Computer Security Symposium 2021
26 - 29 October 2021

Androidアプリケーションにおける暗号ライブラリ利用状況の大規模調査

金岡 晃[1,a)]　阿部 衛[1,b)]

概要：開発者を対象にしたユーザブルセキュリティ研究が活発になっており、その中でも開発者が暗号技術をどのように利用しているかに焦点を当てた研究が注目されている。これらの研究では特定の暗号技術に焦点を当ててその適切利用についての調査や対処が行われていた。ソフトウェアにおける暗号技術の利用は暗号ライブラリを利用することが主となっているが、暗号ライブラリが実際のソフトウェアにおいてどのライブラリがどれほど利用されどう利用されているかはまだわかっていない。本研究では Android アプリケーションを対象に初めて大規模に暗号ライブラリの利用状況を調査する。調査結果により代表的なライブラリや独自ライブラリの利用や、暗号アルゴリズムの固定化などの実情を明らかにする。

キーワード：暗号ライブラリ、Android、静的解析

A Large-Scale Survey of Cryptographic Library Usage in Android Applications

Akira Kanaoka[1,a)]　Mamoru Abe[1,b)]

Abstract: Usable security research targeting developers is becoming more and more active. Among them, research focusing on how developers use cryptography has been attracting attention. These studies have focused on specific cryptographic techniques to investigate and address their appropriate use. The main use of cryptography in software is through the use of cryptographic libraries. However, it is still unclear which cryptographic libraries are used in actual software, how much they are used, and how they are used. In this study, we survey the usage of cryptographic libraries in Android applications on a large scale for the first time. The results of the survey will reveal the use of typical libraries, proprietary libraries, and fixed cryptographic algorithms.

Keywords: Crypto Library, Android Applications, Static Analysis

1. はじめに

開発者向けユーザブルセキュリティはユーザブルセキュリティ研究分野の 1 つとして 2010 年代半ばより活発に研究されてきた。そこではソフトウェア開発者による暗号技術の不適切な利用に着目した研究が多くされている。そしてこれまでの研究結果から、暗号技術の基本的な部分で誤使用をしているケースが多いことが明らかになってきた。また誤使用対策の研究として、API ドキュメントの整理やサンプルコードやチュートリアルの充実さへの提言や、技術としてメタ API や抽象化、統合開発環境における支援技術の研究がされるなど、調査とその応用が盛んに研究されている。

これらは、暗号技術の中でもさらに焦点を絞って特定の技術に着目し、その技術が適切に利用されているかの調査や、それらの技術を適切に利用するための支援技術の研究であった。たとえば本分野の先駆的研究であった Egele らの研究 [1] では、Android での暗号 API に対しその誤使用

[1] 東邦大学
Toho University, Funabashi, Chiba 274–8510, Japan
[a)] akira.kanaoka@is.sci.toho-u.ac.jp
[b)] 6521001a@st.toho-u.jp

の解析手法を提案したがその誤使用の解析対象アルゴリズムがブロック暗号の利用モードと初期ベクトル、またパスワードベース暗号の Salt とストレッチング回数、乱数利用の適切さなど調査対象を絞ったものとなっていた。Chatzikonstantinou らによる研究では同様の解析ながら分析対象が広範囲になったものの、ブロック暗号の利用モードやパディング、古いアルゴリズムなど引き続き基礎的な部分に着目した解析であった [2]。Muslukhow らは同様の調査を 2018 年に再度行い、その傾向の変化を見せた [3]。サードパーティ製ライブラリに着目した分析も行われていたが、それらの調査は限定的なものであった。

Android における暗号の誤使用については、公式 API である JCE（Java Cryptography Extension）に着目した研究が広く進んでおり、サードパーティ製暗号ライブラリについてはまだ広く調査が済んでいない。さらには暗号利用には、それぞれの開発者が独自に実装をするケースがあり、この問題点は指摘されているもののその実態はまだ未調査である。

