Îmbunătățirea Securității în Aplicația Android

folosind Google Nearby Connections și Criptare

Introducere

Pe măsură ce utilizarea aplicațiilor mobile continuă să crească, securitatea datelor devine din ce în ce mai importantă. În cadrul dezvoltării aplicațiilor Android care utilizează Google Nearby Connections pentru comunicarea între dispozitive, protejarea confidențialității și a integrității datelor este crucială. Acest document explorează și demonstrează modul în care implementarea tehnicilor criptografice avansate, cum ar fi AES și RSA, a condus la îmbunătățiri semnificative în securitatea aplicațiilor.

Tehnologii și Mecanisme Utilizate

Google Nearby Connections

Google Nearby Connections este o platformă dezvoltată de Google care permite dispozitivelor mobile să comunice între ele într-un mod simplu și eficient fără a necesita o conexiune la Internet. Serviciul folosește conexiuni fizice între dispozitive precum Bluetooth, Wi-Fi și ultrasunete pentru a facilita schimbul de date.

Caracteristicile principale:

- Descoperirea dispozitivelor: Permite dispozitivelor să descopere alte dispozitive din apropiere și să stabilească conexiuni între ele.
- Schimbul de mesaje: Oferă un sistem robust de mesagerie între dispozitive și facilitează comunicarea în timp real.
- Conectivitate peer-to-peer: Permite dispozitivelor să stabilească conexiuni directe între ele.
- Reziliență la schimbarea circumstanțelor: Se adaptează la schimbările din mediu, cum ar fi mișcarea dispozitivului sau interferența cu alte dispozitive.

Algoritmul de criptare AES (Advanced Encryption Standard)

AES este un algoritm de criptare simetrică, adoptat la nivel global pentru securizarea informațiilor sensibile. Principalele caracteristici ale AES includ:

- Simetrie: Aceeași cheie este utilizată atât pentru criptare, cât și pentru decriptare.
- Blocuri de date: Funcționează pe blocuri de date fixe (de obicei 128 de biți).
- Chei cu lungime variabilă: chei de 128, 192 sau 256 de biți pot fi utilizate, oferind niveluri de securitate flexibile.
- Eficiență: Cunoscut pentru eficiența implementării și a performanței.

Algoritmul de criptare RSA (Rivest-Shamir-Adleman)

RSA este un algoritm de criptare asimetric bazat pe utilizarea unei perechi de chei public-private. Principalele caracteristici ale RSA sunt:

- Chei publice și private: Permite utilizarea cheii publice pentru criptare și cheii private pentru decriptare.
- Securitate ridicată: Oferă securitate puternică bazată pe dificultatea factorizării numerelor mari.
- Semnături digitale: Poate fi folosit pentru a crea semnături digitale pentru a îmbunătăți autentificarea și integritatea datelor.
- Comunicare securizata: Posibilitatea de a facilita comunicarea securizată între dispositive.

Provocări Inițiale

Înainte de implementarea criptării avansate, aplicațiile se confruntau cu anumite vulnerabilități și riscuri de securitate:

- Eavesdropping (interceptarea mesajelor): Comunicațiile dintre dispozitivele din apropiere pot fi interceptate, datele putand fi accesate de către terți neautorizați.
- Atacuri de tip Man-in-the-middle (MITM): Permiterea unui atacator să interfereze între dispozitivele care comunică și să modifice sau să obțină acces neautorizat la datele transmise.
- Lipsa criptării avansate: Transmiterea datelor necriptate prin canalele de comunicare prezintă riscuri semnificative, deoarece informațiile transmise sunt expuse unor potențiale atacuri și acces neautorizat.

Obiective

Următoarele obiective au scopul de a îmbunătăți securitatea aplicației folosind algoritmi de criptare.

- Dispozitivele se conectează numai atunci când fiecare dispozitiv prezintă un jeton de autentificare. Acesta este criptat in momentul in care este primit.
- Criptarea mesajelor pentru comunicare securizată.

Implementare

In cadrul dezvoltarii aplicațiilor Android care utilizează Google Nearby Connections, există mai multe metode și clase care sunt importante pentru comunicarea între dispozitive. In continuare voi prezenta cele mai importante metode si clase care au fost folosite in aplicatie:

- ConnectionsClient: Această clasă reprezintă clientul principal pentru gestionarea conexiunilor prin Google Nearby Connections și este folosită pentru a iniția, accepta și gestiona conexiunile între dispozitive.

```
ConnectionsClient connectionsClient = Nearby.getConnectionsClient(context);
```

- EndpointDiscoveryCallback: este o interfață care definește metodele care sunt apelate în timpul procesului de descoperire a dispozitivelor din apropiere.

```
EndpointDiscoveryCallback endpointDiscoveryCallback = new EndpointDiscoveryCallback() {
  // Metodele callback pentru gestionarea descoperirii dispozitivelor
```

```
};
```

