Smartphone as a security token - NFChat

Network and Computer Security

Alameda, Group 26

83448
Dorin Gujuman

83475
Hugo Guerreiro

92255
Nuno Silva

1 Problem

Chat services have existed in some form or another for decades. In general, it is a process by which users on a computer network can quickly communicate with one another using text, images or sound rather than using email. Each user has a piece of software which communicates with a common server that connects the chat sessions.

Many people overshare their own private information on websites such as Facebook, Twitter, Snapchat and Instagram. What happens is that because of social media and the ease of connecting with new people your contact list may become so large that new problems emerge like impersonators, which might commit fraud or defamation. Another problem are chat-bots or spam accounts, that send you unwanted messages.

In conclusion, there is a need for a personal chat where you can be confident in the security and identity the participants. In other words offering confidentiality, authenticity and integrity to the user.

2 Requirements

The Electronic Frontier Foundation lists some traits that contribute to the security of instant messengers:

- Having communications encrypted in transit between all the links in the communication path.
- Having communications encrypted with keys the provider does not have access to (end-to-end encryption).
- Making it possible for users to independently verify their correspondent's identity e.g. by comparing key fingerprints.
- Having the software's security designs well-documented.

3 Proposed solution

NFChat aims to solve the described problems by offering a chat app that leverages the proximity to your closest friends and their smartphones. You start a conversation by touching two phones enabling the users to share completely private and secure messages.

3.1 Security components

On it's first boot the app generates the following:

User UUID An unique identifier for that specific phone. (128-bit number)

EC key pair An Elliptic Curve key pair used only for signing messages. (SHA256 with ECDSA)

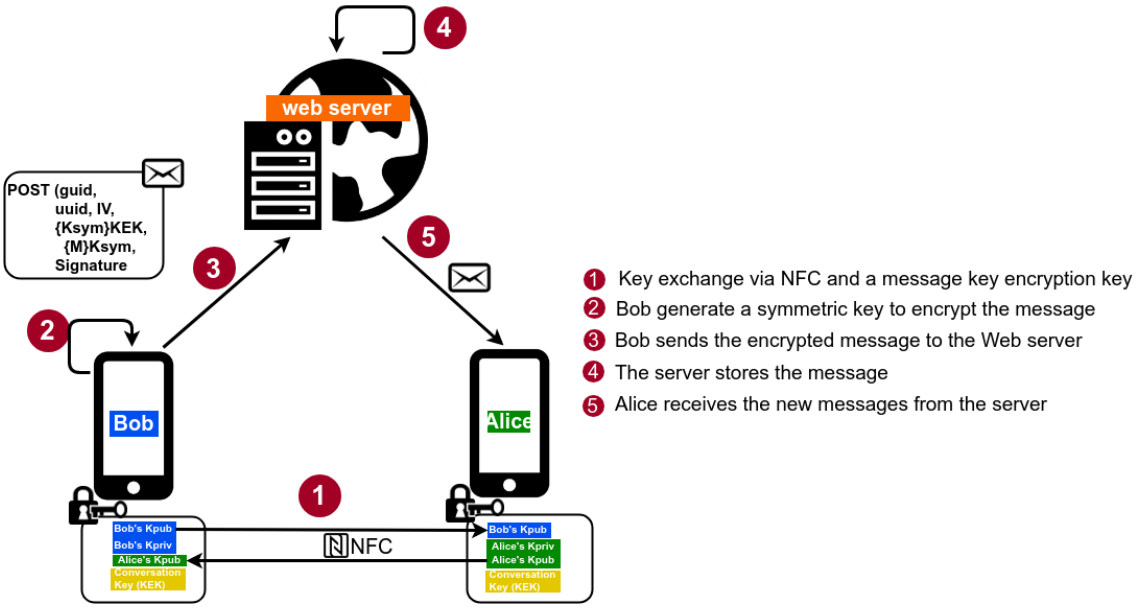
When two users engage on a NFC transmission the following data is exchanged:

User UUID The aforementioned unique identifier of the phone.

Conversation UUID A randomly generated unique identifier of the conversation, used to identify the conversation between the two users. (32 byte alphanumeric string).

Key encryption key (KEK) A randomly generated encryption key whose function it is to encrypt and decrypt the data encryption key (DEK) present in each message. (256-bit AES)

ECDSA Certificate The certificate which will be used to verify signatures of messages. (X509 encoded certificate)



After the NFC transmission we are able to converse with our contact securely since we share a KEK. A message contains:

DEK Key Random data encryption key encrypted with the conversation KEK. (256-bit key encrypted with AES/ECB/NoPadding)

IV The IV of the data cyphertext.

Cyphertext The data encrypted using the DEK key and AES/CBC/PKCS5Padding algorithm.

