger group later). It will cost sender 300*500=146.48 KByte traffic to send the message to every recipients, which would drastically consume user's cellular data, also slow down message distribution speed.

Address Book Synchronization

Address book uploaded to server is one-way hashed by HMAC SHA256.
We need hashed address book to:

1. match and notify user if members inside address book join SOMA
2. share online status with others (only allow friends inside address book to see online status)

Authentication Process

The client generates a GUID during registration. Server encrypts the GUID using AES encryption, then store the SHA hash result inside database.
While client asks for authentication, server will check the hash. This avoid weak password problem, hackers cannot counterfeit user simply after obtaining the password of an account.

Notification Data Security

Android

SOMA server uses GCM Push when client offline, using the same mechanism of message sending to SOMA (with AES + EC/E2EE).

iPhone

Only use APNS when user offline, to notify user there is a new message (without detail).

Those mechanism ensures notification data security, avoids the risks of message leaking through third-party systems.