WhatsAppに対する不正デバイス登録攻撃

白木 章伍[1,a)]　木村 隼人[2,3]　伊藤 竜馬[2,3]　峯松 一彦[4]　五十部 孝典[2,3]

概要：WhatsAppは月間30億人が利用する世界最大のインスタントメッセンジャーであり，複数端末でのログインを許可している．Albrechtら (Eurocrypt 2025) は複数端末でログイン後のメッセージ暗号化処理に対する形式化と安全性評価を実施したが, 端末を追加する際の認証プロトコルは評価されていない．本研究ではMeta社独自の端末認証方式であるLink using an 8-character Codeに対する安全性評価を実施する．本方式では，2台目の端末に表示された8桁のコードを認証済みの1台目の端末に入力することで認証を行う．この認証は素朴なコードの照会ではなく，End-to-end暗号化との互換性を維持するために暗号学的要素を用いる．我々は，認証コードの実効エントロピー，楕円曲線暗号の代数的性質，鍵導出関数や共通鍵暗号方式への入力などを考慮した解析を行う．その結果, 認証強度に重大な影響を与える5つの脆弱性を発見し, 悪意あるサーバーが利用者の協力なしに不正デバイスを登録可能であることを示す．また，効果的な対策方法について議論する．これらの成果は2025年4月にMeta社へ報告され, 我々の提案に基づき仕様と実装が修正された．本研究は，巨大なユーザー数を擁するシステムに対する実用的な解析であり，プロトコルの形式化に加えて具体的な構成に基づく手動解析を実施することの重要性を示すものである．

キーワード：WhatsApp, E2EE, マルチデバイス, デバイス登録, elliptic-curve distinguisher

Malicious Device Registration Attacks on WhatsApp

Shogo Shiraki[1,a)]　Hayato Kimura[2,3]　Ryoma Ito[2,3]　Kazuhiko Minematsu[4]　Takanori Isobe[2,3]

Abstract: WhatsApp is the world's largest messenger with billions of monthly active users and supports multi-device login. While prior research has analyzed the message encryption after multi-device sign-in, the authentication protocol used when adding new devices has not been sufficiently evaluated. We conduct a security assessment of a proprietary device authentication method used in this process and identify several issues that could weaken its security under certain conditions. Our analysis, conducted independently from existing studies, highlights the importance of combining formal analysis with detailed evaluation of actual implementations. The vendor has since updated the protocol based on security considerations.

Keywords: WhatsApp, E2EE, Multi-device, Link device, elliptic-curve distinguisher

1. はじめに

WhatsAppは米国で1億人，全世界で30億人以上の月間アクティブユーザを擁する世界最大のインスタントメッセンジャーである[*1][*2]．WhatsAppの暗号化に関するホワイトペーパーによると，エンドツーエンド暗号方式にSignalプロトコルを採用している [1]．ただし, このプロトコルが適用されるのは1対1通信に限られ, それ以外の機能についてはMeta社が独自に設計・拡張したものである．Signalプロトコルは広く安全性評価が行われている一方で，Meta

[1] 兵庫県立大学 / University of Hyogo
[2] 国立研究開発法人情報通信研究機構 / NICT
[3] 大阪大学 / The University of Osaka
[4] 日本電気株式会社 / NEC Corporation
[a)] 4w3tag185mpja@gmail.com

[*1] https://blog.whatsapp.com/100-million-using-whatsapp-across-the-united-states?lang=en
[*2] https://investor.atmeta.com/investor-events/event-details/2025/Q1-2025-Earnings-Call/default.aspx

社が独自に設計した機能やプロトコルについては十分な第三者評価が行われていない．Albrecht らはマルチデバイスでログインした後のメッセージ暗号化処理に対する形式化と安全性評価を実施した [2]．しかし，その安全性の前提となるデバイス追加時の認証プロトコル「Companion Device Registration（CDR）」の評価は未実施である．

CDR ではユーザが所持する 2 台目のデバイスを認証し，マルチデバイス間でメッセージを同期するためのプロトコルである．認証されたデバイス (PC，タブレット端末, スマートフォン等) は過去・現在のメッセージの取得，および新規メッセージの送信が許可される．最大で 4 台のデバイスを登録可能であり，14 日間毎に再認証が要求される．

CDR プロトコルは, 初期設定, オフラインフェーズ，そしてオンラインフェーズの 3 段階で構成される．

(1) 初期設定：Curve25519 に基づく公開鍵および秘密鍵ペアの生成を行う．
(2) オフラインフェーズ：ユーザが 2 台目のデバイスに表示された 8 桁の認証コードを認証済みの 1 台目のデバイスに入力する．認証コードから共通鍵を派生する．
(3) オンラインフェーズ：暗号化された公開鍵を伴う，デバイス間の ECDH 鍵共有によってデバイスを認証する．このとき，公開鍵は認証コードから派生された共通鍵によって暗号化される．公開鍵を交換した後にデバイス間の ECDH が実行される．ECDH を通した認証に失敗した場合は，オンラインフェーズの `Primary Hello` (2.3.2 章) からやり直しを行う．

なお，今後は認証済みの 1 台目のデバイスは Primary デバイス（PD），2 台目のデバイスは Companion デバイス（CD）と表記する．

1.1 本研究の成果

本論文では，Meta 社が独自に設計した CDR に対して初めて詳細な安全性解析を行い，悪意のあるサーバ攻撃者が不正デバイスを被害者の利用デバイスとして登録可能なことを示す．本攻撃では, オフラインとオンラインでの両フェーズの脆弱性を利用する．オフラインフェーズでは，認証コードのエントロピー不足とオフライン攻撃耐性の欠如を利用することで, 確率的に不正デバイスを登録する手法を示す（基本的攻撃）．また，楕円曲線暗号の代数的性質と組み合わせることで，オフラインで効率的に認証コードの候補を削減する方法を示す（判定オラクル利用攻撃）．さらに，オンラインフェーズでの再試行の仕様を利用することで, 再試行回数に応じて攻撃成功確率を向上させる方法を示す（再試行利用攻撃）．これらの攻撃により，WhatsApp のデバイス登録機能が理想的な安全性要件を満たしていないことを明らかにする．

