令和 2 年度 東邦大学理学部情報科学科 卒業研究

Android アプリケーションにおけるサードパーティー製 API での暗号技術利用動向の調査

学籍番号　5517097

山口千尋

金岡研究室

目 次

1　はじめに

近年開発者向けユーザブルセキュリティ、ユーザブルプライバシーの研究分野においてソフトウェア開発者の暗号技術の利用に関する研究が活発になっている。これまでの研究では、そういった暗号技術の利用が適切にされておらず脆弱性を生み出しているアプリケーション（ソフトウェア?）が多数存在することが判明している。Android アプリケーション内では、SSL/TLS の暗号技術の利用がされているケースが多くあると考えられている。

そこで、Android アプリケーションの暗号技術利用に関する全体像を知るために広くデータ分析をして、現状を明らかにするという目的のもと研究を進めていく。暗号技術の不適切利用に限った調査はいくつか行われているが、それらは SSL/TLS や DES、AES など特定の暗号技術に限った調査だけであり、暗号技術全般の網羅的な調査が行われていないという課題がある。

本研究では、API の分析から行っていく。先行研究である河合による調査では、一部の API のみで行なわれた調査であるので、より網羅的な調査のために他のライブラリや APK のデータ規模を拡大し、調査の幅を広げていく。

本稿の構成は以下のとおりである。始めに第 2 章で、本研究に関連する技術などについての解説を行い、次の第 3 章では関連研究の紹介を行う。第 4 章、第 5 章では本研究を始めるにあたっての前段階の準備と調査方法や環境についての説明をし、第 6 章では結果と考察を行う。第 7 章で本研究をまとめ、残課題について説明する。

2　前提知識

本研究における前提知識を解説する。

2.1　Android

Android とは、Googel 社が 2007 年に開発したスマートフォンやタブレット端末など携帯情報機器向けの Operating System、あるいは Android OS が搭載された端末を指す。主にスマートフォンの OS として広く普及しており、世界的に Apple 社の携帯機器向け iOS と市場を二分している [1]。

2.2　Operating System

Operating System(以後 OS) とは、ソフトウェアの種類の 1 つで、機器の基本的な管理や制御のための機能や、多くのソフトウェアが共通して利用する基本的な機能などを実装したシステム全体を管理するソフトウェアのことである。

2.3　Android アプリケーション

Android アプリケーションとは、Android にインストール可能なアプリケーションである。主に、Java や Kotlin などのプログラミング言語で作成されている。Java プログラムをコンパイルして機械語に変換し、画像などのリソースと合わせてパッケージにすることでインストール可能である。

2.4　APK

APK とは、Android Application Package の略であり、Android 向けのを Android 端末にインストールできる形式にパッケージにしたもの、もしくはそのファイルのことである。入手方法は APK ストアからダウンロードする方法や、単体で公開されている APK ファイルをダウンロードする方法等が存在する。一般的に APK は“.apk”拡張子を持つ。ただし、.apk ファイル自体は zip 形式で圧縮されており、その中にはアプリケーションの動作に必要なさまざまなファイルが納められている。.apk ファイルに対して zip ファイルと同様の解凍処理を行い、得られるファイルのうち本研究に関連する項目を解説する。

- AndroidManifest.xml
  - Android アプリケーションの必要要件や、最初に起動されるアクティビティの記述がされている
  - zip の解凍処理により得られる AndroifManifest.xml はバイナリファイルの状態であるため、テキストエディタ等で内容を閲覧するためにはデコード処理が必要である
- classes.dex
  - Android アプリケーションのソースファイルを変換して Android で実行可能なようにまとめたファイルである

– 1つの dex ファイルに含められるメソッドの数は 65,536 が上限であり、それ以上の数のメソッドが 1 つの Android アプリケーションに含まれる場合は、classes2.dex、classes3.dex …と複数ファイルに分割される

2.5 バイナリファイル

バイナリファイルとは、コンピュータプログラムによって読み書きや処理を行うことを前提に、文字コードの規約を用いずに任意のビット列によって構成されるデータを格納するものである。バイナリファイルはその形式に対応したソフトウェア以外で内容を知ることは不可能である。ただし、バイナリエディタによってどのようなバイト列が並んでいるかを見ることが可能である。

