SIM-Sign: 実用的なAndroid端末向けMitMo対策技術

大塚　玲[1,a)]　佐藤 裕之[2]　櫻木 正一郎[2]　落合 三男[2]　横田 勇一[3]　藤澤 将吾[3]　本間 靖朗[3]
小関 松子[3]

概要：マルウェアによる金融機関等への攻撃が巧妙化し，不正送金等の不正アクセスによる被害が看過できない規模に達している．対策とされるハードウェアトークンの大量配布は安全性は高いが利便性の低さやコストが障害となり，結果として十分な普及率と利用率を達成できないでいる．全体の安全性を高めるためには，安全性のみならず，利便性, 普遍性, コストを重視し，普及率と利用率を高める対策技術が求められる．広く普及しているスマートフォンで対策技術を構成できれば普及率と利用率を大幅に向上させ，不正送金等の被害を抑制することが期待される．本稿では，SIM(Subscriber Identity Module) を利用して安全性高めた対策技術について複数の構成法を検討し，Android 端末のセキュリティ機構と SIM だけを用いて実現する実用的な Man-in-the-Mobile(MitMo) 対策技術 SIM-Sign を提案する．

SIM-Sign: A Practical Countermeasure for Man-in-the-Mobile Attacks on Android Smartphones

Akira Otsuka[1,a)]　Hiroyuki Sato[2]　Shoichiro Sakuragi[2]　Mitsuo Ochiai[2]　Yuichi Yokota[3]
Shogo Fujisawa[3]　Yasuo Homma[3]　Matsuko Koseki[3]

Abstract: Banking trojans are getting smarter and more sophisticated year by year. The financial loss at the targeted online banking services has already reached the non-negligible amount. Secure hardware tokens are already delivered to their customers. Banks are, however, reluctant to force the customers to use the complicated transaction authentication functionality. Once we could find the countermeasures on smartphones against Man-in-the-Mobile (MitMo) attacks, they become almost-universal countermeasure with rich user-experience and lower cost. In this paper, with SIM as the root of trust, we investigate three types of countermeasures against Man-in-the-Mobile attack, and propose a practical countermeasure, called SIM-Sign.

1. はじめに

金融取引等における不正アクセスや MitB(Man in the Browser) 攻撃の被害が深刻化している．PC 環境に対する MitB 攻撃の多くはブラウザ機能拡張や Javascript 等を用いた平易な攻撃だが，中には管理者権限で wininet.dll を改変して通信を改竄する MitB 攻撃も存在している [19].

トランザクション認証に対応した One-time Password トークンや暗号機能と UI 機能を有するハードウェアトークン [20] は MitB 攻撃対策として有効だが，コストや不便さに課題があり十分に利用が進んでいない．高機能で既にほとんどの利用者が所有しているスマートフォンで十分な安全性が確保できれば，利便性, 普遍性, コストを総合して極めて有効な対策技術になることが期待される．

スマートフォンに対する MitMo(Man in the Mobile) 攻撃は，Android OS や iOS のセキュリティ機構に阻まれるため，PC 環境とは異なる攻撃が必要となる．iOS ではソースコード審査を経た，Apple 社のコード署名を持つ App のみが実行可能な，Apple 社が管理するホワイトリストに掲載された App だけを実行可能とするセキュリティ機構が有効に働いており，マルウェア被害は無視できる

[1] 国立研究開発法人産業技術総合研究所
〒 135–0064 東京都江東区青海 2–3–26
[2] 株式会社さくら情報システム
〒 108–8650 東京都港区白金 1–17–3 NBF プラチナタワー
[3] 東北インフォメーション・システムズ株式会社
〒 980–0021 仙台市青葉区中央二丁目 9 番 10 号
[a)] otsuka@ni.aist.go.jp

ほど少ない [5]．一方，Android OS は，iOS のようなホワイトリストの仕組みを持たないため，有名な App に偽装した Repackaging が野放しになっており，マルウェアの 80%以上が Repackaging 攻撃により利用者端末に導入されている [22]．さらに，Repackaging を通じて端末に侵入し，Root に権限昇格するマルウェアも多数確認されている [22]．Android 端末における MitMo 対策は，Root 権限に昇格したマルウェアに対抗しなければならず，極めて厳しい状況にある．

2009 年に Global Platform で SIM アプリ (Java Applet) を遠隔導入する規格 [13] が制定され，Type A/B の NFC 対応サービスも含めて OTA(Over The Air) でアプリケーションを遠隔導入する世界標準のプラットフォームが整った．国内キャリアも Felica 仕様と Type A/B 仕様の NFC の両方に対応したモバイル NFC サービスのプラットフォームを構築し，SIM アプリ等を提供するサービスプロバイダに SIM 領域の一部の貸与と SIM アプリの遠隔導入 [25] を提供するサービスを開始している．この仕組みを利用すれば，既に携帯電話に内蔵されて広範に普及している SIM を遠隔プログラム可能な耐タンパーデバイスとして活用できる．そこで本稿では，SIM を活用して，マルウェア対策が不安視されている Android スマートフォンに認証スキームを構成することで，MitMo 攻撃に対してどの意味で安全を満たす認証スキームを構成可能かを考察する．

