PhoneX system
(confidential, subject to NDA)

Dušan Klinec

1 Product outline

- PhoneX is a solution for a secure mobile communication.
- Currently we support secure voice calls, text messages and a file transfer among users in the PhoneX system.
- PhoneX system is designed as a closed system in a sense that only PhoneX users can talk securely to each other. Main reason is the security. We set high security level and it can be guaranteed only inside our system.
- Encryption is performed directly on end devices, PhoneX is an end-to-end encryption system. Server, administrators and arbitrary third parties are not able to eavesdrop user's communication.
- Voice call is transmitted over a data channel, no SIM card is required.
- Our voice call has low bandwidth requirements. One way (half-duplex) data channel requires 8–20 kbits per second.

Available platforms

- Android platform fully supported, available on Google Play.
- iOS platform in development. Complete client scheduled by the end of 2014.
- Desktop platforms scheduled to Q1,Q2/2015.

2 System architecture

This section briefly describes overall PhoneX architecture. Some aspects are simplified for easy understanding.

- PhoneX system consists of a PhoneX cluster and PhoneX client application. Figure 1 displays basic PhoneX system architecture.
- One PhoneX license corresponds to one user identity, identified by it's login name. User identity is explained more in section 3.
- Each identity has assigned a home PhoneX cluster. This cluster manages particular user identity and it's contact list. Home cluster has minimal information about identities (e.g., no logs are stored, messages are kept in end devices, private key is unknown to the server).
- Identities can communicate between each other, even if they both reside on a different PhoneX cluster.

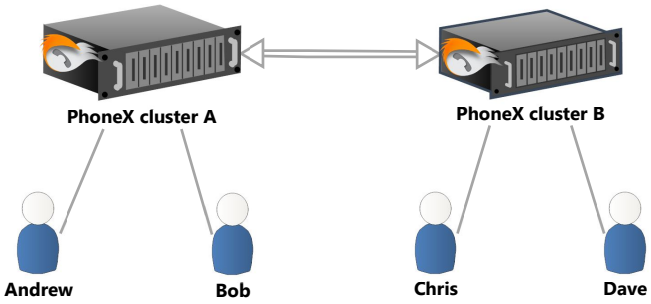

Figure 1: PhoneX system architecture.

3 User identity

This section describes how the user is represented in the PhoneX.

- Each user represents a single identity in the system. E.g., user Alpha, identified by it's resource identifier (like mail address) and service password. Example:
  - `alpha@phone-x.net`
  - `password123456`
- Each user can use PhoneX system on multiple devices. E.g., device A, device B.[1] Example:
  - `alpha@phone-x.net/deviceA`
  - `alpha@phone-x.net/deviceB`
- Each device user Alpha uses, generates own asymmetric key-pair.
  - Currently: RSA 2048 bits.
  - Private key never leaves the device.
  - Private key is stored in a secure way on the device (strong encryption).
  - Next plans: 4096, elliptic curves.
- *X509v3 Certificate*, binding user identity and his public key is generated by the server during the first login. Home PhoneX cluster is a certification authority for the identity.
  - Certificate is stored on the user's device.
  - This client certificate is used for authentication to PhoneX servers, in *TLS* connections.
  - PhoneX uses this certificate for digital signatures verification.

3.1 Password bruteforce protection

- User's password is kept secret, only user knows it. It is unknown to server, administrators and anybody else in a plaintext form.
- PhoneX mitigates weak user passwords problem by using strong password derivation functions.
- From the user password, several cryptographically independent secure passwords are generated using state-of-the art key derivation algorithms:
  - Scrypt[2]

[1] Not implemented yet.
[2] Protects `https://litecoin.org/` crypto currency. Designed to resist hardware crackers.

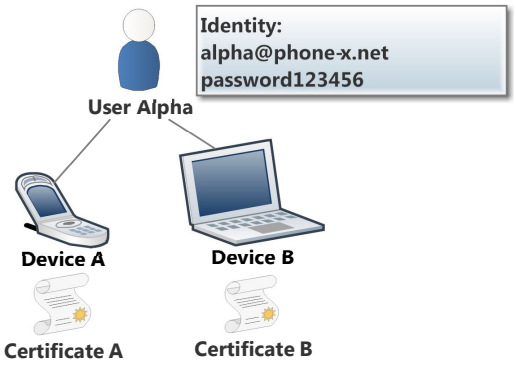

Figure 2: User identity scheme.

  - PBKDF2, 8192 iterations, SHA-256 HMAC mode.

- Dictionary/bruteforce password cracking of our password scheme is very slow,
  - 16 passwords per second on 1.8 GHz, iCore 5.
  - 0.3 passwords per second on iPhone 4.