- ConnectionLifecycleCallback: Această interfață definește metode care sunt apelate în timpul ciclului de viață al unei conexiuni.

```
ConnectionLifecycleCallback connectionLifecycleCallback = new
ConnectionLifecycleCallback() {
   // Metodele callback pentru gestionarea ciclului de viață al conexiunii
};
```

- PayloadCallback: Interfață care definește metodele care trebuie apelate în timpul primirii sau trimiterii payload-ului.

```
PayloadCallback payloadCallback = new PayloadCallback() {
   // Metodele callback pentru gestionarea Payload-urilor
};
```

- Payload: Această clasă reprezintă datele trimise și primite între dispozitive care utilizează Google Nearby Connections.

```
Payload payload = Payload.fromBytes("Data".getBytes());
```

- ConnectionInfo: clasă folosită pentru a furniza informații despre conexiunile . (Numele dispozitivului și ID-ul utilizatorului).

```
ConnectionInfo connectionInfo = new ConnectionInfo(endpointId, endpointName,
authenticationToken);
```

- AdvertisingOptions și DiscoveryOptions: Aceste clase sunt utilizate pentru a specifica opțiunile de publicitate și, respectiv, opțiunile de descoperire.

```
AdvertisingOptions advertisingOptions = new
AdvertisingOptions.Builder().setStrategy(STRATEGY).build();

DiscoveryOptions discoveryOptions = new
DiscoveryOptions.Builder().setStrategy(STRATEGY).build();
```

Stabilirea conexiunii

În contextul Google Nearby Connections, SERVICE_ID se referă la identificatorul asociat serviciului pe care dispozitivele îl utilizează pentru a se descoperi reciproc și a stabili o conexiune. Acest identificator este folosit pentru a distinge serviciul aplicației tale de alte servicii și reprezintă o parte crucială a procesului de descoperire a dispozitivelor și stabilire a conexiunii.

SERVICE_ID este esențial pentru conectarea dispozitivelor care își propun să ofere sau să caute un anumit serviciu. Un dispozitiv care nu are acest SERVICE_ID nu va putea să se conecteze la alte dispozitive care utilizează același identificator de serviciu.

Pentru a preveni conectarea dispozitivului la un dispozitiv extern, după ce dispozitivul Discovery descoperă dispozitivul Advertise, acesta verifică dacă dispozitivul descoperit are token-ul de autentificare corect.

Valoarea token-ului este criptată și transferată celuilalt dispozitiv atunci când conexiunea este stabilită. Dacă token-ul primit este valid, conexiunea este acceptată si dispozitivele pot comunica intre ele.

Pentru criptarea acestei valori se va folosi algoritmul de criptare AES. Cheia simetrica folosita pentru criptarea si decriptarea acestei valori va fi salvata in codul aplicatiei, dispozitivele putand face schimb de mesaje criptate fara a fi nevoie de schimbul unor chei.

Criptarea datelor

Google Nearby foloseste un sistem de criptare in momentul incare mesajele sunt schimbate intre dispositive. In cadrul proiectului am dorit sa imbunatatim aceasta criptare adaugand un nou strat de criptare folosind algoritmii AES si RSA.

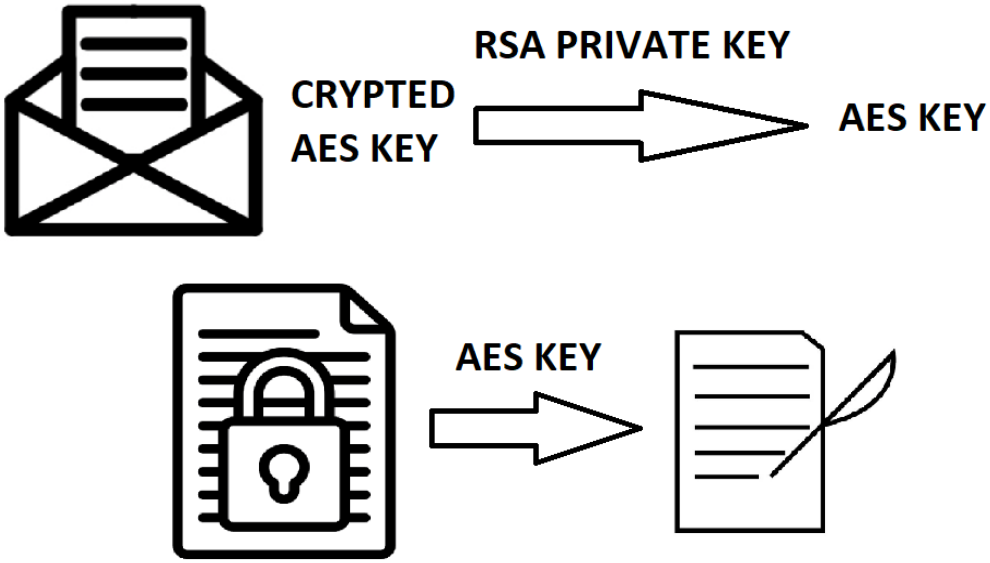
Dimensiunile cheilor sunt:

