ITエンジニアとして
知っておいてほしい
電子メールという
大きな穴

【2025改訂版】

Takuto Nagami
@logica0419

講師自己紹介

- 永見 拓人 (@logica0419)
- 千葉工業大学 情報科学部 情報ネットワーク学科 4年
  - 研究: **分散DB** (etcd/Raft) × **暗号** (秘密分散)
- Go言語・Kubernetes周りにある程度詳しいです
- セキュキャン歴
  - 2022 全国大会 修了
  - 2024/2025 全国大会 チューター
  - 2024 ミニ (山梨) 講師

なぜ、今**メール**なのか

「メールなんて**今どき**
使わないじゃん」と
思っていませんか？

サイバー犯罪検挙件数の推移 by 警察庁

平成28年～令和2年のもの

罪名＼年	14	15	16	17	18	17年から18年にかけての増加
不正アクセス禁止法違反	105	145	142	277	703	426 (153.8%)
コンピュータ・電磁的記録対象犯罪	30	55	55	73	129	56 (76.7%)
電子計算機使用詐欺	18	34	42	49	63	14 (28.6%)
電磁的記録不正作出・供用	8	12	8	17	56	39 (229.4%)
電子計算機損壊等業務妨害	4	9	5	7	10	3 (42.9%)
ネットワーク利用犯罪	1,471	1,649	1,884	2,811	3,593	782 (27.8%)
詐欺	514	521	542	1,408	1,597	189 (13.4%)
児童買春・児童ポルノ法違反（児童買春）	268	269	370	320	463	143 (44.7%)
児童買春・児童ポルノ法違反（児童ポルノ）	140	102	85	136	251	115 (84.6%)
商標法違反	37	95	82	109	218	109 (100.0%)
青少年保護育成条例違反	70	120	136	174	196	22 (12.6%)
わいせつ物頒布等	109	113	121	125	192	67 (53.6%)
著作権法違反	66	87	174	128	138	10 (7.8%)
その他	267	342	374	411	538	127 (30.9%)
合計	1,606	1,849	2,081	3,161	4,425	1,264 (40.0%)

メールが何らかの形で絡んでいるものはどれでしょう？

罪名 ＼ 年	14	15	16	17	18	17年から18年にかけての増加
不正アクセス禁止法違反	105	145	142	277	703	426 (153.8%)
コンピュータ・電磁的記録対象犯罪	30	55	55	73	129	56 (76.7%)
電子計算機使用詐欺	18	34	42	49	63	14 (28.6%)
電磁的記録不正作出・供用	8	12	8	17	56	39 (229.4%)
電子計算機損壊等業務妨害	4	9	5	7	10	3 (42.9%)
ネットワーク利用犯罪	1,471	1,649	1,884	2,811	3,593	782 (27.8%)
詐欺	514	521	542	1,408	1,597	189 (13.4%)
児童買春・児童ポルノ法違反（児童買春）	268	269	370	320	463	143 (44.7%)
児童買春・児童ポルノ法違反（児童ポルノ）	140	102	85	136	251	115 (84.6%)
商標法違反	37	95	82	109	218	109 (100.0%)
青少年保護育成条例違反	70	120	136	174	196	22 (12.6%)
わいせつ物頒布等	109	113	121	125	192	67 (53.6%)
著作権法違反	66	87	174	128	138	10 (7.8%)
その他	267	342	374	411	538	127 (30.9%)
合計	1,606	1,849	2,081	3,161	4,425	1,264 (40.0%)

メールが何らかの形で絡んでいるものはどれでしょう？

罪名＼年	14	15	16	17	18	17年から18年にかけての増加
不正アクセス禁止法違反	105	145	142	277	703	426 (153.8%)
コンピュータ・電磁的記録対象犯罪	30	55	55	73	129	56 (76.7%)
電子計算機使用詐欺	18	34	42	49	63	14 (28.6%)
電磁的記録不正作出・供用	8	12	8	17	56	39 (229.4%)
電子計算機損壊等業務妨害	4	9	5	7	10	3 (42.9%)
ネットワーク利用犯罪	1,471	1,649	1,884	2,811	3,593	782 (27.8%)
詐欺	514	521	542	1,408	1,597	189 (13.4%)
児童買春・児童ポルノ法違反（児童買春）	268	269	370	320	463	143 (44.7%)
児童買春・児童ポルノ法違反（児童ポルノ）	140	102	85	136	251	115 (84.6%)
商標法違反	37	95	82	109	218	109 (100.0%)
青少年保護育成条例違反	70	120	136	174	196	22 (12.6%)
わいせつ物頒布等	109	113	121	125	192	67 (53.6%)
著作権法違反	66	87	174	128	138	10 (7.8%)
その他	267	342	374	411	538	127 (30.9%)
合計	1,606	1,849	2,081	3,161	4,425	1,264 (40.0%)

メールが何らかの形で絡んでいるものはどれでしょう？

何気なく使っているメールは
サイバー犯罪の温床！

						けての増加
						(153.8%)
						(76.7%)
						(28.6%)
						(229.4%)
						(42.9%)
						(27.8%)
						(13.4%)
						(44.7%)
						(84.6%)
	商標法違反	37	95	82	109	218	109 (100.0%)
	青少年保護育成条例違反	70	120	136	174	196	22 (12.6%)
	わいせつ物頒布等	109	113	121	125	192	67 (53.6%)
	著作権法違反	66	87	174	128	138	10 (7.8%)
	その他	267	342	374	411	538	127 (30.9%)
合計		1,606	1,849	2,081	3,161	4,425	1,264 (40.0%)

なぜ電子メールは
様々な**サイバー犯罪で**
用いられるのだろうか？

そのためには、メールの現状を知る必要がある

電子メールのおこり

- **1965年**頃から存在する
  - 非常に**長く使われている**
  - **インターネットがない時代**からある
  - インターネットのもとになった技術の一つ
- 現在の fuga@hogehoge.net のような形のアドレスが使われ始めたのは、1971年から
- 1980年代以降に、**RFC**として送受信方法の**共通化**が行われた

電子メールの特徴

- 特定の**サービス等に依存しない**
  - ここがSNSとは違う
- **プライベート**な 1対1 or 1対多 の通信
  - Webサイト等とは少し別の技術
- **非常に広く**用いられる
  - メールアドレスを持たない人、周りでいますか？
  - 多くのサイトで**ログイン**に使用する
  - **企業同士**のやり取りは、多くが電子メール