本研究の成果を表 1 にまとめる．また，攻撃に用いる脆弱性とそれらを利用した攻撃の詳細を以下に示す．

認証エントロピーの不足　認証コードの 8 桁文字列は 56 ビットのセキュリティを提供するように見えるが，実際には ASCII ではなく Base32 で構成されているため，保証されるのは 40 ビットに留まる．

オフライン攻撃耐性の欠如　認証情報としてデバイスの公開鍵を暗号化して送付するが，暗号化に必要な秘密情報は認証コードのみである．そのため，攻撃者はデバイスから送付された暗号文から, 2^{40} の認証コード候補に対応する公開鍵候補をオフラインで列挙可能である．オンラインフェーズでは, 攻撃者の公開鍵を推測した認証コードで暗号化し，偽造した認証情報とすることで, 正規の認証を 2^{-40} の確率で突破できる．

Identity Key 真正性チェックの欠如　デバイス間で交換される Identity Key に対する真正性チェックが存在しない．このため，攻撃者によって Identity Key がすり替えられても検出できない．

公開鍵候補の判定オラクル　楕円曲線暗号の代数的性質を用いることで，事前に認証コードと公開鍵候補の対から誤ったものを除外する，公開鍵候補の判定オラクルを構成可能であることを示す．このオラクルを利用することで，一度のオフラインフェーズで候補を半数に絞ることが可能である．一度の認証の試行で 2 回のオフラインフェーズが発生するそのため，判定オラクルを使用した攻撃成功確率は 2^{-38} である．これは判定オラクル無しの基本的攻撃と比較して．4 倍の成功確率である．このオラクルはオフラインで実行される．

再試行による攻撃成功確率の向上　オンラインフェーズで認証に失敗すると `Primary Hello` から再試行となるため，攻撃者はその都度判定オラクルを利用して公開鍵候補の探索空間を半減できる．仕様上は 2 回の再試行が許容されているため，オンラインフェーズにおける攻撃成功確率は基本的攻撃の 16 倍にあたる 2^{-36} まで向上する．加えて，攻撃者は悪意のあるサーバであるため，やり直し回数を制御可能と仮定すると，たった 38 回の再試行で高確率に攻撃が成功する．認証コードの空間は 2^{40} であるため, 理想的には任意回数のやり直しを許容した場合でも 2^{40} 回程度の試行が必要となるが, 遥かに少ない試行回数で攻撃が可能である．

さらに，我々は解析過程において仕様の矛盾点や不明瞭な箇所を 11 箇所特定し，脆弱性と併せて報告した．発見した脆弱性や攻撃は，これらの記載上の不備に依存するものではないが，サービスプロバイダーによって公開された仕様と実装上の仕様とのギャップを解消することは，今後の安全性評価研究の発展に寄与する重要な課題である．

1.2 倫理的配慮と情報開示

2025 年 4 月，我々は Meta 社に対して脆弱性とその修正方法を報告した．その結果，Meta 社は一部を除き我々の

表 1 WhatsApp に対する不正デバイス登録攻撃 (MS: 悪意のあるサーバ, CD: Companion デバイス, PD: Primary デバイス)

攻撃名	攻撃者	判定オラクル実行回数 (脆弱性 4)	計算量 (絞り込み処理)	成功確率 (不正 CD)	成功確率 (不正 PD)	参照
基本的攻撃	MS	-	N/A	2^{-40}	2^{-40}	4 章
判定オラクル利用攻撃	MS	2	$\sum_{i=0}^{1} 2^{40-i}$	2^{-39}	2^{-38}	5 章
再試行利用攻撃 (再試行 2 回)	MS	4	$\sum_{i=0}^{3} 2^{40-i}$	2^{-39}	2^{-36}	6 章
〃 (N 回再試行可能)	MS	$N+2$	$\sum_{i=0}^{N+1} 2^{40-i}$	2^{-39}	2^{-38+N}	6 章

報告内容を認め，我々の提案に基づいた修正を行うことを約束した．具体的には，Meta 社は基本的攻撃（4 章），判定オラクル利用攻撃 (5 章) の実行可能性を認め，エントロピー増加によって対処することを約束した．さらに，再試行利用攻撃（6 章）のうち報告時の実装である 2 回までの再試行を使用した攻撃を認めた．対策として，認証失敗時にオフラインフェーズからの再試行を許容せず，初期設定から再試行する仕様へ修正することを約束した．

一方で，悪意あるサーバが再試行回数を制御可能であるという仮定について，Meta 社はこの前提を採用しない立場を示した．しかし，我々は，E2EE 暗号化においてはサーバ攻撃者がプロトコル遂行に必要なパラメータを制御可能であることは現実的であるため，この仮定を考慮した上でも，安全性を保証すべきであると考える．

2. Companion Device Registration

本章では，WhatsApp が提供する Companion Device Registration (CDR) の仕様について述べる．CDR には，QR コードを用いた「Link using a QR-Code」と，認証コードを用いた「Link using an 8-character Code」の 2 方式が存在する．本稿では，脆弱性が確認された後者の「Link using an 8-character Code」に焦点を当て，プロトコルフローを説明する．本プロトコルは初期設定，1 台目のデバイスおよび 2 台目のデバイスによるオフラインフェーズ，認証情報を交換するオンラインフェーズから構成される．

2.1 初期設定

初期設定では，ユーザが 2 台目の新たなデバイス（Companion device, CD）を 1 台目の既存の認証済みデバイス（Primary device, PD）へ登録するための準備を行う．

