The State of Messaging Security 2020: メールおよびメッセージングアプリのセキュリティプロトコルの現在

Hiroki Suezawa（@rung）
https://github.com/rung

公開情報

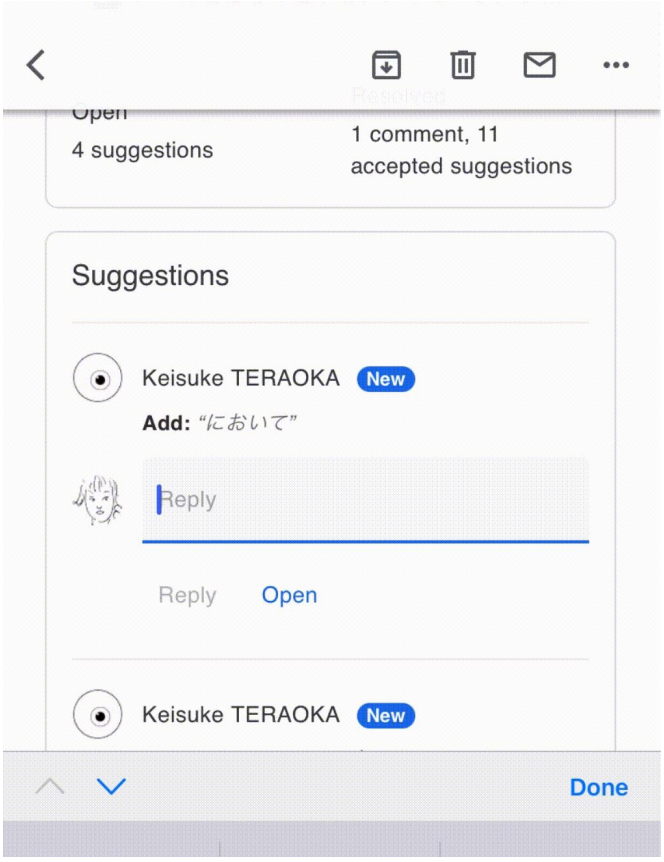
Ver1.50 (2020年11月1日)
初回公開日付: 2020年10月17日
公開URL: https://github.com/rung/messaging-security-2020

目次

Chapter 1: はじめに

本文書では、主に電子メールからWhatsAppやLINE、Facebook Messengerなどのメッセージングアプリまで、メッセージングプロトコルを取り巻くセキュリティについて、2020年現在の状況を、技術的な観点から説明する。

2013年のスノーデン事件以降、インターネットにおけるTLSなどを利用した暗号化通信の利用は急速に広まった。しかしながら、メールに関連する暗号化やセキュリティについての情報はWebに比べると非常に少ないし、議論が十分に行われているとは言い難い。
その一方で、メールは依然としてインターネットを支える根幹の技術として位置し続けている。いくら個人間コミュニケーションとしてWhatsApp、LINE、Facebook Messengerなどが普及しようとも、オープンに標準化され、誰もが使えるメッセージングプロトコルとして、メールは特定のメッセージングアプリでは置き換えることの出来ないインターネットの基盤技術である。たとえば多くのサービスは、メールを使ったパスワードリセットを提供しているし、HTTPSを支えるTLS証明書は、メールを使ったドメイン所有の確認により発行することが出来る[1]。加えて、すべてのメールサービスが対応しているわけではないが、メールの表現できる内容も増えており、Googleが2018年に発表した"AMP For Email"を利用すると、動的なメッセージが使用できるようになっているなど、利用の幅も拡大している。



※ 動的メッセージの利用例 (動的なGoogle Docsへのコメント)

それでは、現状、メールの信頼性はどのように守られているのか？メールのセキュリティに

[1] CAによるTLS証明書の発行ポリシーは、業界団体であるCA/Browser Forumが事実上規定しており、特定のメールアドレスを所有していることによるドメイン認証を認めている。参考: "Constructed Email to Domain Contact" （Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates）

関する通信プロトコルは、何を守り、守らないのか？そして、昨今、国際社会で議論されている「End-to-end暗号化」とは、どのようなセキュリティ技術であるのか？

日本において、メールに関するセキュリティは、メールサービスを提供する事業者向けの情報は多いものの、ユーザ目線やプライバシーの観点から語られることは少なかった。アメリカではEFF（Electronic Frontier Foundation）がSTARTTLS Everywhereというメール暗号化のキャンペーンを2014年より行っていたが、日本ではメール利用者からこういったキャンペーンが行われることもなかった。Webのセキュリティについては、スノーデン事件以後、EFFとMozillaなどが中心となり、Googleなどのサポートも受けて誕生した、無料で証明書を発行するCAであるLet's Encryptなどの立ち上げも後押しして、HTTPSについては一気に普及が進んだ。一方で、メール暗号化については、Googleが外部と平文でやりとりしたメールドメインを確認する限り、暗号化が行われていないメール送信元・送信先として、.jpドメインの多さが目立つ状況にもなっている。アメリカ政府により、アメリカにおいてSIGINT・サーベイランスにより、メールのデータベースが作成されていたとすると[2]、日本でメールを使っている一般市民の個人的なメール通信について、国家機関により監視されたケースもあると想定できる。

本文書では、まずメールの仕様を解説したあとで、暗号化や認証などの現況を説明する。その後、LINE社のLetter Sealingプロトコルおよび、WhatAppやFacebook Messengerで採用されているSignalプロトコルを例にとり、メール以外のメッセージングアプリのEnd-to-end暗号化プロトコルを紹介する。主にPrivacyやTLS、End-to-end暗号化などについて関心のある方々が、メッセージングを支えるセキュリティ技術に対して、2020年現在の現況を把握出来るようにすることを目的として記載している。
各章のはじめには要点を記載しているため、その箇所のみ読むという読み方もできるようになっている。

なお、内容に事実誤認があった場合は、速やかに訂正するので、ご連絡をいただけると嬉しい。

[2] 土屋 大洋『サイバーセキュリティと国際政治』P10（千倉書房，2015年） 2015

Chapter 2: メールの仕組み

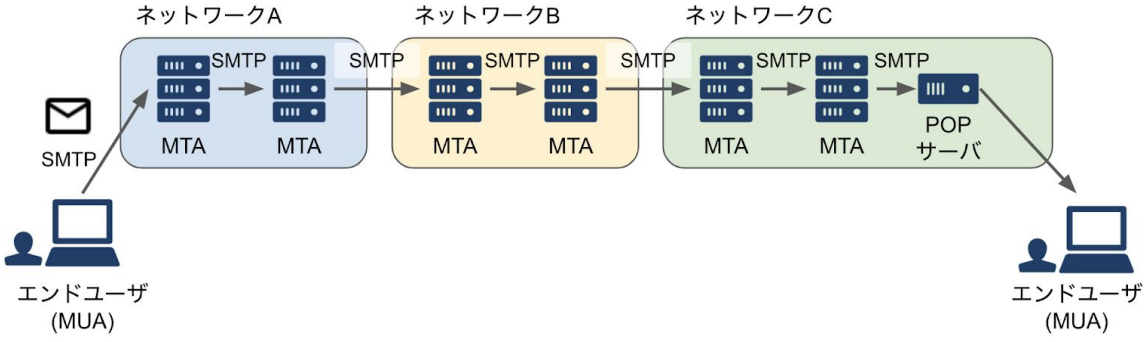
Chapter 2の要点

- メールの配送を行うソフトウェア（Sendmail, Postfixなど）をMTAと呼ぶ
- MTAによるメールの配送にはSMTPという80年代に策定されたプロトコルが使用される
  - そのままの使用だと平文での通信となる
  - 上記の問題の補強を行うために、Chapter 3から説明するセキュリティプロトコルが存在する
- メールにはEnvelope（封筒）と、Headerというものがあり、両方にメール送信元と送信先が記載されている
  - Envelope From/To: メールの配送のためにMTAが使用する送信元/送信先
  - Header From/To: エンドユーザのメーラーに表示される送信元/送信先
- エンドユーザが送信元として認識しているのは、メーラーで表示されるHeader Fromである

メールにおけるセキュリティについて説明するにあたり、本文書では主にSMTP（Simple Mail Transfer Protocol）を取り上げる。

まず前提としてメールの概要を説明する。
IETFでSMTP（RFC821）が策定されたのは1980年代初頭であり、インターネットに暗号技術が普及するはるか以前から存在している。たとえば80年後半の世界初のコンピュータワームと呼ばれるモリスワームはMTA（sendmail）の脆弱性を利用するなど、インターネットにおいてだけでなく、コンピュータウイルスが脅威とみなされる時代より前から存在している歴史の長いプロトコルである。そのプロトコルが、基本的な仕組みは変えないまま、少しずつ拡張されながら現在も利用されている。

メールの基本思想



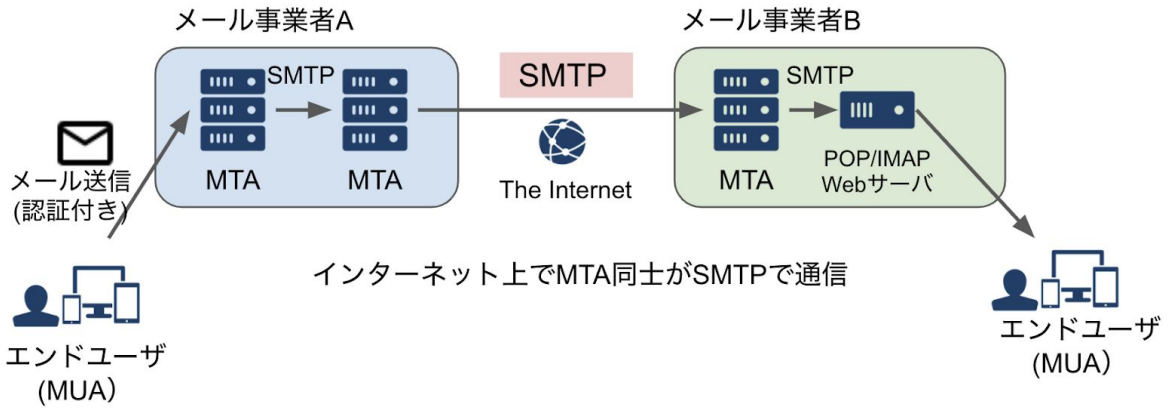
かつては、メールは隣接するネットワーク上のSendmailというMTA（Mail Transfer Agent）にメール配送を繰り返していき、最終的に目的地に届けるという前提で運用されていた。[3]SMTPは、世界中のIPアドレスにデータ配送されることが当たり前の時代以前から存在するプロトコルである（インターネットの基盤プロトコルであるBGPすら標準化されて

[3] 90年代、ネットワーク間を中継するオープンリレーサーバは一般的であったが、90年代中盤以降、スパムメールの送信に使われることが問題となり、非推奨の設定となった（参考: Open mail relay）

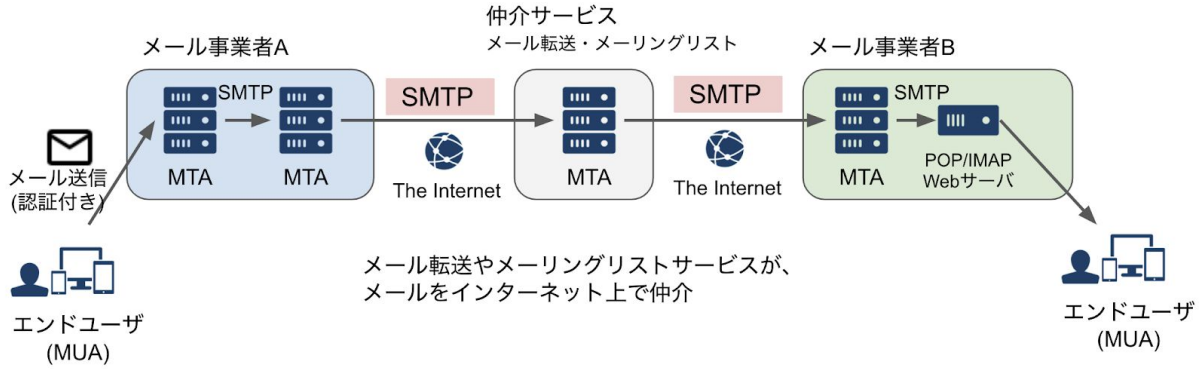
いない時代から存在している。日本でIIJが商用インターネットを始めたのも1992年であり、SMTPの歴史はそれよりも長い）。
SMTPは相互運用性が非常に重視されているプロトコルであり、次段MTAがオフラインの場合でも、ローカルでキューイングを行い定期的に再送を試みることで、次段MTAが復活した際にメール配送を継続する事ができる。こういったリレーを繰り返して最終目的地にたどり着くのが基本的なメールの考え方である。常にコンピュータに電源が入っているとも、ネットワークにつながっているとも言えない80年代を考えると、非常に理にかなった仕様である。MTAとして代表的なオープンソースソフトウェアであり現在も広く使われているPostfixなども、もちろんこのキューイングや再送の仕様を備えている。

現在の基本的な構成

- パターンA （直接通信）

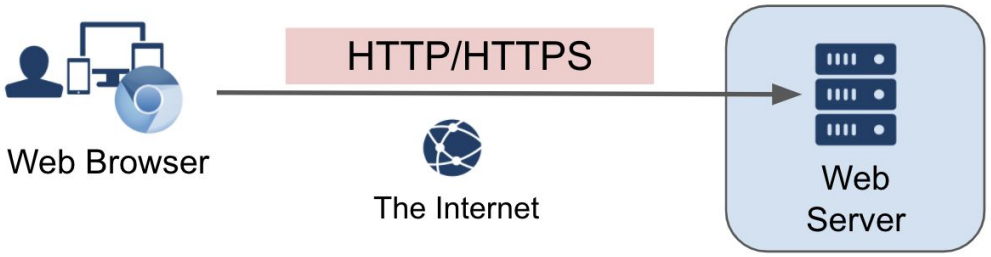


- パターンB （仲介サービス経由）



現在では、主に上記の構成で利用されることが多い。インターネットに面したMTA ClientとMTA ServerがSMTPを用いて通信を行う。世界中のIPアドレスにいつでもどこでも通信が行える現在を考えると、当たり前の状況と言える。注意事項としては、パターンBのようにメール転送やメーリングリストの使用時は、別のサービスが間に入ることになる。メール配送のプロトコルについては、現在でもSMTPを用いて通信が行われている。

参考: HTTPにおける基本的な構成



※Web Serverの前段にCDNやLoad Balancerが入る
ケースも多いが簡単のためWeb Serverのみ記載する

SMTPの仕組み

メール配送先の名前解決

メールの配送先が”example@gmail.com”である場合、インターネットに面したMTA Clientは”gmail.com”にメールを配送することを試みる。

MTA Clientは、”gmail.com”のMXレコードを用いて、メールの配送先を確認する。
優先度の高いgmail-smtp-in.l.google.comのAレコード「64.233.189.27」を配送先と決めSMTP接続を行う。

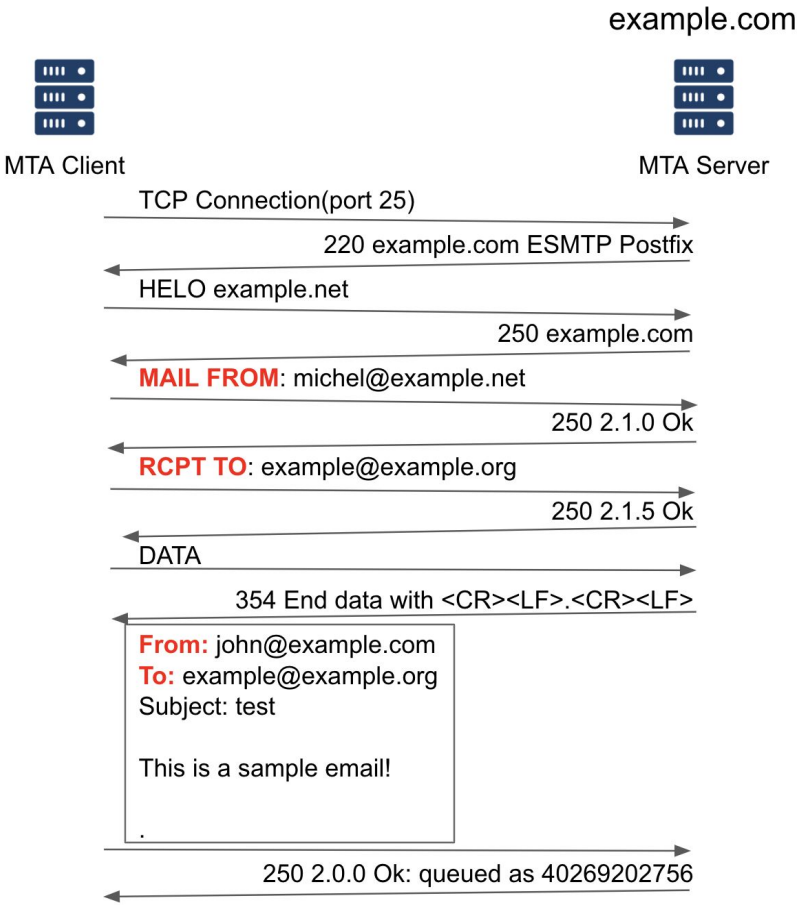
※名前解決の例
% dig gmail.com mx +short
5 gmail-smtp-in.l.google.com.
10 alt1.gmail-smtp-in.l.google.com.
20 alt2.gmail-smtp-in.l.google.com.
30 alt3.gmail-smtp-in.l.google.com.
40 alt4.gmail-smtp-in.l.google.com.
* 最初の数字は優先度である
* MXレコードに直接IPアドレスを書くことは出来ない。必ずドメイン名が指定される