2.6 APK ストア

APK ストアとは、Android アプリケーション開発者の作成した Android アプリケーションの配信を代行するサービス、およびそれを行っている Web サイトのことである。Android の公式 APK ストアは、Android の公式 APK ストアである GooglePlay[2]1 つのみであり、非公式の APK ストアは数多く存在する。

2.7 Android Developers

Android Developers とは、Android アプリケーション開発者向けの Android 公式 Web サイトのことである [3]。Android の詳細やドキュメントが提供されている。公式ドキュメントといった場合 Android Developers を指す。

2.8 smali ファイル

smali ファイルとは、Android の Dalvik 仮想マシンで使用される開発者ファイルである。通常、Android アプリケーションに含まれている実行可能ファイルである。DEX(Dalvik Executable)（Dalvik 実行可能）ファイル（.apk ファイル）を逆コンパイルすることによって作成される。smali ファイルの取得には、Apktool[4] を用いる方法と、Baksmali[5] を用いる方法がある。

2.9 中間言語

中間言語とは、計算機が実行するコードを人間が理解できる形式で表現するための言語である。以下に本研究に関連する Dalvik バイトコードについての詳細な説明を述べる。

2.9.1 Dalvik バイトコード

Dalvik バイトコードとは、Android における中間言語である。Apktool 等を用いて APK より取得できる smali ファイルは、Dalvik バイトコードで記述されている。以下に、ソースコード 1、ソースコード 2 に Dalvik バイトコードの例と、対応するソースコードを示す。

Listing 1: 対応するソースコード

```
public int add(int a, int b) {
    int c = a + b;
    System.out.print(c);
    return c;
}
```

Listing 2: Dalvik バイトコードの例

```
# virtual methods
.method public add(II)I
    .locals 2
    .param p1, "a"    # I
    .param p2, "b"    # I

    .prologue
    .line 3
    add-int v0, p1, p2

    .line 4
    .local v0, "c":I
    sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;

    invoke-virtual {v1, v0}, Ljava/io/PrintStream;->print(I)V

    .line 5
    return v0
.end method
```

2.10 CUI

CUI とは、Character User Interface の略であり、コンピュータやソフトウェアが利用者に情報を提示したり操作を受け付けたりする方法の 1 つで、すべてのやり取りを文字によって行う方式のことである。

2.11 Linux

Linux とは、Linux カーネルを利用している UNIX 系の OS である。主にネットワーク上で他のコンピュータに機能やサービスを提供するサーバコンピュータ用として利用されるほか、スマートフォンなどの携帯端末から一般的なパソコン、家庭用ゲーム機やデジタル家電、スーパーコンピュータまで、様々な種類や用途のコンピュータ製品に組み込まれ広く普及している。

2.12 Linux カーネル

Linux カーネルとは、OS に必要な基本機能を集めた核となるソフトウェアのことである。

2.13 Linux ディストリービューション

Linux ディストリービューションとは、Linux カーネルに加えて OS として機能するよう必要なプログラム群を合わせた配布パッケージを指す。カーネルを利用者がコンピュータに導入して操作可能な状態にするために作成されている。Linux ディストリビューションは自由に開発・配布できるため、個人や数人のグループから企業、大規模オープンソースプロジェクトまで様々な開発主体が様々な機種・用途向けのものを提供している。その中の 1 つに、パソコン向けやサーバ向けとして Ubuntu がある。Ubuntu は、シェルスクリプトや smali ファイル解析に用いる環境が整っていることから本研究では Ubuntu を利用した。

2.14 シェル

シェルとは、オペレーティングシステムと対話するためのインターフェイスであり、コマンドなどを制御する環境のことである。シェルがあることでコマンドを受付、OS との対話が可能である。CUI 環境においてシェルは最も身近なインターフェイスである。

2.15 UNIX コマンド

UNIX コマンドとは、Linux OS 等の UNIX マシンにおいて CUI 上からコンピュータを操作するために使用するコマンドを指す。ファイルのコピーを行う cp、ファイルの内容を表示する cat、ディレクトリの内容を表示する ls などが存在する。

2.16 シェルスプリクト

シェルスプリクトとは、OS を操作するためのシェル上で実行できる簡易なプログラム言語（スプリクト言語）のことを言う。また、スプリクト言語によって書かれた、複数の OS コマンドや制御文などを組み合わせたプログラムを指す。sh コマンドの引数としてシェルスクリプトのファイルを与えて実行すると、ファイルに記述された UNIX コマンドが上から順に実行される。以下のシェルスクリプトを実行すると、a.txt が b.txt にコピーされ、a.txt の末尾に “ hoge ” の文字列が追加される。