まず，WhatsApp をインストール後，すべてのデバイスは Identity Key Pair を次のように生成する．

$$\{idk_{\mathrm{priv}}^{\mathrm{device}}, idk_{\mathrm{pub}}^{\mathrm{device}}\} \leftarrow \mathsf{X25519.KeyGen}(). \quad (1)$$

CD は，デバイスの連携手順を開始する際，40 ビットの認証コード SECLCP を生成する．この認証コードは 8 文字の Base32 形式で CD の画面に表示される．また，CD は X25519 鍵ペア $\{k_{\mathrm{priv}}^{\mathrm{CD}}, k_{\mathrm{pub}}^{\mathrm{CD}}\}$ を生成する．このとき，$k_{\mathrm{pub}}^{\mathrm{CD}}$ はサーバを含む第三者に開示されないことに注意する．

$$\{k_{\mathrm{priv}}^{\mathrm{CD}}, k_{\mathrm{pub}}^{\mathrm{CD}}\} \leftarrow \mathsf{X25519.KeyGen}(). \quad (2)$$

2.2 オフラインフェーズ

オフラインフェーズでは，ユーザは CD に表示された認証コード SECLCP を物理的に PD へ入力する．この操作はオンライン通信を伴わないオフラインでの情報伝達であることに注意する．以降の処理では，両デバイスが同一の認証コードを共有していることを前提とする．

2.3 オンラインフェーズ

オンラインフェーズでは，Companion Hello, Primary Hello, および Companion Finish で構成されるハンドシェイクによってデバイスの認証を行う．

Companion Hello と Primary Hello では, それぞれ CD と PD の X25519 公開鍵を共通鍵暗号で暗号化した上で共有する．この暗号化では AES のカウンターモード (AES-CTR) が使用され，その秘密鍵はオフラインフェーズで共有された認証コード SECLCP から導出される．

Companion Finish では，両デバイスの Identity Key Pair と X25519 鍵ペアから 2 つの共有鍵を生成し，これらの共有鍵から鍵導出関数で秘密の共有値 L_{cpn} を導出する．L_{cpn} は AES-GCM で暗号化した上で共有される．L_{cpn} の共有をもって，デバイス追加時の認証プロセスは完了となる．

2.3.1 Companion Hello

Companion Hello では CD の X25519 公開鍵 $k_{\mathrm{pub}}^{\mathrm{CD}}$ を暗号化して PD に配送する．このために, まず, CD は SECLCP と $N_{\mathrm{KDF}}^{\mathrm{CD}}$ を PBKDF2-HMAC-SHA-256（PBKDF2）に入力し，AES 秘密鍵 $k_{\mathrm{CTR}}^{\mathrm{CD}}$ を導出する．その後，CD 自身の X25519 公開鍵 $k_{\mathrm{pub}}^{\mathrm{CD}}$ を AES-CTR で暗号化する．具体的には，256 ビットの $N_{\mathrm{KDF}}^{\mathrm{CD}}$ と 128 ビットの $IV_{\mathrm{Hello}}^{\mathrm{CD}}$ をそれぞれランダムに生成し，以下の処理を行う．

$$k_{\mathrm{CTR}}^{\mathrm{CD}} \leftarrow \mathsf{PBKDF2}(\mathtt{SECLCP}, N_{\mathrm{KDF}}^{\mathrm{CD}}, 2^{17}, 32), \quad (3)$$

$$C_{\mathrm{pub}}^{\mathrm{CD}} \leftarrow \mathsf{AES\text{-}CTR.Enc}(k_{\mathrm{CTR}}^{\mathrm{CD}}, IV_{\mathrm{Hello}}^{\mathrm{CD}}, k_{\mathrm{pub}}^{\mathrm{CD}}). \quad (4)$$

$\mathsf{PBKDF2}(x, s, i, l)$ において，x は入力（keying material），s は salt, i は内部の反復回数，l は出力長を表す．また，$\mathsf{AES\text{-}CTR.Enc}(k, iv, p)$ は AES-CTR 暗号化関数であり，k が AES 秘密鍵, iv が初期化ベクトル, p が平文（ペイロード）を表す．CD は $\{C_{\mathrm{pub}}^{\mathrm{CD}}, N_{\mathrm{KDF}}^{\mathrm{CD}}, IV_{\mathrm{Hello}}^{\mathrm{CD}}\}$ を Companion Hello として PD へ送信する．PD は式 (3) と同様にして，受信した Companion Hello から AES 秘密鍵 $k_{\mathrm{CTR}}^{\mathrm{CD}}$ を導出し, CD の X25519 公開鍵を復号する：

$$k_{\text{pub}}^{\text{CD}} \leftarrow \text{AES-CTR.Dec}(k_{\text{CTR}}^{\text{CD}}, IV_{\text{Hello}}^{\text{CD}}, C_{\text{pub}}^{\text{CD}}). \quad (5)$$

AES-CTR.Dec(k, iv, c) は AES-CTR 復号関数であり，c が復号対象となる．

2.3.2 Primary Hello

`Primary Hello` では PD の X25519 公開鍵 $k_{\text{pub}}^{\text{PD}}$ と Identity Key の公開鍵 $idk_{\text{pub}}^{\text{PD}}$ を CD に配送する．このとき $k_{\text{pub}}^{\text{PD}}$ のみ暗号化される．まず，PD は自身の X25519 鍵ペア $\{k_{\text{priv}}^{\text{PD}}, k_{\text{pub}}^{\text{PD}}\}$ を生成する．

$$\{k_{\text{priv}}^{\text{PD}}, k_{\text{pub}}^{\text{PD}}\} \leftarrow \text{X25519.KeyGen}(). \quad (6)$$

