Beta Test Plan

Core Functionalities

Action Plan review:

In our previous Action Plan, we listed the following functionnal specifications:

- Phone call encryption between two known pairs, that exchanged keys in person. *Mandatory*
- Phone dialer that is discret and functional, and should not disturb a normal use (clear phone call). *Mandatory*
- Phone call encryption between two unknown pairs, with key exchange on the go. Optional.
- SMS encryption between two known pairs (in person key exchange). Optional.

We now retain only the two first functional specifications.

Core Functionalities

Based on this review, here are all the core functionnalities we set:

Icing protocol

- Advanced protocol documentation, paving the way for a full RFC.

The protocol definition will include as completed:

- Peer ping
- Ephemeral key gestion
- Perfect Forward Secrecy
- Handshakes
- Real-time data-stream encryption (and decryption)
- Encrypted stream compression
- Transmission over audio stream (at least one modulation type)
- First steps in FEC (Forward Error Correction): detecting half of transmission errors

And should include prototype or scratches functionalities, among which:

- Embedded silent data transmission (such as DTMF)
- On-the-fly key exchange (does not require prior key exchange, sacrifying some security)
- Stronger FEC: detecting >80%, correcting 20% of transmission errors

The Icing dialer (based on Icing kotlin library, an Icing protocol implementation)

The Icing dialer should be a fully transparent and almost undistinguishable smartphone dialer. Any Icing-unaware user should be able to use the dialer smoothly to make calls to anyone. The dialer should propose a full set of functionnalities to handle its Icing protocol implementation.

Here is the list of all the functionnalities our dialer will integrate:

- Call
    - Ringtone on incoming call
    - Incoming and ongoing call notification
    - Complete dialer with all numbers, star *, pound #
    - Mute button
    - Speaker button
    - Normal call
    - DTMF transmission
    - SIM choice on call
- Encrypted Call
    - Encrypted call if pair public key is known
    - Encrypted DTMF transmission
    - Data rate indicator
    - Data error indicator
    - Disable encryption button
- Call history
    - Call details (timedate, duration, ring number)
    - Missed calls filter
    - Outgoing calls filter
    - Incoming calls filter
    - Call back function
    - Contact modal on history tap
    - Block call number
- Contacts
    - Sorted contact listing
    - Contact creation / editing buttons
    - Contact sharing via QR code / VCF

    - Contact search bar (application wide)
    - Favorite contacts
    - Contact preview (picture, number, public key...)
- Visual voicemail
    - Play / Pause
    - Notification
    - Quick link to call, text, block, share number...
- Miscellanous
    - Settings menu
    - Version number
    - Storage of user public keys
    - Blocklist gestion (list / add / del / search)
    - Default SIM choice
- Asymetric Keys
    - Secure storage
    - Generation at startup if missing
    - Full key management (list / add / del / search / share)
    - Secure generation (Android Keystore generation)
    - Insecure generation (RAM generation)
    - Exportation on creation (implies insecure generation)
    - Importation
    - Trust shift (shift trust from contacts)

Beta Testing Scenarios

- Clear call from Icing dialer to another dialer (Google, Apple...)
- Clear call from Icing dialer to another Icing dialer
- Clear call from Icing dialer to an icing pubkey-known contact but without Icing dialer
- Encrypted call from Icing dialer to a known contact with Icing dialer
- Encrypted call from Icing dialer to an unknown contact with Icing dialer
- Create / Edit / Save contact with(out) public key
- Share contact as QR code / Vcard
- Import contact from QR code / Vcard
- Listen to voicemail
- Record encrypted call and check the encryption
- Change default SIM

User Journeys

Mathilda, 34 years-old, connects to her PayPal account from a new device. To authenticate herself, PayPal sends her a code on her voicemail. Mathilda being aware of the risks of this technology, she has set up strong Icing authentication with her network provider by registering a pair of her Icing public keys. When she calls her voicemail, Icing protocol is triggered and checks for her key authentication ; it will fail if the caller does not pocesses the required Icing keys. Mathilda is thus the only one granted access, and she can retreive her PayPal code securely.

Jeff, 70 years-old, calls his bank after he had a problem on his bank app. The remote bank advisor asks him to authenticate, making him type his password on the phone dialer. By using the Icing protocol, not only would Jeff and the bank be assured that the informations are transmitted safely, but also that the call is coming from Jeff's phone and not an impersonator.

Elise, 42 years-old, is a journalist covering sensitive topics. Her work draws attention from people who want to know what she's saying - and to whom. Forced to stay discreet, with unreliable signal and a likely monitored phone line, she uses Icing dialer to make secure calls without exposing herself.