そこで本研究では以下を Research Question と置き、その答えを探るべく調査を行う。

> **Research Question:** *Android* アプリケーションでは、公式ライブラリやサードパーティ製暗号ライブラリを含めてどれくらいのライブラリがどれほど利用されているのか？

調査では、Androzoo よりランダムに取得した大量の Android アプリケーションに対して、静的解析を行った。まず 247,000 のアプリを分析し、そこで暗号利用がされていると思われる部分を抽出し、それらが実際に暗号の利用であるかを複数の視点から評価し、サードパーティ製ライブラリであるか独自実装であるかをさらに分析した。その結果、これまでのサードパーティ製ライブラリ調査では明らかになってこなかったサードパーティ製ライブラリの実態が明らかになった。また独自実装についてもその特徴が複数明らかになった。

本研究ではそれらで抽出されたサードパーティ製ライブラリをさらに調査をし、JCE を含めた暗号ライブラリやサードパーティ製ライブラリ内の暗号機能など、計 33 種類のライブラリと 91 の検出ルールにより、Android アプリケーションでこれらのライブラリがどれほど利用されているかを調査した。調査は 1,587,956 のアプリに対して実施された。また、最も使われていると予想される JCE に焦点を当て、共通鍵暗号による暗号化、公開鍵暗号による署名、メッセージダイジェスト機能（ハッシュ関数）、メッセージ認証機能（MAC）、暗号学的な疑似乱数機能といった機能がどれほど使われているかも調査した。

その結果、暗号利用のライブラリには独自実装するものや JCE の利用の簡易化をするラッパークラス・メソッドの 2 種類に大別できることや、アプリケーションの 80.29%が java.security パッケージを利用していること、暗号機能としてはハッシュ関数に代表されるメッセージダイジェスト機能が 78.13%のアプリケーションで利用されているなど、暗号利用の実態が明らかになった。

2. 関連研究

2.1 開発者向けユーザブルセキュリティ

開発者向けユーザブルセキュリティは、2010 年代半ばより研究が本格化され、近年では盛んに研究が行われている。それらの研究は

- 開発者の特徴分析 [1], [2], [3], [6], [7]
- 実装ミス等の原因把握 [9], [10], [11], [12], [13], [15], [16], [20]
- 開発者の特性を利用した実装ミス削減手法 [8], [14], [17], [19]

に大別される。

2018 年に金岡によりサーベイされた文献があるため、そちらも参照されたい [18]。

2.2 Android におけるサードパーティ製ライブラリに関する調査

Android においてサードパーティ製ライブラリに着目した調査や分析手法がいくつか提案されている。Backes らは信頼性の高いサードパーティ製ライブラリを検出する手法を提案し [5]、Derr らはサードパーティ製ライブラリが適切に更新されているかの調査を行った [4]。これらはサードパーティ製の信頼性や安全性について調査を行ったものであるが、暗号ライブラリに限ったものではなくより広範なサードパーティ製ライブラリを対象としていた。そのため暗号ライブラリについての詳細な調査は行われていなかった。

開発者向けユーザビリティ研究の 1 つでもある Muslukhow らの調査では、調査対象にサードパーティ製ライブラリを加えている [3]。そこでは Backes らの手法を応用したとされている。

3. 暗号ライブラリ、サードパーティ製ライブラリの暗号利用クラスの調査

3.1 調査対象

本研究ではまず Android アプリに対して静的解析を行い JCE や android.security といった公式の暗号関連ライブラリ以外のサードパーティ製暗号ライブラリや、暗号ライブラリではないがライブラリの中の機能として暗号機能を提供するライブラリなどの存在を調査する。調査対象となる Android アプリは、Androzoo データセット [21] より 247,000 のアプリをランダムに選んだ。