Signature A signature using the senders EC private key. (SHA256 with ECDSA)

To implement This we created an Android App that in its basic version was able to share secrets through NFC and add them as contacts to a simple chat interface. In the intermediate version we were able to send and receive messages with the format we described above. Finally in the final version we added a support for group chats.

3.2 Group chats

The groups chats follow the same structure as the individual chats.

The biggest difference between group chats and individual chats is that to start the group chat a group key is broadcasted to every member in the group by the person who first created it. The key itself is properly encrypted with the KEK's of each individual member.

Another key difference between the individual chats is that some users might not know each other therefore they cannot validate the messages from that user (they can still read the messages).

However, the user has some level of trust on the messages from an unknown user because he assuredly trusts the user that created the chat in the first place (the user that created the chat trusts every member he added).
We give the choice to the user to trust or not the message.

3.3 Web Server

The web server is one of the most important parts in our system. Without it the entire app wouldn't work as intended.

We don't assure the security of the web server itself because we felt like it was out of the scope of the project. What we assure is the security in the connections between the users and the server by having a valid certificate that allows HTTPS requests.

The server is a very simple application made using the Flask framework and stores the data in a local SQlite database and runs on a virtual machine on Microsoft Azure cloud (https://nfchat.dorin.space). We also provided a very simple API for the requests:

- POST - posts a new message to the server with the parameters previously specified.
- GET - gets a specified number of previous messages stored in a server for a given Guid.

3.4 Security Policies

In this subsection we specify some of the policies we made while developing the project.

3.4.1 General policies

- Each user identifier must be unique.
- An users private key must never be shared.
- An users public key must only be shared by using the NFC technology or other similar one that provides the same level of proximity.
- Every sensible information must be encrypted.
- Every message must be signed by the person who sent it.
- Every message not properly signed should be easily identified.
- To access the app the user must pass through a mandatory lock screen.

3.4.2 Group chat policies

- Each group must be unique.
- The person who creates the group must know every other user involved in it.
- When a group is created no system critic keys should be shared.
- Users inside a group should be able to read every message sent to the group, but will have to choose to trust messages from other users that themselves don't directly know.

4 Results

The mobile application has successfully reached the desired advanced state. When an user opens the app for the first time he is mandated to define an hand drawed pattern authentication. After an authentication process the user gets two options:

- Add a new user through NFC by touching a compatible phone with his own device
- Add a new group

Both were implemented and both create a room where the participants are able to talk via text messages. We implemented a secure communication with the web server by using a library that supports this type of communications and by setting up a certificate on the web server. As specified on the Proposed solution the app should secure the conversation between friends with the sharing of keys and signature verification through certificate. We implemented this by using thechnology provided by Java and Android.

When a user wants to create a new group he can choose from a list of his own friends. If two users happen to not know each other the messages will appear as red to warn each party that those messages are not verifiable. If, both users later add each other as friends, this messages will become normal. In conclusion, we implemented everything that we wished to implement.

5 Evaluation

Considering our requirements we were able to fulfill all of them. We grant confidentiality because communications are encrypted end-to-end and it's impossible to decrypt them without knowing the KEK that was personally shared.

Every message is verified by every user and if something is wrong it will show the message with a red background indicating that something went wrong with the validation. One weakness is that we didn't implement protection from repetition attacks, but that isn't too bad on a chat application (compared with, for example, banking applications) and can easily be solved by adding a freshness element to the message like a nonce. Since we use HTTPS to send and receive messages we have the added protection that it provides, like hiding the arguments a user sends in the POST request to the web server.Otherwise, an attacker could find out the conversation GUID with a man in the middle attack user and monitor when the participants send messages.

The major weakness of our system is that by design we depend too much on our phone, if it gets stolen the chats become compromised and all the previously sent messages could be read if an attacker gets ahold of a KEK. We try our best to mitigate this by storing our keys on the Android Key Store and adding a lock screen to the app. Overall, many security related functionalities of the full system could be optimized, especially on the side of the web server and lock screen. We decided to concentrate on the confidentiality and integrity of the messages, but many other part's security could be better explored.

6 Conclusion

In summary, the goal of NFChat was to build an android application based on security and proximity that also assured privacy in the contents of the shared messages.

We achieved this by leveraging the proximity imposed by the NFC technology that guarantees the identity of each person (avoiding the possibility of impersonators).
On a final note, even though we managed to achieve an overall safe app, no app is ever trully safe, but it doesn't mean we shouldn't aim to try and create one.

7 References

- Type of keys: Symmetric (AES) and Asymmetric (EC)
- Integrated Development Environment : Android Studio and SDK Tools
- Android Beam to share data by NFC
- Java SE Security
- Android Keystore System to store cryptographic keys
- Certificates and Web Servers (HTTPS, Let's Encrypt, Django/Flask)