Paul, a 22 years-old developer, is enjoying its vacations abroad. But everything goes wrong! The company's product he works on, is failling in the middle of the day and no one is qualified to fix it. Paul doesn't have WiFi and his phone plan only covers voice calls in his country. With Icing dialer, he can call his collegues and help fix the problem, completely safe.

Evaluation Criteria

Protocol and lib

1. Security

- Encryption Strength: Ensure that the encryption algorithms used (AES-256, ECC) are up-to-date and secure.
- Key Management: Evaluate the mechanism for generating, distributing, and storing encryption keys (P-256 keys, ECDH).
- Forward Secrecy: Confirm that the protocol supports forward secrecy, meaning that session keys are discarded after use to prevent future decryption of past communication, and that future sessions are salted with a pseudo-random salt resulting or derived from the past calls.
- End-to-End Encryption Integrity: Verify that no clear data is exposed outside the encryption boundary (client-side only).
- Replay Protection: Ensure that the protocol includes strong mechanisms to prevent replay attacks.

2. Performance

- Latency: Measure the round-trip time (RTT) for call setup and audio quality during the call. The system should aim for the lowes latency possible.
- Bandwidth Efficiency: Evaluate the protocol’s ability to optimize bandwidth usage while maintaining acceptable audio quality.
- Audio Quality: Assess the audio quality during calls, including clarity, consistency, and minimal distortion.

3. Usability

- Ease of Integration: Evaluate how easy it is to integrate the library into an Android application, including the availability of well-documented APIs and clear examples.

- Seamless User Experience: Check for smooth call initiation, handling of dropped calls, and reconnection strategies. The app should handle background operation gracefully.
- UI/UX Design: Assess the user interface (UI) of the Android dialer for intuitiveness, accessibility, and if it could be a drop-in replacement for the original dialer.
- Error Handling and Recovery: Evaluate how the system handles unexpected errors (e.g., network issues, connection drops) and recovers from them.

4. Interoperability

- Support for Multiple Protocols: Verify if the protocol can integrate with existing standards (e.g., SIP, WebRTC) for interoperability with other services.
- Cross-device Compatibility: Ensure that calls encryption can be initiated and received across different devices, operating systems, and network conditions.
- Backward Compatibility: Test whether the protocol is backward compatible.

5. Privacy

- Data Storage: Evaluate how the system stores any data (user details, identities). Ensure that sensitive information is encrypted.
- Data Minimization: Ensure that only the minimum necessary data is used for the protocol to function.
- No Call Metadata Storage: Ensure that no metadata (e.g., call logs, duration, timestamps) is stored unless necessary, and, if stored, it should be encrypted.

6. Maintainability

- Code Quality: Review the library for clarity, readability, and maintainability of the code. It should be modular and well-documented.
- Documentation: Ensure that the protocol and library come with thorough documentation, including how-to guides and troubleshooting resources.
- Active Development and Community: Check the active development of the protocol and library (open-source contributions, GitHub repository activity).

Dialer

1. User Interface

- Design and Layout: Ensure that the dialer interface is simple, intuitive, and easy to navigate. Buttons should be appropriately sized, and layout should prioritize accessibility.
- Dialer Search and History: Ensure there’s an efficient contact search, history logging, and favorites integration.
- Visual Feedback: Verify that the app most usefull buttons provides visual feedback for actions, such as dialling, calls available interactions for example.

2. Call Management

- Call Initiation: Test the ease of initiating a call from contact list, recent call logs, contact search or direct number input.
- Incoming Call Handling: Verify the visual and audio prompts when receiving calls, including notifications for missed calls.
- Call Hold/Transfer/Forward: Ensure the dialer supports call hold, transfer, and forwarding features.
- Audio Controls: Check whether the app allows users to adjust speaker volume, mute, and switch between earpiece/speakerphone.

3. Integration with System Features

- Permissions: Ensure the app requests and manages necessary permissions (microphone, camera for scanning QR codes, contacts, call history, local storage).
- Integration with Contacts: Ensure that the app seamlessly integrates with the Android contacts and syncs correctly with the address book.
- Notifications: Ensure that call notifications and ringtone works even when the app is in the background or the phone is locked.

4. Resource Management

- Resource Efficiency: Ensure the app doesn’t excessively consume CPU or memory while operating, during idle times or on call.

5. Security and Privacy

- App Encryption: Ensure that any stored and sensitive data is encrypted, or protected.
- Secure Call Handling: Verify that calls are handled securely through the encrypted voice protocol.
- Minimal Permissions: The app should ask for the least amount of permissions necessary to function.

6. Reliability

- Crash Resistance: Test for the app’s stability, ensuring it doesn't crash or freeze during use.