電子メールのセキュリティ懸念

- 単純に**広く使われている**ので、犯罪に使いやすい
  - ターゲットが多いので、**数打て戦法**が有効に
  - SNSをあまり見ない**高齢者や企業**さえ**標的に**できる
- インターネットの起こり時代に定められた、非常に**脆弱で機能が少ない**仕組みで送られている
  - 素の状態では**送信元偽装・改ざん**とかし放題
  - ≒ **Secure by Default**でない
    - ただ使っているだけで安全、とはならない

サイバー犯罪検挙件数の推移 by 警察庁

メールはどのように絡んでいるのか

罪名＼年	14	15	16	17	18	17年から18年にかけての増加
不正アクセス禁止法違反	105	145	142	277	703	426 (153.8%)
コンピュータ・電磁的記録対象犯罪	30	55	55	73	129	56 (76.7%)
電子計算機使用詐欺	18	34	42	49	63	14 (28.6%)
電磁的記録不正作出・供用	8	12	8	17	56	39 (229.4%)
電子計算機損壊等業務妨害	4	9	5	7	10	3 (42.9%)
ネットワーク利用犯罪	1,471	1,649	1,884	2,811	3,593	782 (27.8%)
詐欺	514	521	542	1,408	1,597	189 (13.4%)
児童買春・児童ポルノ法違反（児童買春）	268	269	370	320	463	143 (44.7%)
児童買春・児童ポルノ法違反（児童ポルノ）	140	102	85	136	251	115 (84.6%)
商標法違反	37	95	82	109	218	109 (100.0%)
青少年保護育成条例違反	70	120	136	174	196	22 (12.6%)
わいせつ物頒布等	109	113	121	125	192	67 (53.6%)
著作権法違反	66	87	174	128	138	10 (7.8%)
その他	267	342	374	411	538	127 (30.9%)
合計	1,606	1,849	2,081	3,161	4,425	1,264 (40.0%)

サイバー犯罪検挙件数の推移 by 警察庁

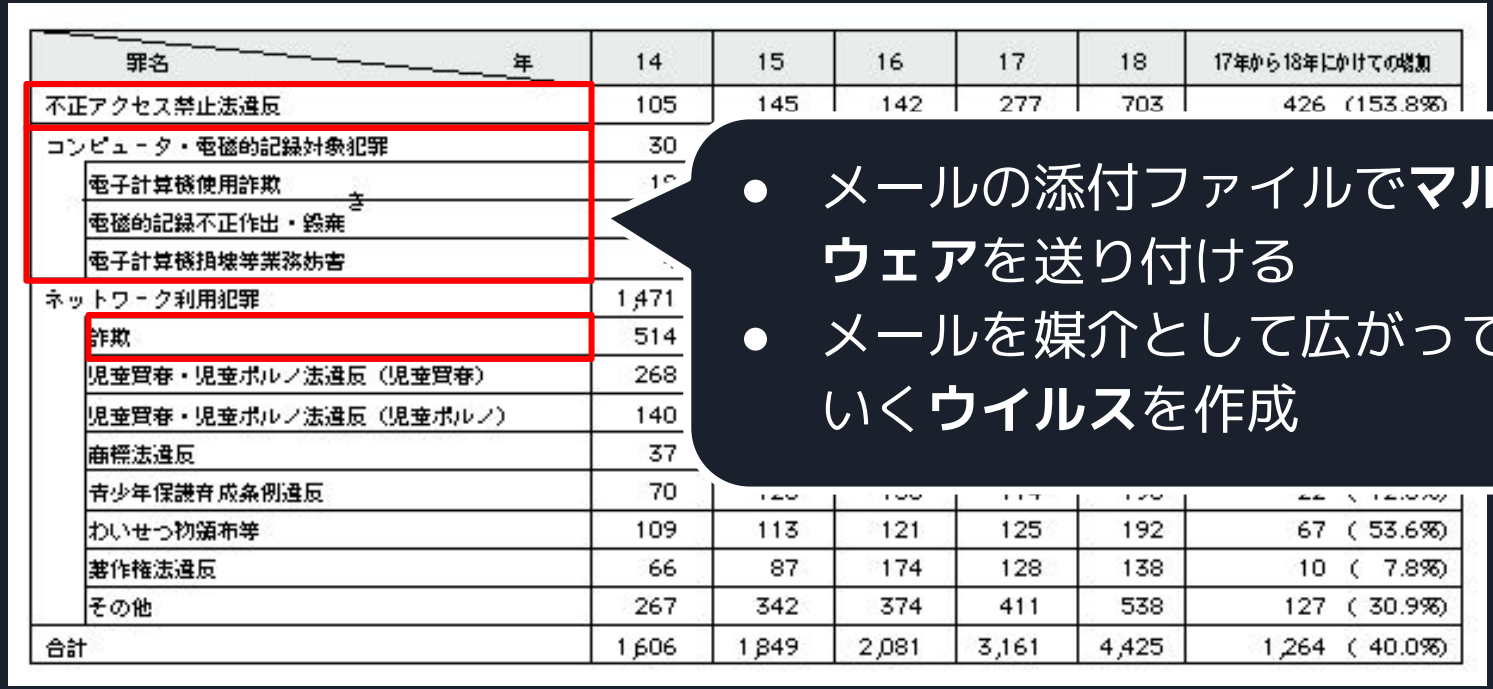
メールはどのように絡んでいるのか

罪名＼年	14					
不正アクセス禁止法違反						
コンピュータ・電磁的記録対象犯罪						
電子計算機使用詐欺	16					
電磁的記録不正作出・供用	8					
電子計算機損壊等業務妨害	4					
ネットワーク利用犯罪	1,471					
詐欺	514					
児童買春・児童ポルノ法違反（児童買春）	268					
児童買春・児童ポルノ法違反（児童ポルノ）	140					
商標法違反	37					
青少年保護育成条例違反	70	120	136	174	196	22 （ 12.6%）
わいせつ物頒布等	109	113	121	125	192	67 （ 53.6%）
著作権法違反	66	87	174	128	138	10 （ 7.8%）
その他	267	342	374	411	538	127 （ 30.9%）
合計	1,606	1,849	2,081	3,161	4,425	1,264 （ 40.0%）