% dig gmail-smtp-in.l.google.com a +short
64.233.189.27

- メールの構造

メールは主に、SMTPで使用されるEnvelopeと、エンドユーザの利用するメーラーが表示するDATA部に分けられる。メール配送にはEnvelope From/Toが用いられる一方で、メーラーはHeader From/Toを送信元/送信先として表示する。

- 下記に、SMTPの基本的な通信の流れを記載している



上記通信のMAIL FROM、RCPT TOで入力されているのがEnvelope、DATAとして続けて入力しているのがHeaderおよびBodyである。

MTAは、Envelope（封筒）をSMTPでのメール配送に利用する。エンドユーザの利用するメーラーは、Envelopeではなく、DATA部のHeader部分に記載されているFromとToを利用する。
Envelope FromとHeader Fromは異なっていてもよい。例えば上記の例では、Envelope Fromではmichel@example.netを名乗っているが、Header Fromではjohn@example.comが指定されている。ここでは、最終的にエンドユーザのメーラーに送信元として表示されるのは”Header From”であることを認識しておいて頂きたい。
Envelope Fromは、メールの送信に失敗した際などのバウンスメール（エラーメール）の返送先として主に利用される[4]。

SMTPのセキュリティ上の課題

SMTP単体だと、全て平文の通信となるし、送信先や送信元の認証はない。
たとえば、あなたのメールボックスに、あなたの友人からメールが届いたとする。SMTP単体では、そのメールを送ったのは本当にあなたの友人なのか、メールの途中が改ざんされていないか、誰かに盗み見られていないかなど、何も保証されていない状態である。

次章からは、SMTPを補強する各セキュリティプロトコルを説明する。そして、それぞれのセキュリティプロトコルの制約や限界についても併せて説明する。

[4] バウンスメール自体のEnvelope Fromは<>（空）であり、バウンスメールの配送に失敗した際は、MTA上で破棄されるようになっている。

Chapter 3: SMTP Over TLS（STARTTLS）の守るもの

Chapter 3の要点

- メールのMTA間での経路暗号化を行う方法としてはSTARTTLSが普及している
- Googleの透明性レポートによると、2020年10月時点で世界的には90%程度が暗号化に対応しているが、日本国内の大手メールサービスなどはサポートしていないケースも多く見受けられる
- STARTTLSの特徴
    - STARTTLSは、盗聴などの受動的攻撃を防ぐ手段であり、通信に介入されるなどの能動的な攻撃には無力である
    - MTA Server（受信側）での送信元の認証、MTA Client（送信側）での接続先の認証などは提供されない
    - エンドユーザが、メール中継する各MTAにTLS暗号化を強制することはできない
- 能動的な攻撃も防ぐための手段として2018年にMTA-STSという仕様がIETFにて策定され、アメリカの大手メールサービスを始めとする一部のサービスが利用を始めている段階である
    - MTA-STSを用いると、MTA Clientが接続先が正しいことを認証出来る
    - ただし、MTA-STSを用いてもエンドユーザがメール中継する各MTAにTLS暗号化を強制することはできない
    - STARTTLSの普及率を考えると、全てのMTAがMTA-STSをサポートする日が来るかは疑問

SMTPをTLSを用いて暗号化通信を行う方法として一般的なものは、STARTTLS（スタートTLS）である。STARTTLSを規定するRFC3207は、SMTPを拡張する技術として2002年に策定された。

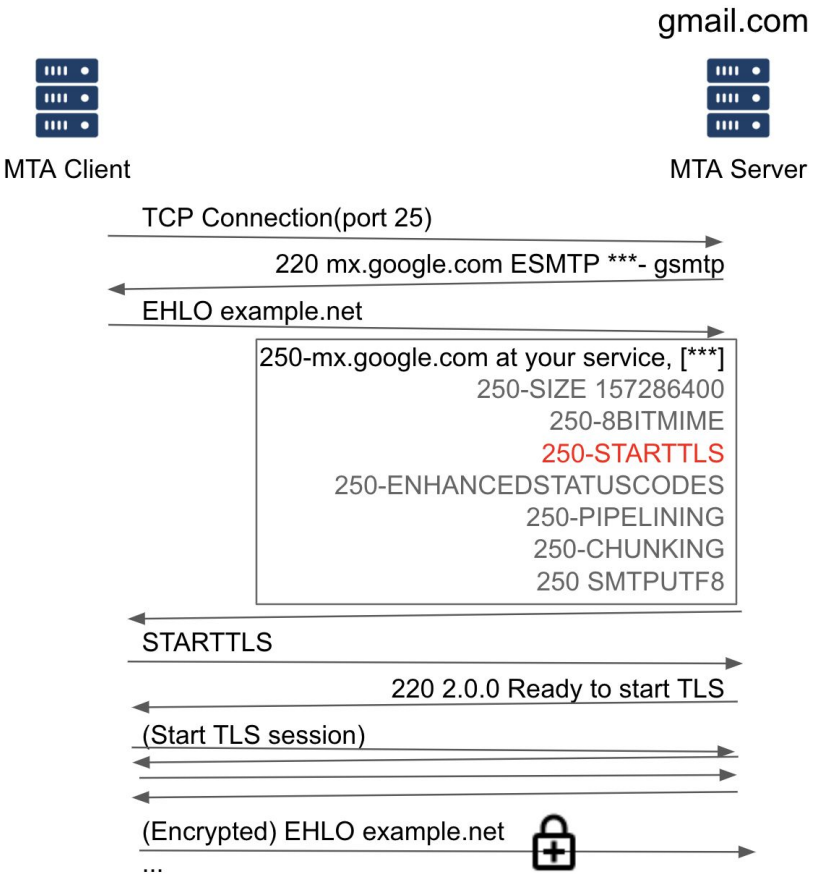
2003年に刊行された『マスタリングTCP/IP SSL/TLS編』[5]にはこういう一文がある。「送信者は、メールが受信者に安全に配送されることを求めます。受信者は、送信者を認証し、メッセージが途中で改竄されていないかどうかを求めます。」「HTTPSではこれらの目標すべてが満たされています。しかし、これまで見てきたとおり、STARTTLSではこのいずれの性質も実現していません」
SSL/TLSの使用されるバージョンに変化はあるが、根本的なSTARTTLSの仕組みは当時から変わっていない。[6]

STARTTLSの仕様

STARTTLSでは、日和見暗号化（Opportunistic encryption）を提供する。暗号化を試みるが、悪い攻撃者により、暗号化を解除されうる。また、暗号通信の失敗時には平文にフォールバックされる動きが一般的である[7]。もちろん、そもそも通信相手が暗号化に対応していない場合は、平文通信が基本である。

---

[5] Eric Rescorla『マスタリングTCP/IP SSL/TLS編』P409（オーム社、2003年）
[6] SMTPをTLS上でやり取りするプロトコルとしてはSMTPSも存在するが、MTA間通信として普及していないため本文書では取り扱わない。
[7] Postfixにおいて一般的なTLS Security Levelである”May”は、STARTTLSに対応していない相手との通信は平文で行う。また、暗号化通信に失敗した場合も平文にフォールバックを行う。（参考: http://www.postfix.org/TLS_README.html#client_tls_may）



上記の通り、最初に平文で通信を開始する。
まず、MTA ClientがSTARTTLSに対応しているかをEHLOコマンドにより確かめる。MTA Serverが「250-STARTTLS」と返したときに、MTA ClientがTLS通信を後から開始するという仕組みになっている。TLSセッションが貼られ暗号化が開始されたあとの通信は、通常のSMTP通信と同様のメッセージのやりとりを行う。
ポイントとしては、平文でTLSに対応しているかを確認することである。Webでは、平文（HTTP）は80番ポート、TLS（HTTPS）は443番ポートが使われる一方、メール（SMTP）の場合は一般的に25番ポートのみが使用される。

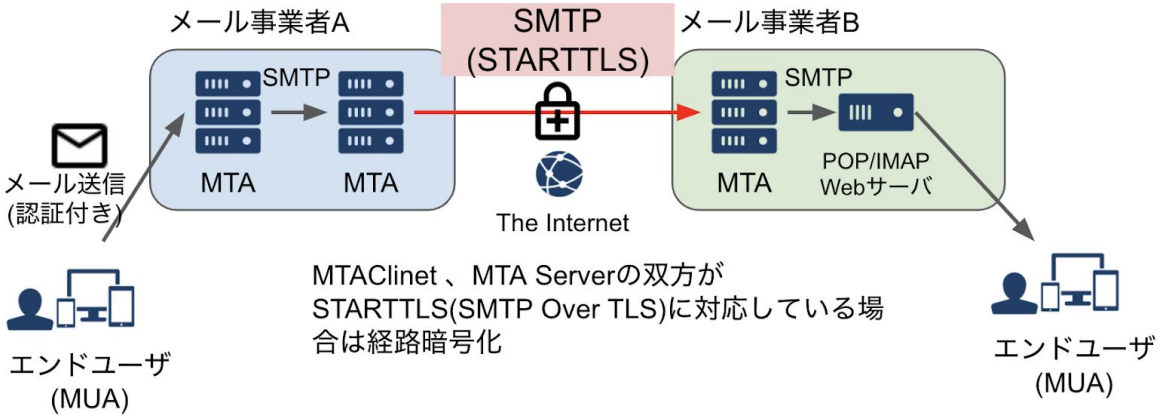
証明書検証

HTTPSにおいて、たとえばhttps://www.example.comにアクセスした際は、証明書はwww.example.comというドメイン名に対して発行されているかを検証することになる。SMTP Over TLSでは、次段配送先のMXレコードに記載されているHostnameを証明書の検証において利用する。たとえばgmail.comにメールを配送する場合、gmail.comのmxレコードに記載されている「gmail-smtp-in.l.google.com」を使って証明書検証することになる。[8]

> 注: そもそも証明書検証はRFC3207上では強制されていない。
> 日和見暗号化であるSMTP Over TLSでは証明書の検証も行われないことが多く、自己署名証明書を使うこともあった。PostfixのManualでは、明確に証明書検証のデフォルト有効化は非推奨とされている。（”On a machine that delivers mail to the Internet, you should not configure mandatory server certificate verification as a default policy.”）
> ただし、昨今は大手メールサービスが証明書を検証するようになっているため、CAが発行した証明書が設定されているケースが基本だろう。

STARTTLSが守るものと制約

暗号化の範囲



STARTTLSにMTA ClientおよびServerが対応している場合、通信するMTA間でTLS暗号化が行われる。適切に暗号化が行われた場合、インターネット上で通信の傍受を行っても、復号することは出来ない（いわゆる受動的攻撃に対して耐性がある）。

制約

STARTTLSには下記の制約がある。

- TLS通信はインターネットに面したMTA間で行われる
  - ブラウザとサーバ間でTLSネゴシエーションが直接行われるHTTPSと違い、STARTTLSはMTA間でのネゴシエーションとなる。
  - エンドユーザがMTA間のSTARTTLSを強制することは出来ない。MTA Clientもしくは Serverのどちらか一方がSTARTTLSに対応していない場合は、TLS暗号化は行われない。
- End-to-end暗号化ではない
  - エンドユーザ間で行われるEnd-to-end暗号化ではない

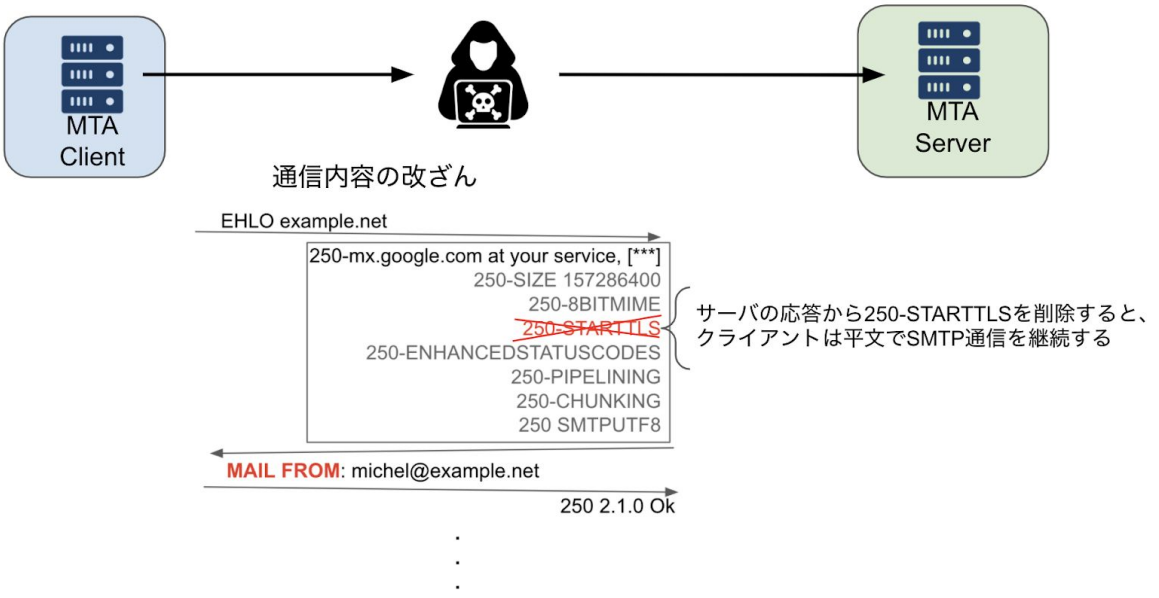
[8] GmailとSTARTTLSによりTLS通信した際に送信されてくる証明書の例: https://crt.sh/?id=3473247616

    - 各メール事業者内のデータセンター内やインターネット上で多段にMTAを経由する場合、全てのMTA間でTLSによる暗号化が行われているとは限らない
    - 経路暗号化であり、各MTAやメールを保管するメールボックス上では暗号化されず平文のまま処理される。
- 証明書検証はMXレコードに記載のHostname
    - 証明書検証は宛先メールドメインそのものではなく、MXレコードに記載のHostnameを用いて検証を行う。そのため、宛先ドメイン名に正しく接続されたことは検証されない。
- STARTTLSは送信元を検証する手段ではない
    - 経路が暗号化されていても、送信元が正しいことを保証しない。

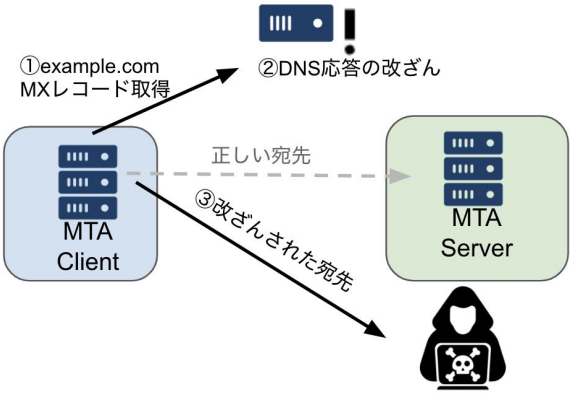
歴史的な経緯により、多段にメールを中継していくMTAに、通信経路の日和見暗号化以上の機能を持たせるのも難しかったと言えるのではないだろうか。実際、RFC策定段階で、次に説明する攻撃手法の一部は明記されている。

主な攻撃手法

主な攻撃手法として、SMTP通信の中間者攻撃（Man in the middle attack）が挙げられる。



また、SMTP通信の途中経路の侵入だけでなく、DNS応答結果の改ざんでも、攻撃が成立する。



9

もちろん、STARTTLSは通信経路の暗号化のためメッセージの内容そのものは暗号化されておらず、中継するMTAのうち一つでも攻撃され不正侵入された場合、メッセージの内容を取得される可能性もある。