2. 準備

2.1 Root 権限昇格を伴うマルウェアに対する Android 端末のセキュリティ機構

Android 4.4 以降の端末は Verified Boot[1] に対応しており，以下のコマンドにより工場出荷時の `LOCKED` 状態から `UNLOCKED` 状態に移行する，いわゆる Root 化することによって正規 Android OS 以外の OS を起動することが可能となる [1]．また，`LOCKED` 状態から `UNLOCKED` 状態への移行の際に，ユーザー領域 (/Data) はプライバシー保護のために初期化される仕様になっている．

```
fastboot flashing [unlock | lock]
```

通常，`LOCKED` 状態では Bootloader の Verified Boot[1] 機能により正規 Android OS しか起動できず，NAND ロック等の機能により OS パーティション (/System) は Read-only に制限され，改竄できないように制限されている．

`LOCKED` 状態で起動する正規 Android OS は Root 権限を一般アプリケーションに解放しない．しかし，表 1 に示すように Android OS の脆弱性を利用して Root 権限を取得する Exploit がこれまでに存在する．Dogspectus[3] が Exploit コードとして Towelroot (CVE-2014-3153) を利用したと言われているように，十分に設計されたマルウェアは表 1 の Exploit 等を利用して Root 権限に昇格をすると想定する．ただし，前述の Android OS の外部に存在する Bootloader の Verified Boot[1] 機能により，Android OS は改竄から保護されている．従って，マルウェアが Root 権限に昇格したからと言って，直ちに正規 Andoird OS に改竄を加えられる訳ではない．

表 1 Root 権限昇格の Exploit と脆弱性 [21]

Vulnerability/Exploit Name	CVE ID
mempodipper	CVE-2012-0056
exynos-abuse/Framaroot	CVE-2012-6422
diagexploit	CVE-2012-4221
perf_event_exploit	CVE-2013-2094
fb_mem_exploit	CVE-2013-2596
msm_acdb_exploit	CVE-2013-2597
msm_cameraconfig_exploit	CVE-2013-6123
get/put_user_exploit	CVE-2013-6282
futex_exploit/Towelroot	CVE-2014-3153
msm_vfe_read_exploit	CVE-2014-4321
pipe exploit	CVE-2015-1805
Ping Pong Root	CVE-2015-3636
f2fs_exploit	CVE-2015-6619
prctl_vma_exploit	CVE-2015-6640
keyring_exploit	CVE-2016-0728

2.2 前提条件

そこで，本稿では以下の 2 つの仮定を置く．すなわち，

仮定 1 利用者が意図的に UNLOCK 状態に移行しない限り，Android OS は改竄されない．

仮定 2 プログラムコードに固有の脆弱性がない限り，プログラムコードを改竄せずにその機能を変更できない．

ここで，“Android OS が改竄されない” とは単に “/System パーティションが書き換え不能である” ことを指す．また，“プログラムコードの機能を変更する” とは，対象となるプログラムコードが “ある入力に対して想定された仕様と異なる結果を出力する” ことであると定義する．仮定 2 を認めた場合，改竄されていない Android OS や，コード署名により Integrity が検証されたプログラムコードは，仕様通りの機能を持っていると考える．ただし，固有の脆弱性が存在する場合には，プログラムコードを改竄せずにその機能を変更できる可能性がある [17], [18] ので，必ずしも仕様通り機能するとは限らない．

本稿では，更に，Android OS 全体に対して仮定 1 と仮定 2 が同時に満たされる，最も強い仮定を仮定 3 とする．

仮定 3 Android OS について仮定 1 と仮定 2 が真．すなわち，利用者が意図的に UNLOCK 状態に移行せず，かつ Android OS の当該機能に関わる脆弱性がない限り，Android OS の機能を変更できない．

仮定 2 もしくは仮定 3 が満たされていても，Root 権限に昇格したマルウェアは，Android OS の仕様に従って Android OS や Android App の管理する機密情報を読み出すこと，並びに，読み出した機密情報を用いて Repackage

表 2　略語の定義

App	Application
NFC	Near Field Communication
OTA	Over The Air
SIM	Subscriber Identity Module
	(本稿では UICC, USIM と同義で扱う．)
SIM Applet	SIM 上で動作する Java Applet
SMS	short message service
SP-TSM	Service Provider - Trusted Service Manager
TEE	Trusted Execution Environment
UI	User Interface
UICC	Universal Integrated Circuit Card
USIM	Universal SIM

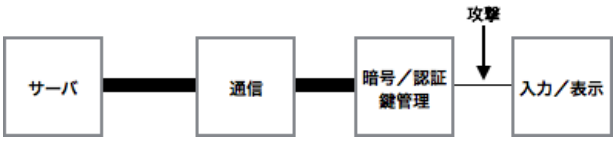

図 1　取引認証の構成と攻撃

した偽装 App を導入すること等が可能なことに注意する．

2.3　Man-in-the-Mobile 攻撃対策に求められる安全性

Android 端末を利用した取引認証においては，インターネットおよび Android 端末内に潜むマルウェアにより，全ての通信が盗聴／改竄される状況を考える必要がある．すなわち，サーバーから送られた通信内容が，利用者に正確に表示され，利用者の入力内容が改竄されずにサーバに伝達される安全な通信路 Trusted Path[4] を構成することが求められる．