3.2 調査方法

3.2.1 暗号ライブラリ、サードパーティ製ライブラリの暗号利用クラスの抽出

暗号ライブラリやサードパーティ製ライブラリの暗号利用クラスの調査にあたり、Android アプリケーション内で呼び出されるメソッドのパッケージ名、クラス名、メソッド名と引数に着目した。暗号に関連する機能を提供しそれを開発者に認識してもらうためにこれらの名称には暗号技術であることが明記される可能性があると考えた。そこで、暗号技術の機能提供時に利用される可能性と考えられるキーワード”encryp”,”decryp”,”cipher” の 3 つに対し、これらのキーワードを含むパッケージ名、クラス名、メソッド名、引数が呼ばれた箇所を調査した。そして抽出されたパッケージ名、クラス名、メソッド名、引数を著者らが確認し、パッケージの調査等を Web 検索などで行うことにより暗号ライブラリであるか、あるいは別用途ライブラリにおける暗号機能提供用パッケージ・クラスであるかを判断し、Android アプリケーションで頻度が高く利用される暗号ライブラリやサードパーティ製ライブラリの暗号利用クラスのリストアップを行う。

3.2.2 Smali ファイルの抽出と静的解析

Android アプリケーションは ApplicationPackage（APK）として提供される。APK ファイルは ZIP 形式で圧縮がされており、その中に Android で実行可能なバイトコードとして DEX ファイルが格納されている。Linux 等で利用可能な apktool を利用することでこの DEX ファイルを Smali ファイルに変換することが可能である。

Smali ファイルでは、アプリケーションのプログラムにおいて、どのクラスのどのメソッドが呼ばれたかがテキストで記載されている。これらの情報ではクラス名だけでなくそのパッケージ名も含んでいるために、そのアプリケーションで利用されているクラスとメソッドのパッケージ情報を取得することが可能である。

本調査ではこの記載に注目した。これらの名称に暗号に関連した名称が使われているかを抽出することで暗号に関連するライブラリやパッケージ、クラスの利用を把握できると考えた。

暗号に関連した名称として、考えうるキーワードをリスト化し、検討した。検討の結果、先述した 3 つのキーワード “encryp”, “decryp”, “cipher” を含んだパッケージ名、クラス名、メソッド名、引数のいずれかがあった場合はその情報をさらに分析する対象とした。キーワードの一致検索にあたっては、大文字と小文字を区別しないものとした。

3.2.3 繰り返し作業による精緻化

アプリケーションのメソッド呼び出し時に記載される「パッケージ名、クラス名、メソッド名、引数タイプ」を 1 つのデータレコードとし、そこに 3 つのキーワードが含まれるかを調べた。そしてそのデータレコードが調査対象のアプリケーションの中でのべ何回出現するかを計上して頻度が高いものに着目し、著者らによりそのデータレコードに含まれるパッケージやクラス、メソッドが暗号関連であるかを判断した。判断にあたり、Web 検索等を用いてライブラリの公式ドキュメントやライブラリ提供組織の情報などを精査した。暗号関連であるものと判断したものと、暗号関連ではないと判断したものをフィルターのリストとして、再度アプリケーションの分析を行い、いずれのフィルターにも合致しないが頻度が高いものに関してさらに調査をおこないフィルターを充実化させていった。これらを繰り返し、フィルターに合致しないが頻度の高いデータレコードの発生がのべ 10,000 回を切るまでフィルターを構築した。

3.3 調査結果

3.3.1 フィルター構築

247,000 のアプリに対して 3 つのキーワードを含むデータレコードを計上し、暗号関連か判定をしていくことを繰り替えした結果 104 のフィルタールールが構築された。

3.3.2 暗号ライブラリ、サードパーティ製ライブラリの暗号利用クラスの整理

104 のフィルタールールのうち、暗号関連機能であると判断された 77 のルールについて、それぞれのパッケージ名をもとに Web 検索を行い、ライブラリであるかの判定を行った。ライブラリである可能性のあるものについては、API ドキュメントがあるかどうかを調べ、API ドキュメントがある場合は検出されたフィルタルール以外にも暗号関連のパッケージやクラス、メソッドがあるかを調査し、本調査に向けた暗号関連機能のリストを整理、拡充した。