Protection extension

- We have designed a new password scheme protocol that provides total protection against password bruteforcing.
- It works as a PIN code with ATM card, several attempts are allowed, then timeout increases.
- After too many attempts, CAPTCHA solving is required.
- Server interaction is required, password cannot be cracked offline, server limits login attempts.
- Implementation of this feature scheduled to Q1/2015.

3.2 Secure storage

- Private key is stored in an encrypted container *PKCS12*, protected by a strong encryption password derived from the user password.
- User's private data, (e.g., contacts, messages, call logs) are stored in encrypted *SQLCipher*[3] database.
  - SQLCipher is an open source extension to SQLite that provides transparent 256-bit AES encryption of database files.
- In case of a theft or a device loss, data is protected in an encrypted form. It cannot be read without user password.

3.3 Client communication

- PhoneX clients do not communicate directly (in a P2P manner) by default but through their home PhoneX servers.
  - Only exception for direct communication is during voice call, if system manages to establish a direct connection between users.
- Communication with the PhoneX clusters is protected by *TLS* protocol, using both server and client certificate verification.

[3] `https://www.zetetic.net/sqlcipher/`

3.4 Contact list

- Each identity defines it's own *contact list*. It is a list of identities that are allowed to contact list owner.
- Contact list establishes an asymmetric trust relation. By adding user B to my contact list I allow user B to contact me and I express my intention in receiving B's presence updates.
  - Users can add arbitrary users to their's contact lists.
  - In order to communicate A and B with each other they have to at first add each other to their contact lists.
- Example contact list is shown in figure 3.
- Contact list is stored on the server so it can block incoming requests that are not authorized.
- Contact list contains public certificate for each user so user can verify digital signatures produced by contacts and/or use it for asymmetric encryption.
- Contact list is managed by users via PhoneX application. In a corporate setup, the central administration control panel is planned on Q1/2014. The control panel will have the following features:
  - Centralized user management.
  - Centralized policy management for users.
  - Definition of user groups, levels and privileges.
  - Integration with customer's user database.

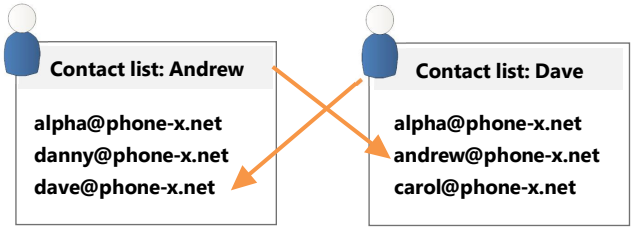

Figure 3: Example contact list for two users.

3.5 Presence

- Each user can define it's own presence, displayed to other PhoneX users. Typical presence statuses are: Online, Away, Invisible.
- Presence is based on *XMPP*.
- Presence updates are sent from user to home PhoneX cluster. Then, PhoneX cluster pushes presence updates to all subscribed users.
- Presence information conveys additional information about the user (e.g., his certificate version, capabilities of the client / supported features.) so subscribers can react on changes and choose the best protocol for communication.
- Presence updates from A to B are sent only if A has B in the contact list and vice versa.

4 Voice call

- *SIP* is used for signaling (i.e., initiating a voice call / multimedia session among PhoneX users).
- *ZRTP* is used to establish a shared symmetric encryption keys prior voice call.
- *SRTP* is used to protect a multimedia session between users by a symmetric cipher, using password derived by ZRTP.
- SIP packets are digitally signed so they cannot be forged by attackers. The authenticity of the requests is guaranteed.

4.1 ZRTP

- An unique encryption key is established for each voice call.
- Encryption keys are destroyed right after finishing voice call.
- ZRTP internally uses *Diffie-Hellman* key establishment protocol.
- Perfect forward secrecy property. I.e., even if long term secret is compromised (i.e., private key), the past voice call session recorded by an eavesdropper cannot be decrypted.
- ZRTP is protected against man-in-the-middle attacks by *SAS* hash. SAS has to be verified verbally by both sides during the first voice call. After this verification it is no more required since shared secrets from past are used to protect further communication from attackers. Example of the SAS verification dialog is depicted in figure 4.
- ZRTP negotiation happens in the RTP media channel, it can happen directly between users, without server interaction. It is a different flow from SIP signaling that uses server.
- Added value: Another layer of man-in-the-middle attack protection is employed. *ZRTP hash* sent by the SIP layer is digitally signed. If the signature matches, it is guaranteed there is no man-in-the-middle attacker in the session.