次に，式 (3) と同様に，オフラインフェーズで入力された `SECLCP` を用いて，`Primary Hello` 用の AES 秘密鍵 $k_{\text{CTR}}^{\text{PD}}$ を導出する．この際，PBKDF2 に入力される salt は $N_{\text{KDF}}^{\text{CD}}$ ではなく，PD が新たに生成する乱数 $N_{\text{KDF}}^{\text{PD}}$ である．その後，式 (4) と同様に，PD の X25519 公開鍵を AES-CTR で暗号化する．

$$C_{\text{pub}}^{\text{PD}} \leftarrow \text{AES-CTR.Enc}(k_{\text{CTR}}^{\text{PD}}, IV_{\text{Hello}}^{\text{PD}}, k_{\text{pub}}^{\text{PD}}). \quad (7)$$

PD は $\{C_{\text{pub}}^{\text{PD}}, N_{\text{KDF}}^{\text{PD}}, IV_{\text{Hello}}^{\text{PD}}, idk_{\text{pub}}^{\text{PD}}\}$ を `Primary Hello` として CD へ送信する．

式 (5) と同様，CD は受信した `Primary Hello` から秘密鍵 $k_{\text{CTR}}^{\text{PD}}$ を導出し，PD の X25519 公開鍵 $k_{\text{pub}}^{\text{PD}}$ を復号する．

$$k_{\text{pub}}^{\text{PD}} \leftarrow \text{AES-CTR.Dec}(k_{\text{CTR}}^{\text{PD}}, IV_{\text{Hello}}^{\text{PD}}, C_{\text{pub}}^{\text{PD}}). \quad (8)$$

2.3.3 Companion Finish

`Companion Finish` では，最終的なデバイス認証用の秘密の共有値である L_{cpn} を両デバイス間で共有する．このために，まず，CD は PD の X25519 公開鍵 $k_{\text{pub}}^{\text{PD}}$ と自身の秘密鍵 $k_{\text{priv}}^{\text{CD}}$ から共有鍵 k_{shr} を生成する．具体的には，128 ビットの $Salt_{\text{KDF}}^{\text{CD}}$ をランダムに生成し，HKDF-SHA256（HKDF）を使用して以下の処理を行う．

$$k_{\text{ECDH}} \leftarrow \text{X25519.ECDH}(k_{\text{priv}}^{\text{CD}}, k_{\text{pub}}^{\text{PD}}), \quad (9)$$

$$k_{\text{shr}} \leftarrow \text{HKDF}(k_{\text{ECDH}}, \text{Info}, Salt_{\text{KDF}}^{\text{CD}}, 32). \quad (10)$$

HKDF(x, y, s, l) において，x は keying material，y はコンテキスト情報，s は Salt，l は出力バイト長を表す．なお，Info は “ink_code_pairing_key_bundle_encryption_key” である．

次に，各デバイスの Identity Key を使用し，2 つ目の共有鍵 idk_{shr} を生成する．

$$idk_{\text{shr}} \leftarrow \text{X25519.ECDH}(idk_{\text{priv}}^{\text{CD}}, idk_{\text{pub}}^{\text{PD}}). \quad (11)$$

デバイス認証のために共有される L_{cpn} は 2 つの共有鍵 k_{shr} と idk_{shr}，および 256 ビット乱数 $\text{Secret}_{\text{Root}}$ を連結した値から導出される．

$$k_{\text{mat}} \leftarrow k_{\text{shr}} || idk_{\text{shr}} || \text{Secret}_{\text{Root}}, \quad (12)$$

$$L_{\text{cpn}} \leftarrow \text{HKDF}(k_{\text{mat}}, \text{“}adv\_secret\text{”}, null, 32). \quad (13)$$

その後，CD から PD へ L_{cpn} を共有するために，CD 上で生成された $\text{Secret}_{\text{Root}}$ を，共有鍵 k_{shr} を使用して AES-GCM で暗号化する．具体的には，128 ビットの $IV_{\text{Fin}}^{\text{CD}}$ をランダムに生成し，以下の処理を行う．

$$k_{\text{bdl}} \leftarrow idk_{\text{pub}}^{\text{CD}} || idk_{\text{pub}}^{\text{PD}} || Secret_{\text{Root}}, \quad (14)$$

$$C_{\text{bdl}}^{\text{CD}} \leftarrow \text{AES-GCM.Enc}(k_{\text{shr}}, IV_{\text{Fin}}^{\text{CD}}, k_{\text{bdl}}). \quad (15)$$

AES-GCM.Enc(k, iv, p) は AES-GCM 暗号化関数であり，k が AES 秘密鍵，iv が初期化ベクトル，p が平文を表す．なお，暗号化仕様書にこのときの AD (Associated Data) は明記されていない [1]．そのため，本稿では empty として扱うが，提案手法は AD に依拠しないことに注意する．

CD は $\{C_{\text{bdl}}^{\text{CD}}, Salt_{\text{KDF}}^{\text{CD}}, IV_{\text{Fin}}^{\text{CD}}\}$ を `Companion Finish` として PD に送信する．PD は `Companion Finish` を受信し，式 (10) と式 (11) のとおり，2 つの共有鍵 $k_{\text{shr}}, idk_{\text{shr}}$ を計算する．その後，AES-GCM.Dec で 256 ビット乱数 $\text{Secret}_{\text{Root}}$ を復号した後，式 (13) のとおり L_{cpn} を計算する．L_{cpn} の共有をもって，デバイス追加時の認証プロセスは完了する．デバイス追加後は L_{cpn} を用いたメッセージ同期時の認証が実施されるが，紙面の都合により省略する．