結果はすべてで 28 種類であった。表 1 に調査結果の抜粋を示す。結果の詳細や各ライブラリの説明は付録にて解説する。

4. 暗号ライブラリ、サードパーティ製ライブラリの暗号利用クラスの利用状況調査

4.1 調査方法

前章で整理された 28 種類のライブラリや暗号利用クラスに加え、Android や Java が公式に提供しているライブラリやパッケージを加えた 33 種類に対しパッケージ名、クラス名、メソッド名、引数の組み合わせを表すデータレコードのフィルタルールを作成し、Android アプリケーション内でそれらのデータレコードと一致する箇所を探索し、それぞれのフィルタルールが延べ何回出現でるかを調べる

一致は文字列検索で行い、Android アプリケーションは前章と同様に apktool を用いて APK を展開し、smali ファイルを対象に検索を行う。検索に用いたフィルタルールは 91 個であった。

33 種類のパッケージや 91 のフィルタルールについては

表 1 抽出された暗号ライブラリ、サードパーティ製ライブラリの暗号利用クラス（抜粋）

ライブラリ名称	パッケージ名	分類
Spongy Castle	org.spongycastle	暗号ライブラリ
Bouncy Castle	org.bouncycastle	暗号ライブラリ
SQL Cipher	net.sqlcipher	暗号ライブラリ
JOSE: Javascript Object Signing and Encryption	com.nimbusds.jose	暗号ライブラリ
Conceal	com.facebook.crypto.cipher	暗号ライブラリ
Conscrypt	org.conscrypt	暗号ライブラリ
Amazon AWS S3 Client	com.amazonaws.services	暗号機能提供ライブラリ
okhttp	okhttp	暗号機能提供ライブラリ
AWS Key Management Service	com.amazonaws.services.kms.model	暗号機能提供ライブラリ
ExoPlayer	com.google.android.exoplayer2	暗号機能提供ライブラリ
Apache HTTP Client	org.apache.http.impl.auth	暗号機能提供ライブラリ
Visual Studio App Center	com.microsoft.appcenter.utils.crypto	暗号機能提供ライブラリ
Realm	io.realm	暗号機能提供ライブラリ
Zip4j	net.lingala.zip4j.crypto	暗号機能提供ライブラリ
Apache Common Codes	org.apache.commons.codec.digest	暗号機能提供ライブラリ
iText	com.itextpdf.text.pdf.crypto	暗号機能提供ライブラリ
Tencent Open Platform	com.tencent.utils	独自ライブラリ
Alipay wireless SDK	com.alipay.sdk.encrypt	独自ライブラリ

付録にて詳細を説明する。

4.2 調査対象

調査対象となる Android アプリは、Androzoo データセット [21] より 1,587,956 のアプリをランダムに選んだ。

4.3 調査結果：各パッケージ利用アプリ数

調査の結果の抜粋を表 2 に示す。公式パッケージである java.security の利用が 80.29%、javax.crypto が 63.37%、javax.net.ssl が 43.79%と、多くのアプリケーションで公式の暗号ライブラリが利用されていることが分かった。多くの利用によりデータ通信はデータ保管、認証等で安全性が保たれている状況にあると考えることもできるが、一方でこれまでの開発者向けユーザブルセキュリティの調査にあるように、利用はしているが正しく動作していない可能性もある。その場合は暗号技術を適用しているという過信によりむしろリスクが高まる可能性があるため、より深い調査が必要であることが示されたと言えよう。

暗号ライブラリ以外では okhttp の利用が 84.68%と大きいことが特徴的である。okhttp は広く使われていることが本調査で目立った。okhttp は、さまざまなパッケージに同梱されているケースも多く、Android 向けの http ライブラリとして一般的に利用されていることがうかがえた。同じく ExoPlayer のライブラリ利用の多さも特徴的であった。ExoPlayer は Android 用のサードパーティ製の音声・動画のライブラリであり、API の一部に暗号関連が存在している。こういった音声や動画に対しても暗号技術が広く採用されていることが伺える。