- **フィッシング**メールで
  パスワードを奪う
- 警告メールを装ってアカウント
  リセットを強要し**アカウントを
  乗っ取る**

サイバー犯罪検挙件数の推移 by 警察庁

メールはどのように絡んでいるのか



罪名＼年	14	15	16	17	18	17年から18年にかけての増加
不正アクセス禁止法違反	105	145	142	277	703	426 (153.8%)
コンピュータ・電磁的記録対象犯罪	30					
電子計算機使用詐欺						
電磁的記録不正作出・供用						
電子計算機損壊等業務妨害						
ネットワーク利用犯罪	1,471					
詐欺	514					
児童買春・児童ポルノ法違反（児童買春）	268					
児童買春・児童ポルノ法違反（児童ポルノ）	140					
商標法違反	37					
青少年保護育成条例違反	70					
わいせつ物頒布等	109	113	121	125	192	67 (53.6%)
著作権法違反	66	87	174	128	138	10 (7.8%)
その他	267	342	374	411	538	127 (30.9%)
合計	1,606	1,849	2,081	3,161	4,425	1,264 (40.0%)

- メールの添付ファイルで**マルウェア**を送り付ける
- メールを媒介として広がっていく**ウイルス**を作成

サイバー犯罪検挙件数の推移 by 警察庁

メールはどのように絡んでいるのか

罪名＼年	14	15	16	17	18	17年から18年にかけての増加
不正アクセス禁止法違反	105	145	142	277	703	426 (153.8%)
コンピュータ・電磁的記録対象犯罪	30	55	55	73	129	56 (76.7%)
電子計算機使用詐欺	18	34	42	49	63	14 (28.6%)
電磁的記録不正作出・供用	8					
電子計算機損壊等業務妨害	4					
ネットワーク利用犯罪	1,4					
詐欺						
児童買春・児童ポルノ法違反（児童買春）	2					
児童買春・児童ポルノ法違反（児童ポルノ）	140					
商標法違反	37					
青少年保護育成条例違反	70					
わいせつ物頒布等	109					
著作権法違反	66					
その他	267					
合計	1,606					

- **フィッシング**メール
- 送り主を大手サービスだと偽りお金を払わせる
- 家族のふりをしてメールを出しオレオレ詐欺をする

様々な犯罪に絡んでしまう
電子メールを
学ぶことは大事！

自分たちが使わないからこそ、狙われることも

アジェンダ

- **なぜ、今メールなのか**
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

アジェンダ

- なぜ、今メールなのか
- **プロトコルスタックとRFC**
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

プロトコルスタックとRFC

プロトコル

- コンピューター同士が**通信する時の約束事**
  - どんなデータを渡す必要があるか
  - どんな順番でデータを渡すか
  - それぞれのデータはどんなサイズか
  - …などなど
- 人間界の「**言語**」と同じもの
- 世界中のあらゆる機器（どんな企業製でも）が相互通信できる**インターネット**を作るのに**必要不可欠**

プロトコル

プロトコルを使って「会話」する

プロトコル

プロトコルが無いと…

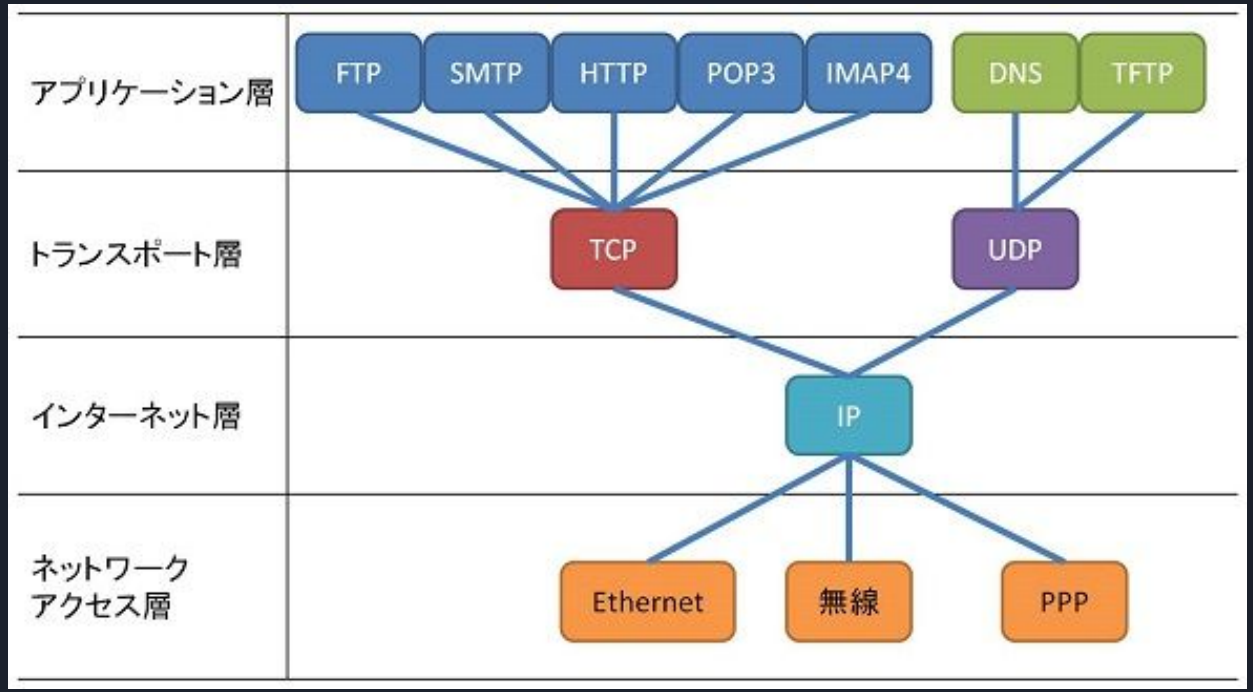
言ってることが
わからないので、
通信できない！

プロトコルは、**通信の**
「カオス」を解決してくれる
素晴らしい仕組み

プロトコルという仕組みに感謝しよう

プロトコルスタック

様々なプロトコルを、**積み重ね**ひとまとめにしたもの



なぜ「積み重ねる」(階層化する) のか

- 通信に必要な**知識は膨大**
  - 極論を言えばみんなが指から電気信号を出して通信しなければいけなくなる
  - 人間に**分かりやすくする**ため / **あらゆる場面に対応する**ために、様々な知識が必要
- **関心の分離**をし、専門分野を分ける
  - LANケーブルを扱う人と、Webページを作る人は一緒ではない