Listing 3: シェルスクリプトの例

```
cp a.txt b.txt
echo "hoge" > a.txt
```

2.17 正規表現

正規表現とは，ある文字列の規則を表現する方法である。正規表現ではメタ文字と呼ばれる特別な意味を持つ文字や記号が存在する。基本的なメタ文字を表 1 に示す。ある文字列の中から通常の文字とメタ文字によって作られた特定の規則に当てはまる文字列を検索するときに利用される。正規表現の例を表 2 に示す。

表 1: 正規表現における基本的なメタ文字の一覧

.	任意の 1 文字
∗	直前のパターンの 0 回以上繰り返し (最長一致)
+	直前のパターンの 1 回以上繰り返し (最長一致)
?	直前のパターンの 0〜1 回繰り返し (最長一致)

表 2: 正規表現の例

正規表現の例	正規表現の例の意味	マッチする例
.	任意の 1 文字	a
and∗roid	an と d の 0 回以上の繰り返しと roid からなる文字列	anroid
and+roid	an と d の 1 回以上の繰り返しと roid からなる文字列	andddddroid
and?roid	an と d の 0 回〜1 回の繰り返しと roid からなる文字列	android

2.18 暗号技術

2.18.1 MD5

MD5 とは、Message Digest algorithm 5 の略であり、ハッシュ値を計算するためのハッシュ関数の 1 つである。RSA 暗号の開発者の 1 人、ロン・リベスト氏らによって開発された。IPsec や、POP before SMTP など、さまざまなセキュリティプロトコルで使われている一方、最近になって脆弱性も指摘されている。ハッシュ関数により生成された値は「ハッシュ値」と呼ばれる。MD5 のハッシュ値は、128bit である。

2.18.2 SHA-1

SHA-1 とは、アメリカ国家安全保障局が考案し、1995 年から米国政府の標準として使用されているハッシュ関数である。任意のデータから 160bit のハッシュ値を生成する。2017 年、Google が SHA-1 でハッシュ値が衝突する事例 [6] を発見したため、より安全なハッシュ関数を使用することが推奨されている。

2.18.3 SHA-2

SHA-2 とは、SHA-1 を改良したハッシュ関数である。このハッシュ関数は、バリエーション豊富であり以下を総称して SHA-2 と呼ばれている。

- SHA-224（ハッシュ値：224bit）
- SHA-256（ハッシュ値：256bit）
- SHA-384（ハッシュ値：384bit）
- SHA-512（ハッシュ値：512bit）
- SHA-512/224（ハッシュ値：224bit）

- SHA-512/256（ハッシュ値：256bit）

基本となるアルゴリズムは、SHA-256 と SHA-512 である。SHA-224 は SHA-256 で出力されたハッシュ値を 224bit に切り詰めたものであり、SHA-384 は SHA-512 で出力されたハッシュ値を 384bit に切り詰めたものである。SHA-512/224 と SHA-512/256 についても SHA-512 で出力されたハッシュ値を 224bit、256bit に切り詰めたものである。大きな違いとしては、SHA-256 は 32bitCPU、SHA-512 は 64bitCPU に最適化されている点がある。ハッシュ長が長い方がセキュリティ的な強度が高いが、負荷が高くなる。ただし、現状 SHA-256 でも必要十分な強度となっているため、SHA-256 が利用されている。

2.19 API

API とは、Application Programming Interface の略であり、あるコンピュータプログラム（ソフトウェア）の機能や管理するデータなどを、外部の他のプログラムから呼び出して利用するための手順やデータ形式などを定めた規約である。APK に使われている暗号技術は大きく 3 種類に分けられる。詳細は、第 4 章で説明する。

2.20 API ドキュメント

API ドキュメントとは、API による開発方法やクラス内のメソッドの使用方法を解説した説明書である。API リファレンス [7] とも呼ばれる。

3　関連研究

本研究における関連研究を紹介する。

3.1　河合らの調査

河合による調査は、Android アプリケーションを調査対象とし、Android アプリケーションの暗号技術利用に関する現状を明らかにするために、401,971 個 APK より展開された smali ファイルと 4,324 個の公式 API より取得した暗号・セキュリティに関するクラスが持つメソッドのリストを使用し、暗号で用いられるメソッド名や特徴のある用語によるフィルタリングアルゴリズムが指定可能な代表的箇所の抽出や API の利用傾向分析の調査を行った。