Weigold ら [20] は，表示機能と OK とキャンセルの 2 つの入力ボタンを備えた USB デバイスを試作し，PC に接続された USB デバイスがサーバとの間で TLS 相互認証を行うことで，PC から分離された環境で利用者にサーバのメッセージを表示し，Trusted Path を構成している．高木ら [23] は，NFC と電子ペーパーを用い，より利便性に優れたデバイスで Trusted Path を構成している．通信機能を備える取引認証システムを図 1 に示すようにサーバと 3 つの機能で構成すると考えると，Weigold ら [20] や高木ら [23] らの方式は，通信，暗号／認証／鍵管理，入力／表示の全ての機能を一体のデバイスとして Blackbox を構成することにより，現実的な攻撃を通信チャネル経由に限定することで問題を解決していると見ることができる．他方，Android 端末で構成する場合には，各機能を Android OS や SIM, App に分割して割り当て，部分的に Whitebox として Trusted Path を構成する必要が生じる．それでも，SIM で暗号／認証に用いる鍵を管理すれば，サーバと SIM の間で適切な暗号／認証を行うことにより，図 1 の太線で示した通信路を保護することは比較的容易である．従って，主な問題は，SIM で復号した取引内容を改竄されずに

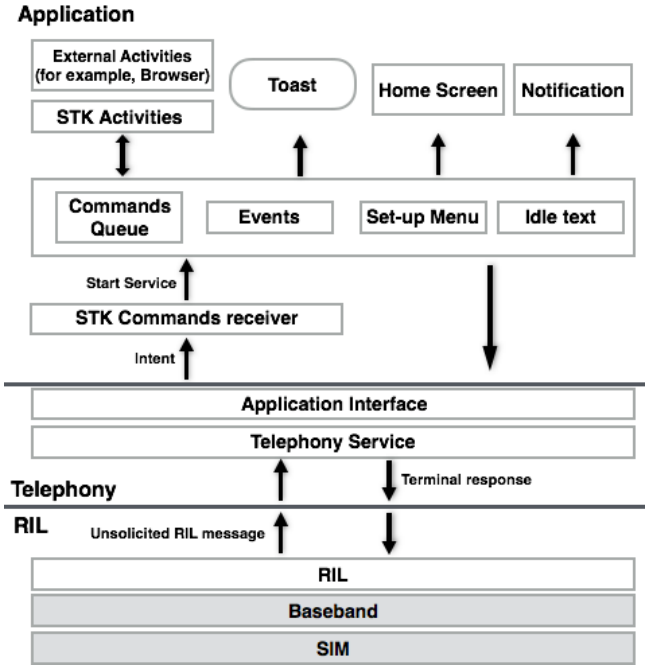

図 2　Android 上の SIM Toolkit 構成 [7]

画面に表示し，利用者の入力を正確に SIM に伝達することに集約される．続く章において，SIM を活用した 3 通りの実現方法を述べる．

3.　SIM を活用した MitMo 対策の構成方法

3.1　SIM Application Toolkit + SIM

SIM Application Toolkit(以下，SAT) は，SIM (Subscriber Identity Module) に導入された SIM 用 Java Applet(以下，SIM Applet) で携帯電話の各種機能を制御するための API 等で構成されるプラットフォームである [6], [14], [15]．SAT は Feature Phone や iPhone を含むほぼ全ての携帯電話で利用できると考えられているため，本稿で述べる 3 方式の中で最も普遍的 (Universal) である．SAT を利用した取引認証は実際にノルウェーの金融機関等で利用されている．

SMS を用いて OTA(Over The Air) で SIM Applet の遠隔導入が可能なため [24]，既に普及している携帯電話に展開できる．SIM Applet はサーバから送信された SMS で起動し，SIM Applet が以下に示した SAT API を通じて携帯電話を制御して，Web ブラウザやダイアログを通じて利用者にサーバのメッセージを表示し，利用者の入力をサーバに送信する．SIM に格納できるプログラムの容量制限が厳しい上，Web ブラウザ機能やメッセージの表示機能は簡素であり，スマートフォンで利用した場合でも必ずしも使いやすいとは言えない欠点がある．