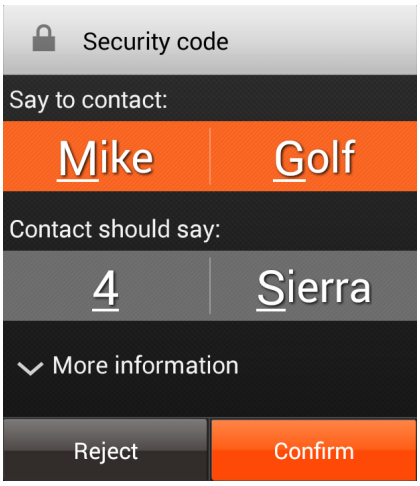

Figure 4: SAS verification dialog.

4.2 Call establishment

- Assume A wants to call B.
- A starts SIP INVITE session with B according to SIP protocol, using it's home PhoneX cluster.
- Call session is forbidden if they to not have each other in their contact lists.
- *ICE*, *STUN* and *TURN* protocols are used to establish a direct connection between A and B.
  - Server does not have to forward encrypted communication between parties.
  - Better server infrastructure scalability.
  - No direct bottleneck for multimedia sessions.
  - Connection usually has better parameters (e.g., jitter, latency).
  - Added security benefits – no central forwarding element, DoS protection.
  - Added value: We designed and published a new tunneling protocol for a symmetric NAT. This improves success ratio of establishing a direct connection between users.
  - System has a fallback option, if direct connection cannot be established due to strict firewall settings. Relay servers are used in this case.
  - Relay servers have small code and memory footprint, they can be deployed to the cloud to improve PhoneX system scalability.
- ZRTP establishes a shared symmetric encryption key over negotiated media channel.
- SRTP uses symmetric encryption to protect transmitted voice data, using key derived by ZRTP.

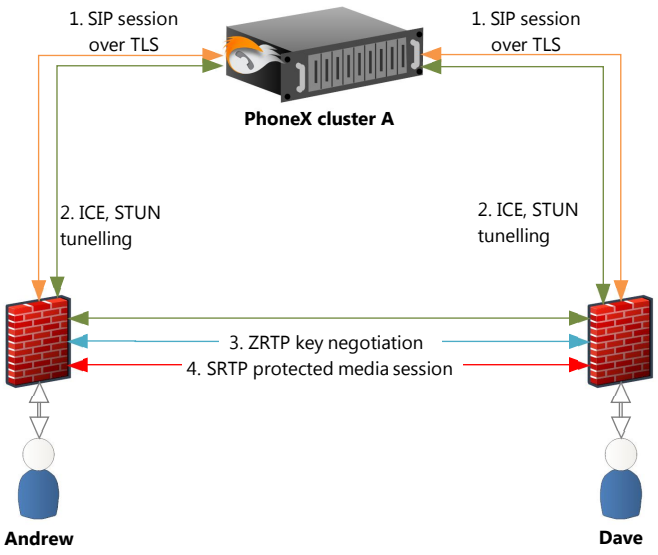

Figure 5: Simplified call setup.

4.3 Conference call

- This feature is planned on Q2/2015.
- Multiple modes of operations are designed.

5 Text messages

- Text messages work very similar to PGP. End-to-end encryption is used. Server does not see message content.
- Messages are encrypted using asymmetric public key of a recipient. Hybrid encryption with *AES-256-GCM* is used.
- Text messages are delivered by SIP protocol. It enables delivery acknowledgements and offline messages.
- Offline messages are stored on server and forwarded to the user when he is connected again.
- Message read notification (i.e., message was really seen by user in the PhoneX application) are currently being tested and will be released in a next version.
- We designed a highly flexible messaging protocol that enables us to change encryption scheme of the messages in future releases to protocols with better properties.

6 File transfer

- File transfer protocol enables secure asynchronous file transfer between PhoneX users using strong end-to-end encryption.
- Arbitrary files can be sent between users (images, music, binary data).
- Our protocol has perfect forward secrecy property. For each file, a new unique encryption password is derived with contribution of both parties.

6.1 Basic principle

- For each user in the contact list PhoneX client generates *N half-keys* and stores them on the server.
- These half-keys can be seen as an *open file slots*, that we provide for given users. I.e., if we generate 5 half-keys for user Andrew, then Andrew can send us 5 files and no more.
- This system design has a very good anti-SPAM and anti-DoS properties since we generate half-keys only for the contacts we wish to receive files from.
- System is highly flexible. Number of generated half-keys can vary from contact to contact, depending on the current needs.
- Protocol works in an asynchronous manner. Contacts does not have to be online at the same time in order to perform file transfer.

Setup

- In the initialization phase, half-keys are generated for each user in the contact list and stored on the home PhoneX cluster.
- Keys are generated in such a way, that only a recipient can understand them, nobody else (e.g., server, administrators, other users) can understand the half-key. This makes our scheme resistant to a server compromise.
- Figure 6 depicts setup phase for a file transfer.