表 2 暗号ライブラリ、サードパーティ製ライブラリの暗号利用クラスの利用状況

ライブラリ名称	利用アプリ数	
java.security	1224474	80.29%
javax.crypto	966326	63.37%
javax.net.ssl	667743	43.79%
android.security	3003	0.20%
androidx.security.crypto	2257	0.15%
Spongy Castle	62098	4.07%
Bouncy Castle	86128	5.65%
SQL Cipher	20798	1.36%
JOSE	25463	1.67%
Conceal	8402	0.55%
Conscrypt	27487	1.80%
AWS S3 Client3	32148	2.11%
okhttp	1291362	84.68%
Apache HTTP Client	241681	15.85%
AWS KMS	8342	0.55%
ExoPlayer	580507	38.07%
Visual Studio App Center	10861	0.71%
Realm	26605	1.74%
Zip4j	6439	0.42%
Apache Common Codes	62578	4.10%
iText	12268	0.80%
Tencent Open Platform	12453	0.82%
Alipay wireless SDK	13507	0.89%

4.4 暗号機能の利用動向調査

前節の調査により、暗号関連ライブラリとして Java や Android が提供している公式のライブラリが広く使われていることがわかった。ここではさらに詳細の調査として、java.security と javax.crypto、androidx.security.crypto ライブラリにある代表的な暗号機能に焦点をあて、それらの

表 3　暗号機能調査の対象パッケージとクラス

暗号機能	パッケージ名	クラス名
共通鍵暗号	androidx.security.crypto	MasterKey
共通鍵暗号	androidx.security.crypto	EncryptedFile
共通鍵暗号	javax.crypto	KeyGenerator
共通鍵暗号	javax.crypto	Cipher
公開鍵暗号	java.security	PrivateKey
公開鍵暗号	java.security	Signature
公開鍵暗号	java.security	Publickey
公開鍵暗号	java.security	KeyPair
メッセージダイジェスト	java.security	MessageDigest
乱数	java.security	SecureRandom
メッセージ認証	javax.crypto	Mac

機能がどれくらいのアプリケーションで利用されているかを調査した。調査の対象の機能とそれに対応するパッケージ名とクラスの一覧を表 3 に示す。

調査方法はこれまでと同様であり、1,587,956 のアプリを展開し Smali ファイル群に対して文字列の一致検索をして計上した。

結果を表 4 に示す。調査結果をみてわかるように、利用される機能は強く偏っていることがわかる。最も多く利用されている機能はメッセージダイジェスト機能を行う MessageDigest クラスであり、次いで共通鍵暗号に用いられる Cipher クラスであった。これらのことから、アプリケーションで利用される情報の内部保護等に用いられていることが考えられる。

5. 議論

5.1 制約

本研究により、これまでは明確になってこなかった暗号ライブラリの利用状況や暗号機能の利用状況が大規模なデータセットによる評価とともに明らかになった。一方で、本研究で行った手法にはいくつかの制約が伴う。ここではそれらの制約とその影響の可能性について議論する。

5.1.1 難読化の影響

本調査では Android アプリケーションの APK ファイルを apktool により展開された smali ファイルに対して文字列検索をすることでライブラリ情報や暗号関連機能の抽出やその利用動向調査を行った。これらの調査結果は apktool が Android アプリケーションの正確な情報展開をしていると仮定してのものである。しかし APK によってはこういった展開を避けるために難読化が施されているものもあり、一部のアプリケーションでは正確な情報の展開がされていない恐れがある。今回の評価は大規模なデータセットで行ったために影響は少ないことが期待される一方で、難読化が施されるアプリケーションについてはより高いセキュリティの設定がされる可能性はあり、これらの評価結果の数値は真の値と離れた値となっている可能性は存在する。しかしそれらが発生したとしても、暗号の利用状況は高まる方向に変化することが考えられるために、本研究で得られた結果は十分な意味を持つと言える。