2.4 再試行処理

Companion Device Registration 機能は認証処理の再試行を最大で 2 回まで許容している．CD が `Companion Finish` (2.3.3 章) の受信後，AES-GCM.Dec での認証タグの検証に失敗すると，PD-CD 間の認証は失敗したとみなされる．このような認証失敗は CD が PD に提示した認証コード (`SECLCP`) と PD が入力した認証コードが打ち間違えによって一致しないとき発生する．認証失敗時には PD は追加で最大 2 回まで同一の認証コード (`SECLCP`) を再入力する機会を与えられる．また，`SECLCP` の打ち直し後は `Primary Hello` (2.3.2 章) の生成から処理が再開される．

3. 攻撃者モデルと安全性要件

WhatsApp の Companion Device Registration（CDR）仕様に対する攻撃者モデルと安全性要件について定義する．

3.1 攻撃者モデル

WhatsApp の暗号化仕様書は攻撃者モデルを明示的に定義していないため [1]，本章では Unger らによって定義された悪意のあるサービスプロバイダーが運営する中継サーバ (悪意のあるサーバ) を定義する [3]．エンドツーエンド暗号化（E2EE）において，ユーザ間の通信は暗号化された状態で中継サーバを通過する．この際，正当な中継サーバはメッセージの暗号化・復号に関与しないが，敵対的なサービスプロバイダーが管理する中継サーバはプロトコルの逸脱を試みる可能性がある．そのため，Unger らはサーバは潜在的に「信頼できない存在」と見なすべきであると

している．本研究のターゲットはマルチデバイス環境においてE2EEとの互換性を維持するためのプロトコルであることから，攻撃者モデルとしても妥当である．

悪意のあるサーバ (Malicious Server, MS)：MSは，利用者にとっては正規のWhatsAppサーバと同様に動作する．そして，悪意を持ってプロトコルから逸脱することで，ユーザから受信した暗号文の盗聴，改ざんを試みる．具体的には，次の能力を有し，利用者の協力なしにデバイス登録の認証手順を迂回し，攻撃者が用意したデバイスを正規デバイスとして登録することを試みる．

- Primaryデバイス（PD）とCompanionデバイス（CD）間で送受信される全てのプロトコルメッセージの取得・改ざん・遅延・再送を試みる能力
- プロトコル仕様に準拠しない形式のメッセージ生成・送信
- 登録処理中に送信される暗号化公開鍵や認証コードを保持・解析するためのオフライン計算の実行

3.2 安全性要件

攻撃者はPDとCD間の登録処理における認証の突破を目指す．この際，CDR仕様が満たすべき安全性要件を次の通りに定義する．

認証性：登録処理に参加するデバイスが，正当な利用者の所有する正規デバイスであることを相互に確認できること．すなわち，PDは正当なCDと，CDは正当なPDとリンクされることを保証する必要がある．MSが，攻撃者のデバイスを正規デバイスとして登録しようとした場合にそれを検出し登録処理を中止しなければならない．

4. 基本的攻撃

本章では，オフラインフェーズおよびオンラインフェーズ の脆弱性に基づく基本的攻撃について述べる．これはPDおよびCDの両デバイスに対し，不正なデバイスを正規のデバイスとして登録させる攻撃である．

4.1 オフラインフェーズにおける脆弱性

脆弱性 1. 認証エントロピーの欠如.

認証コードは8桁のBase32文字列で構成される．8桁文字列は56ビットのセキュリティを提供するように見えるが，実際にはASCIIではなくBase32で構成されているため，保証されるのは40ビットに留まる．

脆弱性 2. オフライン攻撃耐性の欠如.

CDRプロトコルにおいて，認証情報は各デバイスのX25519公開鍵を暗号化して送付するが，この暗号化に必要な秘密情報は認証コードのみである．脆弱性1より，認証コードは2^{40}の候補に限定されるため，認証コードに対応するX25519公開鍵の候補を確率的に復元することが可能となる，さらに，暗号化されたX25519公開鍵を偽造することで，正規の認証を迂回することが可能となる．

4.2 オンラインフェーズにおける脆弱性

脆弱性 3. Identity Key の真正性チェック欠如.

Identity Keyの真正性を確認する手段がない．たとえば，攻撃者によって`Primary Hello`中のIdentity Keyがすり替えられたとしても，CDは不正なPDのIdentity Keyを検出することができない．

4.3 攻撃手順

悪意のあるサーバ攻撃者（MS）により，不正なCDを被害者のPDへ登録する手順について述べる．

不正 CD 登録攻撃：以下の手順により，MSが不正なCDを正当なPDに接続できることを示す．

(1) MSは2^{40}の認証コード候補の中から1つを攻撃用認証コードとして選択する．
(2) MSは`Malicious Companion Hello`を生成する．これは攻撃者のX25519公開鍵と攻撃用認証コードから生成される．
(3) MSは`Malicious Companion Hello`をPDへ送信する．
(4) PDは`Primary Hello`以降の処理を継続する．このときMSはPDが受信するCDのIdentity Key $idk_{\mathrm{pub}}^{\mathrm{CD}}$を攻撃者の$idk_{\mathrm{pub}}^{\mathrm{Adv}}$に置き換える．
(5) MSが選択した攻撃用認証コードが，実際に正当なCDに表示されていた認証コードと一致していた場合は`Companion Finish`が完了する．攻撃者が選択した認証用コードが誤っていた場合は`Companion Finish`におけるPD側のAES-GCM復号処理は失敗する．成功および失敗はサーバに通知される．
(6) 認証コードの推測が成功した場合，最終的にPDと不正なCDは認証用の共有値L_{cpn}を共有する．

なお, MSは`Companion Hello`の代わりに`Primary Hello`に対して同様の攻撃を実施することで，不正PD登録攻撃を実行できる．詳細は紙面の都合で省略する．

4.4 攻撃評価

オンラインフェーズでは，攻撃者は40ビットの認証コードから一つを選択して，攻撃を実施するが攻撃チャンスは一度のみであるため，攻撃成功確率は2^{-40}である．

5. 判定オラクル利用攻撃

基本的攻撃（4章）では，被害者に追加の操作は要求しないが，攻撃成功確率が低いという欠点がある．そこで，本章では，オンラインフェーズでの攻撃成功確率を向上させるための手法を示す．具体的には，オフラインフェーズで事前に認証コード候補の絞り込みを行うことで，攻撃成功確率の向上を図る．

5.1 オフラインフェーズの脆弱性

脆弱性 4. 公開鍵候補の判定オラクル.