多くの攻撃手法があるなかで、STARTTLSはあくまでネットワークの盗聴などの受動的な攻撃を防ぐものである。しかしながら、インターネットの大規模な盗聴からPrivacyや情報を守れるという観点で見ると、STARTTLSだけでも大きなメリットがあると言える。

補足: TLSバージョン

MTA ClientおよびServerのサポートするTLS Versionは、一般的にHTTPSに比べて古いバージョンをサポートすることが多い。主な理由として、STARTTLSでは暗号化通信が行われないときに平文にフォールバックする[10]方針が基本であることにより、新しいTLS Versionのみをサポートするメリットが少ないことなどが挙げられる。さすがに、平文よりもTLS1.0を利用する方が、盗聴に対する耐性は強い。また、各MTAは相互運用性を重視することから、新しいセキュリティ機能の追加されているOpenSSLのバージョンアップを渋っているケースなどもあり、旧バージョンのサポートアウトはそんなに単純な話ではない。

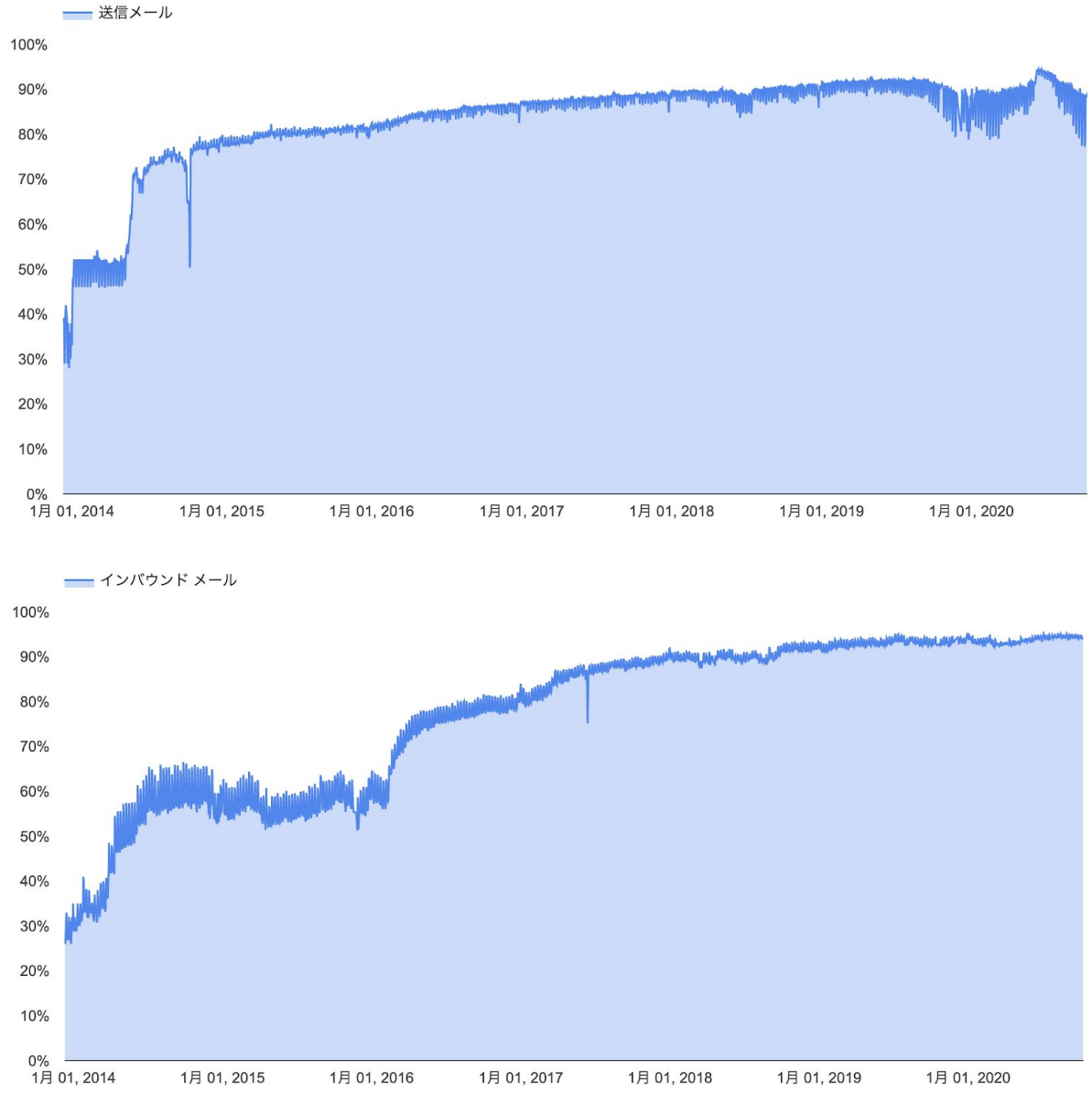
たとえば、Microsoftは2018年10月より、Office365全体でSMTP Over TLSも含むTLS1.0および1.1のサポートアウトを表明していた。しかし、その方針は延期し続けており、2020年10月17日現在でもTLS1.0での接続が可能な状態である[11]。また、SMTPにおいて平文はいまだなお一般的に使用されているため、平文より安全なTLS1.0、TLS1.1のサポートアウトをしたところで平文自体のサポートアウトは事実上不可能なのが現状である。

STARTTLSの普及率

Googleは2014年頃より、Googleと他社MTA間でのSTARTTLS（メール通信の暗号化）の普及率を透明性レポートにて公表している。当初は30%程度であったその比率は、スノーデン事件以後、急速に高まり、現在ではGoogleからの送信メール・受信メールの約90%が暗号

[9] DNS応答の改ざんについてはDNSSECの普及により改善する可能性はあるが、現状普及していない
[10] Sendmailなどの場合、TLS通信失敗時にメールが破棄されることもある
[11] Office 365 to enforce TLS 1.2 per October 15, 2020 - Dave Stork's IMHO
2020年10月15日よりTLS1.0および1.1の無効化を行うとアナウンスされているが、17日現在だと接続できる状況である。（例: openssl s_client -connect microsoft-com.mail.protection.outlook.com:25 -starttls smtp -tls1）

化されている状況となった。2014年よりSTARTTLS Everywhereプロジェクトを行っていたEFFは、2020年4月にプロジェクトを終了した。
もちろん、これはアメリカを中心としたGoogle全体の世界中とのメールのやりとりでの比率であり、日本国内での普及率ではない。単純な比較ではないが、アジアでの暗号化がなされていないメールドメインの例については透明性レポートで確認することができ、日本国内の大手メールサービスがSTARTTLSに対応していないケースも確認することができる。

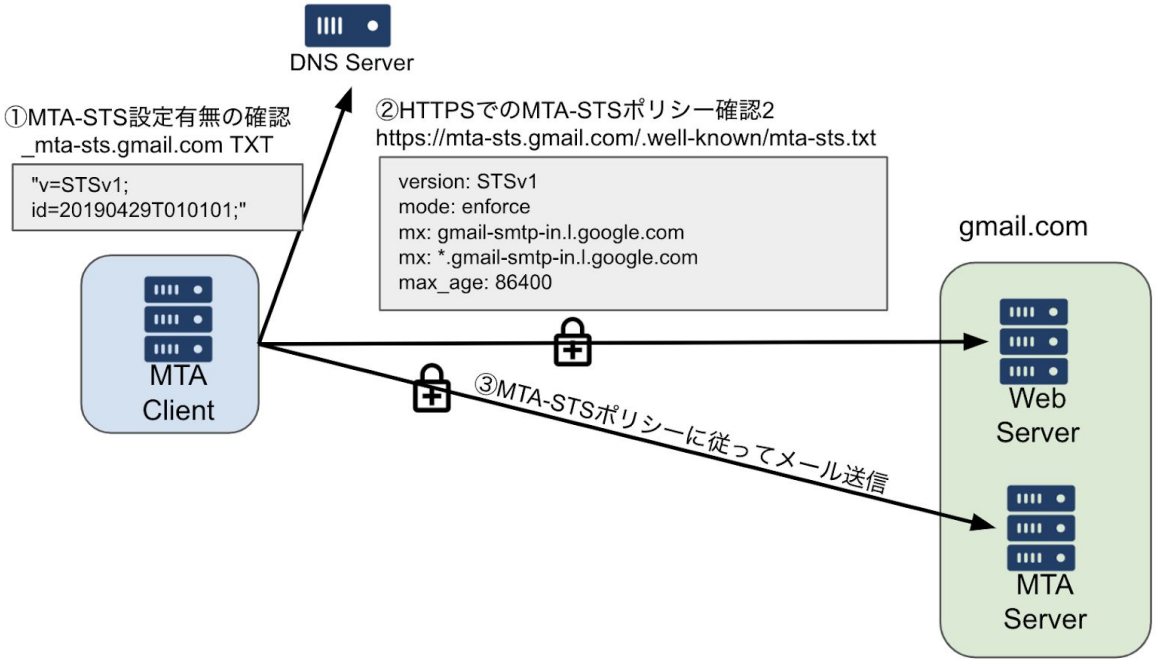


引用元: Email encryption in transit （Google）

解決案としてのMTA-STS

2018年、STARTTLSが能動的な攻撃を防げないという問題についても、解決するためのプロトコルであるMTA-STS（RFC8461）が標準化された。策定には主にGoogleやMicrosoft、Yahoo（US）、ISPなどの人々が関わっており、Draft段階からMTA-STSの実証実験を行っていたと記憶している。

HTTPSには、HSTS（HTTP Strict Transport Security）という、特定のドメインに対するTLSの使用を強制化するための仕様があるが、MTA-STSはいわばそのメール版であり、TLS暗号化を強制するための手段である。



MTA-STSの特徴としては下記が挙げられる

- MTA Clientが、宛先ドメインの認証を行うことにより、接続先サーバが正しいことを検証できる。
    - 宛先ドメインのMTA-STS情報を確認する
        - @gmail.comにメールを送る場合、gmail.comのMTA-STS情報を確認する。
        - （MXレコードに記載されている「gmail-smtp-in.l.google.com」ではない）
    - 配送先として許可するMXレコードの一覧をMTA-STS情報として記載する。
        - @gmail.comの場合は、”gmail-smtp-in.l.google.com”、”*.gmail-smtp-in.l.google.com”のみが記載されている。※証明書検証は、配送先のドメイン（gmail-smtp-in.l.google.comなど）に対して行う
- MTA-STSに対応しているサーバに対しては、接続時にTLS1.2以上のみを用い、証明書検証も実施する必要がある
- 宛先ドメインのWebサーバ（https://mta-sts.宛先ドメイン/.well-known/mta-sts.txt）に、設定ファイルを置く必要がある。
    - DNS情報の改ざん（DNSキャッシュポイズニング）は、HTTPSに比べると攻撃が容易である。TLSを用いた証明書検証の行われるHTTPSをPolicyの確認に採用することにしたのは、よりセキュアな選択と言える。
- TOFU（Trust-on-first-use）のプロトコルである。最初の接続時に情報が取得できなかたりした場合は、MTA-STSは有効化されない。

- Reporting機能があり、クライアント側でMTA-STSの設定が正しく取得できなかった際やTLS接続ができなかった場合は、サーバが指定した宛先にクライアントMTAから失敗レポートが送信される。

上述の通りTOFU（Trust-on-first-use）であり、MTA-STSの初回接続時のポリシー取得を防ぐことによりMTA-STSは無効化しうる。しかし、特にDNSサーバについて、長いMTA-STSのキャッシュ時間（TTL）を指定することなどにより、能動的な攻撃は非常に難しい状況になると言える。なお、MTA-STSはクライアント認証を提供する機能ではなく、届いたメールの送信元が正しいことを保証しない。

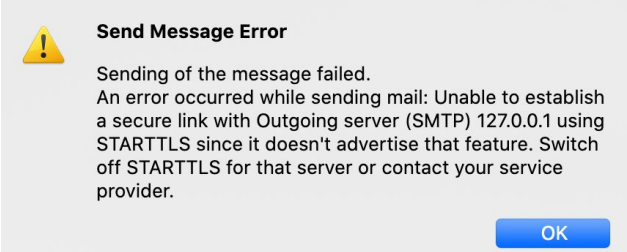
Googleを始めとするアメリカの大手メールサービス業者はSTARTTLSの強制設定や検証を始めており、その状況は各社の”/.well-known/mta-sts.txt”にアクセスすることで確認することができる[12]。現状、MTA-STSによるTLS強制設定（enforce mode）にしている事業者は少ないが、Gmail、大手ISPのComcast、高セキュリティを謳うProtonMail、2020年に登場したメールサービスであるHeyなどは既にTLS強制設定となっている。
また、ビジネスメールとしては、GSuiteで既にMTA-STSは設定可能となっており各社が使用できる状況になっている。[13][14][15]

しかし、メールにおける新規プロトコルの普及には、ステークホルダの多さから非常に時間がかかる。そもそもSTARTTLSすら対応していないメールサービスも存在する中、すべてのMTAがMTA-STSをサポートし、メールにおけるインターネット上の経路暗号化を完全に信じられる日が来るかと言われると、甚だ疑問である。

[12] 2020年10月時点で、gmail.comはenforce、outlook.comはtesting、yahoo.comはtestingの状況である。これらの企業はRFCの標準化が完了する以前から協力している。もちろんtestingにしている業者も、対応している会社からReportを受け取っている。
[13] MTA-STS と TLS レポートについて - Google Workspace 管理者 ヘルプ
[14] HSTSはtrust-on-first-useであるHeaderでのHTTPS強制化と、ブラウザ自体への組み込みでのHSTS強制化の二種類がある（HSTS Preload List）。一般的に使われているとは言い難いが、EFFもSTARTTLS Everywhereプロジェクトにおいて、MTA-STS preload listを提供している。
[15] Googleによるアナウンス（2019年）
https://security.googleblog.com/2019/04/gmail-making-email-more-secure-with-mta.html

小話:
基本的に、本文書ではMTA間の経路暗号化を対象として説明している。エンドユーザ（MUA）とメール送信待ち受け用のSMTPサーバ間では、SMTP-AUTHというSMTP拡張の機能を使い、エンドユーザの認証を行う。Port 465（SMTPS）が用いられる場合、HTTPSのようにTLS Handshakeがすぐに行われ、TLSの使用が強制される。Port 587（STARTTLS）が用いられる場合、本章で説明した平文からのSTARTTLSが同じように使用される。しかしなら、Gmailなどを始めとしたアメリカの大手メールサービスでは、エンドユーザ-MTA間でPort 587を利用した場合でも、TLS暗号化を必須化しており、TLSセッションを貼らずにはメールを送信できないようになっている。もしMTA側でTLS強制化されていない場合も、基本的にはエンドユーザの用いるメーラー側でTLSの使用が強制されている（例: Thunderbird）。たとえば、中間者攻撃のリスクが非常に高い公衆Wifiを用いて、ダウングレード攻撃可能なMTAにエンドユーザが繋いだとしても、平文へのダウングレード攻撃に対して耐性がある。受信側のエンドユーザとメールボックス（POP3サーバ・IMAPサーバ）との通信についても同様の議論が成り立つ。

歴史的な経緯により、メールサービスによってはエンドユーザとやりとりするMTA側での平文ダウングレードなどを許可しているケースもあるが、エンドユーザ側で利便性を損なわずに自衛できるということもあり、本文書では省略する。