難読化に対応した調査も今後の課題と言えよう。たとえば [4] や [5] の研究応用や、[3] の手法により難読化に対応した調査が可能になる可能性もある。

5.1.2 名称に反映されない機能

今回の調査では、暗号に関連したメソッドについてはパッケージ名、クラス名、メソッド名、引数のいずれかに暗号の機能を示すキーワードが入る可能性がある、と想定をしてキーワードを暗号に関連する 3 つにしてライブラリの整理などを行ったが、これらのフィルターに検知されないまま見逃されている暗号関連ライブラリがまだ存在する可能性がある。こういったライブラリの発見の手法は今後の課題である。

5.1.3 独自実装とラッパークラス

暗号関連のライブラリでは、暗号のプリミティブな部分の実装がライブラリの開発者自身で行われている独自実装のケースと、既存の暗号ライブラリの利用を簡易化するためにあらかじめいくつかのパラメータを設定済みにして既存のライブラリでの暗号を利用するようなラッパークラスのケースが考えられる。それぞれのライブラリがこのどちらのケースにあたるかの判断は困難である。ソースコードを確認しなければならず、そのソースコードが公開されていないライブラリも多いことが理由である。

今回の調査では、Zip4j や kObjects は AES を独自に実装していることが github 上のソースコードを確認することで判明した。同じく okhttp や Visual Studio App Center、Java MP4 Parser、iText はラッパークラスであることが同じくソースコードにより判明したがそれ以外のライブラリについては確認が取れなかった。

独自実装であるかどうかを明確に知ることにより、より公式ライブラリの利用動向が明確になる。それにより、たとえば公式ライブラリに脆弱性が発見された場合などは影響範囲の特定と適切な対処が可能になるだろう。また、独自実装やメンテナンス性や脆弱性の生みやすさの面で推奨はされないため、こういった点でライブラリ利用をやめる検討を行うことも良いだろう。

5.2 利用アルゴリズムの特定

今回の調査では利用されるメソッドにより暗号機能の利用を調査したが、実際の暗号利用ではこれらの機能の利用にあたって暗号アルゴリズムを設定する。ここで設定される暗号アルゴリズムがすでに脆弱とされているものであったばあい、暗号技術が適用されていても安全とは言えない。利用アルゴリズムの状況も調査されるべき問題である。

一方で、利用アルゴリズムの同定は簡単ではない。たと

表 4　暗号機能調査の対象パッケージとクラスの利用状況

暗号機能	パッケージ名	クラス名	利用アプリ数	
共通鍵暗号	androidx.security.crypto	MasterKey	1208	0.08%
共通鍵暗号	androidx.security.crypto	EncryptedFile	1325	0.09%
共通鍵暗号	javax.crypto	KeyGenerator	69	0.00%
共通鍵暗号	javax.crypto	Cipher	963682	63.19%
公開鍵暗号	java.security	PrivateKey	40567	2.66%
公開鍵暗号	java.security	Signature	702922	46.09%
公開鍵暗号	java.security	Publickey	0	0.00%
公開鍵暗号	java.security	KeyPair	18	0.00%
メッセージダイジェスト	java.security	MessageDigest	1191530	78.13%
乱数	java.security	SecureRandom	0	0.00%
メッセージ認証	javax.crypto	Mac	46	0.00%

えば、ソースコードにおいて同じクラス上で直接アルゴリズムを指定している場合は、分析は容易である。しかし、アルゴリズムの設定はさまざまなケースが考えられる。たとえば設定用のクラスにより設定されたあと、そのクラスのメンバ変数やインスタンスを介してアルゴリズムが設定される場合や、初期値としてアルゴリズムがハードコードされているが、そのアルゴリズムは変更可能になっていることや、ソースコードを超えて Android のアプリケーションのリソース設定を介して設定されるケースや、SharedPreference を利用して設定されるケースなど、組み合わせは多岐にわたり、かつその追跡は再帰的になることが考えられ、分析はコストが高いことが予想される。これらも今後の課題であろう。

