QUANTUM-RESISTANT CRYPTOGRAPHIC APPLICATION FOR SECURE DATA STORAGE ON OS ANDROID

Pavol Michalec
Bachelor Degree Programme (3), FEEC BUT
E-mail: xmicha62@vutbr.cz

Supervised by: Lukáš Malina
E-mail: malina@feec.vutbr.cz

Abstract: Data security is an essential part of many Android OS applications. This paper presents a secure implementations including Post-Quantum Cryptographic (PQC) protocols that are going to be very important in the near future with the development of quantum computers.

Keywords: Android, Authentication, Cryptography, Post-Quantum Cryptography, Security

1 ÚVOD

V dnešnej dobe je platforma Android základným OS u väčšiny smartfónov, preto je nevyhnutné dbať na bezpečnosť dát uložených na týchto zariadeniach. Široká dostupnosť kryptografických knižníc nám umožňuje dáta zabezpečiť avšak je nutné vhodne zvoliť kryptografické schémy a správne ich implementovať s čo najmenším zaťažením koncového užívateľa. Náš projekt sa zameriava na implementáciu bezpečného uloženia dát na platforme Android. Cieľom projektu je aj zabezpečenie dát do budúcnosti formou post-kvantovej kryptografie ktorá je odolná aj voči kvantovým počítačom.

2 APLIKOVANÁ KRYPTOGRAFIA NA OS ANDROID

V praxi sa najčastejšie využívajú dve techniky pre ukladanie šifrovacích kľúčov. Bezpečný hardware (Keystore) alebo vo forme súboru na disku (najčastejšie formou SharedPreferences), prípadne v RAM zariadenia. Možné je aj využiť Secure Element na externej microSD karte, avšak tieto karty nie sú bežne dostupné, preto nie sú v tejto práci zahrnuté.

2.1 ANDROID KEYSTORE

Keystore funguje ako bezpečný kontajner pre kryptografické operácie [1]. Tieto operácie sa nevykonávajú priamo v aplikácií ale v bezpečne oddelenej časti systému. Citlivé údaje, napríklad šifrovací kľúč, sa teda nikde v aplikácií nevyskytuje a nie je možná jeho extrakcia zo zariadenia. Jednotlivé kľúče uložené v Keystore sú viazané na aplikácie, ktoré ich vytvorili. Aplikácia A teda nemôže pristúpiť ku kľúču aplikácie B a naopak. Taktiež je možné jednotlivým kľúčom priradiť ich funkciu. Napríklad môžeme kľúč označiť ako šifrovací a jediná operácia, ktorú nám Keystore s daným kľúčom povolí je šifrovanie.

2.2 ULOŽENIE ŠIFROVACIEHO KĽÚČA NA DISK ZARIADENIA

Šifrovacie kľúče môžeme na disk uložiť v otvorenej (plaintext) forme alebo v zašifrovanej. Nakoľko otvorená podoba nepredstavuje žiadne zabezpečenie, nie je vhodná. Lepšou alternatívou je daný kľúč zašifrovať a uložiť do súkromných súborov aplikácie alebo do shared preferences. Prvá línia obrany pred zneužitím týchto kľúčov je samotný Android a hlavne SELinux [2]. Práve vďaka nemu iné aplikácie nemajú prístup k súkromným súborom inej aplikácie. Samo o sebe to však nie je dostačujúce a