OSI参照モデル

今のインターネットを支えるプロトコルの、**よく使われる階層分け方法**

7	アプリケーション層
6	プレゼンテーション層
5	セッション層
4	トランスポート層
3	ネットワーク層
2	データリンク層
1	物理層

階層ごとの役割

7	アプリケーション層	アプリケーション間のやり取り
6	プレゼンテーション層	データの表現形式
5	セッション層	接続の手順
4	トランスポート層	データ通信の制御
3	ネットワーク層	インターネットワークでの通信
2	データリンク層	同一ネットワーク上での通信
1	物理層	ケーブルや電気信号やコネクタなど

それぞれの階層でプロトコルを守ることで、インターネットは成り立っている

インターネットはこれらの頑張りによる奇跡の産物

プロトコルを決める過程 - RFC

- **RFC** (Request for Comments)
  - インターネットにおける**プロトコルの仕様書**のこと
- RFCのフェーズ
  - **Internet Draft** - 仕様を煮詰める
  - **Proposed Standard** - 標準化すべきと認められた
  - **Draft Standard** - 多くの機器で使える
  - **Standard** - インターネットで広く使われる
- 「決めてから普及」ではなく、「**普及したら標準**」

プロトコルを決める過程 - RFC

- **IETF**という団体で議論され、標準化される
- 思想
  - **オープン**である - Request for “Comments”
    - RFCはメーリングリストで決まる
  - 仕様よりも**実装重視**
    - RFCのフェーズも、この思想の下にある
- 内容は**改定できない**
  - 新しいRFCの発行と、古いRFCの無効化で変更

RFCの例 - http

https://www.rfc-editor.org/rfc/rfc9110.html で公開

RFC 9110
HTTP Semantics

Abstract

The Hypertext Transfer Protocol (HTTP) is a stateless application-level protocol for distributed, collaborative, hypertext information systems. This document describes the overall architecture of HTTP, establishes common terminology, and defines aspects of the protocol that are shared by all versions. In this definition are core protocol elements, extensibility mechanisms, and the "http" and "https" Uniform Resource Identifier (URI) schemes.

This document updates RFC 3864 and obsoletes RFCs 2818, 7231, 7232, 7233, 7235, 7538, 7615, 7694, and portions of 7230.

インターネットが好きなら
是非**プロトコルスタック**と
RFCを理解して使おう！

「RFCを読む会」なんてイベントもあるそう

アジェンダ

- なぜ、今メールなのか
- **プロトコルスタックとRFC**
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- **メールが送られる仕組み**
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

メールが**送られる仕組み**

hoge@gmail.com から fuga@icloud.com への メール送信を図解してみる

メールが送られる仕組みを理解しよう

登場人物たち

ユーザー

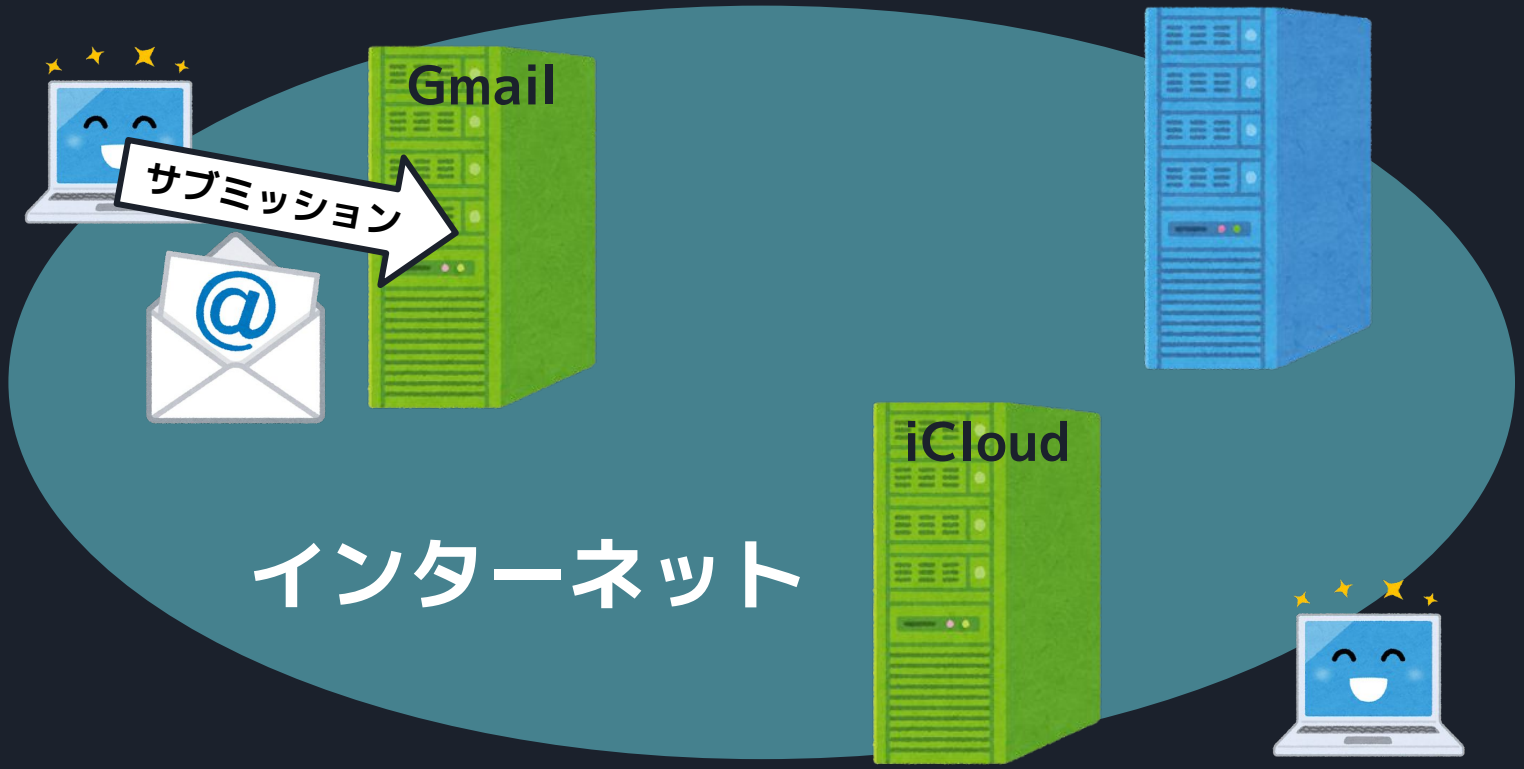
メールサーバー

DNSサーバー

ユーザーが送信する (サブミッション)

