Security Whitepaper

Siilo develops a secure, asynchronous communication platform for healthcare professionals. We implement proven cryptography based on open source components in consultation with academic cryptographic experts. In addition, **Siilo** was audited by FoxIT in April 2016. In this whitepaper, our cryptographic design decisions as well as our threat model are demonstrated. We aim to strike the perfect balance between security and ease of use.

Contents   Page

Overview end-to-end encryption

Siilo's end-to-end encryption protocols for data-in-transit between "Alice" and "Bob" is as follows:

- Alice uses Elliptic Curve Diffie-Hellman (ECDH) over the Curve25519 and hashes the result with Hsalsa20 to derive a shared secret from her own private key and Bob's public key. Elliptic curve properties allow for key pair swapping: the combination of Alice's private key and Bob's public key achieves the same as Alice's public key and Bob's private key
- Alice generates a random nonce
- Alice uses the Xsalsa20 stream cipher with the shared secret as the key and the random nonce to encrypt the plaintext (packet) into ciphertext (encrypted packet).
- Alice uses Poly1305 to compute a Message Authentication Code (MAC) and attaches it to the ciphertext. A part of the key stream from XSalsa20 is used to form the MAC key.
- Alice sends the MAC, nonce and ciphertext to Bob.

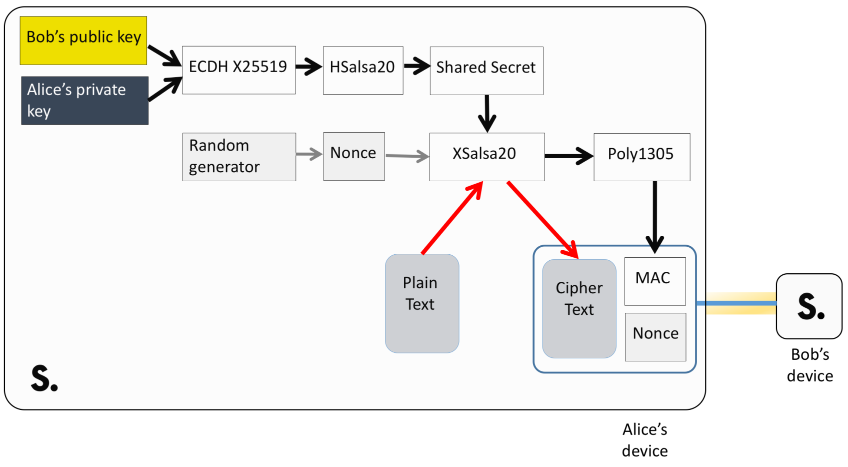

Figure 1 General scheme of Siilo's implementation of NaCl cryptography

By reversing the above steps and using his own private key and Alice's public key, Bob can first verify its authenticity and only then decrypt the message.

Cryptography

Types of keys

1. MasterKey
   - based on a 5 digit code chosen by the user, this input is stretched using hashing into a 32 byte symmetric key.
     - Android - > pbkdf2
     - iOS -> blake-256
   - This key is used by the pin code UI to block user access to the application.
   - This key is also used to symmetrically encrypt/decrypt the other keys mentioned in this section. This forms the "MasterKey" for the system.
   - This key is updated whenever the user defines / changes a new pin code.
2. SignKey
   - 32 byte curve25519 public/private keypair
   - Used for client authentication of api calls
   - The public key is stored by the server and available to any authenticated client.
   - This key is generated as part of the registration process is completed and remains the same for the lifetime of the installation.
   - This key can be revoked by the server either as a result of a new installation or manual action taken by an authorized Siilo operator. The invalidation mechanism is a contact update message sent on the same communications channel as the rest of the protocol.
3. BoxKey
   - 32 byte curve25519 public/private keypair
   - Message encryption uses the private BoxKey of Alice and Bob's public BoxKey and the NaCl crypto_box implementation, the nonce is 24 bytes, randomly generated using the NaCl provided random function, and concatenated to the encrypted message as part of the transmitted payload.
   - The public key is stored by the server and available to any authenticated client.
   - This key is generated during installation and cannot be changed without reinstalling the application.
4. Database SymmetricKey
   - 32 byte symmetric key

- DatabaseKey is used as input into the platform specific SQLCipher implementation which leverages AES-256 to protect the underlying flat files which make up the SQLite database.
- The Database SymmetricKey is generated as part of the registration process is completed and remains the same for the lifetime of the installation.
- This key is used by SQLCipher to encrypt the client database which contains contacts, messages, and associated metadata.
- This key is generated during installation and cannot be changed without reinstalling the application.