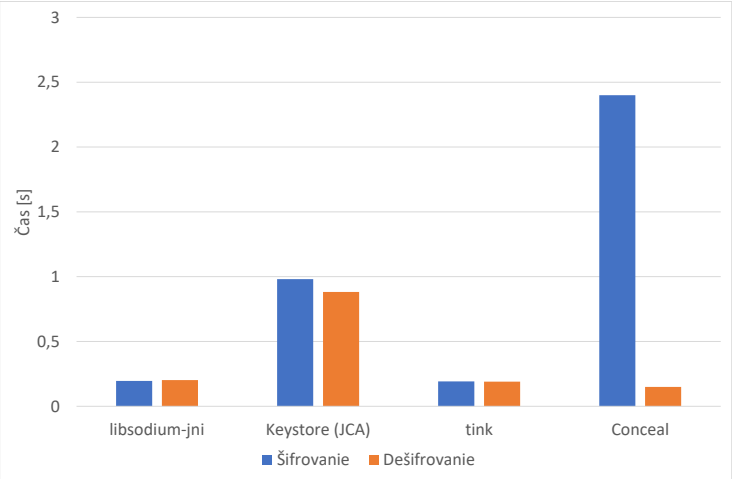
tieto súbory zároveň šifrujeme kľúčom derivovaným od hesla zadaného užívateľom. Výhodou tohoto prístupu je kompatibilita s inými zariadeniami. Šifrovací kľúč nemusí byť derivovaný od hesla užívateľa, môže byť umiestnený v kóde aplikácie a skrytý (obfuscation). Avšak taktiež nejde o bezpečnú metódu. Podobne je možné využiť na uloženie šifrovacieho kľúča Keystore. V tomto prípade by sa ale kľúč krátkodobo nachádzal v pamäti zariadenia odkiaľ by ho mohol útočník získať.

2.3 Experimentálne porovnanie kryptografických knižníc

Pre výkonové porovnanie najpopulárnejších knižníc sme vytvorili testovaciu aplikáciu. Pomocou jednotlivých knižníc sa načíta súbor o veľkosti 6 MB, zašifruje sa pomocou autentizovanej šifry (AES-GCM alebo ChaCha20poly1305) a uloží sa na disk. Následne sa zašifrovaný súbor načíta, dešifruje a znova uloží. Integrita výsledného súboru s pôvodným bola manuálne overená pomocou SHA-256. Do porovnania bola zahrnutá východzia implementácia JCA (Java Cryptography Architecture) s využitím Android Keystore. Jednotlivé knižnice mali nasledovnú konfiguráciu (nebolo možné nájsť zhodnú konfiguráciu vo všetkých knižniciach):

- **libsodium-jni** – Algoritmus: *ChaCha20poly1305*, Dĺžka kľúča: *256 b*
- **Keystore (JCA)** – Algoritmus: *AES-GCM*, Dĺžka kľúča: *128 b* (na testovacom zariadení nebola podpora pre 256 bitovú dĺžku)
- **tink** – Algoritmus: *AES-GCM*, Dĺžka kľúča: *256 b*
- **Conceal** – Algoritmus: *AES-GCM*, Dĺžka kľúča: *256 b*

Každá knižnica pracovala s rovnakým vstupným súborom a šifrovanie aj dešifrovanie celkovo bežalo 10× za sebou. Výsledný čas bol spriemerovaný. Ako testovacie zariadenie bol použitý Samsung Galaxy S5 (Qualcomm Snapdragon 801 MSM8974-AC, 4 jadrá @ 2,5 GHz, 2 GB RAM) s Android 6. Výsledné časy je možné vidieť na obrázku 1.



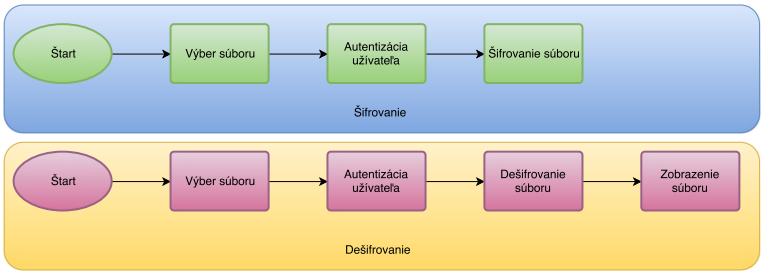
Obr. 1: Porovnanie výkonu kryptografických knižníc.