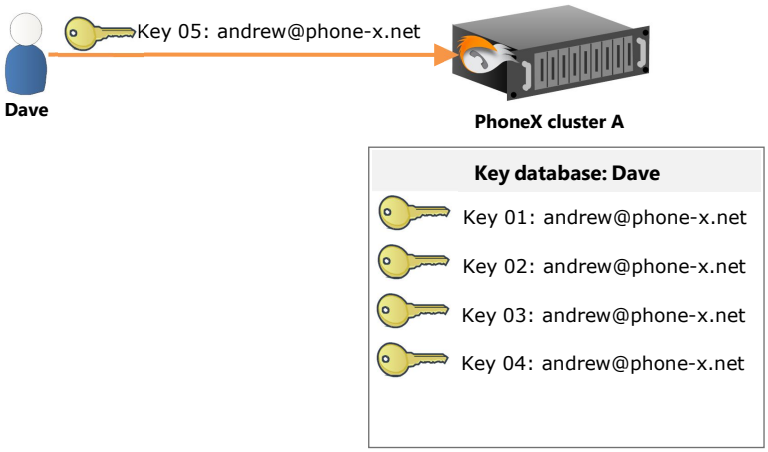

Figure 6: File transfer, key setup phase.

Transfer

- A wants to send a file to B.
- A fetches half-key from the B's server, generated by B and stored for A.
- A generates a full key using half-key.
- A encrypts a file with a strong encryption and uploads this file to the server.
- Server stores file for later download. It does not know encryption password.
- Later, when B connects to the PhoneX cluster, he is notified about a new file.
- B downloads encrypted file from the server.
- B generates full encryption key for the file, verifies file authenticity and decrypts the file.
- Figure 7 depicts file transfer protocol.

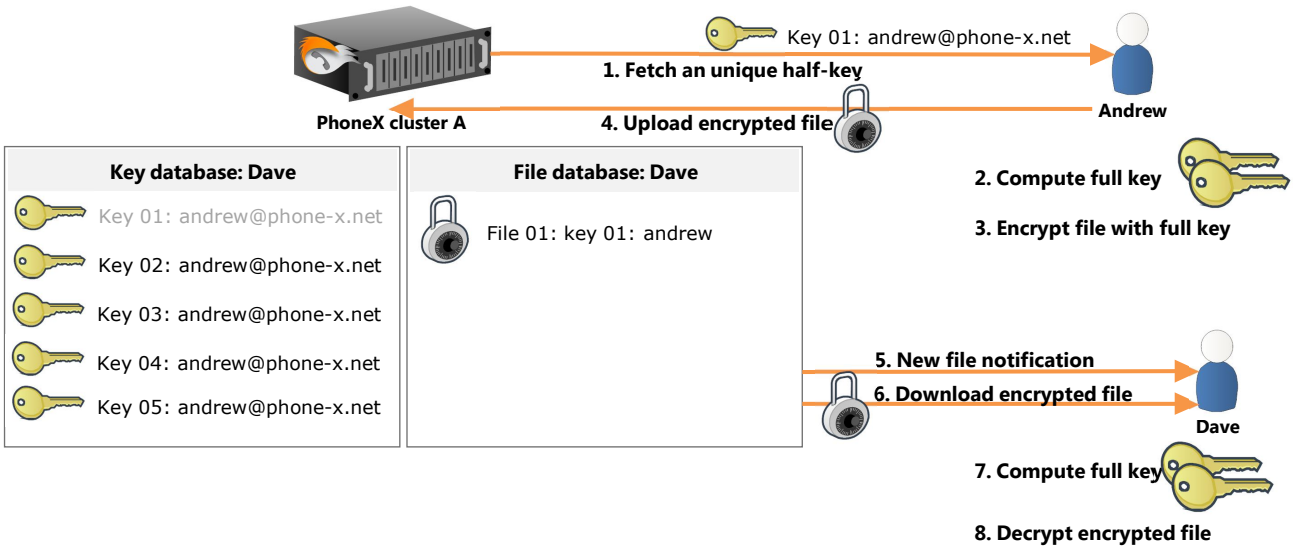

Figure 7: File transfer.

7 Further extensions

We prepare more extensions to our PhoneX system, targeted on a corporate customer. Here is the example of a few:

- Centralized corporate administration panel for PhoneX administration.
- Advanced server monitoring and usage statistics.
- Use of hardware security modules for increasing a security level against attackers.
- Secure web client.
- Sending of a voice messages via file transfer, in case of poor internet connection for voice call.
- QR codes business cards for PhoneX system.
- Integration with other IM/VoIP platforms.
- Identity aliasing. E.g., support@phone-x.net would be redirected to currently available identity in the network.
- Conference calls, messages.