TÉCNICO LISBOA

Network and Computer Security

Secure messaging using SMS

Report

MEIC-A
Group 6

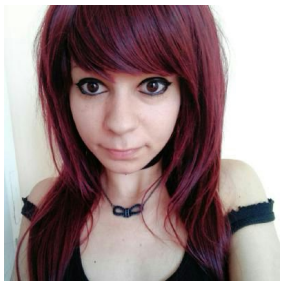
70171

João Miguel Neves
jsoaresmatosneves@tecnico.ulisboa.pt

72904

Luís Ribeiro Gomes
luis.ribeiro.gomes@tecnico.pt

74190



Ana Beatriz Alves
anabeatrizalves@tecnico.ulisboa.pt

Problem

Normal SMS exchanging security relies exclusively on the security of the underlying cellular network protocol. This poses two security concerns, first the network operator is able to look into the plaintext content of messages, and secondly, attacks on cellular networks themselves are known to be possible, allowing an attacker to intercept, tamper with and spoof messages.

Solution Design

Assumptions

We assume that CA certificate that the user installs on the application to verify the validity of other end-user certificates is legitimate. All users who want to exchange messages have to share their certificates.

Architecture model

When a user installs the application, he must first set it up with vital information such as the credentials he's going to use to login, the CA certificate and his, and the private keys previously generated. The login password is also used to cipher the stored private keys and the exchanged messages database.
Now the user is able to unlock the application, where he can add contacts and exchange messages with them.
A user can only message someone whom he previously added as a contact - which means he must have imported the contacts certificates . This will be necessary when the application validates any interaction with this contact.
After adding a contact, the user can send and receive sms messages, but what happens when a message is sent is the following:
The application searches for the contact in the database, then checks the session status with that contact. If no valid session exists yet, a new one is created (the parameters are described in the diagram below) for the duration of 1h. In this case, the message isn't sent until the receiving party confirms the request.
However, if both users have already a session established, it's only necessary to update the sequence numbers, sign the information and encrypt everything with the session key.

Meanwhile, when a user has incoming messages, a similar process occurs:
The session is validated to check its status. If the session with that contact expired or wasn't created yet, the application will check if the message type is a request. Then the application prompts the user to accept or decline this new session.

In this case, a new session is created with the parameters received, and it's marked as established. The original sender has to know if the session was confirmed, so the receiving user sends a message with acknowledgment type.
When the original sender receives and validates this last message, the session is officially established, the pending message is sent and both users can exchange sms messages until the session expires.
To defend against replay attacks the Timestamp in the request message has to be within the timeframe of a session, and the acknowledgement message has to include the sequence number sent before. (The sequence numbers are randomly generated by each party). Regular text messages are validated by checking the signature and the sequence number.
The session request message is ciphered with the contacts 2048 bit RSA public key, and the included session key is a 256 bit AES key. The messages are ciphered using AES in CBC mode with Ciphertext Stealing and a new random IV is sent along with every message. The signature is produced with ECDSA using 224 bit keys and SHA-224.

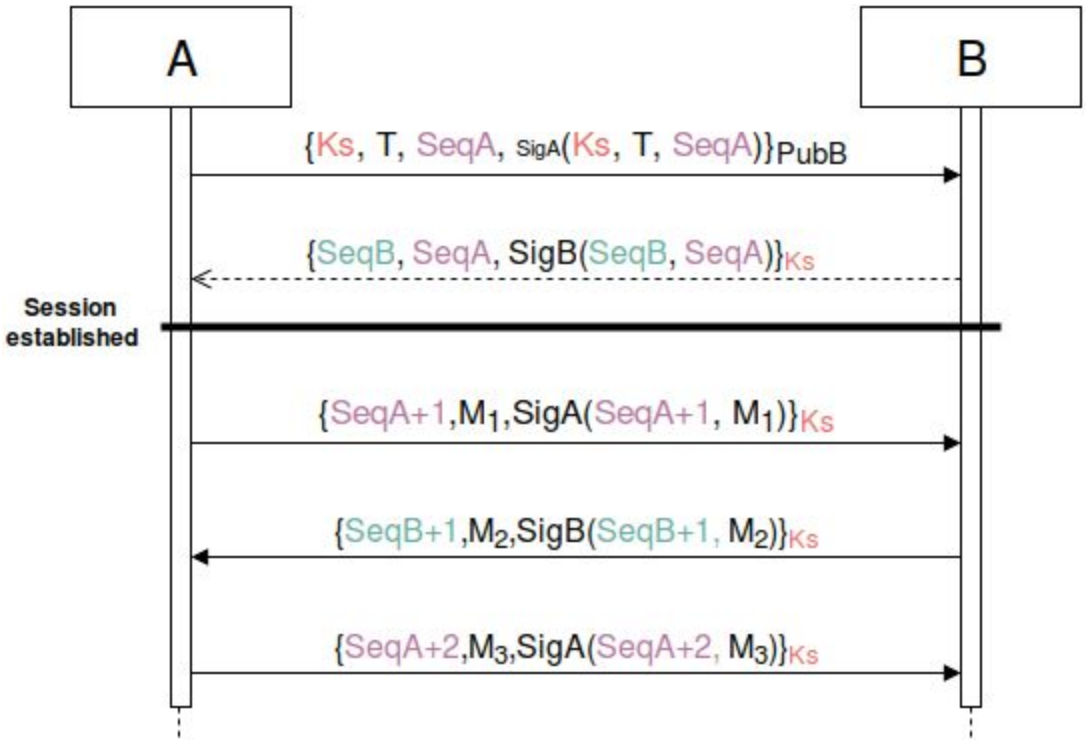


Diagram 1 - Protocol diagram

Ks - Session key	T - Timestamp
Seq - Sequence number	Pub - Public key
M - Message	Sig - Signature