- 端末へのテキストの表示
- 端末へのダイアログの表示
- 端末画面のフレーム分割
- 通話の開始

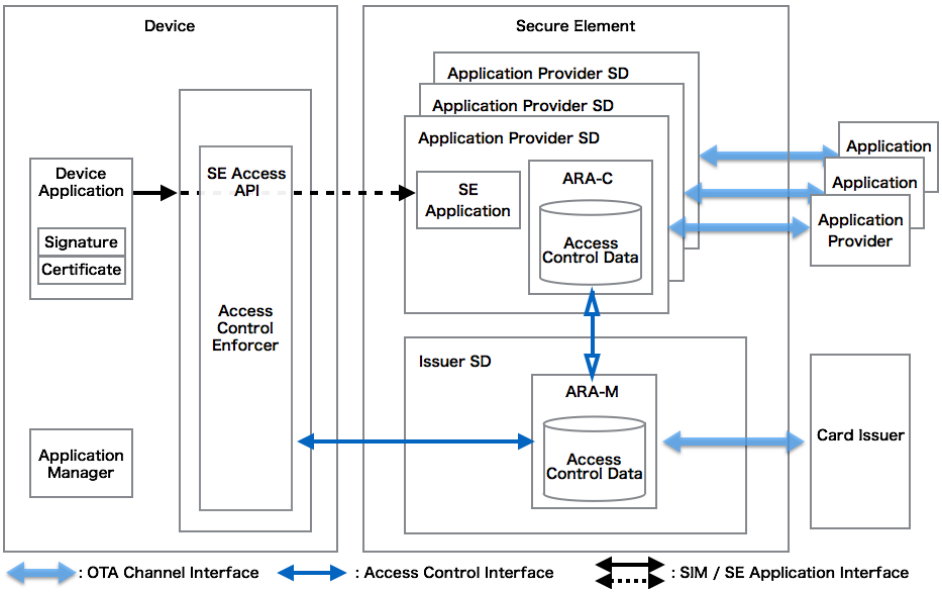

図 3　Global Platform: Secure Element Access Control Architecture[10][16]

- SMS の送受信
- URL を指定したブラウザの起動
- 端末でのアプリケーションの起動
- GPS 情報の取得

Android 端末上での SAT の構成を図 2 に示す．灰色で示した機能以外，SMS 等の一部機能を除く SAT の多くの機能は Android OS の一部として実装されている．SIM と Android OS の間の通信は平文であり，SIM の端子を外部に取り出すことで通信内容を盗聴できることが報告されている [9]．従って，Root 化された `UNLOCKED` 状態の Android OS を改竄すれば SIM との通信を盗聴／改竄することも可能と予想される．従って，仮定 3 を認め，正規 Android OS が改竄去れずに正しく機能することが，SIM Application Toolkit+SIM の構成が安全であるための条件となる．

3.2　NFC UI Application + SIM

Android App から SIM Applet への安全なアクセス制御を実現するために GSMA(Global System for Mobile communications Association) において NFC Handset APIs & Requirements[16] が規定され，Android OS に既に実装されている．NFC Handset APIs & Requirements[16] におけるアクセス制御の目的は，予め指定された Android App 以外からの SIM Applet へのアクセスを禁止することである．

NFC Handset APIs & Requirements[16][26] におけるアクセス制御は図 3 のような構成になっており，Android App のコード署名機構を用いてアクセス制御を実現している．Android App には開発者のコード署名が施されており，Android App のインストール時に Android OS が Android App のハッシュ値を計測してコード署名を検証する．このため正しくないコード署名を持つ Android App は Android OS 上で実行できない仕様になっている．NFC Handset APIs & Requirements[16] におけるアクセス制御は，コード署名の検証に用いる公開鍵証明書のチェインを検証する．すなわち，SIM Applet 領域（図 3 中 Application Provider SD) へのアクセス要求を発行した Android App のコード署名を確認し，コード署名に用いられた公開鍵証明書と同一かより上位の公開鍵証明書が SIM に事前に登録 (図 3 中 ARA-M に登録) されている場合に限って Access Control Enforcer が当該 SIM Applet へのアクセスを許す．この機構が SIM Applet にアクセス可能な Android App のホワイトリストを構成し，コード署名に用いる公開鍵証明書の発行をコントロールすることでホワイトリストを制御できる．

NFC UI Application+SIM における取引認証の実装は，一般の Android App で実現可能な全ての機能を用いて UI を構成できるため 3 方式の中で最も利便性に優れた UI を構成できる．他方，UI 部分が一般ユーザ権限で動作する Android App で実現されるため，Repakage 攻撃に晒されるリスクは避けられないが，SIM Applet と連動することで機密情報の管理を SIM Applet に任せれば正規 Android App は機密情報を管理する必要がなく，上述の SIM Applet へのアクセス制御機構により，正規のコード署名を持つ Android App 以外は SIM Applet にアクセスできなくなる．従って，仮定 3 を認めた場合，アクセス制御機構が安全に動作する限り，Android マルウェアの大部分を占める Repackage 攻撃に対して耐性がある．　その他の攻撃に対する安全性評価を表 3 に示す．コード署名が施された正規 Andorid App は Android OS のコード署名検証機能により，改竄を加えると実行不能になる．コード署名の偽造は計算量的に困難なことを考えると，正規 Android App の