5. Local Attachment SymmetricKey
   - 32 byte symmetric key
   - Attachments which are stored locally on the device are encrypted using the NaCl crypto_secretbox, the nonce being randomly generated and prepended to the encrypted byte array.
   - This key is generated during installation and cannot be changed without reinstalling the application.
6. Attachment SymmetricKey
   - 32 byte symmetric key
   - These keys are single use and generated (by the client) when uploading an attachment such as: image/video/audio to the server.
   - The nonce used is the 1-based index of the attachment part, attachments are "chunked" into 2 Mb fragments. Because the nonce can be derived by the recipient, it is not transmitted. So in case of a two chunk attachment the first nonce is "1" and the second nonce is "2".
   - For symmetric cryptography the NaCl crypto_secretbox implementation is used.
   - The key is shared with each intended recipient using their BoxKey.
7. TLS Certificate
   - Issued by AlphaSSL CA
   - Root CA GlobalSign nv-sa
   - The certificate uses: RSA 2048 bit key, SHA256

Key Invalidation

The BoxKey and SigningKey can be revoked by the server either as a result of a new installation or manual action taken by an authorized Siilo operator. The invalidation mechanism is a contact update message sent from the server on the same communications channel as the rest of the protocol. This message is sent either in response to a dedicated invalidation api call only available to operations personnel or as a result of a successful installation with the same phone number as an existing key.

Open Source Cryptography Implementations

- SQLCipher Project
    i. https://www.zetetic.net/sqlcipher/
- NaCl Project
    i. http://nacl.cr.yp.to/index.html
    ii. http://cr.yp.to/highspeed/coolnacl-20120725.pdf (whitepaper)
- Android
    i. https://github.com/joshjdevl/kalium-jni
    ii. https://www.zetetic.net/sqlcipher/sqlcipher-for-android/
- iOS
    i. https://github.com/jedisct1/swift-sodium
    ii. https://github.com/stephencelis/SQLiteCipher.swift
- Backend
    i. Tweet NaCl is used in the proof of concept client.

Key Storage

iOS

- All keys are managed by a class called **BrineKeysProvider**
- All keys are stored in the iOS Keychain
- In addition to the encryption provided by the iOS keychain the key data is also encrypted using crypto_secretbox and the MasterKey to ensure that the keys are protected even if the device does not have a pin code set at the OS level.

Android

- All keys are managed by a class called **CryptoControl**

- Keys are encrypted using AES-256-CBC and the MasterKey, the IV is generated using the JCE provided "AES/CBC/PKCS5Padding" implementation.
- All keys are stored in the application shared preferences, a file which is accessible only by the application itself and Android processes with root privileges.

Protocol

The wire format and domain objects are generated using Google Protocol Buffers see Envelope.proto

The highest level object is an envelope which, protected by TLS 1.2, provides metadata which allows the backend component to route the message in order to “store and forward” until they can be "Ack"ed by the recipient and then deleted from the backend.

Messages are uniquely identified by a combination of message-id and timestamp. Note: an authorised client can rewrite history at present.

Receiving a message

HTTP POST: /api/{api version}/messages/receive HTTP HEADERS: x-siilo-uid, x-siilo-sig HTTP BODY: Protos.Ack HTTP Response: Protos.EnvelopeList

The “Ack” is used for the client to signal to the server that it has received that particular message as well as all messages prior to it. The server will delete those messages upon receiving the Ack. The EnvelopeList is a list of Envelopes in the same format as the send API.