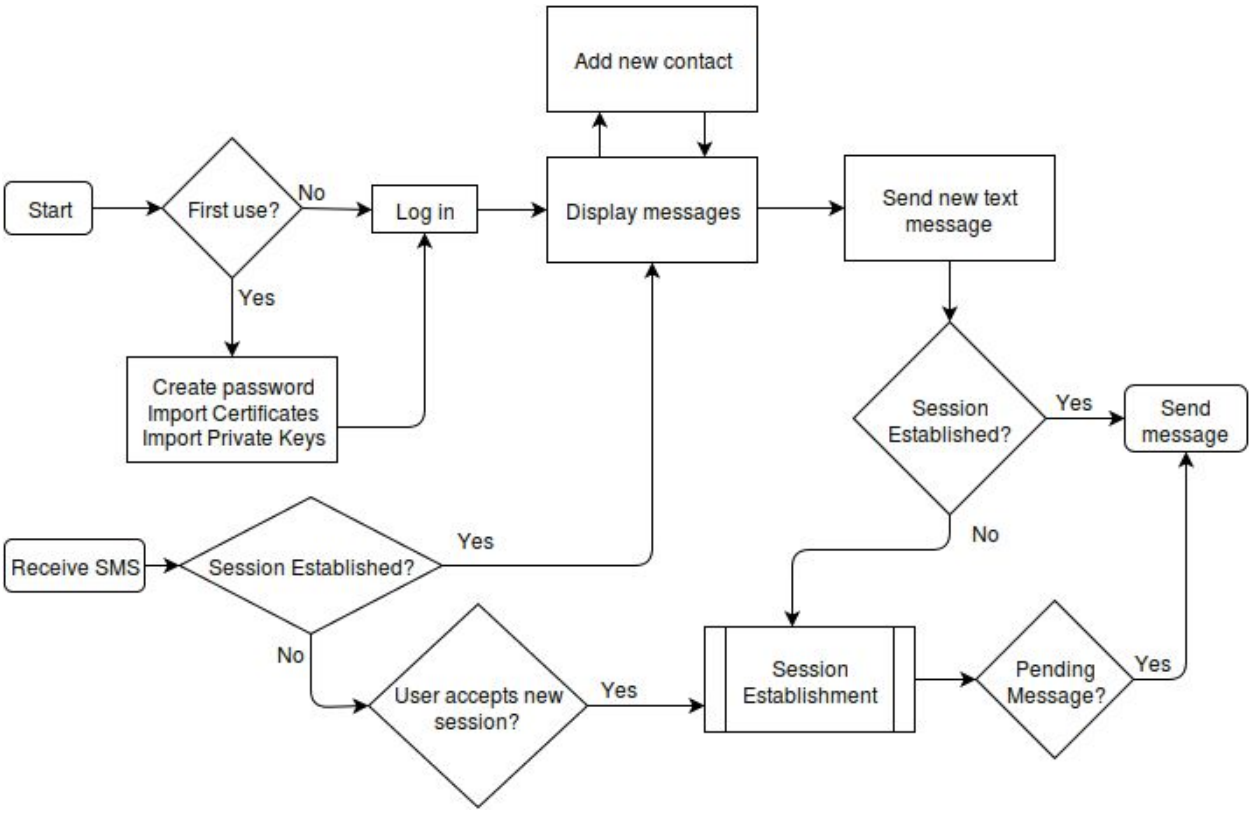


Diagram 2 - Structure diagram

Solution Implementation

Results

The application is capable of importing X.509 certificates (end-users and a single CA) as well as the user's private keys (encrypted in PEM format). It stores all keys in an encrypted key store on the phone's external memory.

The application is capable of managing its own contact list and store all previously exchanged messages (these are encrypted with the user's password).

It can establish a session and exchange text messages, implementing the previously described protocol, with any contact previously added by the user.

Evaluation

The application has certain weaknesses, some caused by compatibility issues between the tools chosen and the platform in which we were developing. Some of the functionality we wanted to use is

only natively available on the latest version of Android (Marshmallow), therefore we had to use an external library to provide cryptographic functions and a secure key store.
We also had to make choices so that security was never compromised, which means all messages are very limited: only about ⅓ of the total sms size is available for the user's text message.
This is, in part, an issue with the SMS itself, however we did our best to minimize the number of bytes sent that would be used by the protocol. The biggest flaw is perhaps the triple SMS message exchange while establishing a session: because the first message has to be encrypted with RSA, it takes up the space equivalent to 2 messages.
The acknowledgment message could be replaced by a normal text message that implicitly established the session whenever the user replied to the original sender, but we figured that the receiving user could choose not to reply and receive all incoming messages from that contact anyway. Therefore, it was a necessary sacrifice.

Conclusion

We have obtained an application that allows SMS exchanging while taking into account security properties, such as confidentiality, integrity, authentication, non-repudiation and freshness.
Our architecture allows allows users to establish a secure channel in which parties can communicate privately.

Tool References

Java SE
Android SDK Tools
Bouncy Castle Crypto API