表 3　NFC UI Application+SIM に想定される攻撃と安全性評価

攻撃の種類	安全性評価
Repackage 攻撃	NFC Handset APIs & Requirements[16] における SIM Applet へのアクセス制御機構により，正規のコード署名を持つ Android App 以外は SIM Applet へのアクセスが禁止される．このため Repackage 攻撃は無効化される．また正規 Android App は機密情報の管理を SIM Applet に委託できるので，正規 Android App を解析しても有用な情報は得られない．コード署名の偽造は計算量的に困難．コード署名検証機構の無効化，アクセス制御機構の無効化等が必要．
正規 Android App の表示改竄	仮定 3 により正規 Android OS が動作し，コード署名の検証が正しく行われる条件の下では，正規 Android App の改竄は Repakage 攻撃と等価である．仮定 2 を認めれば，正規 Android App に表示を改竄可能な脆弱性がない限りプログラムコードの機能変更は不可能．
Screen Overlay 攻撃	Android 6.0 以降の OS では `ACTION_MANAGE_OVERLAY_PERMISSION` を取得した Android App でなければ Screen Overlay できない仕様に変更された[*1]．正規 Android OS の機能を用いた Screen Overlay 攻撃に対しては，Android OS の機能で対応が可能と考えられる[*2]．

[*1] Android OS 6.0 以前は Window Manager の優先度を悪用して Screen Overlay 攻撃を実現できる可能性が残されていた [2][8]．App を設計する際に `FLAG_WINDOW_IS_OBSCURED` を確認することで操作イベントの盗取 (Screen Overlay 攻撃) を Android App で検出できる．

プログラムコードを一切改竄せずに，表示内容を改竄する攻撃を考える必要がある．仮定 3 を認めて Android OS 全体の機能が保全される状況の下では，関連する有効な脆弱性がない限り，表示の改竄は困難と考えられる．

Android OS の機能を用いて改竄した表示を重ねて錯誤させる Screen Overlay 攻撃については，正規の機能を用いるので仮定 3 の条件下でも攻撃が成立する．正規 Android App の設計の際に，適切に `FLAG_WINDOW_IS_OBSCURED` を確認して操作イベントの盗取を検出すること，並びに Android OS 6.0 以降であればマルウェアが `ACTION_MANAGE_OVERLAY_PERMISSION` を取得する必要が生じたことから利用者が適切にパーミッションを与えること等の対策を確実に実施する必要がある．この問題はよく認知されており，Android OS の進化に伴って Screen Overlay 攻撃の困難さは増している．

SAT と同様，図 3 に示すように NFC Handset APIs & Requirements[16] におけるアクセス制御機構 (図 3 中 Access Control Enforcer) は Android OS(図 3 中 Device) の一部として実装されている．このため，Root 化された `UNLOCKED` 状態の Android OS を改竄すれば，SIM Applet へのアクセス制御機構を無効化することも可能となる．従って，NFC UI Application+SIM の構成においても，仮定 3 を認め，正規 Android OS の機能が変更されずに正しく実行されることが安全性の条件となる．

3.3　TEE: Trusted Execution Environment + SIM

TEE(Trusted Execution Environment) はスマートフォン端末上の独立な領域であり，従来の OS が動作する REE(Rich Execution Environment) と隔離して動作する．TEE 上で動作するアプリケーション (Trusted Application) は管理するデータの機密性が保たれる．Global Platform 等の規格としても採用されている．

SIM は TEE における Secure Element の一つとして定義されており，現在の規格 [12] では Rich Application (REE 上のアプリケーション, Android App) を経由して Trusted Application と SIM が通信するか，あるいは Trusted Application が TEE 環境の中で直接に SIM にアクセスすると記載されている．前者の場合には，Trusted Application の間でセキュアチャネルを構成して，Android OS の安全性に依存しないように構成しておく必要がある．

さらに，Global Platform[11] では，Trusted User Interface(Trusted UI) が定義されている．Trusted UI は表示制御デバイスも TEE に対応することが求められ，Trusted UI が動作している間は REE からの表示制御デバイスへのアクセスは禁止される．また，Trusted UI を表示するスクリーンは TEE 専用に用意することが求められ，常に入力を受け付け，オーバレイ表示は許されない等の厳しい制限を課している．Trusted UI を装備したスマートフォンが普及すれば，取引認証に関わる MitMO 攻撃に対する耐性は相当高まると期待される．

TEE+SIM の構成においては，SIM と Trusted Application の間にセキュアチャネルを構成し，Trusted UI 機能を備えた端末を用いることにより，Android OS の安全性に依存しない，最も安全な取引認証を構成できると考えられる．

4.　SIM-Sign の構成

4.1　3 方式の比較

3 方式の比較を表 4 に示す．比較のために SIM を用いず，単純に Android App で実現した場合を加えた．単純な方式は Repackage 攻撃に対して脆弱だが，SIM を用いるいずれの方式も後に述べる構成証明 (Remote Attestation) を実装可能なので Repackage 攻撃に対して頑健な方式を構成できる．他方，意図的な Root 化に対しては，TEE+SIM を除く方式は実装の一部を Android OS に依存しているので OS が改竄された際に正しく機能しなくなる．ただし，いずれの場合も SIM が鍵を含む機密情報を管理しているので，鍵が漏洩するリスクは極めて小さい．従って，Medium