与えられた公開鍵候補のうち平方剰余判定により明らかに誤ったものを除外することで，候補数 2^M 個の認証コードを約半数に削減できる．基本的攻撃のオフラインフェーズ（4.3 章）では，全ての認証コード候補を列挙し，それぞれから派生する AES 秘密鍵で暗号化された X25519 公開鍵を復号するが，誤った認証コードを選ぶと AES-CTR.Dec は誤った公開鍵候補を返す．この出力を乱数とみなし，正規の X25519 公開鍵形式との識別手法を用いれば，事前に候補数をさらに削減できる．

本研究では平方剰余による判定に着目する [4]．この判定は X 座標のみに基づいて実行可能であり，X 座標のみで構成される X25519 公開鍵に直接適用できる．公開鍵候補の削減率は 2^{-1} であり，これは一様乱数列を X25519 の公開鍵 (X 座標) u と解釈した場合，曲線方程式の右辺が平方剰余となる確率がおよそ 2^{-1} であることに起因する．

5.2 攻撃手順

脆弱性 4 を用いて基本的攻撃 (4 章) の成功確率を向上させる．攻撃手順における通信を図 1 に示す．本章では簡単のために図の左半分の PD に対する不正 CD 登録攻撃のみを紹介する．また，平方剰余判定を用いた公開鍵の削減手順を以下に示す．攻撃者 (MS) はあらかじめ正当な CD から送信された `Companion Hello` を保存している．そして，オフラインフェーズの最初に暗号化された X25519 公開鍵に対して，2^{40} の認証コード候補を用いて対応するすべての復号結果の候補を事前計算する．次に攻撃者はオフラインフェーズで，公開鍵候補 u を用いて，X25519 を構成する Montgomery 曲線の右辺を計算する．

$$E : v^2 = u^3 + Au^2 + u \pmod p \tag{16}$$

ここで $p = 2^{255} - 19$，$A = 486662$ である．次に，$r = v^2$ としたときに，r が平方剰余であるかを判定する．平方剰余判定は ルジャンドル記号 $\left(\frac{r}{p}\right)$ を用いて次のように定義される：

$$\left(\frac{r}{p}\right) \equiv r^{\frac{p-1}{2}} \pmod p = \begin{cases} 1 & (\text{rは平方剰余}), \\ -1 & (\text{rは非平方剰余}), \\ 0 & \text{if } r \equiv 0 \pmod p. \end{cases} \tag{17}$$

攻撃者はオフラインフェーズでこの手順を 2^{40} の公開鍵候補全てに適用する．そして，非平方剰余と判定されるおよそ半数の公開鍵候補と，それに対応する認証コード候補を破棄する．その後，残りの半数の公開鍵候補を使用し，オンラインフェーズを実行する．オンラインフェーズ以降の攻撃手法は基本的攻撃と同様であるため説明は省略する．

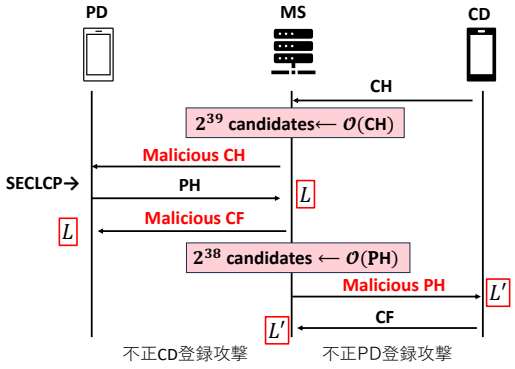

図 1 PD, CD に対する判定オラクル利用攻撃（CH: `Companion Hello`, PH: `Primary Hello`, $\mathcal{O}(X)$: X に対して判定オラクル (脆弱性 4) を実行, L : 認証完了後の共有値）.

5.2.1 攻撃成功評価

オンラインフェーズにおいて，攻撃者は削減後の公開鍵候補から一つの公開鍵を選択し，認証を試みるため，削減数に比例して確率は向上する．候補数の削減回数は，認証コードを用いた X25519 公開鍵の暗号化回数に依存する．例えば，不正 CD 登録攻撃では `Companion Hello` のみを使用する．X25519 公開鍵の暗号化回数は 1 回のみであるため，削減後の X25519 公開鍵候補数は $2^{40-1} = 2^{39}$ 個となり，成功確率は 2^{-39} である．一方，不正 PD 登録攻撃では `Companion Hello` と `Primary Hello` を使用するため，X25519 公開鍵の暗号化回数は 2 回となる．削減後の X25519 公開鍵候補数は $2^{40-2} = 2^{38}$ 個となり，その攻撃成功確率は 2^{-38} である．

オフラインフェーズで実施する平方剰余判定は定数時間で実行可能であるため，1 回あたりの判定にかかる計算量は $O(1)$ となる．したがって，候補数の削減回数を X 回とすると，オフラインの計算回数は $\sum_{i=0}^{X-1} 2^{40-i}$ となる．不正 PD 登録攻撃で 2 回の削減を行う場合，$2^{40} + 2^{39}$ 回の計算コストがかる．一方，オンラインフェーズでは，通常の認証手順と比較して追加の計算や通信を必要としない．そのため，攻撃の過程で被害者が不正を検知したり，通常の認証フローと本攻撃フローを見分けることが難しい．