インターネット

ユーザーが送信する (サブミッション)

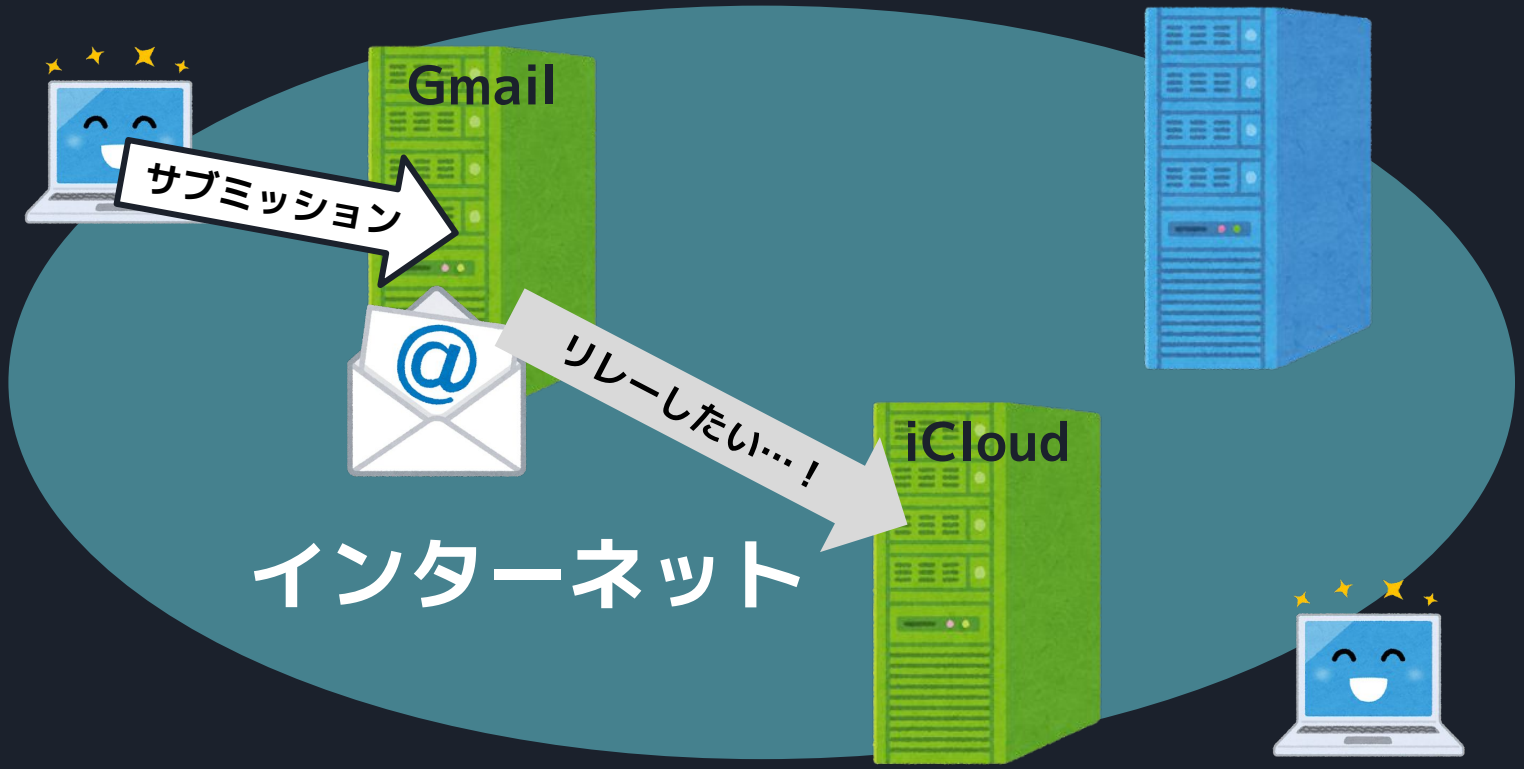


メールサーバー

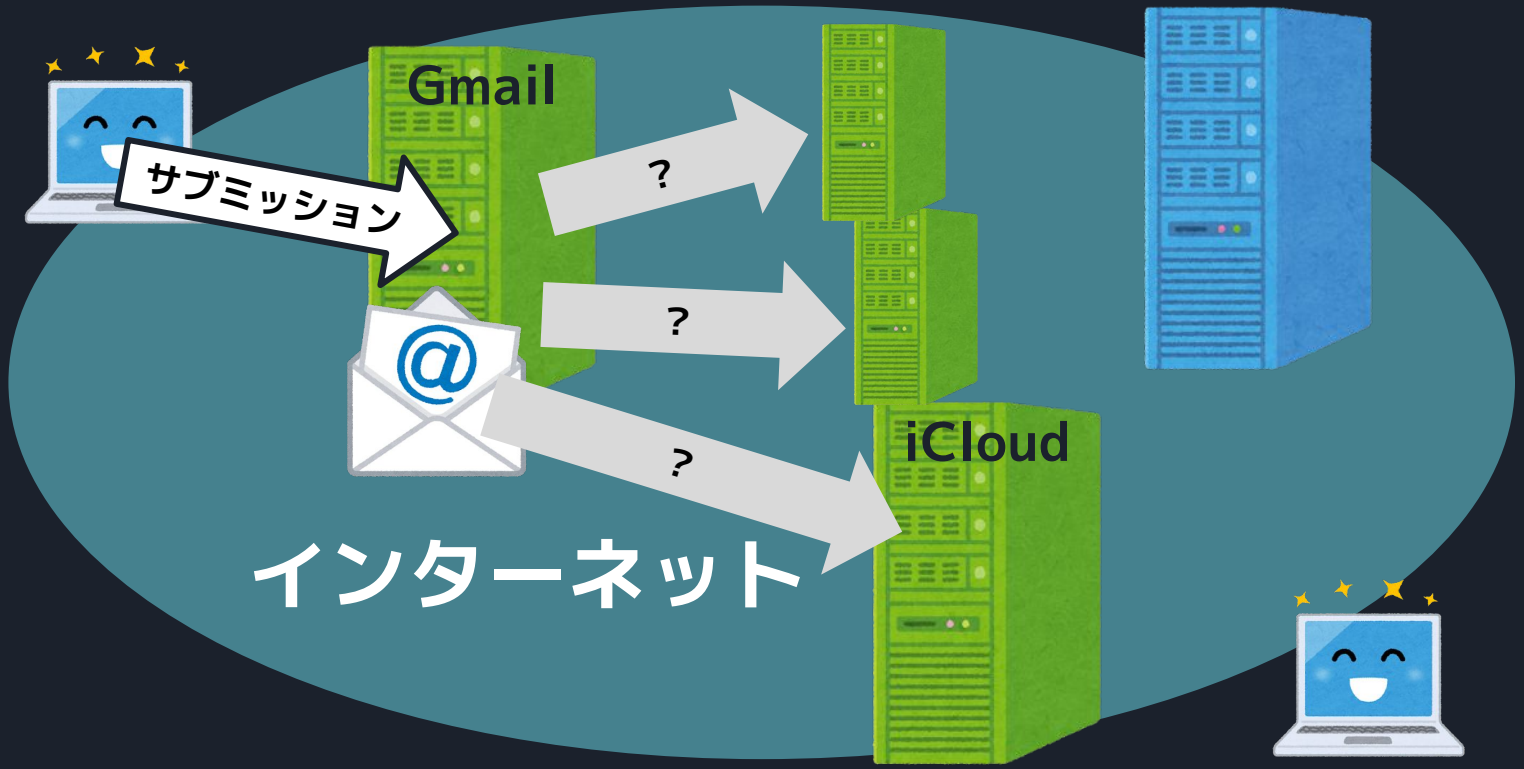
- メールの **送信** / **受信** / **保存** 全てを執り行う
- メールの**受信**
  - あらゆるメールを受け取る
  - 自分宛のメールなら**ユーザーごとのメールボックス**に**保存**し、提供する
- メールの**送信**
  - 自分宛でなければ正しい宛先に**届ける**
  - **リレー**と呼ばれる

宛先に届ける (リレー)

宛先に届ける (リレー)



でも、iCloudのサーバーの場所が不明…



IPアドレスとドメイン名

- **IPアドレス**