- RSA – 2048 bits
- AES – 256 bits

La stabilirea unei conexiuni, dispozitivele schimbă chei publice pentru algoritmul RSA. Aceste chei vor fi folosite pentru criptarea chei simetrice de la algoritmul AES prin intermediul careia vom cripta/decripta mesajele trimise.

Când se trimite un mesaj, acesta este criptat folosind algoritmul AES.Am folosit acest algoritm în loc de algoritmul RSA, deoarece nu poate cripta mesajele de dimensiunea unei imagini. Dupa criptarea imaginii, se va cripta si cheia simetrica de la algoritmul AES folosind cheia publica a dispozitivului catre care vom trimite mesajul. Atat imaginea cat si cheia algoritmului AES vor fi trimise catre dispozitiv.

In momentul in care mesajul este receptionat de catre dispozitiv, acesta va fi impartit in cheia criptata a algoritmului AES si mesajul criptat. Se va folosi cheia privata a algoritmului RSA pentru decriptarea cheii criptate a algoritmului AES. Dupa ce cheia AES a fost decriptata, se va folosi pentru obtinerea mesajului decriptat.



Rezultate

Impactul asupra timpui de executie

Deoarece in cadrul unui singur transfer de mesaje exista multiple criptari si decriptari, timpul de executie este afectat. Pentru a vizualiza cat de impactat este timpul de executie, vom analiza cat dureaza transferul unei imagini de dimensiunea 166 x 500. In continuare vor fi prezentate timestamp-urile pentru operatiile de trimis si primit mesaje.

/// Fara criptare/decriptare

2024-01-15 00:26:43.043 6111-6111 System.out com.example.bluetoothconnection I
Time passed: Payload received in Advertise (Start) - 1705271203043

2024-01-15 00:26:48.052 6111-6111 System.out com.example.bluetoothconnection I
Time passed: Send message in Advertise (Start) - 1705271208051

2024-01-15 00:26:48.057 6111-6111 System.out com.example.bluetoothconnection I
Time passed: Send message in Advertise (End) - 1705271208057

2024-01-15 00:26:48.057 6111-6111 System.out com.example.bluetoothconnection I
Time passed: Payload received in Advertise (End) - 1705271208057

2024-01-15 00:28:48.833 26613-26613 System.out com.example.bluetoothconnection I
Time passed: Payload received in Discovery (Start) - 1705271328833

2024-01-15 00:28:50.314 26613-26613 System.out com.example.bluetoothconnection I
Time passed: Send message to single endpoint in Discovery (Start) - 1705271330314

2024-01-15 00:28:50.316 26613-26613 System.out com.example.bluetoothconnection I
Time passed: Send message to single endpoint in Discovery (End) - 1705271330316

2024-01-15 00:28:50.316 26613-26613 System.out com.example.bluetoothconnection I
Time passed: Payload received in Discovery (End) - 1705271330316

/// Folosind criptare/decriptare

2024-01-15 00:33:51.734 10944-10944 System.out com.example.bluetoothconnection I
Time passed: Payload received in Advertise (Start) - 1705271631734

2024-01-15 00:33:56.775 10944-10944 System.out com.example.bluetoothconnection I
Time passed: Send message in Advertise (Start) - 1705271636775

2024-01-15 00:33:56.794 10944-10944 System.out com.example.bluetoothconnection I
Time passed: Send message in Advertise (End) - 1705271636794

2024-01-15 00:33:56.794 10944-10944 System.out com.example.bluetoothconnection I
Time passed: Payload received in Advertise (End) - 1705271636794

2024-01-15 00:35:01.766 28785-28785 System.out com.example.bluetoothconnection I
Time passed: Payload received in Discovery (Start) - 1705271701766

2024-01-15 00:35:03.346 28785-28785 System.out com.example.bluetoothconnection I
Time passed: Send message to single endpoint in Discovery (Start) - 1705271703346

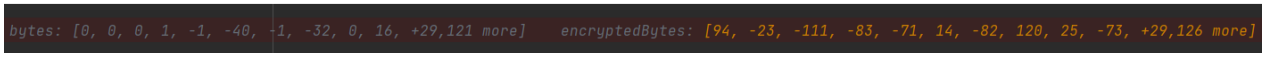
2024-01-15 00:35:03.362 28785-28785 System.out com.example.bluetoothconnection I
Time passed: Send message to single endpoint in Discovery (End) - 1705271703362

2024-01-15 00:35:03.362 28785-28785 System.out com.example.bluetoothconnection I
Time passed: Payload received in Discovery (End) – 1705271703362

Analizand aceste operatii, putem observa ca nu exista diferente mari intre transmiterea mesajelor folosind criptarea si transmiterea mesajelor nefolosind criptarea nu este una mare. Diferenta la trimiterea mesajelor este de cateva milisecunde (in medie 10ms) iar la primire si procesarea acestora este de 100 de ms.