Sending a message

HTTP POST: /api/{api version}/messages/send HTTP HEADERS: x-siilo-uid, x-siilo-sig HTTP BODY: Protos.Envelope HTTP Response: 204 No Content

An envelope contains the “address” of the sender as well as the related group if relevant. Content messages can be sent as well as Typing and Read receipts. The envelope allows the client to send multiple messages in one payload, for example when sending to a group the same envelope is used and then a list of payload objects is sent with different cypher text generated using the public key of the intended recipient.

Signature

All api calls are protected by a digital signature sent in the html headers x-siilo-uid and x-siilo-sig.

The x-siilo-uid must match the user_id in the x-siilo-sig.

Steps to sign a request: 1. construct the SignatureHeader protobuf object and convert to bytes 2. Using Brine.signatureHeader (iOS) or CryptoControl.signMessage (Android) which use the signingKey and the NaCl crypto_box public key cryptography. 3. Base64 encode the result and send in the header

---

System Diagram

Security Mechanisms

- TLS 1.2 with the following ciphers which support forward secrecy.
  - ECDHE-ECDSA-AES128-GCM-SHA256
  - ECDHE-RSA-AES128-GCM-SHA256
  - ECDHE-ECDSA-AES128-SHA256
  - ECDHE-RSA-AES128-SHA256
  - ECDHE-ECDSA-AES256-GCM-SHA384
  - ECDHE-RSA-AES256-GCM-SHA384
  - ECDHE-ECDSA-AES256-SHA384
  - ECDHE-RSA-AES256-SHA384
  - ECDHE-RSA-AES256-SHA
- Client side certificate Pinning of the server's public key.
- End-to-End encryption based on NaCl primitives
- Pin code access control to the application
- Local storage is encrypted using SQLCipher (AES-256) / NaCl (xsalsa20poly1305)
- OS level pin code and device encryption (at user's discretion)

Threat Model

The security goals of this platform are:

- Protect the confidentiality and integrity of messages while in transit between sender and receiver.
- Protect the confidentiality of messages stored on the device when the device is lost or stolen.
- Protect the confidentiality of messages while temporarily stored by the server awaiting delivery.
- Do not store data personal information (phone number, emails) for non-Siilo users.
- Verify the identity of users of the network.

This document will model threats using the following template:

1. **Description** - a brief outline of the attack/attacker
2. **Prerequisites** - conditions required to implement the attack
3. **Prevention** - measures in place to prevent or decrease the likelihood of an attack
4. **Difficulty** - an assessment of the practicality of implementing the attack
    - Easy -> No specialised knowledge, skills, or equipment is required for the attack.
    - Moderate -> Moderate knowledge or information is required. For example a person with a strong computer science background (knowledge of protocols, debuggers, hardware, and etc) or information about the intended target which would not be publicly available.
    - Difficult -> Specialised knowledge, or detailed information about the target is required. For example: years of experience in Informatics Security, or personal knowledge of the target which could only come from prolonged social contact with the intended target.
    - Very Difficult -> Highly specialised knowledge, or detailed information about the target is required. This level of knowledge is approaching that of a nation state or person in the top 3% of the security field.
5. **Severity** - an assessment of the impact of a successful attack, independent from the associated **Difficulty**.
6. **Conclusions** - An assessment of what the impact of a successful attack could accomplish and where applicable a explanation our design decisions regarding this attack vector.

Intercepted network traffic (for example compromised wifi)

Description - An attacker intercepting network traffic of one participant. For example a user connects their device to a wifi endpoint that the attacker has compromised so that he can intercept traffic before forwarding it on to the server.

Prerequisites - The attacker was able to compromise a device in the network path such as a wifi router. Further more to leverage this they must also be able to break the transport layer encryption.

Prevention - We use certificate pinning on the clients to prevent many forms of man-in-the-middle attacks. Each request is cryptographically signed with a timestamp which is valid only for 5 minutes meaning that even replay attacks have a limited window of applicability. End-to-end encryption is used in addition to TLS so even after a failure of that level the end user content is still safeguarded.