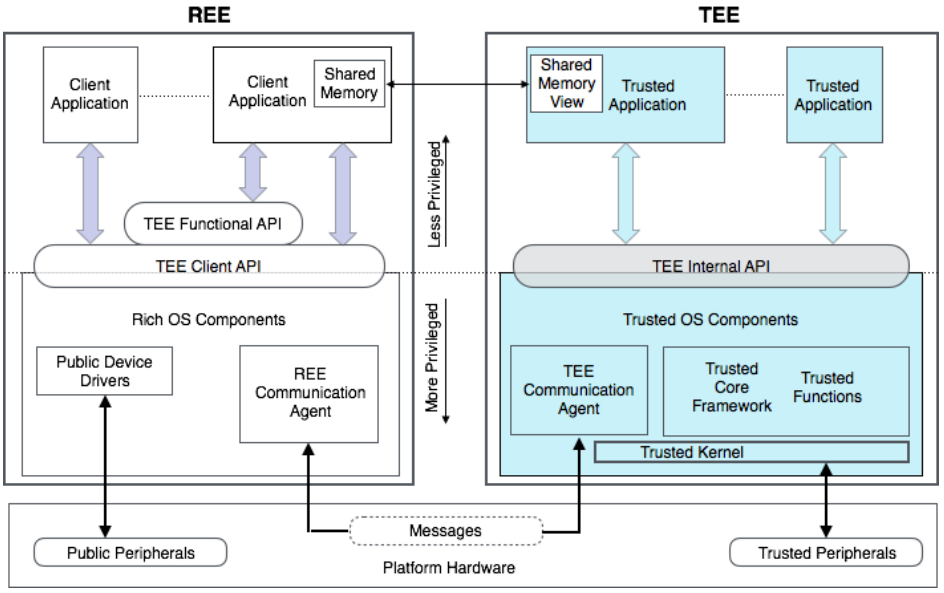

図 4　Trusted Execution Environment アーキテクチャ [7]

とした．

SIM Application Toolkit は Feature Phone を含むほぼ全ての携帯電話で利用できるので，最も広い普遍性 (Universality) を有する．他方，TEE(特に Trusted UI) を実装した Android 端末は僅かである．利用者を広げるために重要な UI の品質 (Quality) は，Android App と NFC UI Application+SIM が Android OS の機能を全て活用できるため優れており，TEE+SIM 及び SIM Application Toolkit は独自の UI 実装が求められるため，Android OS の UI 品質と比較して相対的に貧弱になると考えられる．

以上をまとめると，3 方式は安全性，普遍性，UI の品質の全ての点で一長一短があるが，現時点では NFC UI Application+SIM のバランスが最も良い．従って，次節では，NFC UI Application+SIM のプロトコル構成を中心に考えるが，プロトコル構成の大部分は 3 方式に共通である．また，将来 Trusted UI を実装した TEE 対応 Android 端末が利用可能になれば，順次，TEE+SIM 方式に移行することも考えられる．

4.2　SIM-Sign の構成法

SIM-Sign プロトコルは SP-TSM 機能を持つ認証局，銀行等サービス提供者のサーバ，SIM を有する Android 端末，利用者の 4 者で構成される．SP-TSM と SIM の間はプラットフォームで提供されるセキュアチャネルで結ばれており，サーバは認証局から証明書 S_i(i はサーバの ID) を受けているとする．また，SIM の記憶領域を拡張するために，SIM で管理すべき機密情報の一部を SIM 固有の鍵で暗号化して Android 端末内およびクラウド上に記憶していると仮定する．SIM-Sign の SIM Applet には認証局の証明書と，アクセス制御に使用するための App のコード署名に用いる証明書[*1]が組み込まれている．

4.2.1　SIM Applet の遠隔導入

Android 端末からの SIM Applet の遠隔導入要求を受け，SP-TSM がキャリア経由で Android 端末上の SIM に SIM Applet を遠隔導入する [25]．その際，SP-TSM は SIM とのセキュアチャネルを通じて必要な鍵を SIM に配布する．

(1) 利用者が Google Play から SIM-Sign 対応 Andoird App(以下, App) を導入して起動する．
(2) App が端末情報を認証局 (SP-TSM) に送る．
(3) SP-TSM からキャリアに端末情報が送られ，キャリアから当該 SIM に SIM Applet が遠隔導入される．
(4) SP-TSM から SIM Applet にサーバの証明書リストを送付する．
(5) SIM Applet が複数の鍵ペアを生成し，セキュアチャネルを通じて SP-TSM に送信して証明書リスト $C = \{C_1, \ldots, C_n\}$ を受ける．(または，SP-TSM が証明書と秘密鍵のリストを生成し，セキュアチャネルを通じて SIM Applet に送付する．)
(6) 認証局と SIM が App を経由してチャレンジレスポンスを行い，App と SIM Applet の間のアクセス制御機能の動作と SIM Applet の導入完了を確認する．

4.2.2　Remote Attestation(Repackage 攻撃対策)

Android 端末の Verified Boot の結果を遠隔サーバに対して証明 (Remote Attestation) できれば，遠隔サーバからの Root 化検知に使用できる．同機能の実装は現時点で確認できないので，本稿では Android OS を正規と仮定し，SIM とアクセス制御機構を用いた Remote Attestation を考える．以下にそのプロトコルを示す．