5.3　時系列での分析

今回の分析では Androzoo データセットを用いた。Androzoo データセットは 1600 万を超えるアプリデータを保有し、その規模に申し分はないが、それぞれの APK が実際にリリースされていた時期などの情報に関しては不完全になっており、調査したアプリが最近のアプリなのか古いアプリなのかの判断は別途調査しなければならない。これは簡単なことではなく、特に大規模な分析となると 1 つ 1 つのアプリの時期を特定することは非常にコストのかかる作業となる。

一方で、それぞれのアプリの時期を知ることで時系列で暗号の利用動向の変化がつかめるなど、分析として重要な点がまだ存在する。これらの低コストな解決方法を考え、時系列で分析することも課題である。

6.　まとめ

本研究では、これまで大規模な調査が行われてこなかった暗号ライブラリの利用状況調査を Android アプリケーションを対象に 150 万をこえるアプリをもちいて調査を行った。調査は複数にわたった。まず暗号関連のライブラリやパッケージ、クラス、メソッドに利用される可能性のあるキーワードを設定し、そのキーワードを持つパッケージ等を調査することで暗号ライブラリや暗号関連のクラスを抽出し、その整理を行った。こういった整理はこれまでは明確にされていたことは筆者の知る限りなく、1 つの成果と言えよう。本研究はされにこの整理された暗号ライブラリや暗号関連のクラスよりフィルタルールを作成し、Android アプリケーションにおいてこれらのライブラリやクラスがどれほど利用されているかを大規模の調査した。その結果、公式のライブラリが多く利用されていることや、代表的な http ライブラリが広く利用されている状況があきらかになった。さらに公式ライブラリに焦点を当て、暗号技術のうちどの機能が利用されているかの状況も調査した。その結果、ハッシュ関数に代表されるメッセージダイジェスト機能が最も利用され、次いで共通鍵暗号が利用されている状況があきらかになった。こういった暗号機能の利用実態が大規模な調査で明らかになったのも筆者の知る限りではない。これも重要な成果と言える。

謝辞　本研究の一部は JSPS 科研費 JP19K11972、JP19H04111、JP19H04101 の助成を受けたものです。

参考文献

[1] Manuel Egele, David Brumley, Yanick Fratantonio, and Christopher Kruegel. 2013. An empirical study of cryptographic misuse in android applications. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13). ACM, New York, NY, USA, 73-84. DOI: http://dx.doi.org/10.1145/2508859.2516693

[2] Alexia Chatzikonstantinou, Christoforos Ntantogian, Georgios Karopoulos, and Christos Xenakis. 2016. Evaluation of Cryptography Usage in Android Applications. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS) (BICT'15), Junichi Suzuki, Tadashi Nakano, and Henry Hess (Eds.). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 83-90. DOI: http://dx.doi.org/10.4108/eai.3-12-2015.2262471

[3] Ildar Muslukhov, Yazan Boshmaf, and Konstantin

Beznosov. 2018. Source Attribution of Cryptographic API Misuse in Android Applications. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18). Association for Computing Machinery, New York, NY, USA, 133–146. DOI:https://doi.org/10.1145/3196494.3196538

[4] Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes. 2017. Keep me Updated: An Empirical Study of Third-Party Library Updatability on Android. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17). Association for Computing Machinery, New York, NY, USA, 2187–2200. DOI:https://doi.org/10.1145/3133956.3134059

[5] Michael Backes, Sven Bugiel, and Erik Derr. 2016. Reliable Third-Party Library Detection in Android and its Security Applications. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA, 356–367. DOI:https://doi.org/10.1145/2976749.2978333

[6] David Lazar, Haogang Chen, Xi Wang, and Nickolai Zeldovich. 2014. Why does cryptographic software fail?: a case study and open problems. In Proceedings of 5th Asia-Pacific Workshop on Systems (APSys '14). ACM, New York, NY, USA, , Article 7 , 7 pages. DOI=http://dx.doi.org/10.1145/2637166.2637237