  - インターネットでそれぞれの機器が**一意に持つ住所**
- **ドメイン名**
  - IPアドレスは数字でわかりづらいので、これを**わかりやすい文字列**で表せるようにしたもの
  - https://www.**security-camp.or.jp**/
  - info@**security-camp.or.jp**
    - 実は**メールアドレスにも**ドメインは入ってる！

DNSサーバー

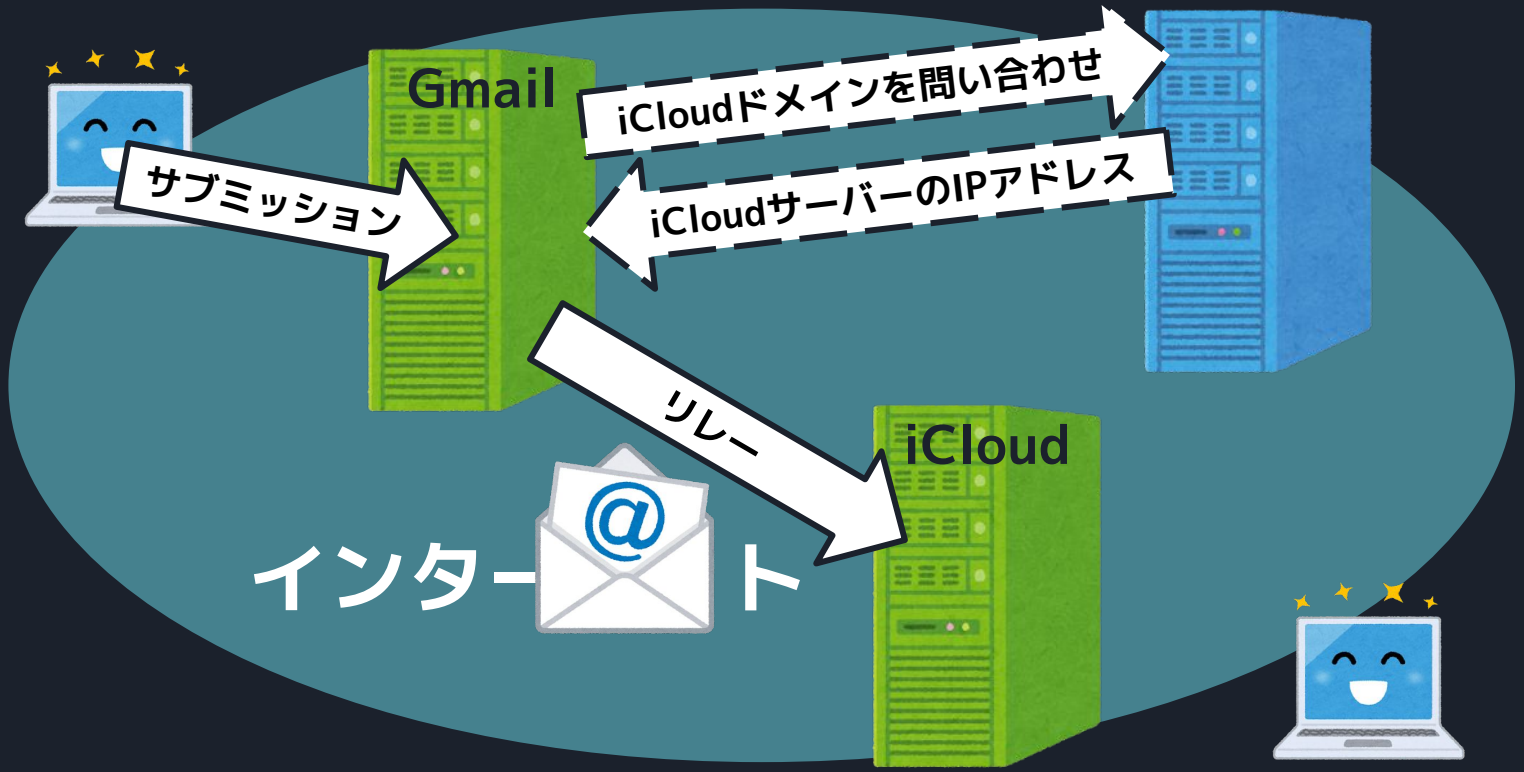
- **ドメイン名**と**IPアドレス**を紐付けるための仕組みを提供する
  - これ無しでドメイン名は成り立たない
- おまけ機能として、**ドメイン名**と**任意のテキストデータ**を紐づける仕組みも提供している
  - これが、**メールを安全にするための**
    プロトコルで活躍する