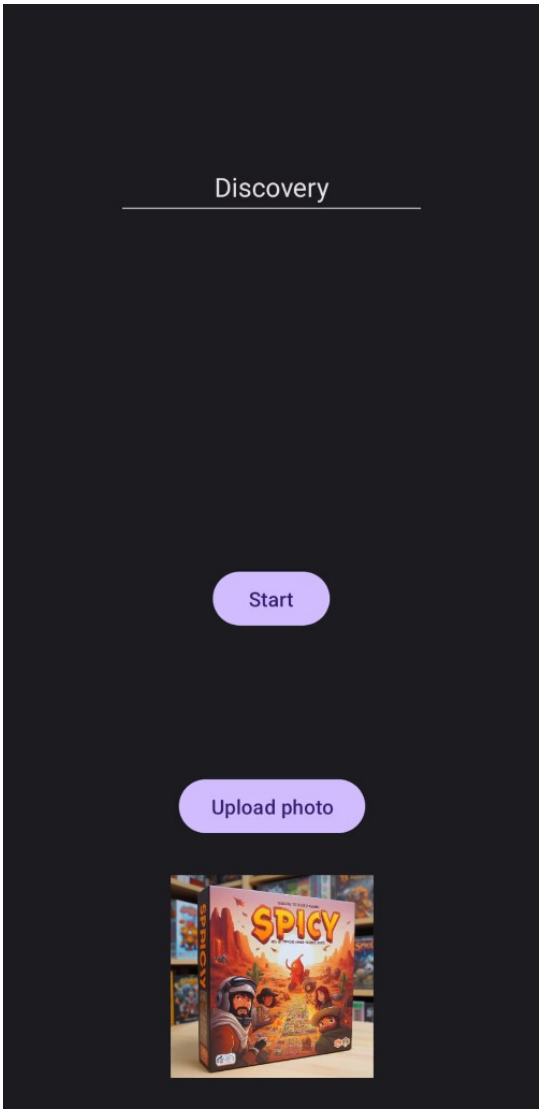
Rezultatul criptarii/decriptarii

Pentru a verifica ca implementarea propusa este una corecta, trebuie testat ca rezultatul decriptarii mesajului primit este corect. In urmatoarele imagini avem atat rezultatul criptarii cat si rezultatul decriptarii. In prima imagine se pot observa mesajul normal (variabila bytes) si mesajulcriptat (encryptedBytes),



In dispozitivul care primeste acest mesaj se va face decriptarea. Cum se poate observa, mesajul decriptat este acelasi cu mesajul initial.

Deoarece lucram cu imagini, aceasta testare se poate face mult mai usor doar analizand imaginea primita si verificand ca imaginea arata ca imaginea originala singura schimbara fiind faptul ca a fost supusa unui filtru grayscale.



Concluzii

In concluzie implementarea unor tehnici de criptare în cadrul unei aplicații Android, precum cele discutate anterior, poate aduce îmbunătățiri semnificative în ceea ce privește securitatea și confidențialitatea datelor. S-a observat că, prin aplicarea optimizărilor potrivite și selectarea algoritmilor adecvați, impactul asupra performanței poate fi redus la minimum.

Analizând metodele de generare a cheilor și algoritmii de criptare, precum RSA pentru criptarea asimetrică și AES pentru criptarea simetrică, s-a observat că performanța este în general acceptabilă în contextul aplicațiilor Android. Implementarea corectă a acestor tehnici de securitate nu numai că contribuie la protejarea datelor sensibile, dar, în același timp, oferă rezultate precise și integre.

Deși există o anumită complexitate adăugată de procesele de criptare, aceasta nu afectează semnificativ timpul de execuție în majoritatea scenariilor. Mai mult, avantajele obținute prin

asigurarea confidențialității datelor și prevenirea accesului neautorizat depășesc cu mult impactul potențial asupra performanței.

Implementarea măsurilor de criptare în cadrul aplicațiilor Android reprezintă un pas esențial pentru garantarea securității și integrității informațiilor. Cu atenția corespunzătoare la detalii și adaptarea la cerințele specifice ale aplicației, se poate realiza un echilibru eficient între securitate și performanță.