Difficulty - Difficult

Severity - Low

Conclusions - Assuming that the attacker has managed to gain access to traffic at the network level, they must still break the transport level encryption (TLS 1.2). If they manage that there are three potential benefits they can achieve.

1. The attacker could replay messages within the server configured signature validity window which would allow them to cause duplicate triggers of simple system messages such as the typing indicator. The attacker cannot spoof content, read notifications, or access end user content.
2. If the transport layer is compromised then message metadata such as the Siilo user id of the participants and the approximate length of the message are exposed to the attacker.
3. The attacker can intercept the encrypted content which is transmitted while they are able to intercept network traffic. Because there is at present no forward secrecy at the message encryption level the user can use this data to attempt to break the user's private key. But even then they would gain access only to the messages which were intercepted.

Non-technical attempt to access the application (for example asking to borrow your phone)

Description - The attacker has brief access to your device through some social engineering attack. For example someone asks to make a call because their battery has died. The key trait here is that the attacker is time constrained and thus cannot perform more invasive attacks such as connecting to the device with a debugger.

Prerequisites - The attacker has physical access to the device. And attempts to access your data at the application level, using a non-technical attack such as manually guessing the pin code.

Prevention - The first line of defense is the OS level pin code. After that the app is protected by a 5 digit numeric cipher which has to be entered if the app has not been opened within 10 minutes. There are timeouts of increasing duration after incorrect pin code attempts.

Difficulty - Difficult

Severity - High

Conclusions - A concession to security has been made by allowing access to the application if the pin code has been entered into the app within the last 10 minutes. This is only relevant in the case that the attacker is actively targeting the individual with the intent to access their specific Siilo data. In other cases, such as the device being left behind, it is likely that the pin code will have timed out before anyone attempts to access the app. If the user wishes to adopt a stronger security posture they can enable their OS level pin code to activate when the device sleeps.

Unrestricted physical access to the device

Description - The device has been left unattended in a situation in which the attacker has time to access the device with specialised tools such as a debugger.

Prerequisites - In this scenario the attacker is not under time pressure to return the device to the target without the intrusion being detected. They can perform any tasks upon the device such as "rooting" or physically opening the device to gain access to the underlying circuitry.

Prevention -

Difficulty - Moderate

Severity - High

Conclusions - The 5 digit pin code is known to provide an insufficiently small addressing space for the hashing to pose a significant counter measure and is considered by us as a form of obfuscation. In this situation the best protection for the device is the OS level pin lock. On iOS devices which support touch id the target can configure his device such that it would be extremely resistant to attempts to crack the OS level passcode. And since all of our key material is protected by the iOS keychain it is implausible that an attacker can gain access on a device which has been well configured this way. On Android the application's data directory is encrypted by the OS level security code; however, there exist bootloader attacks which make it possible on some devices to bypass this mechanism.

Physical access to Siilo servers (Amazon).

Description - This outlines the scenario in which an attacker has physical access to the various components of our server infrastructure. This can be because the attacker is our trusted host provider or because teh attacker has successfully gained entry to the hosting facility.

Prerequisites - Physical access to the Siilo servers.

Prevention - As a trusted provider of cloud services our datacenter provider, Amazon (ec2.amazon.com), has processes in place to protect their customers. Additionally they hold ISO 27001 certification

Difficulty - Difficult

Severity - High

Conclusions - If the breach impacts the messaging server then user metadata such as the hashes of address book telephone numbers, and the graph of their communications history (i.e. who was talking to whom) would potentially be compromised for the length of time that the attacker was able to log traffic to/from the compromised server. However the actually message content would remain safe. If the breach impacted the elastic search server, then only information which was already publicly available to end users via the application would be exposed, admittedly without the benefit of rate limiting. And lastly if the breach involved a database server then the information which is exposed is the same as in the case of the compromised messaging server; however, the scope is the entirety of the system because the attacker would be able to dump the entire database rather than having to intercept pieces of data as they were accessed by authenticated users. Message confidentiality is only assured if the users have mutually verified their key fingerprints, without this safeguard it is possible for the attacker to have put his own key in place to allow for impersonation.