DNSに問い合わせ

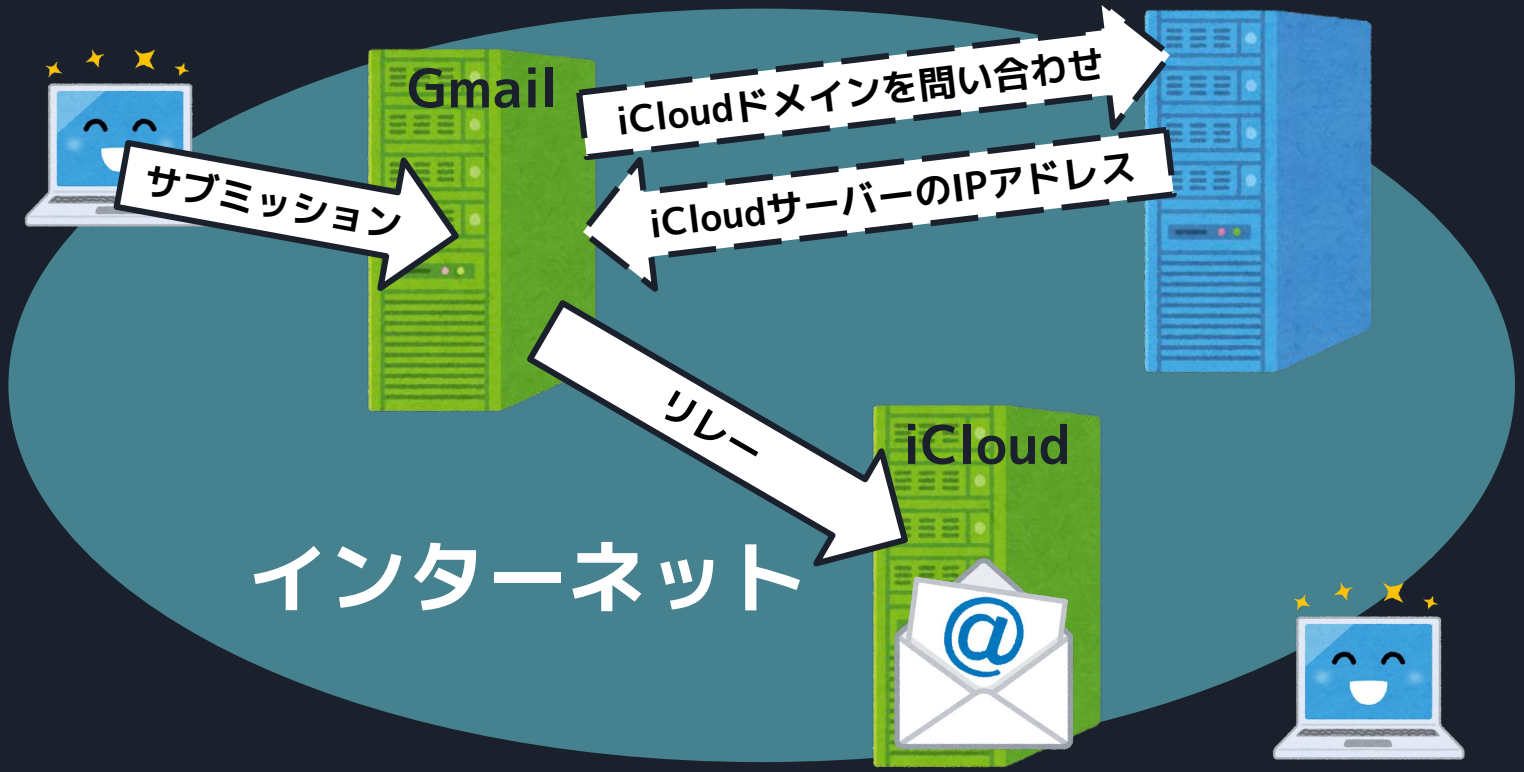
DNSに問い合わせ


Gmail
サブミッション
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
iCloud
インターネット