6. 再試行利用攻撃

本章では PD に対する不正 CD デバイス登録攻撃において，攻撃確率を向上させ，攻撃の実用可能性を向上させる手法を述べる．2.4 章より，CDR が認証コードの入力ミスを許容し，認証コードの再入力を許可していること，制限回数内の再試行では CD からユーザに提示される認証コードが変更されない仕様を利用する．まず，本章で利用する脆弱性（脆弱性 5）について述べ，その後に再試行を考慮する攻撃の手順を述べる．その後，以下の 2 つの設定における攻撃成功確率と計算量評価を行う．

- 再入力が 2 回まで許容されている設定の攻撃（実際の

設定に基づく）

- 悪意のあるサーバ攻撃者が再入力回数を任意回に設定可能な場合の攻撃

なお，本攻撃による候補数削減は PD に対する不正 CD デバイス登録攻撃のみ影響を与える．これは再試行時に Primary Hello から再開し，Companion Hello が再送されないことに起因する．

6.1 オンラインフェーズの脆弱性

脆弱性 5．再試行による攻撃成功確率の向上．

認証コードの再入力は 2 回まで許容されており，初回の入力を含めて最大 3 回，同一の認証コードで再試行可能である．これにより，公開鍵候補の判定オラクル（脆弱性 4）を繰り返し適用することで，攻撃成功確率を向上させることができる．

6.2 攻撃手順

脆弱性 5 を用いて再試行を利用する悪意のあるサーバ攻撃者 (MS) により，不正な PD を被害者の CD に登録する手順について述べる．このとき，再試行回数の上限を N 回とする．これまで，正規の CD または PD どちらか一方との通信を介した攻撃を紹介した (4，5 章)．本攻撃はこれまでの攻撃と異なり，正規の CD および PD を同時に利用して不正 PD を登録する攻撃である．

攻撃の概要を図 2 に示す．ここで，矢印上の黒字は正規の CD または PD から送信されたデータを指す．矢印上の青字は候補数削減のために送信されるデータである．矢印上の赤字は攻撃者が CD に送信する攻撃用データである．

詳細の手順を以下に示す．各行の末尾の括弧は認証コードの候補数を示す．手順 (1)〜(5) では再試行によって再送される Primary Hello に脆弱性 4 を適用し，認証コード候補の削減を行う．これを制限回数 N まで繰り返す．手順 (6)，(7) では削減した認証コード候補から 1 つ認証コードを選び，オンラインフェーズで CD に対して不正 PD 登録攻撃を行う．

(1) MS は正規の CD から Comapnion Hello を受信し，Malicious Companion Hello を PD に送信する．このとき，判定オラクル攻撃の認証コード (SECLCP) 候補数の削減を行う (5 章と同様) (候補数：2^{39})．

(2) MS は正規の PD から Primary Hello を受信し，改ざんせずに正規の CD へ転送する．Primary Hello を用いて認証コードの候補数を削減する (候補数：2^{38})．

(3) MS は正規の CD から Companion Finish を受信し，この AES-GCM のタグをランダム値に改ざんする．改ざん後の Companion Finish’ を PD へ送信する．

(4) PD は Companion Finish’ の検証に失敗することから，Primary Hello から再試行を行う (2.4 章)．このとき，初回の Primary Hello と同じ認証コードに基づいて暗号化されるが，AES-CTR が確率的暗号であることから再試行時に異なる暗号文で構成される．

(5) MS は Primary Hello の受信から Companion Finish の受信までを 1 回として繰り返しこの処理を行う．このとき，追加の認証コードの削減を行う．Primary Hello が常に異なる暗号文で構成されるため，すでに削減済みの認証コード候補に対してもさらなる削減を行うことが可能である．

(6) MS は N 回目の再試行時の Primary Hello を破棄し，Malicious Primary Hello を作成し，CD に送信する．これは，2^{38-N} 候補から選択された一つの認証コードから派生した共通鍵と $IV_{\mathrm{Hello}}^{\mathrm{Adv}}$ を入力として，攻撃者の公開鍵 $k_{\mathrm{pub}}^{\mathrm{Adv}}$ を AES-CTR によって暗号化した値である (候補数：$N=2$ のとき 2^{36})．

(7) MS が推測した認証コードが CD が持つ認証コードと一致するとき，攻撃に成功し，不正 PD が正規の CD と同じ認証情報 L'_{cpn} を共有する．一致しない場合は攻撃に失敗し，攻撃回数が上限に達したとこから再試行は行われない．

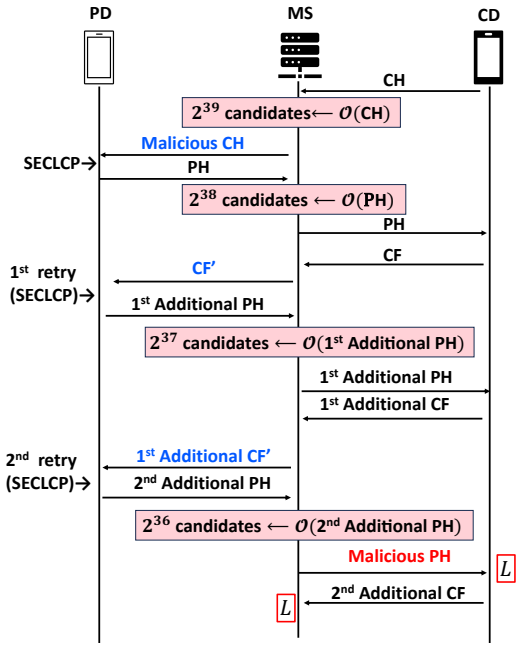

図 2 CD に対する再試行利用攻撃（CH: Companion Hello, PH: Primary Hello, CF: Companion Finish, Additional PH/CF: 再生成し再送された PH/CF, $\mathcal{O}(X)$: X に対して判定オラクル（脆弱性 4）を実行, L : 認証完了後の共有値.