[7] Li Y., Zhang Y., Li J., Gu D. (2014) iCryptoTracer: Dynamic Analysis on Misuse of Cryptography Functions in iOS Applications. In: Au M.H., Carminati B., Kuo CC.J. (eds) Network and System Security. NSS 2015. Lecture Notes in Computer Science, vol 8792. Springer, Cham

[8] Steven Arzt, Sarah Nadi, Karim Ali, Eric Bodden, Sebastian Erdweg, and Mira Mezini. 2015. Towards secure integration of cryptographic software. In 2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!) (Onward! 2015). ACM, New York, NY, USA, 1-13. DOI: http://dx.doi.org/10.1145/2814228.2814229

[9] Y. Acar, S. Fahl and M. L. Mazurek, "You are Not Your Developer, Either: A Research Agenda for Usable Security and Privacy Research Beyond End Users," 2016 IEEE Cybersecurity Development (SecDev), Boston, MA, 2016, pp. 3-8.

[10] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek and C. Stransky, "You Get Where You're Looking for: The Impact of Information Sources on Code Security," 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 2016, pp. 289-305.

[11] Soumya Indela, Mukul Kulkarni, Kartik Nayak, and Tudor Dumitraş. 2016. Helping Johnny encrypt: toward semantic interfaces for cryptographic frameworks. In Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2016). ACM, New York, NY, USA, 180-196. DOI: https://doi.org/10.1145/2986012.2986024

[12] S. Indela, M. Kulkarni, K. Nayak and T. Dumitraş, "Toward Semantic Cryptography APIs," 2016 IEEE Cybersecurity Development (SecDev), Boston, MA, 2016, pp. 9-14. doi: 10.1109/SecDev.2016.014

[13] S. Nadi, S. Krüger, M. Mezini and E. Bodden, ""Jumping Through Hoops": Why do Java Developers Struggle with Cryptography APIs?," 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, 2016, pp. 935-946.

[14] Siqi Ma, David Lo, Teng Li, and Robert H. Deng. 2016. CDRep: Automatic Repair of Cryptographic Misuses in Android Applications. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16). ACM, New York, NY, USA, 711-722. DOI: https://doi.org/10.1145/2897845.2897896

[15] Luigi Lo Iacono, Peter Leo Gorski: I Do and I Understand. Not Yet True for Security APIs. So Sad, EuroUSEC, 2017

[16] Ksenia Ermoshina, Harry Halpin, Francesca Musiani: Can Johnny Build a Protocol? Co-ordinating developer and user intentions for privacy-enhanced secure messaging protocols , EuroUSEC, 2017

[17] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. 2017. A Stitch in Time: Supporting Android Developers in WritingSecure Code. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17). ACM, New York, NY, USA, 1065-1077. DOI: https://doi.org/10.1145/3133956.3133977

[18] 金岡 晃, "開発者向けユーザブルセキュリティ・プライバシー研究の現状調査"、コンピュータセキュリティシンポジウム 2018（CSS2018)、2018

[19] Stefan Krüger, Sarah Nadi, Michael Reif, Karim Ali, Mira Mezini, Eric Bodden, Florian Göpfert, Felix Günther, Christian Weinert, Daniel Demmler, and Ram Kamath. 2017. CogniCrypt: supporting developers in using cryptography. In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017). IEEE Press, Piscataway, NJ, USA, 931-936.

[20] Ukrop M., Matyas V. (2018) Why Johnny the Developer Can't Work with Public Key Certificates. In: Smart N. (eds) Topics in Cryptology – CT-RSA 2018. CT-RSA 2018. Lecture Notes in Computer Science, vol 10808. Springer, Cham

[21] Allix, Kevin and Bissyandé, Tegawendé F. and Klein, Jacques and Le Traon, Yves. AndroZoo: Collecting Millions of Android Apps for the Research Community. In Proceedings of the 13th International Conference on Mining Software Repositories(MSR '16), 2016

付　　録

A.1 詳細データ

詳細なデータを Github で公開している。`https://github.com/kanaoka-laboratory/CryptAPISurvey_inAndroidApps`