※ThunderbirdでのDowngrade攻撃検知時のエラー

Chapter 4: メール送信元認証が守るもの

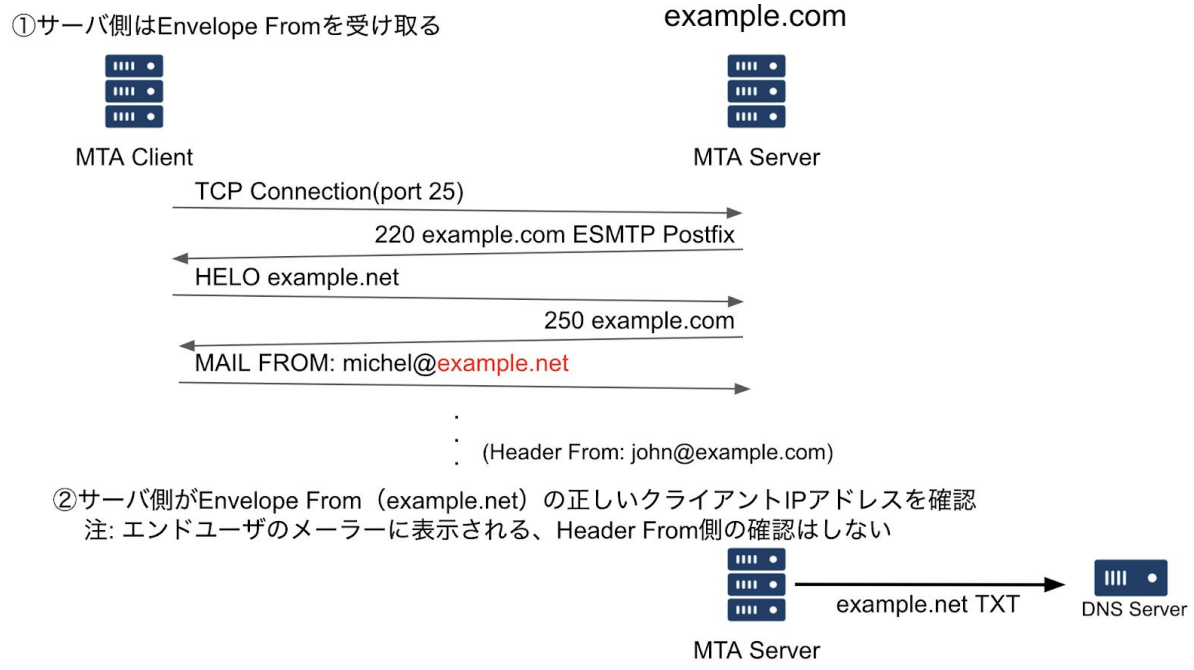
Chapter 4の要点

- 送信元メールアドレスを認証する場合、エンドユーザのメーラーに表示されるHeader Fromを確認する必要がある
- 2011年に策定されたDMARCは、メールのHeader From（送信元）ドメインをインターネット上のMTA間で認証する手段を提供する
    - メールの送信元メールアドレス全体をEnd-to-endで認証する手段ではない
- ただし2020年10月時点で送信元メールドメイン認証の強制化は普及しきっておらず、いまだに強制を望む送信者（MTA Client）は少数派である。
- 受け取ったメールの送信元が正しいと信用することは、いまだに非常に困難な状況である。

本章では、メールの送信元認証としてDMARCを紹介する。その前提として、SPFとDKIMについても説明する。本章は、「あなたが受け取ったメールの送信元は信用できますか？」と聞かれた際に答えになる情報をプロトコルの観点から説明している。

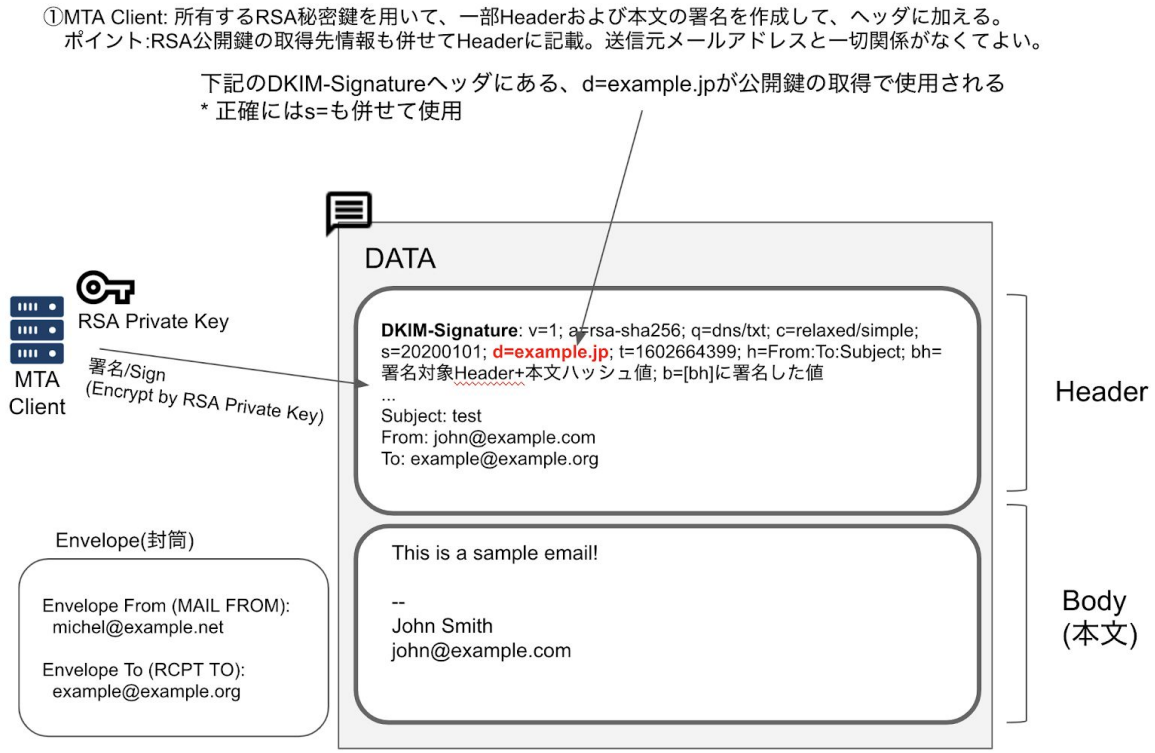
なお、Chapter 1で説明したように、メールにはEnvelopeとHeaderという概念がある。メール配送に使用されるのがEnvelope、エンドユーザのメーラーに表示されるのがHeaderである。
メールの送信元をエンドユーザが認証したい場合、正しいかどうかを確認したいのは、HeaderのFromに対してである。

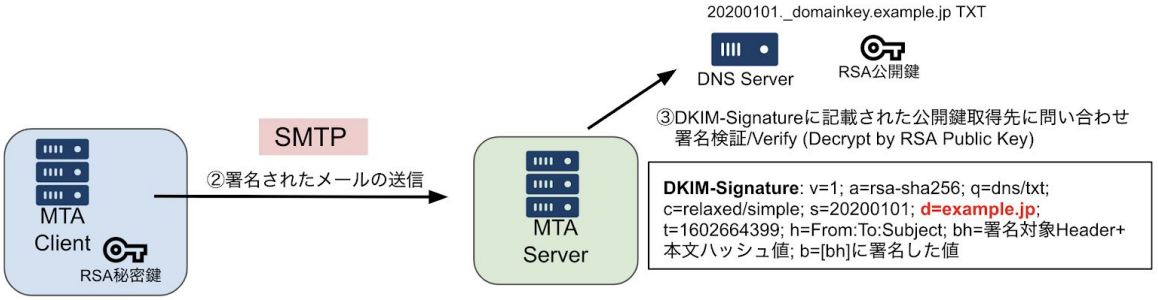
SPF: Envelope Fromのドメインを用いて送信元を確認



SPFは、Envelope Fromのドメインが適切なIPアドレスで送られてきていることを検証するプロトコルである[16]
単体ではHeader From認証のための技術ではない。
参考: @gmail.comのSPF設定例

DKIM: メールが改ざんされていないことを確認





DKIMは、MTA ClientからServerにメールが中継される過程で、メールヘッダ・本文が改ざんされていないことを署名により保証する。単体ではHeader From認証のための技術ではない。

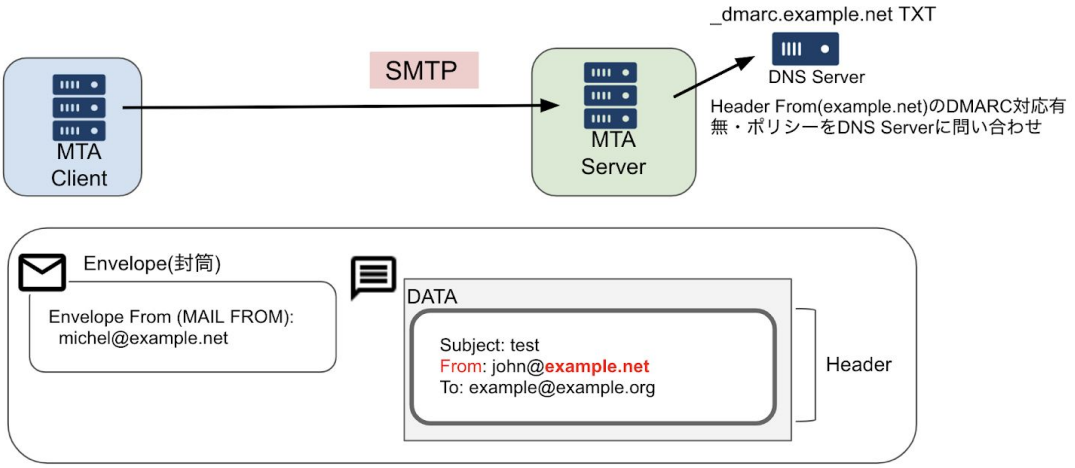
[16] SPFをHeader Fromに適用するメールサービスが一部あるが、一般的ではなくRFCにも沿っていないと認識している。Sender-IDというHeader Fromを検証するプロトコルも存在したが、普及していないため省略する

DMARC: Header Fromのドメイン認証

DMARCは、2011年にRFC標準化されたプロトコルである。MTA-STSと同様にGoogleやMicrosoft、Yahooやアメリカの大手ISPを中心に策定された。

- DMARCの設定

・DMARC: MTA Client側でDMARC設定・MTA Server側で送信元を検証
・MTA Serverは、Header Fromに記載された送信元ドメインに、DMARC対応有無を問い合わせる



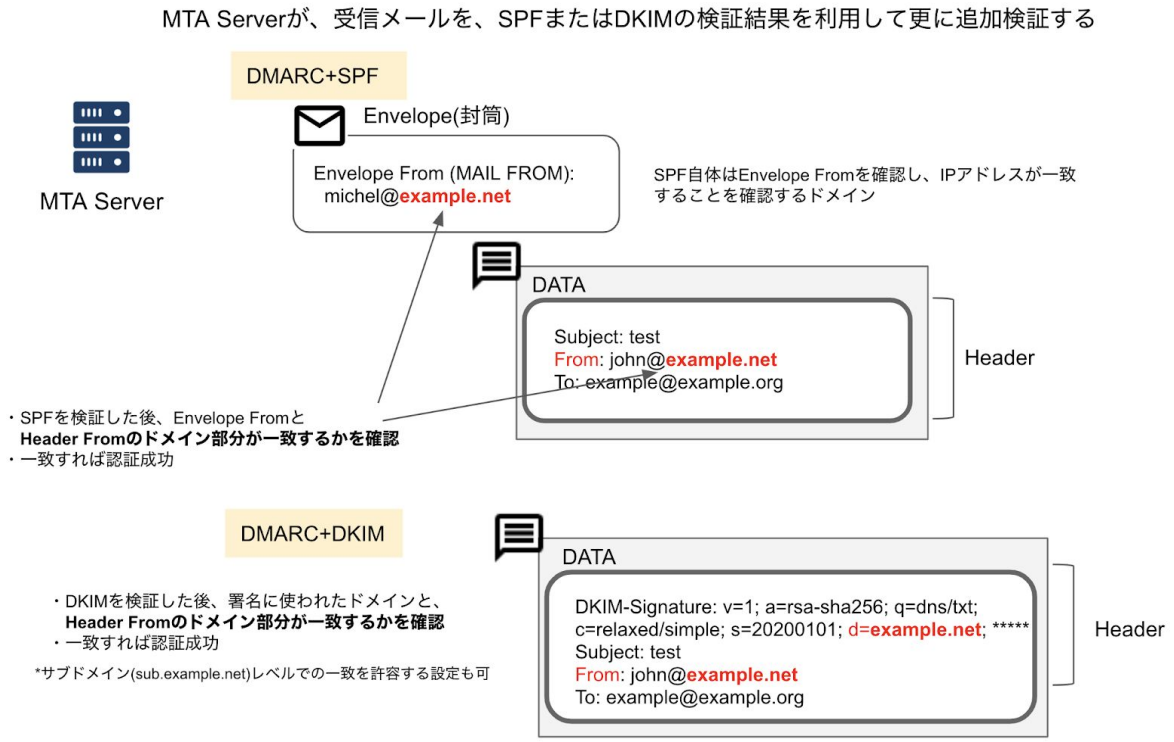
DMARC設定例 (gmail.comの例: _dmarc.gmail.com TXTレコード)

Gmailから送信されるメールについて、Gmail側が下記のポリシーで、受信側(MTA Server)でのDMARCの利用を求めている

"v=DMARC1; p=none; sp=quarantine; rua=mailto:mailauth-reports@google.com"
p=noneは、メール送信元認証に失敗しても、メールを拒否したりせずに配送する設定
rua: DMARC検証の失敗レポート送信先

p(policy)にはquarantine(隔離)やreject(拒否)も設定できるが、Gmailはnone(DMARCでのメール隔離・遮断なし)を設定している

- DMARCによるHeader Fromドメインの検証



上記の通り、DMARCはSPFとDKIMを利用し、Header Fromのドメイン部の認証を提供する。また、検証結果のレポートを送信することも出来る。

制約と普及率について

DMARC（遮断設定）の制約

- MTA Server側でDMARCのDNS名前解決を改ざん・遮断された場合、検証を行うことはできない
- DMARCはEnd-to-endでの送信元認証のための技術ではない。あくまでもインターネットに面したMTA間で送信元を検証する仕組みである。
- メールアドレス全体の認証ではなく、メールドメイン単位の検証である。同じメールボックスから、アカウント部（account@domain.exampleのaccountの部分）を偽装したメールが送られてきても、受信側（MTA Server）は正しいものとして処理することになる

DMARC（遮断設定）の普及率

アメリカを中心とするメールサービス業者のセキュリティ強化のための会合であるM3AAWG（Messaging, Malware and Mobile Anti-Abuse Working Group）は、メール認証のベストプラクティスを公開している[17]。そのベストプラクティスではDMARCによるメールのHeader Fromドメイン検証の失敗時ポリシーとして、reject（遮断）を行うことを推奨している。では、実際のメールサービスはどのように設定しているだろうか。
アメリカの大手サービスを例に出すと、2020年10月現在、Gmail（@gmail.com）はnone（メール配送の継続）、Microsoft（@outlook.com）はnone（メール配送の継続）、Yahoo

[17] M3AAWG Email Authentication Recommended Best Practices Table of Contents

US（@yahoo.com）はreject（遮断）、Apple（@icloud.com）はquarantine（隔離）に設定している。少しずつ、隔離や遮断設定が増えているものの、Gmailのポリシーはいまもnone（メール配信の継続）の状態である。

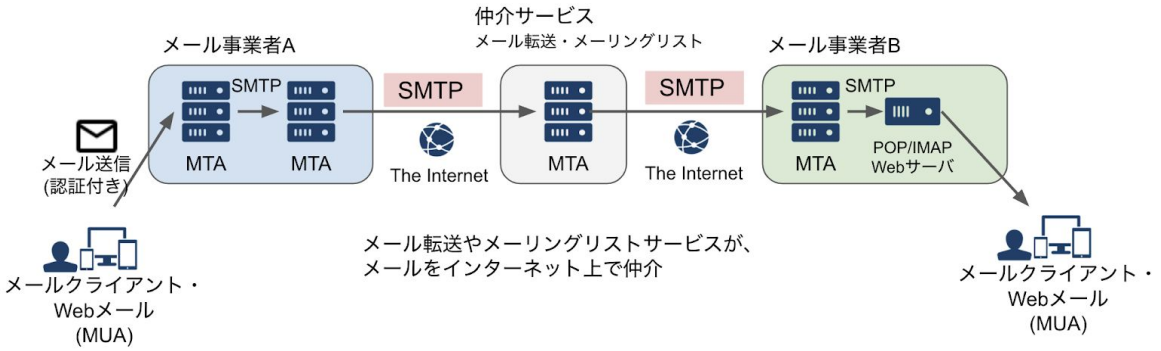
メール事業者の個人向けメールサービス以外を見ると、フィッシング対策業者であるValimailの”Email fraud landscape, Summer 2020”[18]レポートを確認する限り、2020年におけるビジネスメールでのDMARCの普及状況は大まかに下記の通りである。[19]

- Fortune 500企業において、約30%がDMARCにて遮断か隔離を選択。
- 大規模な銀行の36%がDMARCにて遮断か隔離を選択

DMARC設定は、メールを送信する側から受信側への「お願い」である。もちろん送信側（MTA Client）側での設定が普及しても、受信側（MTA Server）でDMARC設定を確認しなければ、送信元の検証は行われない。