調査結果として、調査対象の APK 群に最も利用された数が多かったメソッドは android.net.Uri.parse(java.lang.String) の 248,145 個であり、次が java.net.URL.URL(java.lang.String) の 120,601 個であった。調査対象の APK 群に利用された数が多かったメソッドの上位は java.net や android.net といったネットワークに関わるものであり、調査対象の APK 群のうち 61.73 %の APK が android.net.Uri.parse(java.lang.String) を使用しており、何かしらの通信を行っていると考えられる。暗号･セキュリティに関するメソッドとして、java.security.MessageDigest.digest() の 43,418 個が最も利用されており、次に java.security.MessageDigest.getInstanc の 37,405 個であった。java.security.MessageDigest クラスは主に SHA-1 や SHA-256 といったアルゴリズムを使用したハッシュ値を提供するものである。調査対象の APK 群のうち 10.80 %がハッシュ値を利用していることがわかる。また、javax.crypto.spec.SecretKeySpec.SecretKeySpec(byte[],java.lang.String) がでメッセージダイジェストに関するクラスの次に利用数が多い。javax.crypto.spec.SecretKeySpec クラスは秘密鍵に関する機能を提供するクラスである。調査対象の APK 群において最も利用されている暗号化方式は公開鍵暗号であることが考えられる使用しており、何かしらの通信を行っていると考えられる。

しかし、河合による先行研究では Android Developers に記載されている公式の API についての調査以外は行っていない。暗号利用動向の網羅的調査のために今後これらの更なる調査、分析が必要である。

4　調査対象と手法

4.1　APK の取得

APK において利用される API の調査対象として AndroZoo のデータセットより APK を 411,486 個用意し、APK をダウンロードした。APK から smali ファイルを展開する方法として、Apktool を使用していた。本研究でも、同じ APK を使用する。

AndroZoo の参考文献

4.2　Android API の分類

APK に使われている暗号技術は大きく分けて 3 種類ある。

4.2.1　公式 API

Android の開発者向け公式 Web サイトである Android Developers の API リファレンスに記載されている API のことを本研究では公式 API と呼ぶこととする。

4.2.2　サードパーティー製 API

サードパーティー製 API とは、サードパーティが提供する API のことである。サードパーティとは、特定のハードウェア、OS、ソフトウェア、あるいはサービスなどを対象として、それに対応する製品を販売、提供している組織や企業のことを指す。・Android にもサードパーティ API が多くあるよ。

4.2.3　独自実装等の API

API 開発者が既存の API を利用せずに独自に実装した API や、先述 2 つに含まれないものを独自実装等の API と本論文では呼ぶこととする。

4.3　API の分析

河合による先行研究では、公式 API から、暗号・セキュリティに関するパッケージ、クラス、メソッドを抽出しリスト化を行った。このリストをもとに APK においてどれほど暗号技術が利用されているかの分析を行った。

独自実装等の API はドキュメントが公開されている可能性が低いため API のリスト化が困難である。これは、RSA や ECC、Crypto といった暗号、セキュリティに関するキーワードを API のリストの代わりとし検索する必要があるため APK の網羅的調査を行う上で困難である。

比較して、サードパーティー製 API ではドキュメントが公開されているものもあるのでリスト化の困難性が少ない。サードパーティ製の API の分析ではまず API のリスト化を行う必要がある

が、サードパーティー製 API は公式 API とは違いドキュメントが作成されていないものがある。存在しない場合はサードパーティ製の API のソースコードを解析し、API のドキュメントを作成してから API のリストの作成を行う。

独自実装等の API は今後の課題とする。

4.4 サードパーティ製 API の分析

4.4.1 サードパーティ製 API 例

どんなサードパーティ製 API が存在するのか紹介する。

- Tink

  Tink は、Google の暗号技術者とセキュリティエンジニアのグループが開発した、多言語でクロスプラットフォームな暗号ライブラリである。

- Conceal

  Conceal は、Facebook が開発したライブラリである。共通鍵暗号アルゴリズム AES(256bit) と暗号利用モード GCM を用いた暗号化処理を代行している。