Social Engineering attacks - impersonating someone else

Description - This attack describes the family of attacks related to social engineering and impersonating another person within the system. This includes impersonating a real Siilo user, a person who is not a Siilo user, and impersonating a Siilo employee.

Prerequisites - The ability to install the application. Also if attempting to impersonate a specific user the ability to falsify the origin of an SMS.

Prevention - Users start out in an "unverified" state which is communicated to other users with several UI indicators throughout the application. In order to become verified the user fills in profile information such as their job title, work place, and specialisation(s). This information is verified by Siilo employees and if the information is consistent then the user is marked as verified. Additionally the application supports key *fingerprint* verification in which users can meet in person and verify the hash, or *fingerprint* of their public keys to confirm that the identity supported by the cryptography corresponds to the human being they think it does.

Difficulty - Easy

Severity - Low to Medium

Conclusions - Social engineering attacks are prevalent in any medium in which their is sufficient incentive to attempt them. The problem of correctly identifying an individual and then asserting claims on that identity in the absence of a centralised authority remains unsolved to this day. Siilo is investigating potential solutions to this such as https://www.irmacard.org/irma/, but at this time the best safeguard is for user's to be discrete with their patients' information and to be vigilant about verify the identity of the person they are corresponding with when treating a patient who's data maybe of interest to third-parties.

The case in which an attacker chooses to impersonate a Siilo employee is similar to the common phishing scam used to obtain bank account information. In the case of Siilo the only confidential information a user might disclose to an attacker who had convinced them that they were a Siilo employee is their pin code. This information then could be combined with an attack in real life such as stealing the target's device and using the acquired pin code to unlock the device. The fact that this attack requires the physical involvement of the attacker restricts it in terms of practicality and it will only be used when the target's device is believe to contain data which warrants this level of effort and potential risk on the part of the attacker. The best mitigation of this attack is for the target to realise that a Siilo employee should never need that information and resist giving it out even if the person appears to legitimately represent Siilo.

An attacker who is a trusted Siilo employee

Description - For the sake of completeness we include the scenario in which a Siilo employee with elevated privileges abuses their access the system in an attempt to gain access to end user data.

Prerequisites - The attacker is a Siilo employee who has been granted full operational access. At present this pertains to 3 personnel.

Prevention - The number of employees who have this level of access is minimised and code is peer reviewed by at least 1 other individual so that a code based attack would require collusion between two trusted employees. Further more access credentials to the production environment are issued at the individual level so that actions have an audit trail. Further more every employee with this level of trust has worked for the company for a minimum of 4 years.

Difficulty - Moderate

Severity - Medium

Conclusions - Because an employee at this level of trust has direct access to the database all user metadata including their contact list and phone number. They still do not have access to content such as text messages or images unless they can also break the end-to-end encryption. An attacker at this level would have access to the full SSL certification and thus from this perspective they are no more privileged then an attacker with access to the full network traffic who has broken the transport level encryption. Message confidentiality is only assured if the users have mutually verified their key fingerprints, without this safeguard it is possible for the attacker to have put his own key in place to allow for impersonation.

Compromised root certificate authority

Description - The attacker has gained access to a globally trusted Root Certificate Authority and is capable of issuing their own TLS certificates which third-parties will consider as valid due to the Public Key Infrastructure.

Prerequisites - The attacker has compromised one of the Root Certificate Authorities which a majority of internet trusts so that they can issue their own certificates allowing them to misrepresent themselves as another organisation or service provider.

Prevention - This attack is mitigated to a degree because we use TLS certificate pinning which will reject certificates which do not come from the expected provider from whom we have obtained our certificates.

Difficulty - Very Difficult

Severity - Medium to High

Conclusions - The level of sophistication required to mount this attack decreases the likelihood that our service specifically would be targeted. The most interesting certificates to forge would be Apple/Google (allowing you to alter builds which users download), Amazon (allowing you to impersonate the Amazon APIs we use to manage our servers), and of course the Siilo certificate itself.