DMARC（遮断設定）が普及しない原因

- パターンB （仲介サービス経由）



DMARCは、MTA ClientとMTA Serverの直接通信であれば、適切に動作する。しかしながら、Chapter 2で紹介したパターンB（仲介サービス経由）の場合、適切に動作しないケースが増える。送信側（MTA Client）でDMARCを設定しても、仲介サービスが間に入ると、受信側（MTA Server）から見た送信元IPアドレスは変わるので、SPFの検証に失敗する。また、たとえばメーリングリストでは、件名（Subject）にメーリングリスト名を入れるなどして件名の変更を行うこともある。その場合、DKIMの検証にも失敗することになってしまう。両方の検証に失敗した場合、DMARCの検証としては失敗という扱いになる。
もし、p（Policy）としてreject（遮断）などをMTA Client側で設定していた場合、MTA Server側ではメールを遮断することになってしまう。これがDMARCが遮断設定で大きく普及しない原因であると言える。

今後の展望

DMARCの問題を解決する解決案として、現在、M3AAWGにおいてARCというプロトコルが議論され[20]、IETFではARCのRFC標準化が議論されているようだ。このプロトコルは、各

---

[18] Email fraud landscape, Summer 2020
[19] 250okなどもDMARCの普及状況のレポートを公開している ThreatList: DMARC Adoption Nonexistent at 80% of Orgs
[20] Global Mailbox Providers Deploying DMARC to Protect Users M3AAWGの議事録や議論はIETFと違い非公開であるが、DMARCプロトコルのために用意されたWebサイトには議論が行われた旨が記載されている。

MTA間通信に署名を加えることで、最終的にインターネット上でメール通信を中継するMTAの各ステップを検証可能にするというものであり、これを用いると仲介サービスの問題は解決する。まだ標準化には至ってないが、Googleなどは実証実験を始めており、Gmailで受け取ったメールヘッダなどを確認するとARCの文字を確認することが出来る。

Chapter 5: End-to-end暗号化のための仕様（PGPとS/MIME）

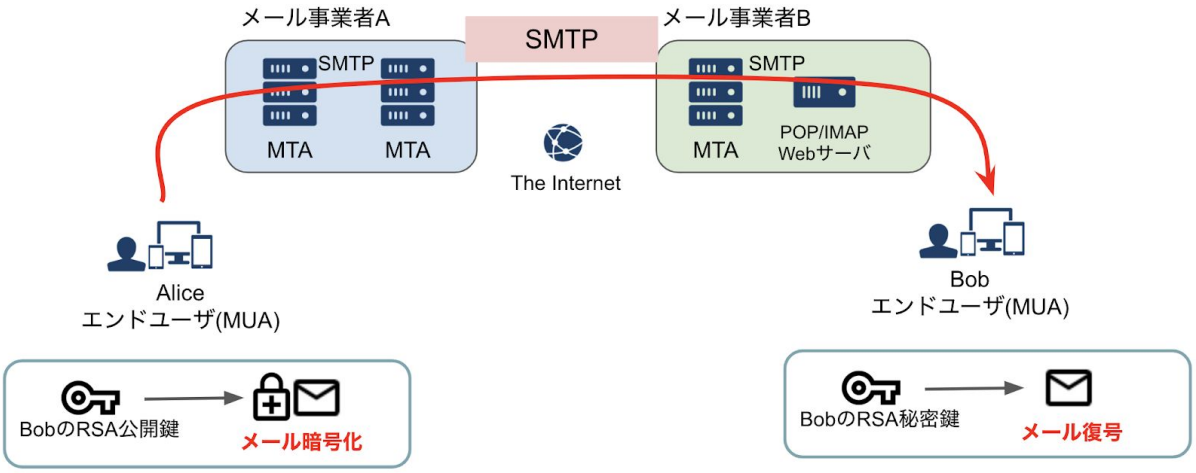
Chapter 5の要点

- PGPおよびS/MIMEは、End-to-end暗号化もしくは署名を提供する
- 暗号化の制約
    - End-to-end暗号化での暗号化範囲は本文（Body）である。メール配送のために必要なEnvelopeおよびメールヘッダ（Date, From, To, 件名など）は暗号化されていない。つまり、「誰が誰に、どういう件名のメールをいつ送ったのか」という情報は暗号化されない。各MTAやメールを保管するメールボックスサーバ上では、それらの情報は平文で取り扱われる。もちろん、各サーバでメール配送のログやヘッダ情報を保管することが出来るし、Chapter 3で説明したように、SMTP Over TLS（STARTTLS）による経路暗号化が行われている保証もない。
    - その他、PFSと呼ばれる特性がないなど、暗号化には様々な制約がある。
- S/MIMEにおける署名は利用しているメール送信者がいるものの、暗号化という観点では、PGPおよびS/MIMEは利便性が悪く一般的に普及しておらず、一部のメッセージを秘匿しなければならない人々にのみ利用されている
- 大きな脆弱性が2018年に見つかったこともあり、WhatsAppやSignalといった、利便性も高くPGPよりもセキュアなEnd-to-end暗号化を提供するメッセージングアプリが普及している今、PGPを利用するケースは非常にまれと言える

本章で紹介するPGPおよびS/MIMEの歴史は非常に長い。PGPは、90年代始めに開発された、ファイルやメールの内容を暗号化したり署名することの出来るEnd-to-endでの暗号化プロトコルである。開発当時のアメリカには暗号技術の輸出規制があり、アメリカ国外での配布が許されていなかった。しかし90年代後半、書籍であれば輸出規制の対象にならないことを逆手に取って、ソースコードを全て印刷した書籍を出版することで、PGPを合法的にアメリカ国外に配ることが可能になった。そういった時代から今日に至るまで、PGPは使用されている。たとえば2013年、エドワード・スノーデンはジャーナリストとの連絡においてPGP暗号化したメールを用いて連絡を取り合っていた。
その他、2015年に匿名で南ドイツ新聞社に提供された「パナマ文書」の分析には、日本からは朝日新聞社のジャーナリストが参加した。パナマ文書などに関する記事を朝日新聞社のWebページなどで確認すると[21]、記者がPGP公開鍵を記載している例が見受けられる。
前章までで紹介したように、Emailを支えるSMTP Over TLSなどの仕様は経路暗号化であり、各メールサービスなどは平文のメールを取り出せる。PGP、S/MIMEを用いたEnd-to-end暗号化は、機密情報のやり取りにおいてジャーナリスト自身の身を守る手段としても利用されている。

[21] 「パナマ文書」で世界に衝撃を与えたＩＣＩＪと朝日新聞はなぜ提携したか

- PGPおよびS/MIMEの概要図

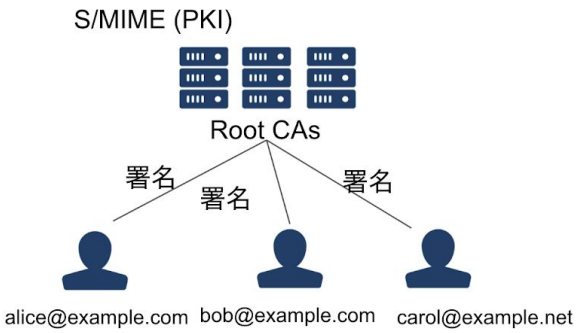


End To End暗号化: 暗号化が、Aliceの手元のメーラー上で行われ、
復号はBobの手元のメーラー上で行われる

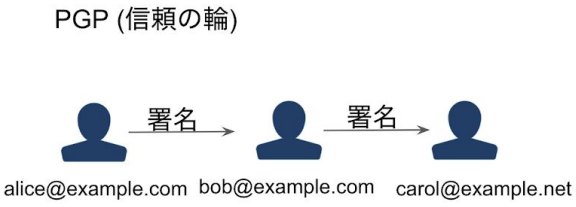
注意点としては、メールの本文は暗号化されるものの、メールの配送に必要なEnvelope From/Toおよび、Header情報（From, To, 件名など）は暗号化されない。各MTAでメール配送（Envelope From/To）のログやヘッダ情報を保管することが出来るし、Chapter 3で説明したように、SMTP Over TLS（STARTTLS）による経路暗号化が行われている保証もないため、第三者に配送情報・ヘッダ情報を盗聴される可能性がある。

信頼モデルの違い

PGPとS/MIMEの大きな差として、信頼モデルの違いがある。S/MIMEは、現在HTTPSで広く用いられているような、PKI（公開鍵基盤）を利用しCAによるルート証明書を辿っていく形の信頼モデルである。それに対しPGPは、知り合いの知り合いは信用するという、”信頼の輪”モデルでの信頼モデルである。



Root CAを信頼する。
その上で、Root CAが信頼できる各メールアドレスの持ち主の公開鍵に、署名を行う

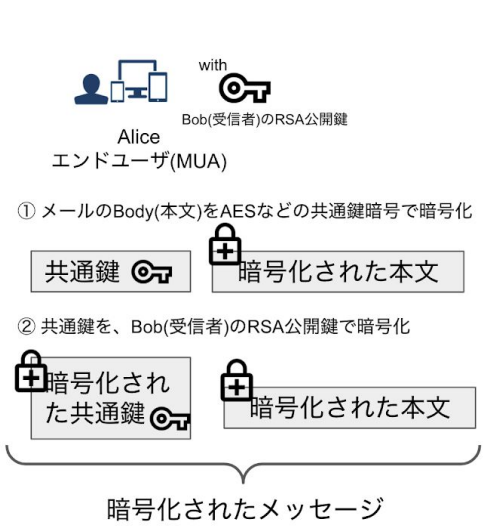


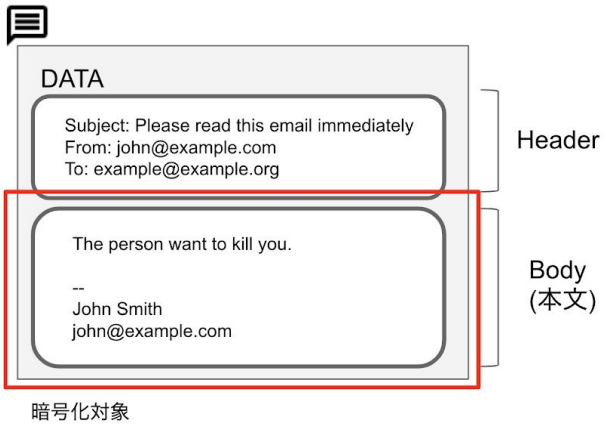
個人間で信頼モデルを構築する。
上記の例だと、真ん中のbobはcarolを信頼している。
そのとき、bobを信頼しているaliceは、carolも信頼することになる。（信頼の輪と呼ぶ）

署名・暗号化

S/MIMEもPGPは、いわゆる公開鍵暗号（RSAなど）と共通鍵暗号（AESなど）を組み合わせて署名・鍵交換・暗号化を行うプロトコルである。
以下の説明では、公開鍵暗号としてはRSAを用いる前提で説明する。

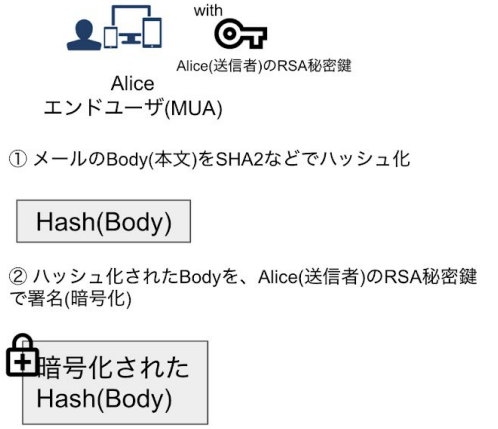
- 暗号化の流れ

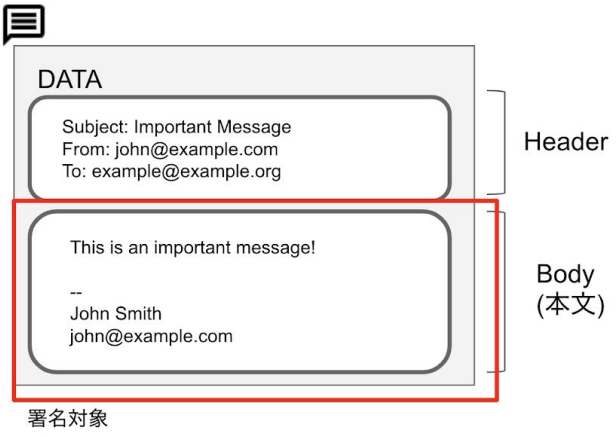




PGPを用いる送信者は、共通鍵暗号で本文を暗号化した後、受信者（Bob）のRSA公開鍵を用いて暗号化を行う。暗号化されたメッセージを復号できるのは、RSA秘密鍵を持っている受信者（Bob）のみである。

- 署名の流れ





署名に用いる事のできる公開鍵暗号（RSAやDSAなど）で、送信者（Alice）の秘密鍵を用いて署名を行う。署名できるのは、秘密鍵を持っているAliceだけである。上記の図の例では、受信者（Bob）は、AliceのRSA公開鍵を用いることで検証が出来る。

制約

End-to-end暗号化を提供するS/MIMEおよびPGPだが、多くの制約が存在する。

- 暗号化の範囲
  - 暗号化の範囲は本文（Body）である。メール配送のために必要なEnvelopeおよびメールヘッダ（From, To, 件名など）は暗号化されていない。つまり、「誰が誰に、どういう件名のメールをいつ送ったのか」という情報は暗号化されない。各MTAやメールを保管するメールボックスサーバ上ではそれらの情報は平文で取り扱われる。もちろん、メール配送のログやヘッダ情報を保管することが出来るし、Chapter 3で説明したように、SMTP Over TLS（STARTTLS）による経路暗号化が行われている保証もなく、盗聴のリスクがある。
  - Privacyの観点からは、ヘッダや送信先が暗号化されていないことは、非常にリスクである。そもそも諜報活動などでも重要になるのは本文より送信元IPやヘッダなどのメタデータという議論があるが[22]、私生活においても、「誰が、誰に、いつ、どういった件名のメッセージを送ったか」が流出することは、大きなPrivacy上の問題となる。具体例として、EFFがWhy Metadata Mattersという文書を公開している。
- 認証手段としての弱さ
  - CAによるPKIを用いてのS/MIMEはともかく、PGPによる信頼の輪は認証手段としては非常に弱い。
- 利便性
  - PGPについてもS/MIMEについても、利用のハードルが非常に高い。2020年になってThunderbirdにデフォルトで機能が組み込まれることになった[23]が、以前はエンドユーザ端末へのOpenPGP（GnuPG）のインストールなども必要で、非常に利用のハードルが高かった。また、利用環境が整っていても、暗号化のためにはまず送信相手の公開鍵を入手する必要もある。
- PFS （Perfect Forward Secrecy/前方秘匿性）がない
  - 特にスノーデン事件以後に有名になった概念にPFSがある。
  - PGPやS/MIMEのように、RSA公開鍵などを用いて共通鍵暗号の鍵交換を行う場合、一度秘密鍵を盗られてしまうと、その秘密鍵を用いて全てのメッセージの復号が可能になる。
  - たとえば2018年に標準化されたTLS1.3などでは、基本的に鍵交換についてはECDHE（楕円曲線Diffie-Hellman key exchange）のみ許可している。この場合は相手のRSAなどの公開鍵を用いて鍵の交換を行うわけではないため、もし秘密鍵を盗まれても、過去の暗号化された通信内容を復号することが出来ない。
- 鍵の更新
  - PGPおよびS/MIMEのための鍵ペアは、定期的に交換することが望ましい。その際は再度、公開鍵を暗号化メールを使いたい人々に配り直さなければならない。
- 普及率
  - PGP・S/MIMEともに、暗号化する方法として、ほとんど普及していない。一部のジャーナリストや、セキュリティ関係者などの通信を守る必要のある人たちのみに使用されているような状況である。

---

[22] 土屋 大洋『サイバーセキュリティと国際政治』P48（千倉書房，2015年）
[23] OpenPGP in Thunderbird 78

- ただし、署名する方法としては、S/MIMEはある程度使われている。Chapter 4で見たように、送信元を認証する手段としてDMARCなどが普及していない状況で、S/MIMEは送信元を証明する方法として有効である。
- たとえば、日本でもメガバンクなどからS/MIMEで署名されたメールが送信されている。
  - 注意点としては、もしS/MIMEで署名が行われていても、同じメールアドレスから、署名が行われていないメールを受信することも可能であることが挙げられる
  - 署名が行われていることは、正しい送信元であることを証明する
  - 署名が行われていない場合、正しい送信元であるかもしれないし、メール送信元詐称の行われているメールかもしれない。