- Geduldig

  Geduldig は、Twitter が開発した

Tink は、Android OS を提供している Google 社によるサードパーティ製 API であるため、Android アプリケーション開発者にも利用されている可能性は高いと考えられるため本研究の調査対象とする。

4.4.2 Tink の分析

Tink は現在、それぞれのプリミティブを使って実装された、4 つの暗号化操作を提供している。

- 関連データを備えた認証付き暗号 (プリミティブ: AEAD)
- メッセージ認証コード (プリミティブ: MAC)
- ディジタル署名 (プリミティブ: PublicKeySign と PublicKeyVerify)
- ハイブリッド暗号化 (プリミティブ: HybridEncrypt と HybridDecrypt)

プリミティブとは、単純あるいは基本的な構造や要素のことを言う。Tink には、ドキュメントが存在するので、ドキュメントが存在しないサードパーティ製 API よりリスト化の困難性が少ない。

4.4.3 それぞれのプリミティブの詳しい説明いれるか迷っている

暗号の説明をしていく。

- AEAD

  AEAD は、次の 3 つのアルゴリズムの組 AE = (AE-K, AE-E, AE-D) で定義される。

- MAC

  メッセージ認証コード MAC は、Message Authentication Code の略であり、ネットワークを通じて伝送されたメッセージが途中で改竄されていないかを確認することである。

- PublicKeySign と PublicKeyVerify
- HybridEncrypt と HybridDecrypt

4.5 API の取得方法

Tink のドキュメント [8] から API を抽出する。そのクラス（計 167 個）が持つメソッド計０００個のリスト化を行った。このリストは、2020 年００月のものである。リストの 1 部を抜粋し、表０に示す。リスト全体は付録 A に示す。

5　調査結果

- APK で暗号、セキュリティに関する API がどれぐらい使われているのか（割合）　- アルゴリズムを指定して取得できた中で最も使われているもの（回数）　- 今回の調査結果と河合さんの結果との比較

- まずは
- この章のなかで書くことを
- 箇条書きで書き出してみる
- ことから始めましょう

6　今後の課題

- Google 以外のサードパーティー製 API の調査　- 独自実装等の API の調査

7　まとめ

まとめえええええええええええええええええええ

参考文献

[1] Mobile Operating System Market Share Worldwide — ”StatCounter Global Stats”, http://gs.statcounter.com/os-market-share/mobile/, （参照 2021-01-21）

[2] Google LLC, ”Google Play”, https://play.google.com/store, （参照 2021-01-21）

[3] Android Developers, ”Android Developers”, https://developer.android.com/index.html?hl=ja, （参照 2021-01-21）

[4] iBotPeaches, ”Apktool”, https://ibotpeaches.github.io/Apktool/, （参照 2021-01-21）

[5] JesusFreke, ”Smali/baksmali”, https://github.com/JesusFreke/smali, （参照 2021-01-21）

[6] INTERNET Watch, ”Google 事例”https://internet.watch.impress.co.jp/docs/news/1046144.html, (参照　2021-01-21)

[7] Android Developers, ”API reference”, https://developer.android.com/reference?hl=ja, （参照 2021-01-21）

[8] Tink Cryptography API for Android,”Tink Cryptography API for Android”,https://google.github.io/tink/javadoc/tink-android/1.5.0/, （参照　2021-01-21）

[9] だれだれ, ”文献 3”, 年度

[10] だれだれ, ”文献 4”, 年度

[11] だれだれ, ”文献 5”, 年度

[12] だれだれ, ”文献 6”, 年度

[13] だれだれ, ”文献 7”, 年度

[14] だれだれ, ”文献 8”, 年度

[15] だれだれ, ”文献 9”, 年度

[16] だれだれ, ”文献 10”, 年度

[17] だれだれ, ”文献 10”, 年度

[18] だれだれ, ”文献 10”, 年度

[19] だれだれ, ”文献 10”, 年度

[20] だれだれ, ”文献 10”, 年度

[21] だれだれ, ”文献 10”, 年度

[22] だれだれ, ”文献 10”, 年度

[23] だれだれ, ”文献 10”, 年度

[24] だれだれ, ”文献 10”, 年度

[25] だれだれ, ”文献 10”, 年度

[26] だれだれ, ”文献 10”, 年度

[27] だれだれ, ”文献 10”, 年度

[28] だれだれ, ”文献 10”, 年度