3 NÁVRH A IMPLEMENTÁCIA APLIKÁCIE POSKYTUJÚCEJ OCHRANU DÁT NA OS ANDROID

Ako kryptografickú knižnicu sme zvolili východiskovú implementáciu JCA s využitím Keystore. Vďaka tomu sa budú všetky kľúče uchovávať v bezpečnom hardware, kde sa zároveň budú vykonávať kryptografické operácie. Aplikácia vyžaduje nastavené bezpečné uzamykanie obrazovky. Užívateľ sa musí do zariadenia autentizovať pri využití šifrovania/dešifrovania, avšak je implementovaná

doba platnosti autentizácie, po ktorú nie je nutné znova autentizovať užívateľa. Použitý algoritmus je AES-GCM s dĺžkou kľúča 256 bitov (Android aktuálne nepodporuje žiadny algoritmus s väčšou dĺžkou kľúča). Výhodou tohoto riešenia je, že užívateľ si nemusí pamätať dodatočné heslá a je to preňho jednoduchšie. Zjednodušenú schému je možné vidieť na obrázku 2.

Súbory je možné zálohovať na server. Prenos súborov je zabezpečený symetrickou šifrou a pre ustanovenie kľúča bola využitá asymetrická kryptografia vrátane post-kvantovej. Využité sú schémy *NewHope* a *MSR LN16* založené na probléme RLWE (ring learning with errors). Aj keby útočník odchytil šifrovanú komunikáciu medzi zariadením a serverom, nebude ani v budúcnosti schopný prelomiť post-kvantovú kryptografiu využitím kvantových počítačov a Shorovho algoritmu.



Obr. 2: Bloková schéma aplikácie.

4 INTEGRÁCIA POST-KVANTOVEJ KRYPTOGRAFIE NA OS ANDROID

Väčšina post-kvantových algoritmov je zatiaľ implementovaná najmä pre jazyk C/C++. Naše riešenie využíva knižnicu *liboqs* (projekt Open Quantum Safe [3]). Ide o zoskupenie viacerých algoritmov do jednej multiplatformovej knižnice (v jazyku C). Pre využitie na platforme Android je nutné použiť Java Native Interface (JNI) vďaka ktorému je možné z jazyka Java volať natívne funkcie knižnice v jazyku C. Pre použitie JNI je nutné čiastočne modifikovať pôvodný kód aby ho bolo možné využívať pomocou Javy. Obe schémy boli testované z hľadiska času, potrebného na ustanovanie kľúča. Testovanie prebehlo na rovnakom zariadení ako v kapitole 2.3. Výsledky: *MSR LN16* (JNI) – 1,73 ms, *NewHope* (Java) – 123,46 ms.

5 ZÁVER

Android ponúka širokú škálu dostupných kryptografických knižníc, je však na vývojárovi aby ich implementoval bezpečne a efektívne. Predložená aplikácia využíva perspektívne kryptografické schémy pre bezpečné a efektívne ukladanie dát na OS Androide. Hlavnou výhodou aplikácie bude aj odolnosť voči útokom využívajúce kvantové počítače. Šifrované dáta a kľúče budú pomocou postkvantovej kryptografie dlhodobo zabezpečené.

LITERATÚRA

[1] *Android Keystore System* [online]. [cit. 2017-10-30]. Dostupné z: https://developer.android.com/training/articles/keystore.html

[2] COOIJMANS, Tim, Joeri DE RUITER a Erik POLL. Analysis of Secure Key Storage Solutions on Android. In: *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones*. New York, New York, USA: ACM Press, 2014, s. 11-20. DOI: 10.1145/2666620.2666627. ISBN 9781450331555. Dostupné také z: http://dl.acm.org/citation.cfm?doid=2666620.2666627

[3] MOSCA, Michele a Douglas STEBILA. *Open Quantum Safe* [online]. [cit. 2018-03-06]. Dostupné z: https://openquantumsafe.org/