便利なEnd-to-end暗号化への移行

2016年、著名な暗号関係のソフトウェアエンジニアであるFilippo Valsorda（現在、Go言語の暗号ライブラリのメンテナ）は、”I'M GIVING UP ON PGP”というブログを投稿した。そのブログでFilippoは、よりよいEndpointでの暗号化手段であるSignalかWhatsAppを使う旨を述べている。また、暗号学者として代表的な存在であるBruce Schneierも、Filippoのブログへの応答として、PGPではなくSignalを使う旨を述べた。[24]
スノーデン事件以降、End-to-end暗号化の要求が高まっていた一方、PGPという手段は少しずつ使用されなくなった。アメリカでは2016年にWhatsAppやFacebook Messengerといった代表的な個人間メッセージングアプリがEnd-to-end暗号化をサポートし、日本でも2015年にLINEによるEnd-to-end暗号化がスタートし、PGPを使うメリットが減ったのである。[25]

2018年: 脆弱性「EFAIL」

2018年、EFAILというS/MIMEおよびPGP暗号化されたメールへの攻撃手法が見つかった。攻撃手法としては、盗聴などで盗んだ暗号化メールに細工を加え、再度、受信者に送信する。その受信者が自身の持つ秘密鍵でメールの復号を行ったとき、攻撃者のサーバに平文データが送られるという攻撃である。

技術的な詳細については述べないが、PGPやS/MIMEで暗号化するテキストの最初には必ず”Content-Type: multipart/signed...”などの固定文字が含まれている。復号したあとの平文部分が分かっていれば、暗号化に共通鍵暗号（ブロック暗号）のCBCモードが利用されている場合、任意の平文の文字を暗号化された文に付与することができる。例えば最初に、「<img src=attacker.example.com/」というメッセージを差し込んだとする。そうすると、HTMLメールを解釈したメールクライアントは、「attacker.example.com/平文メッセージ」というURLにアクセスを行う。こうして、攻撃者は平文メッセージを取得することができる。

これは、PGPやS/MIMEというプロトコルそのものの信頼性を大きく下げることになる脆弱性だった[26]。そして、現代ではそもそもメッセージングアプリが広く普及しており、加えてメッセージングアプリでのEnd-to-end暗号化が提供されていることから、PGPやS/MIMEは暗号化手段としてはほとんど使われなくなっている。

---

[24] The Pro-PGP Position
[25] データセキュリティ （LINE）
[26] We're calling it: PGP is dead

Chapter 6: LINE - メール以外のメッセージングアプリの普及とEnd-to-end暗号化の始まり

Chapter 6の要点

- 国家によるSIGINT・サーベイランスプログラムなどによる通信の監視が国際的な問題となっていた中、LINE社は2015年にEnd-to-end暗号化を開始した
- LINEのメールとのメッセージ配信モデルの違いとして、主に下記がある
    - 経路暗号化の必須化
        - インターネット上の盗聴に対する耐性がある。
    - 送信元・送信先の保証
        - メールのような分散型ではなく、中央サーバと通信する形式であり、中央サーバによりメッセージの送信元・送信先は保証されている
    - デフォルトでEnd-to-end暗号化が有効になっている
- LINEのEnd-to-end暗号化仕様であるLetter Sealingの仕様の特徴
    - 相手がオフラインであっても、相手との鍵交換が可能
    - 固定の長期秘密鍵を利用するためPFS（前方秘匿性）はない
    - 中央サーバが信用できない場合も手動での相手の認証が可能
    - 送信者ID, 受信者IDなどのMetadataは、メッセージ配送に用いられるため、プロトコル上、End-to-end暗号化されない
- Metadata
    - LINEの情報利用規約に同意した場合は、LINE社はメッセージの送信相手、日時、遷移したアクセスURL情報などはLINE社サーバにて保存され、電気通信事業法（いわゆる「通信の秘密」）を遵守した上で、不正防止対策などの同意の範囲で利用するとしている
    - 警察機関などへの第三者提供の件数や内容は透明性レポートにて公開されている

メールまとめ

今までの章で、メールの状況をみてきた。2020年10月時点のメールのセキュリティは、重要な点を上げると、下記の状況である。

- 中身が平文のメールが、複数のメールサーバを中継して配信される
- 経路暗号化については、インターネット上の通信も日和見暗号化（普及率約90%）
- 送信元メールアドレスは信頼できない
- PGPなどを用いたEnd-to-end暗号化はほぼ利用されていない

Messaging App

スマートフォンの普及と共に、世界中で個人間のコミュニケーション手段として、メッセージングアプリの利用が急速に広まった。欧米諸国では、主にWhatsAppやFacebook Messengerの利用が急速に広まった。日本においては、LINEが主に利用されている。これらのサービスは、現状、個人間メッセージングにおけるコミュニケーションインフラとも言える状況になっている。

特に2013年以降、スノーデン事件を始めとして、各国で、国家によるSIGINT・サーベイランスプログラムによる通信の監視が、国際的なPrivacy上の問題となっていた。エンドユーザー間で暗号化を行う、End-to-end暗号化の実施は、原理的に各サービス運営者によるメッセージの内容確認を不可能にするものであり、エンドユーザからはPrivacyの観点、サービス運営企業からすると、多くの報道が出る中での企業ブランド保護戦略の観点からも、実装の必要性が高まっていた。その中、2015年にLINEは、Letter Sealingという独自のプロトコルを利用したEnd-to-end暗号化を開始した[27][28]。
本章では、LINEの暗号化仕様を説明する。なお次章では、2016年にEnd-to-end暗号化を実施したWhatsAppなどで利用されているSignal Protocolを説明する。

LINE: Letter Sealing

LINEはメールのような分散モデルではなく、メッセージを処理する中央サーバと通信を行う形でメッセージ交換を行う。
LINEのメールとのメッセージ配信モデルの主な違いとしては、主に下記が挙げられる。

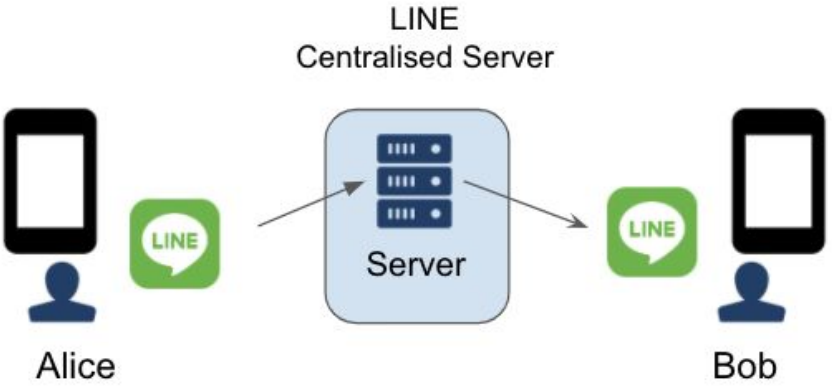
- 経路暗号化の必須化
    - 経路暗号化の必須化されていないメールと違い、各エンドユーザ - LINE社サーバ間の経路暗号化は必須化されている。そのため、インターネット上の盗聴に対する耐性がある。
- 送信元・送信先の保証
    - メールでは送信元メールアドレスを信頼できる状況にないが、LINEの場合、LINE社の中央サーバでのメッセージ配送となるので、LINE社により正しく配送される。
      ※後述するように、もしLINE社が信用できないという前提に立つ場合も、LINE以外の手段を使って、通信相手をエンドユーザ側で認証する手段が提供されている
- デフォルトのEnd-to-end暗号化
    - 利便性の観点もあり、ほとんど利用されていないメールのPGP暗号化などとは違い、デフォルトでEnd-to-end暗号化が実施されている

※本章では、LINEにおける個人間メッセージングを対象として説明している。LINEは企業アカウントなどとのメッセージ交換もできるが、個人間メッセージング以外の機能は対象としていない。

Letter Sealingは、LINE独自のEnd-to-end暗号化プロトコルであり、鍵交換およびメッセージの暗号化を行うことができる。エンドユーザ端末とLINE社との通信は経路暗号化した上で、エンドユーザ間ではデフォルトでEnd-to-end暗号化が行われる仕様となっている。ソー

[27] 参考: 【LINE】LINE Introduces Letter Sealing Feature for Advanced Security 参考: メッセージの安全性新時代：Letter Sealing - LINE ENGINEERING
[28] LINEのEnd-to-end暗号化の採用には、スノーデン事件だけでなく、2014年に、韓国政府機関がLINE社の通信を傍受しているという報道記事が出たことも強く影響していると考えられる。参考:「本日報道の一部記事について」（当時LINE社の社長であった森川亮氏によるブログ記事、なお韓国政府がLINE社を監視していたという確からしい根拠は、研究者や他の報道機関からは出ていない）。その他、タイ政府による会話の監視の報道（LINE社は技術的に不可能と否定）なども2013年から2014年にかけて報道されていた。

スコードは公開されていないが、Whitepaperにおいてプロトコルの仕様が公開されている。本文書では2020年10月現在の最新バージョンである2.0を取り扱う[29]。



経路暗号化は、TLSもしくはTLSベースの独自方式を用いる

※End-to-end暗号化されたメッセージは、LINE社の中央サーバでの復号はできない
※経路暗号化の詳細は脚注に記載している[30]

Letter Sealingの特徴

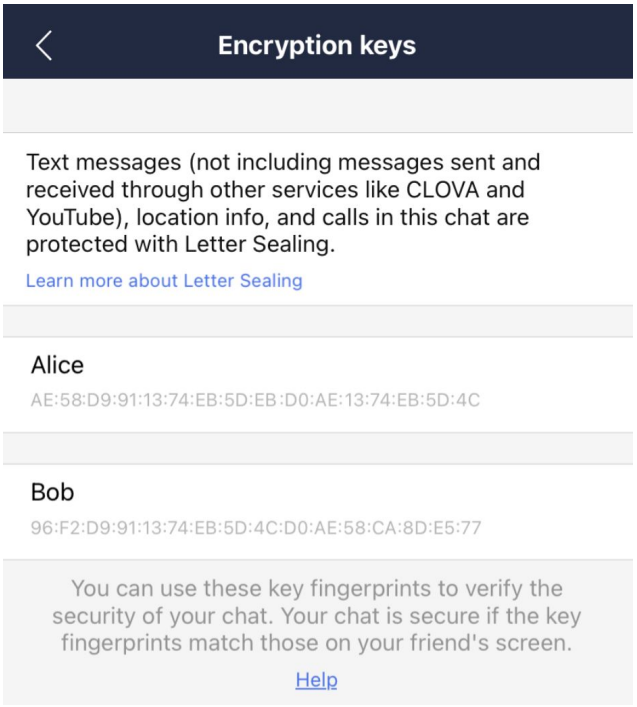
- メッセージを送信したい相手がオフラインであっても、相手との鍵交換が可能
- PFS（前方秘匿性）はない
    - 固定の長期間利用する秘密鍵を利用し暗号化を行うため、端末から秘密鍵が漏れた際は、中央サーバにおいて通信を保存した場合、過去のメッセージ内容も復号可能になる。
- 暗号化の認証手段の提供
    - メッセージの伝送を行う中央サーバが信用できないと仮定した場合も、メッセージの送信元や送信先が本当にやりとりしたい相手なのか、オフラインなどでクライアント同士の公開鍵のFingerprintsを比較することでお互いを認証することが出来る。

---

[29] 研究者により、バージョン1の暗号プロトコルに対する指摘があり、対策の行われたバージョン2が2019年10月にリリースされた(参考: LINE Transparency Report、Breaking Message Integrity of an End-to-End Encryption Scheme of LINE)

[30] 筆者の認識している限り、経路暗号化は、通常のTLSおよび、Whitepaperにて紹介されている”LEGY Encryption FS”という、TLS1.3の0-RTT通信をベースにした独自方式を併用している。なおLEGY（LINE Event Gateway）とはLINE社のTLS終端を担うGatewayの名称である。

LEGY Encryption FSの概要: プロトコルとしてはTLS1.3の0-RTT通信をベースにしている。LINEは、アプリ内に保存されているサーバ（LEGY）の公開鍵を利用することで、TLSハンドシェイクの開始と同時にアプリケーションデータの暗号化通信を始められる（0-RTTでの暗号化）。接続の1往復目はサーバ側が固定鍵で暗号化通信を行うため、PFS（前方秘匿性）がないが、1往復目にメッセージ送信と併せて新たに鍵交換を行うため、2往復目からはPFSがある。
(参考: LINE Transparency Report、LINE Encryption Overview: Technical Whitepaper、SECURITY x LINE PLATFORM: LINE Platform Security)



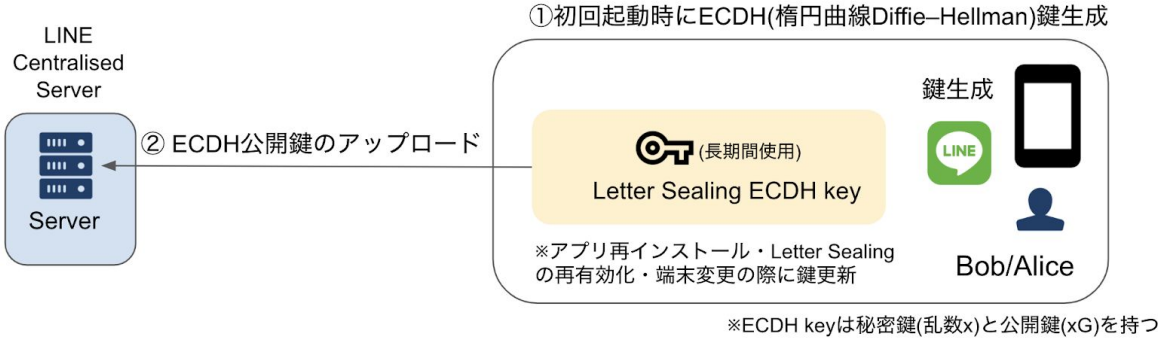
    - 上記画面にはそれぞれの公開鍵のFingerprintが表示されており、送信者・受信者は、例えば対面などのLINE以外の手段を使って、通信している相手が正しいか確認することが出来る
- 暗号化範囲
    - Letter Sealingのプロトコル仕様としては、メッセージの内容はEnd-to-end暗号化され、送信者ID、送信先IDなどのメッセージ配送に利用する情報はEnd-to-end暗号化されない
    - 具体的なLINE社の情報取得範囲については、LINE 暗号化状況レポートにて詳細が公開されている

Letter Sealingのプロトコル概要

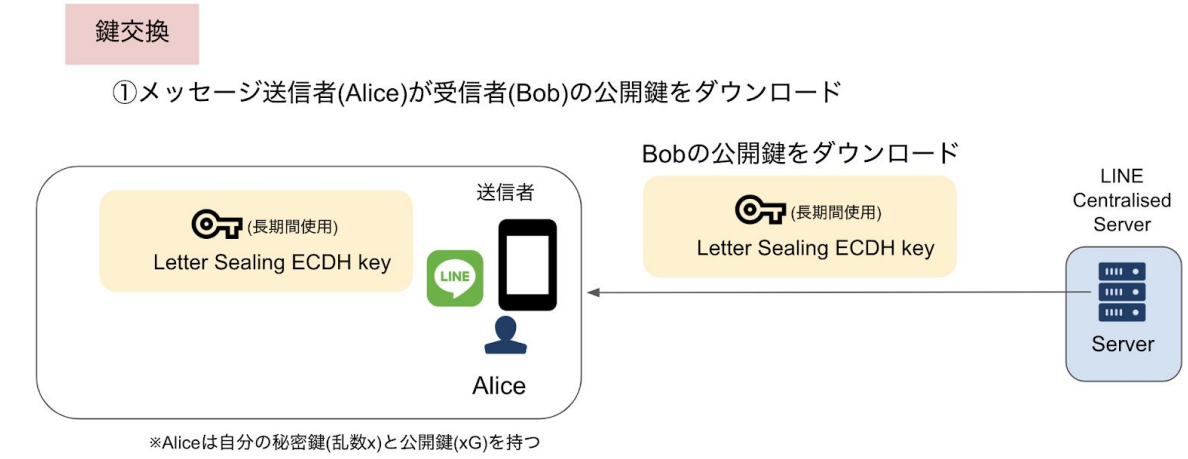
End-to-end暗号化を行うために、各エンドユーザはまず鍵生成を行う必要がある。
※下記はグループでのメッセージング（1:N通信）ではなく、1対1でのメッセージング（1:1通信）の例である