(1) App を起動してサーバに接続する．

[*1] 実際はコード署名に用いる証明書の上位の CA 証明書．

表 4　SIM を用いた MitMo(Man in the Mobile) 攻撃対策の比較

構成法	UI の実装	Security		UI Quality	Universality
		Repackage 攻撃	意図的な Root 化 (UNLOCKED 状態)		
Android App[*1]	Android App	Weak	Weak	Rich	Android Smartphone
SIM Application Toolkit+SIM	SAT	Strong	Medium[*1]	Poor	Smartphone + Featurephone
NFC UI Application +SIM	UI Application (Android App)	Strong	Medium[*2]	Rich	NFC Android Smartphone
TEE + SIM	Trusted UI	Strong	Strong	possiblly Poor[*3]	TEE-equipped Smartphone

[*1]SIM を用いない Android App の実装. 比較のために掲載. [*2]意図的な Root 化により取引認証の安全性が損なわれるが，鍵を含む機密情報が漏洩するリスクは小さい．[*3]Trusted UI の仕様は未確定だが独自の実装を行う必要があるため初期は Poor の可能性が高い.

(2) サーバの証明書 C_i を取得し，SIM Applet に送る．
(3) サーバの証明書に対応する SIM Applet の証明書 S_j を選択する．(対応する証明書がなければ，取得済みの証明書から S_j を新規に割り当て，$i \to j$ を対応とする.)
(4) App を経由してサーバと SIM Applet が相互認証を行う．

サーバが App を経由して TLS 相互認証に成功するのは，App のコード署名が検証され，かつ App が SIM Applet のアクセス制御機構を通過している場合に限られる．このため，Android OS が LOCKED 状態にありアクセス制御機構が正常であれば，サーバが TLS 相互認証で通信可能な相手は正規 App に限られるため Remote Attestation として利用できる．ここで，SIM Applet においてサーバ毎に新規の鍵ペアを選択することで，複数のサーバの結託による SIM の特定を避けている．

4.2.3　アクティベーション

既に利用者がサーバに開設しているアカウントに，SIM を結びつけるために以下のプロトコルを実行する．新規開設は重複が多いので割愛する．

(1) App を用いてサーバにアクセスし，アクティベーションを申請する．
(2) サーバ (銀行等) が利用者の本人確認を行う．
(3) App を経由して SIM とサーバが Remote Attestation を行い，サーバが S_j を取得する．
(4) App の UI を通じて利用者の同意を確認し，結果をサーバに送信する．
(5) サーバが S_j とアカウント k の対応 $j \to k$ を登録する．

サーバ (銀行等) による本人確認は，銀行等のポリシーに従い電話，SMS，PW の郵送，ID/PW でのアカウントへのログイン等で行う．ステップ (4) において，利用者から同意を得る際に，暗証番号や生体認証による本人確認を行うこともある．

4.2.4　取引認証

(1) App を用いてサーバにアクセスし，サーバと SIM Applet の間で Remote Attestation を行う．
(2) サーバが取得した SIM Applet の証明書 S_j がアクティベーション済であれば，登録された $j \to k$ の対応に従いアカウント k へのログインを完了する．
(3) 利用者が App を用いて取引を指示する．この際，App は SIM Applet 経由の TLS 通信路を通じて取引を要求する．
(4) サーバが取引確認を TLS 通信路を用いて SIM Applet に送信する．
(5) SIM Applet は App を通じて取引内容を画面に表示し，利用者の入力を求める．
(6) SIM Applet は利用者の入力を TLS 通信路を用いてサーバに送信する．
(7) 利用者の入力に従いサーバが取引を処理する．

ステップ (5) の取引内容の表示は，NFC UI Application+SIM の場合は Android OS のアクセス制御機構により検証された正規 Android App を通じて行われ，TEE+SIM の場合は，Trusted UI を通じて行われることを想定している．また，同じステップ (5) において，利用者から取引実行の同意を得る際に，暗証番号や生体認証による本人確認を行うこともある．

4.3　考察

本節では標準的な PKI を用いた構成法を述べた．同様の考え方で共通鍵暗号を用いても構成可能である．共通鍵で構成すれば SIM Applet の応答速度を高められる利点がある．また本節では，SIM Applet の遠隔導入の際に，事前に複数の鍵ペアに対する証明書を発行する方式を示した．プライバシー保護のためにサーバ毎に証明書を使い分ける必要があることから，SIM Applet で複数の証明書を管理している．この際，SIM での鍵生成は時間を要するので，大量に鍵ペアを生成する場合には SP-TSM で鍵ペアを生成して SIM に送付する代替案を検討する必要がある．

前述のように Android OS では Repackage 攻撃によるマルウェアの導入率が高い．銀行等の Android App も Repackage 攻撃により容易に取引を改竄されるリスクが高まっている．一般の応用でも 4.2.2 節で述べた Remote Attestation を利用すれば，本人を特定せずに正規 Android

App の改竄 (Repackaging) を遠隔サーバから SIM を利用して検出できる．

5. まとめ

本稿ではスマートフォンと SIM を利用した取引認証の仕組みについて，SIM Application Toolkit, NFC UI Application および TEE を用いた構成法について検討し，現時点で最も実用性が高い NFC UI Application を用いた構成をベースにしたプロトコルの構成法を提案した．Android OS にも SE Linux が本格的に取り入れられ，年々，Android 端末の安全性も高まっている．しかし，誰でも自由に Android App を開発できる方針は変わっておらず，Android OS を単体で利用する限り，Repackage 攻撃は防止できないように思われる．本稿で示した SIM と Android 端末のセキュリティ機構を用いた MitMo 対策技術は，Repackage 攻撃や Banking Trojan の対策に大きな効果が期待できると考えている．

参考文献

[1] Android Open Source Project: Verified Boot, Google Inc. (online), available from ⟨https://source.android.com/security/verifiedboot/⟩ (accessed 2016-08-01).