6.3 攻撃評価

最大試行回数 $N=2$ の設定：これは脆弱性報告時における仕様上の設定である．この設定では，最終的に候補数を $2^{38-2}=2^{36}$ 回まで削減できるため，成功確率は 2^{-36} となる．これは基本攻撃の 16 倍に相当し，無視できない成功確率である．なお，本攻撃は不正 CD 登録時の判定オラクル

攻撃を内包するため，フォールバック時の成功確率は 5 章と同じ 2^{-39} となる．

最大試行回数が任意の設定：再試行上限回数を N とすると，候補数は 2^{38-N} であり，攻撃成功確率は 2^{-38+N} となる．つまり，悪意のあるサーバが再試行上限回数を制御可能であれば，高確率で攻撃可能となる．例えば，$N = 38$ とした場合，攻撃成功確率を 1 まで向上させることができる．

不正 CD 登録時の成功確率は 2^{-39} である．また，候補削減に係る計算量は $O(1)$ である (5.2.1 章)．よって再試行回数の上限値 N の計算回数は $\sum_{i=0}^{N+1} 2^{40-i}$ となる．

ここで，本攻撃に対する Meta 社の見解を次に示す．

> “Specifically, Attack 3 as described is not possible because WhatsApp primary applications rely on hard-coded retry values baked into the binary that can not be modified/altered by the server.”

なお，文中の Attack 3 は悪意のあるサーバ攻撃者が任意の再試行回数 N を設定できる仮定の攻撃を指す．このように，Meta 社は再試行回数がソースコードにハードコードされていることを理由に，攻撃者が実行ファイルのみから試行回数を $N = 2$ 以外に変更することは不可能であると主張している．確かにハッシュ値やコード署名，Trusted Platform Module といった実行ファイル保護手法は，サーバ侵害時の防御手段として有効な場合があるが，具体的な保護方法は開示されておらず，我々が仮定する悪意あるサーバ攻撃者に対して有効かは不明である．さらに，本稿の対象は E2EE 互換システムであり，サーバであっても認証を突破できないことが前提となるため，この攻撃者モデルを考慮することは妥当であると考える．

7. 提案対策と適用状況

本章では，我々が Meta 社に提示した対策手法を述べ，次に Meta 社が我々に約束した修正方針について述べる．

7.1 提案した対策

我々は即時に適用可能かつ効果的な対策として以下の 2 点を提案した．

脆弱性 1 への対策：認証コード (SECLCP) に関して，40 ビットよりも十分に大きいエントロピーの設定を提案する．ユーザ体験を考慮し，文字数の増加よりも 1 文字あたりのエントロピーを増やすことを推奨する．例えば，Base64 形式を採用することで，現行の Base32 形式よりも 1 文字あたりに対する高いエントロピーを実現できる．

脆弱性 5 への対策：再試行機能を廃止し，認証失敗後に同一認証コード (SECLCP) の使用を許容しないことを提案する．加えて，認証失敗時には，必ず新しい認証コードを CD 上に表示することを提案する．

7.2 Meta 社の対策

我々の提案した対策を基に，Meta が採用した改善策はい以下の通りである．

認証コードのエントロピー増大：Meta 社は我々の提案に対して認証コードのエントロピーの増強を約束した．

同一認証コードの打ち直しを廃止：認証コードの打ち直し可能回数が 2 回から 0 回へと変更された．修正後は，再試行時に認証コードは必ず再生成される．この修正は 2025 年 6 月に実際の環境に導入された．

8. まとめ

本研究では，WhatsApp の Companion Device Registration 機能における端末追加プロトコルを解析し，悪意あるサーバが利用者の協力なしに，現実的な計算量と成功確率で不正デバイスを登録可能であることを示した．

Meta 社は我々の報告を受けてエントロピー増大や再試行機能の廃止を導入し，我々の修正提案の一部を実際の環境に反映した．これにより，本研究で提示した攻撃は実行困難となり，WhatsApp の安全性が大幅に改善された．本研究の結果は，既存の形式化による解析で発見されていなかったことから，巨大なユーザ基盤を有するシステムにおいては，形式的手法と実装に即した手動解析を組み合わせた包括的な評価が求められることを示した．

謝辞　我々の脆弱性報告に対する Meta Security 及び WhatsApp エンジニアチームの迅速な対応及び修正作業に感謝申し上げます．また，本研究を遂行するにあたり多大なる貢献をしていただいた，兵庫県立大学 暗号・情報セキュリティ研究グループの川上翔馬氏, 佐々木皓亮氏に深く感謝します．本研究は, JSPS 科研費 JP24H00696 と JST AIP 加速課題 JPMJCR24U1 の支援を受けたものである．

参考文献

[1] Meta: WhatsApp Encryption Overview, Version 8 (2024). https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

[2] Albrecht, M. R., Dowling, B. and Jones, D.: Formal Analysis of Multi-device Group Messaging in WhatsApp, *Advances in Cryptology - EUROCRYPT 2025, Proceedings, Part VIII* (Fehr, S. and Fouque, P., eds.), Lecture Notes in Computer Science, Vol. 15608, Springer, pp. 242–271 (online), DOI: 10.1007/978-3-031-91101-9_9 (2025).

[3] Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I. and Smith, M.: SoK: Secure Messaging, *2015 IEEE Symposium on Security and Privacy, SP 2015, May 17-21, 2015*, IEEE Computer Society, pp. 232–249 (online), DOI: 10.1109/SP.2015.22 (2015).

[4] Bernstein, D. J., Hamburg, M., Krasnova, A. and Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013* (Sadeghi, A., Gligor, V. D. and Yung, M., eds.), ACM, pp. 967–980 (online), DOI: 10.1145/2508859.2516734 (2013).