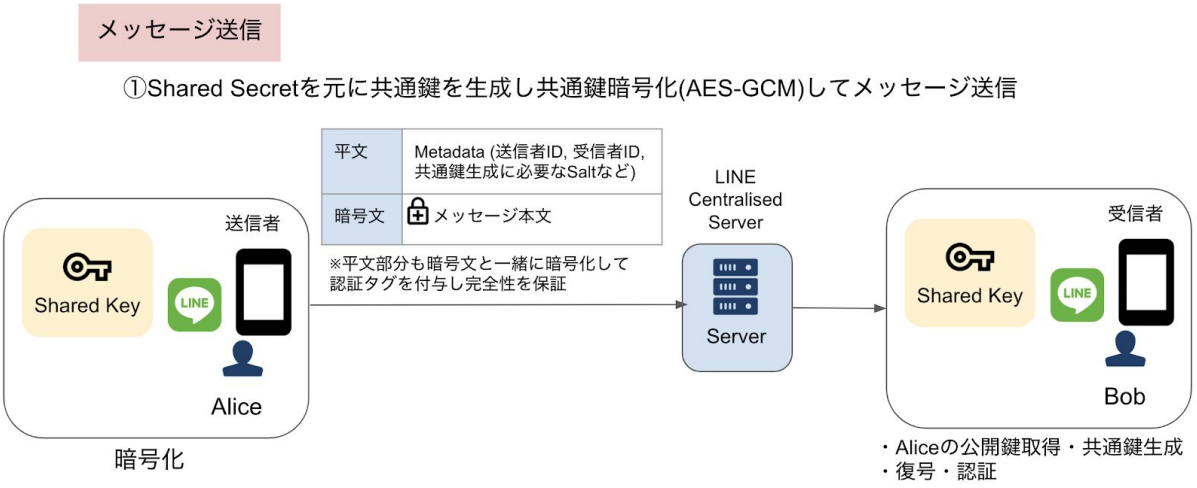
事前準備

ユーザはLINEの初回起動時に鍵ペアを生成し、公開鍵をLINE社サーバに登録



※まずは事前準備として、LINE初回起動時に公開鍵をLINE社のサーバに登録する





上記の図の通り、End-to-end暗号化されている部分はメッセージの本文で、送信者ID、受信者IDなどは平文で送信され、LINE社によるメッセージ配送に利用される。

Metadataの取り扱い

上記で説明した通り、メッセージ本文は暗号化されるものの、送信者IDや受信者IDなどのMetadataは、メッセージ配送に用いるプロトコルの仕様上、End-to-end暗号化されない。なお、プライバシーポリシーでは、メッセージ内容については基本的にEnd-to-end暗号化されるものの、LINEの情報利用規約に同意した場合、LINE社はメッセージの送信相手、日時、遷移したアクセスURL情報などをLINE社サーバにて保存し、電気通信事業法（いわゆる「通信の秘密」）を遵守した上で、不正防止対策などに同意の範囲で利用すると明記している[31]。実際、LINE社は捜査機関への情報提供件数・理由を透明性レポートにて公開してい

[31] LINEの情報利用に関するご案内、LINE プライバシーポリシー

る。2019年7〜12月の捜査機関への情報開示について、日本では児童被害（36%）が理由の開示請求が最多となっている。End-to-end暗号化は国家安全保障文脈でのサーベイランスに対する対策として語られることも多い一方で、一般的な犯罪の解決などにも関わる技術でもある。既にコミュニケーションインフラとなったLINEがMetadataなどを保存していなかった場合、解決に至らなかった犯罪事例もあると想定できる。その一方で、もちろんMetadataは個人のPrivacyと切り離せない要素であり、取り扱いには慎重になるべきである[32]。そこで次章では、欧米を中心に使用されているWhatsAppおよび、可能な限りPrivacyを保護する方針で開発されているアプリであるSignalの事例を主に説明する。

[32] Why Metadata Matters

Chapter 7: WhatsAppとSignal - 世界的なE2E暗号化の普及

Chapter 7の要点

- 2016年、アメリカにおいて主に利用されているメッセージングアプリであるWhatsApp（Facebook傘下）はSignal Protocolという暗号化プロトコルを採用したデフォルトEnd-to-end暗号化を開始した
- Signal ProtocolによるEnd-to-end暗号化には、下記の特徴がある
    - メールのような分散型ではなく、中央サーバと通信する形式であり、中央サーバさえ信用できれば、メッセージの送信元・送信先は保証されている
    - 中央サーバを信用しない場合でも、手動での認証を提供する
    - 暗号鍵はPGPと違い、メッセージ送信先ごとに違う鍵が使用される。秘密鍵から全ての過去のメッセージを復号できる状態ではなく、前方秘匿性（PFS）がある
    - また、過去だけでなく、将来のメッセージも読めないようになっている。（Post-Compromise Security）
- 同じSignal ProtocolでEnd-to-end暗号化をしていると言っても、各社でMetadataの収集については方針が異なっている。Signalは可能な限り全てのMetadataをEnd-to-endで暗号化する方針である一方、Facebook社のWhatsAppは、中央サーバで一定の情報を収集する。
- Facebook社は、会社全体での政府機関への情報提供などについて、透明性レポートを公開している

WhatsAppおよびSignal

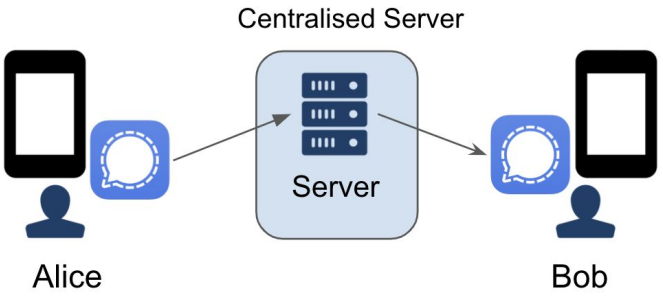
北米諸国やヨーロッパ諸国で主に利用されているメッセージングアプリとしては、WhatsApp（Facebook社傘下）やFacebook Messengerを挙げることができる。
2013年にスノーデンが告発したNSAのサーベイランスプログラム「PRISM」において、協力していると名指しで記載された企業の一つにFacebookがある。こういった、いわゆるTech Giantsと呼ばれる大企業への不信感が高まる中、アメリカで主に使われているメッセージングアプリであるWhatsApp（Facebook社傘下）、Facebook Messengerは2016年より、End-to-end暗号化の提供を始めた[33][34]。WhatsAppではデフォルトで有効化されている。このEnd-to-end暗号化には、Signal Protocolと呼ばれるプロトコル群が利用されている。これは、エドワードスノーデンなども推奨しているSignalという、Privacyを可能な限り重視しているアプリにおいて利用されているEnd-to-end暗号化プロトコルである。このプロトコルには鍵交換やメッセージ暗号化の仕様が含まれており、ライブラリがGitHub上で配布されている。
※Signalという用語の使われ方には、Privacyを最大限配慮したアプリの名前である”Signal”と、そのアプリ開発元によって作成された”Signal Protocol”というEnd-to-end暗号化のプロトコル名がある。”Signal Protocol”は、今までいわゆるTech Giantに採用されてきた。

[33] End-to-end encryption （WhatsApp）
[34] Facebook MessengerはInstagramのDirect Message機能との統合も発表されている

Signal Protocolを採用している主なアプリ[35]				
Signal ※アプリ名	WhatsApp (Facebook)	Facebook Messenger (Facebook)	Skype (Microsoft)	Google Allo (Google) ※サービス終了

本章では、Signal Protocolの構造を読み解き、セキュリティ上の特性を紹介する。

Signalプロトコルの仕様



経路はTLSにて暗号化

Signal Protocolは、各社のメッセージングアプリ利用できるEnd-to-end暗号化プロトコル群である。LINEと同様、一社の管理する中央サーバがいることが前提である。End-to-end暗号化であるため、中央サーバはやりとりするメッセージの復号はできない。
なお、Signal Protocolのライブラリはオープンソースとして公開されているが、SMTPを中心としたメールのセキュリティ強化や置き換えを意図したプロトコルではない。多様なステークホルダーが存在することによりセキュリティ強化の進みづらいメールを考えると、各社独自のメッセージングアプリ向けにプロトコルとライブラリをオープンソースで提供しているSignalの方向性は、インターネット全体の個人間コミュニケーションのセキュリティ強化という観点では非常に有効に働いていると言えるだろう。

Signal Protocolでは、メッセージのやりとりを行うために、まず中央サーバを介した暗号化のための鍵交換からスタートする。

X3DH: 鍵交換プロトコル

Signal Protocolにおいて、鍵交換にはX3DH（Extended Triple Diffie-Hellman）というプロトコルを用いる。主に、下記の特徴がある。

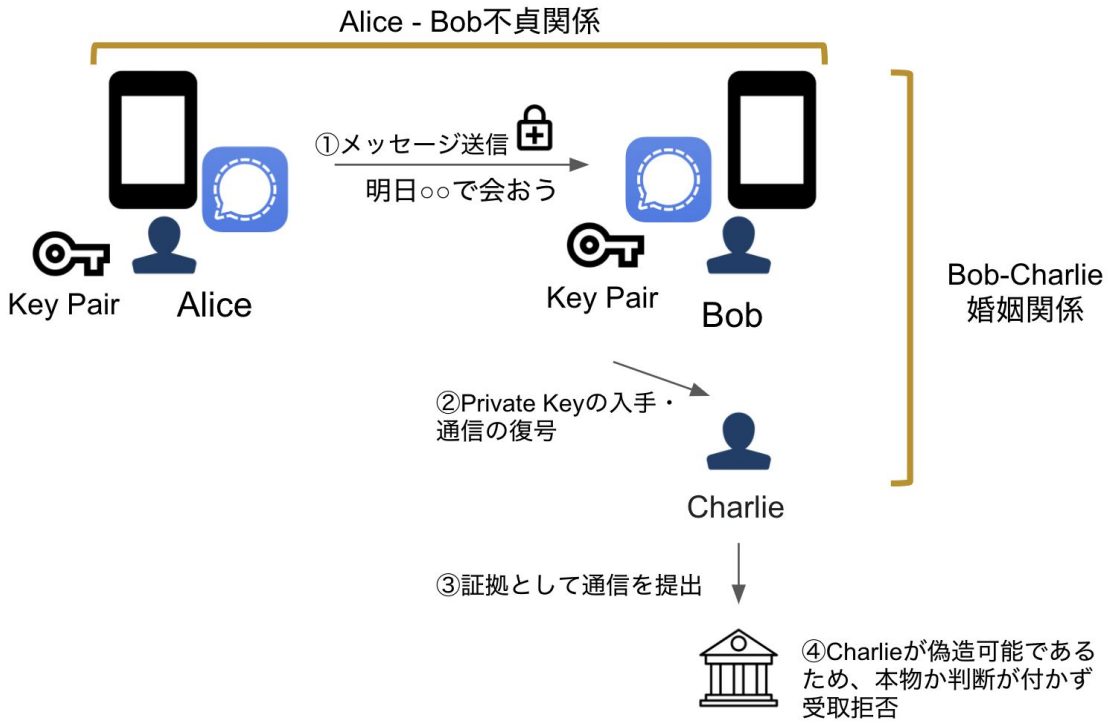
X3DHの特徴

- メッセージを送信したい相手がオフラインであっても、相手との鍵交換が可能
- PFS（Perfect forward secrecy）の提供

[35] WhatsApp Encryption Overview、Messenger Secret Conversations Technical Whitepaper、Signal partners with Microsoft to bring end-to-end encryption to Skype、Open Whisper Systems partners with Google on end-to-end encryption for Allo

    - One time keyはShared Keyを作成後速やかに破棄される
- 暗号化の認証手段の提供
    - メッセージの伝送を行う中央サーバが信用できないと仮定した場合も、メッセージの送信元や送信先が本当にやりとりしたい相手なのか、オフラインなどでクライアント同士の公開鍵のFingerprintsを比較することでお互いを認証することが出来る。

      送信者と受信者で同じSafety Numberを持っていることを確認することで認証できる
- 否認可能性 （Deniability）
    - PGPでの署名などでは否認不可性（Non-repudiation）を提供し、Aliceがメッセージを送ったことを認証する仕組みを提供するが、Signalプロトコルの場合は否認可能性を持つ。
    - イメージ例: 不貞関係にあるAliceとBobおよび、Bobと婚姻関係にあるCharlieを例にする。もしCharlieがBobの秘密鍵を入手しCharlieによるメッセージの復号が行われた後にも、メッセージを送信したAliceは裁判所に「私はやっていない」と言うことが出来る。

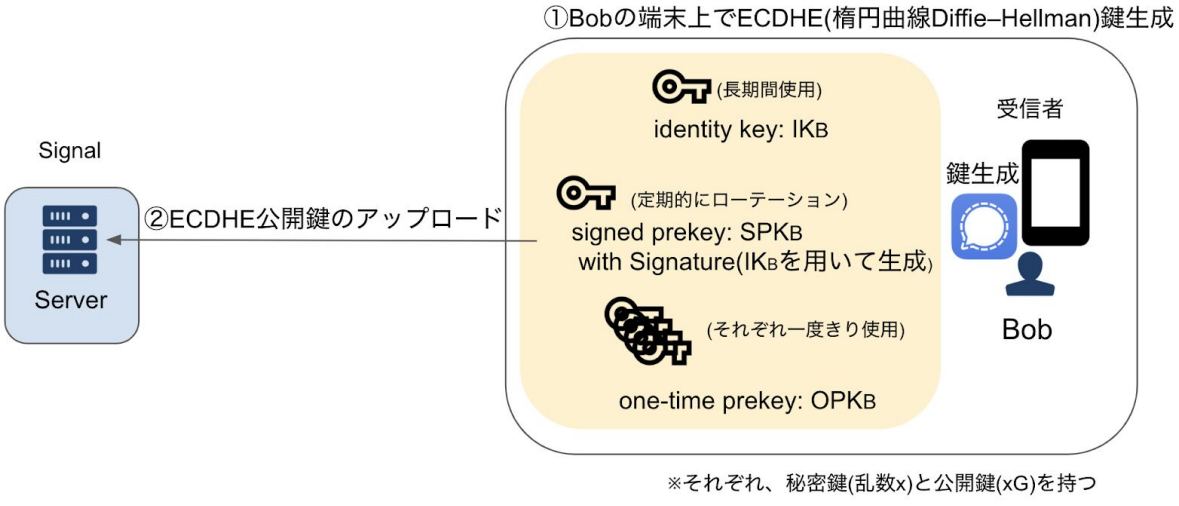


X3DHプロトコルの概要

X3DH鍵交換の流れは下記の図の通りである。Alice（送信者）とBob（受信者）の双方が一度きりだけ使用される鍵を使うことにより、同じ鍵を利用して共通鍵を生成することがないようになっている。（=PFSがある）
なお、「X3DHの特徴」で説明した、認証で用いることの出来るSafety Numberは、Alice（送信者）とBob（受信者）のIdentity key公開鍵を用いて生成されたFingerprintである。

事前準備

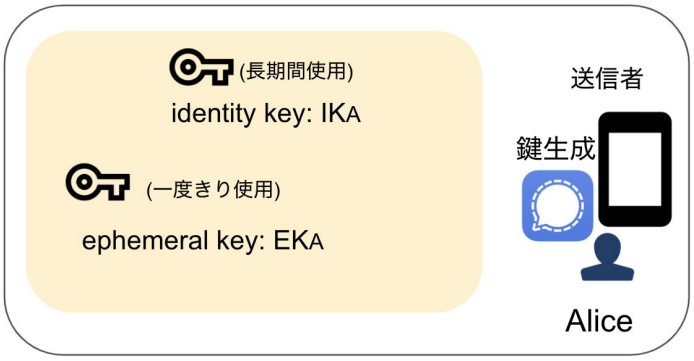
メッセージ受信者(Bob)は3種類の鍵を生成をしてSignal Serverにアップロードしておく