[2] Bianchi, A., Corbetta, J., Invernizzi, L., Fratantonio, Y., Kruegel, C. and Vigna, G.: What the App is That? Deception and Countermeasures in the Android User Interface, *SP '15: Proceedings of the 2015 IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 931–948 (2015).

[3] Brandt, A.: Android Towelroot Exploit Dogspectus Ransomware, BLUE COAT LABS (online), available from ⟨https://www.bluecoat.com/security-blog/2016-04-25/android-exploit-delivers-dogspectus-ransomware⟩ (accessed 2016-08-01).

[4] Computer Security Center (U.S.): Trusted Computer System Evaluation Criteria (1985).

[5] DHS and FBI: Threats to Mobile Devices Using the Android Operating System, (online), available from ⟨https://info.publicintelligence.net/DHS-FBI-AndroidThreats.pdf⟩ (accessed 2016-08-01).

[6] ETSI: *ETSI TS 102 223 V9.1.0 (2010-04) Smart Cards; Card Application Toolkit (CAT) (Release 9)*, ETSI (2010).

[7] ETSI: Smart Cards; Card Application Toolkit (CAT) (Release 9) (2010).

[8] Fernandes, E., Chen, Q. A., Paupore, J. and Essl, G.: Android UI Deception Revisited: Attacks and Defenses, *International Conference on Financial Cryptography and Data Security*, Barbados (2016).

[9] Gielen, S. and Poll, E.: SIM Toolkit in Practice, *Bachelor Thesis*, Norwegian University of science and technology (2012).

[10] GlobalPlatform: *Secure Element Access Control* (2012).

[11] GlobalPlatform: *Trusted User Interface API* (2013).

[12] GlobalPlatform: *TEE Secure Element API v1.1* (2015).

[13] GlobalPlatform: *Remote Application Management over HTTP v1.1.3* (2016).

[14] GSM: *GSM 11.14 version 5.4.0 Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment*, ETSI (1997).

[15] GSM: *3GPP TS 31.111 version 12.10.0 Release 12 Universal Subscriber Identity Module (USIM) Application Toolkit (USAT)*, ETSI (2016).

[16] GSM Association: *GSMA NFC Handset APIs Requirement Specification Version 2 0* (2011).

[17] Hu, H., Shinde, S., Adrian, S., Chua, Z. L., Saxena, P. and Liang, Z.: Data-Oriented Programming: On the Expressiveness of Non-Control Data Attacks, *2016 IEEE Symposium on Security and Privacy*, pp. 969–986 (2016).

[18] Prandini, M. and Ramilli, M.: Return-Oriented Programming, *IEEE Security and Privacy*, Vol. 10, No. 6, pp. 84–87 (2012).

[19] Siebert, T.: Advanced Techniques in Modern Banking Trojans, *Botconf* (2013).

[20] Weigold, T., Kramp, T., Hermann, R., Höring, F., Buhler, P. and Baentsch, M.: The Zurich Trusted Information Channel — An Efficient Defence Against Man-in-the-Middle and Malicious Software Attacks, *Trust '08: Trusted Computing and Trust in Information Technologies*, Berlin, Heidelberg, IBM Zurich Research Laboratory, Springer-Verlag, pp. 75–91 (2008).

[21] Zhang, H., She, D. and Qian, Z.: Android Root and its Providers, *the 22nd ACM SIGSAC Conference*, New York, New York, USA, ACM Press, pp. 1093–1104 (2015).

[22] Zhou, Y. and Jiang, X.: Dissecting Android Malware: Characterization and Evolution, *2012 IEEE Symposium on Security and Privacy*, IEEE, pp. 95–109 (2012).

[23] 高木浩光，渡辺　創：Man-in-the-Browser の脅威と根本的な解決策，AIST（オンライン），入手先 ⟨https://www.risec.aist.go.jp/files/events/2014/0313-ja/risec-sympo2014-takagi.pdf⟩（参照 2016-08-01）．

[24] 南本早苗，保　谷：UIM バージョン 3 の開発，NTT DoCoMo テクニカル・ジャーナル (2007).

[25] 秋山友宏，丹野哲宏，笹川哲広，山口久美子：進化するおサイフケータイ–ドコモ UIM カードへの NFC(Type A/B) 対応サービス発行のしくみ–，Vol. 21, No. 1, NTT DoCoMo テクニカル・ジャーナル (2013).

[26] 菅野利博，塩野入匡宏，木川真孝：進化するおサイフケータイサービスを実現する技術–NFC 対応移動機およびドコモ UIM カードの開発–，Vol. 21, No. 1, NTT DoCoMo テクニカル・ジャーナル (2013).