鍵交換1

メッセージ送信者(Alice)も鍵生成

Aliceの端末上でECDHE(楕円曲線Diffie–Hellman)鍵生成

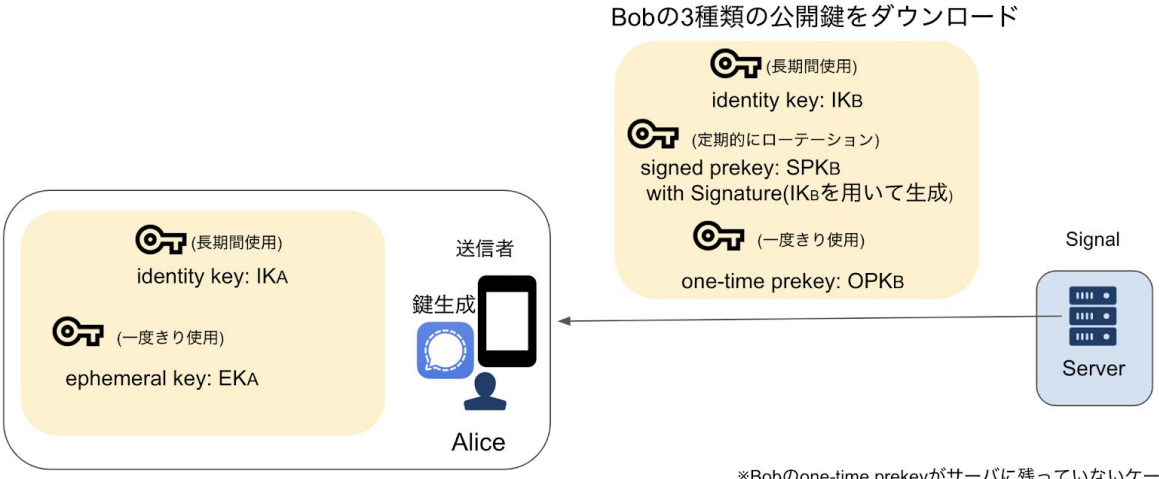


※それぞれ、秘密鍵(乱数x)と公開鍵(xG)を持つ
※EKAは実際はBobの署名検証を行った後に生成されるが、簡単のため簡略化している

鍵交換2

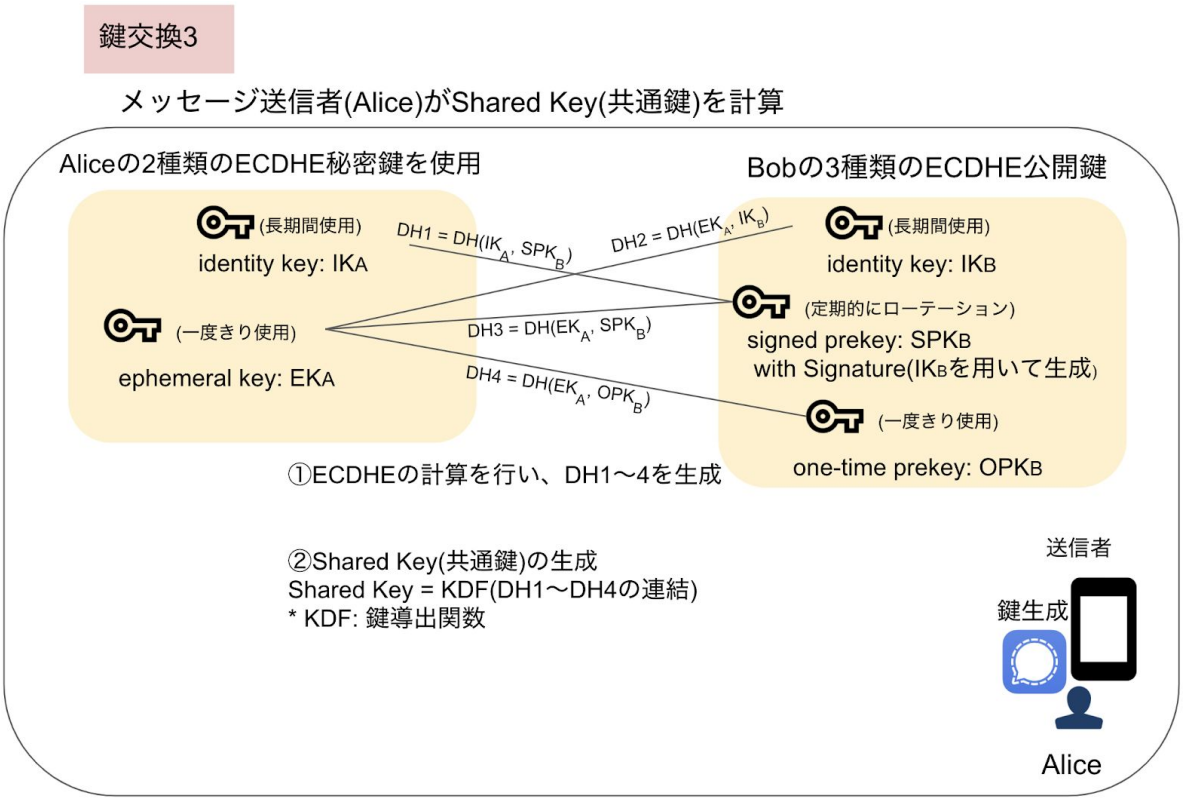
メッセージ送信者(Alice)が受信者(Bob)の公開鍵をダウンロード

Bobの3種類の公開鍵をダウンロード



※それぞれ、秘密鍵(乱数x)と公開鍵(xG)を持つ

※Bobのone-time prekeyがサーバに残っていないケースもあるが、今回は残っているものとする



ここまでで、Shared Key（共通鍵）をAliceとBobが手に入れた。

Double Ratchetの説明: メッセージの暗号化仕様

Signalプロトコルの暗号化はここで終わらない。X3DHにより共有されたShared Key（共通鍵）を元に,
さらにDouble Ratchetというメッセージ暗号化プロトコルで暗号鍵を生成する。

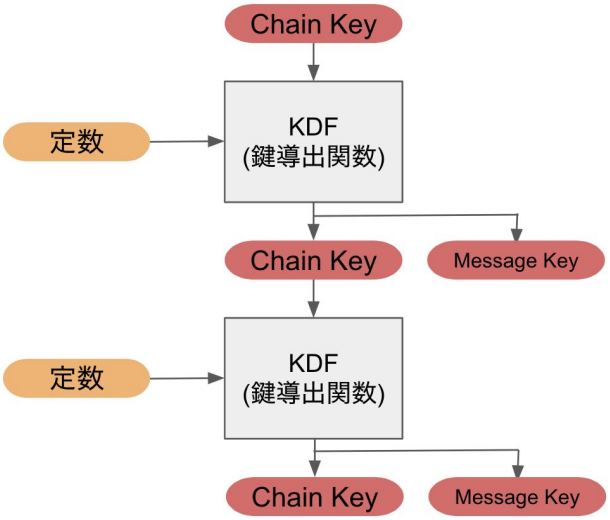
Double Ratchetの特徴

- やりとりするメッセージごとに、独自の鍵で暗号化を行う
- PFSがある
    - 現在の鍵で過去のメッセージを復号できない
- Post-Compromise Securityという特性もある
    - 現在の鍵で復号できるメッセージに限りがある

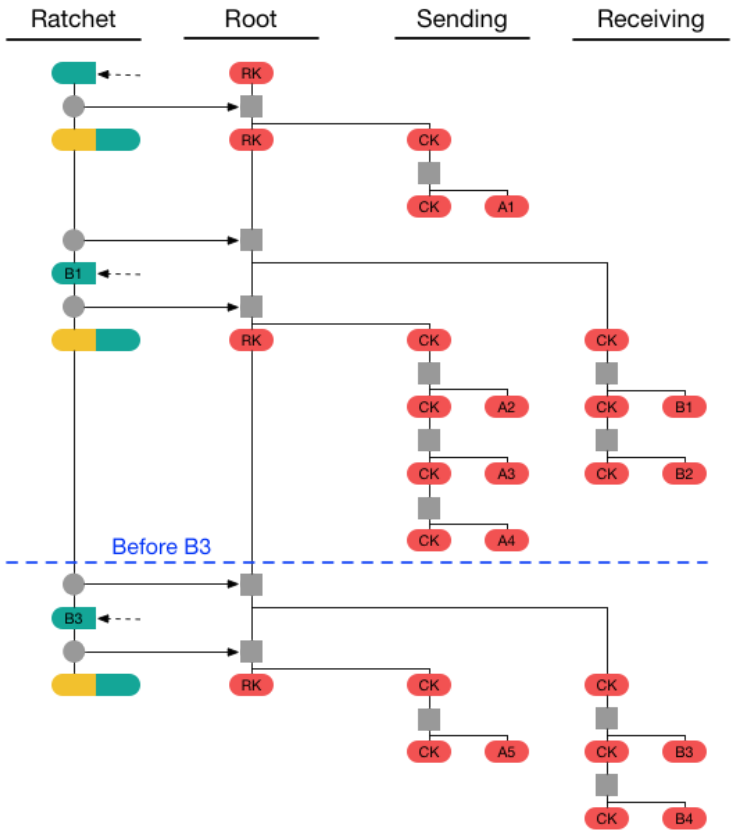
Double Ratchetの概要

Double Ratchetは、KDF（鍵導出関数）チェーンを用いたSymmetric-key ratchetと、
ECDHE鍵交換を利用したDiffie-Hellman ratchetの2つの仕組みを組合わあせて用いる。

Symmetric-key ratchetでは、下記のようなKDF Chain（鍵導出関数チェーン）を用いる。



上記のように、大元のChain Keyから次のChain KeyとMessage Key（共通鍵）を生成して暗号化に用いるという仕様になっている。Chain Keyを盗まれても、過去のMessage Keyを取り出すことはできない（PFS特性）。
しかしながら、これだけだと、Chain Keyを盗まれるとその後のメッセージを全て復号できることになってしまう。
そのため、メッセージに鍵交換のための公開鍵を送付することで、メッセージの送受信の度に鍵交換を行い大元の鍵を変更し、もしChain Keyを盗まれても復号できるメッセージに限りがあるようにしている。（Post-Compromise Security）



The Double Ratchet Algorithmより

上記のように、一番左側の列でメッセージの送受信の度にECDHE鍵交換を行いつつ大元の鍵を更新、さらに大元の鍵が同じあいだもKDF Chainを組み合わせて暗号化するための鍵を毎回生成するのがDouble Ratchetである。

各社のMetadataの取り扱い

上記で説明したように、Signalプロトコルでは、PFSやPost-Compromise Security、毎回の鍵更新や、認証などを、相手がオフラインであっても行えるという、今まで議論してきたEnd-to-end暗号化と比べても、非常にセキュリティ強度の高い暗号化を提供する。
WhatsApp、Signalを用いた場合、全てのメッセージは非常にセキュアな状態でEnd-to-end暗号化されると言える。しかし、同じSignalプロトコルを利用していると言っても、WhatsAppとSignalとでは暗号化の範囲が完全に同じであるわけではない。Signal ProtocolはEnd-to-end暗号化の手段を提供するが、あくまで鍵交換やメッセージ暗号化などの通信仕様にすぎず、実際にどこまで暗号化を行うかなどは、アプリケーション実装者の都合次第である。

可能な限りPrivacyに配慮している、Signalの場合、Metadataを可能な限りサーバ上で保持できないように設計されている。たとえば、コンタクトリストや場所、プロフィールなどをサーバに保管しない。また、”sealed sender”という仕組みを使い、送信元情報も暗号化され、メッセージ送信者がSignalサーバ上では分からないようになっている。

では、WhatsAppはどうか。WhatsAppのPrivacy Policyを確認すると、一定のMetadataの収集が確認できる。たとえば、アカウント情報、参加グループなどの参加などの利用状況、端末のアドレス帳、位置情報などが収集されている。End-to-end暗号化されていると言っても、Signal社と比べるとサーバ側で収集する情報に差があることが分かるだろう。
なお、Facebook社は政府などの要請により、Facebook社の提供するサービス全体（Facebook, Facebook Messenger, Instagram, WhatsAppなど）について、ユーザ情報を提供した件数・理由などについて、透明性レポートで公開している。

Chapter 8: おわりに

さて、本文書では、メールのセキュリティプロトコルから、End-to-end暗号化プロトコルであるLINE社のLetter Sealing、WhatsAppなどで利用されているSignalプロトコルまでを説明した。メッセージング全体のセキュリティという、コミュニケーションを支えるインターネットのセキュリティ技術全体を、スノーデン事件以後のPrivacyの文脈も含めて捉え直すことを目的として書いた。

2020年10月11日、アメリカ・イギリスを始めとする機密情報の交換ネットワークであるファイブ・アイズおよび日本・インドから、「国際声明: End-to-end暗号化と公共の安全（原題: International statement: End-to-end encryption and public safety）」という声明が発表された。この声明では、End-to-end暗号化を行っているメッセージングアプリに対して、暗号化された内容にアクセスする手段を設けることを要請している[36][37]。

市民のPrivacyと、国家インテリジェンス機関による諜報活動は、常に緊張関係にある。国家の安全はもちろん重要である。また、コミュニケーションの情報は国家安全保障だけでなく、一般的な犯罪の解決に役立つこともある。メールサービスは基本的にEnd-to-end暗号化されておらず、昨今だとWeb経由で利用するメールボックスに保管されるようになっている。たとえば大規模メールサービスであるGmailを運営するGoogleは透明性レポートにおいて、政府からのユーザ情報のリクエスト件数を公開しているが、もしEnd-to-end暗号化が行われており、情報を提供できなかった場合、特定の犯罪の解決に至らなかったケースもありうる[38]。その一方で、個人のPrivacyは、必ず尊重されなければならない重要な権利でもある。

End-to-end暗号化については、本文書で説明したように、様々なアプローチや制約がある。読者がどういった意見を持つにせよ、End-to-end暗号化に賛成・反対といった単純な議論では終わらないテーマであろう。End-to-end暗号化を行うにせよ、「Signalのように、サービス提供者側でのあらゆるコミュニケーションの保管を不可能にする方向を目指すべき」「WhatsAppのようなMetadataの取得は認めるべき」「メールのPGP暗号化のレベルのように、PFS（前方秘匿性）のない状態のみ認めるべき」、「LINEのURL遷移先の取得のような、合意を取った上での一定のメッセージ内容の取得は認めるべき」、「Metadata取得はGoogle、Facebook、LINEのような透明性レポートの提供を条件とするべき」など、様々な立場を取りうる。

本文書が、日本において2020年現在のメッセージングのセキュリティの現状がなるべく正確に把握されること、より発展した議論が今後行われていくことに寄与することを期待する。

[36] 対話アプリ、暗号化見直しで声明 日米英など7カ国
[37] International Statement: End-To-End Encryption and Public Safety
[38] Google は政府からのユーザー情報リクエストにどう対応しているか – ポリシーと規約

Appendix: 参考文献

各脚注で触れた書籍・URL以外に、下記を主に参考にした

- スノーデン事件を始めとする諜報活動などの確からしい事実関係の把握
  - 土屋 大洋『サイバーセキュリティと国際政治』（千倉書房，2015年）
  - 土屋 大洋『暴露の世紀　国家を揺るがすサイバーテロリズム』（KADOKAWA、2016年）
    * スノーデン事件により告発されたインテリジェンス機関による諜報活動について、一個人には事実関係を把握することが難しいため、研究者による書籍を参考にしている

- SMTPおよびTLSに関する書籍・Webサイト
  - Philip Miller『マスタリングTCP/IP 応用編』（オーム社、1998年）
  - Kevin Johnson『電子メールプロトコル詳説—インターネット電子メールアーキテクチャからIETF標準プロトコル群の詳細』（ピアソンエデュケーション、2000年）
  - Eric Rescorla『マスタリングTCP/IP SSL/TLS編』（オーム社、2003年）
  - Ivan Ristić『プロフェッショナル SSL/TLS』（ラムダノート、2017年）
  - M3AAWG Recommendations for Senders Handling of Complaints
  - M3AAWG Email Authentication Recommended Best Practices
  - Postfix Documentation
  - STARTTLS Everywhere
  - RFC821: SIMPLE MAIL TRANSFER PROTOCOL
  - RFC3207: SMTP Service Extension for Secure SMTP over Transport Layer Security
  - RFC8461: SMTP MTA Strict Transport Security （MTA-STS）
  - RFC7208: Sender Policy Framework （SPF） for Authorizing Use of Domains in Email, Version 1
  - RFC6376: DomainKeys Identified Mail （DKIM） Signatures
  - RFC7489: Domain-based Message Authentication, Reporting, and Conformance （DMARC）

- PGPおよびS/MIME
  - 電子メールのセキュリティ S/MIMEを利用した暗号化と電子署名（IPA）
  - The PGP Problem
  - 結城 浩『暗号技術入門 第3版　秘密の国のアリス』（SBクリエイティブ、2015年）

- LINE
  - LINE Encryption Overview: Technical Whitepaper
  - SECURITY x LINE PLATFORM: LINE Platform Security
  - LINT (LINE Improvement for Next Ten years)
  - LINE プライバシーポリシー
  - LINEの情報利用に関するご案内
  - 「サービス向上のための情報利用に関するお願い」について、よくあるご質問および詳細情報

- Signal Protocol
  - The State of Private Messaging Applications
  - Specifications >> The X3DH Key Agreement Protocol
  - Blog >> Simplifying OTR deniability.
  - Blog >> Technology preview: Sealed sender for Signal
  - Specifications >> The Double Ratchet Algorithm
  - WhatsApp Encryption Overview
  - CS 465 Signal （Brigham Young University）
  - CS 255: Intro to Cryptography （Stanford University）