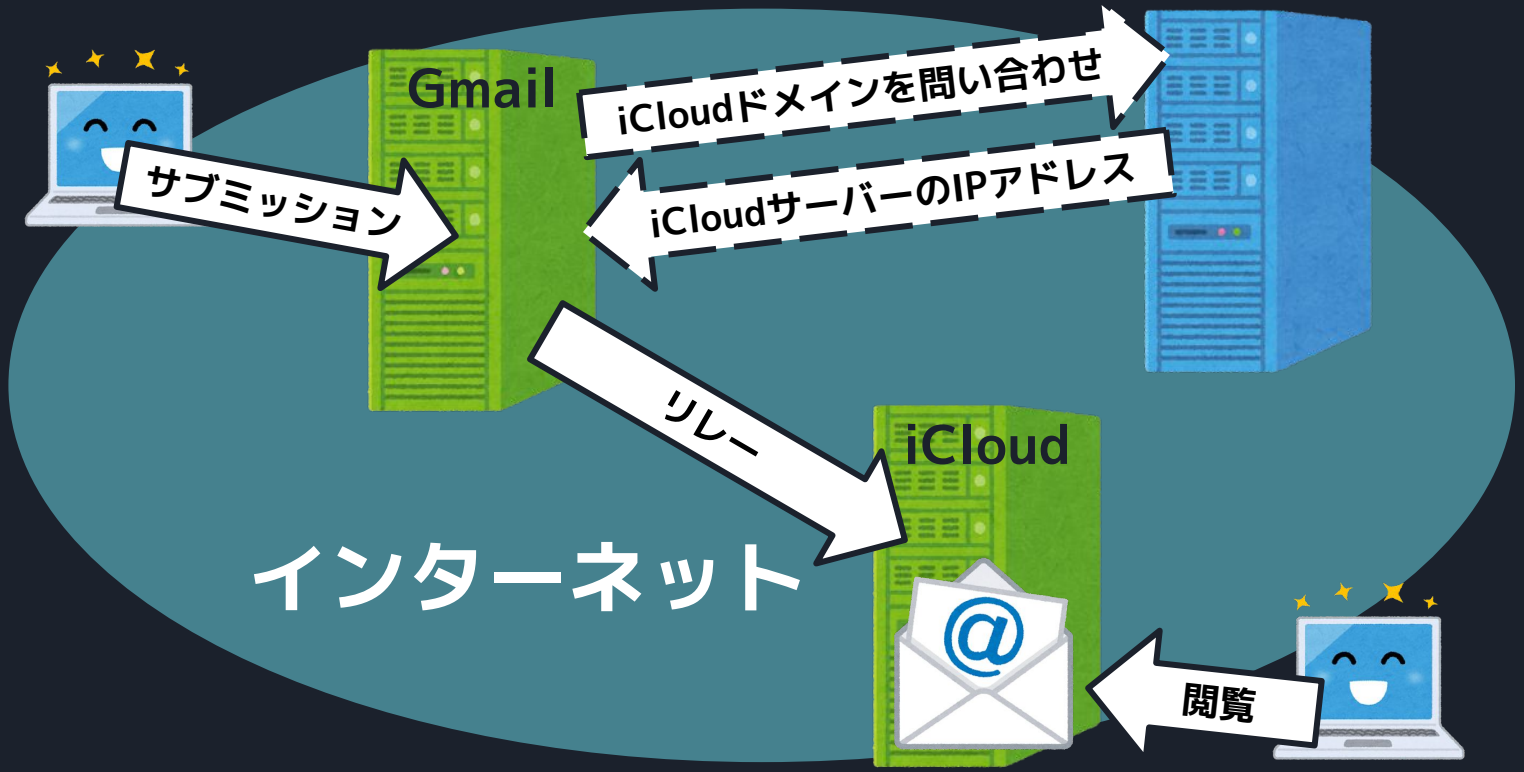
宛先に届ける（リレー）


Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
リレー
iCloud
インター
ト

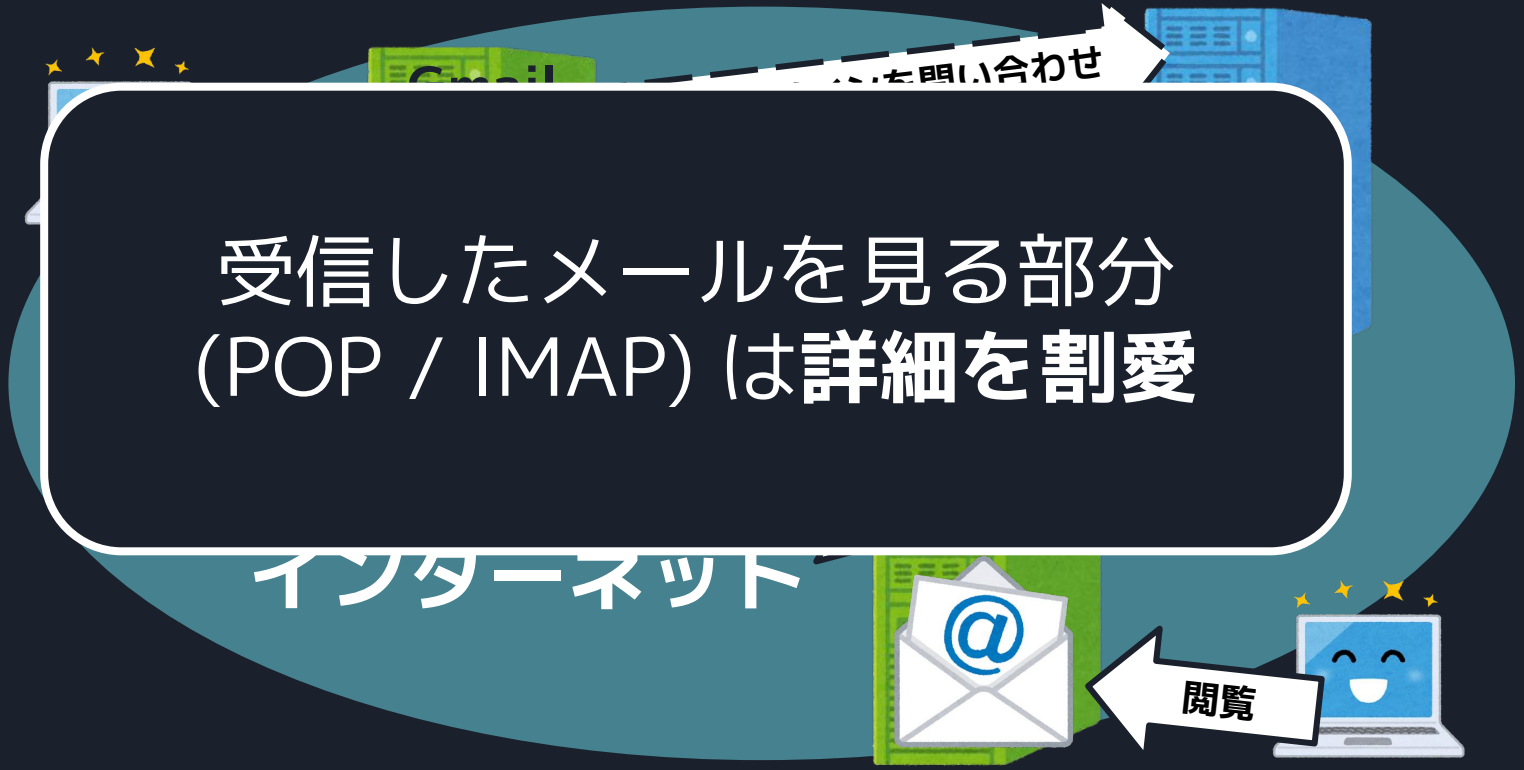
iCloudがメールを受信



ユーザーが受信したメールを見る

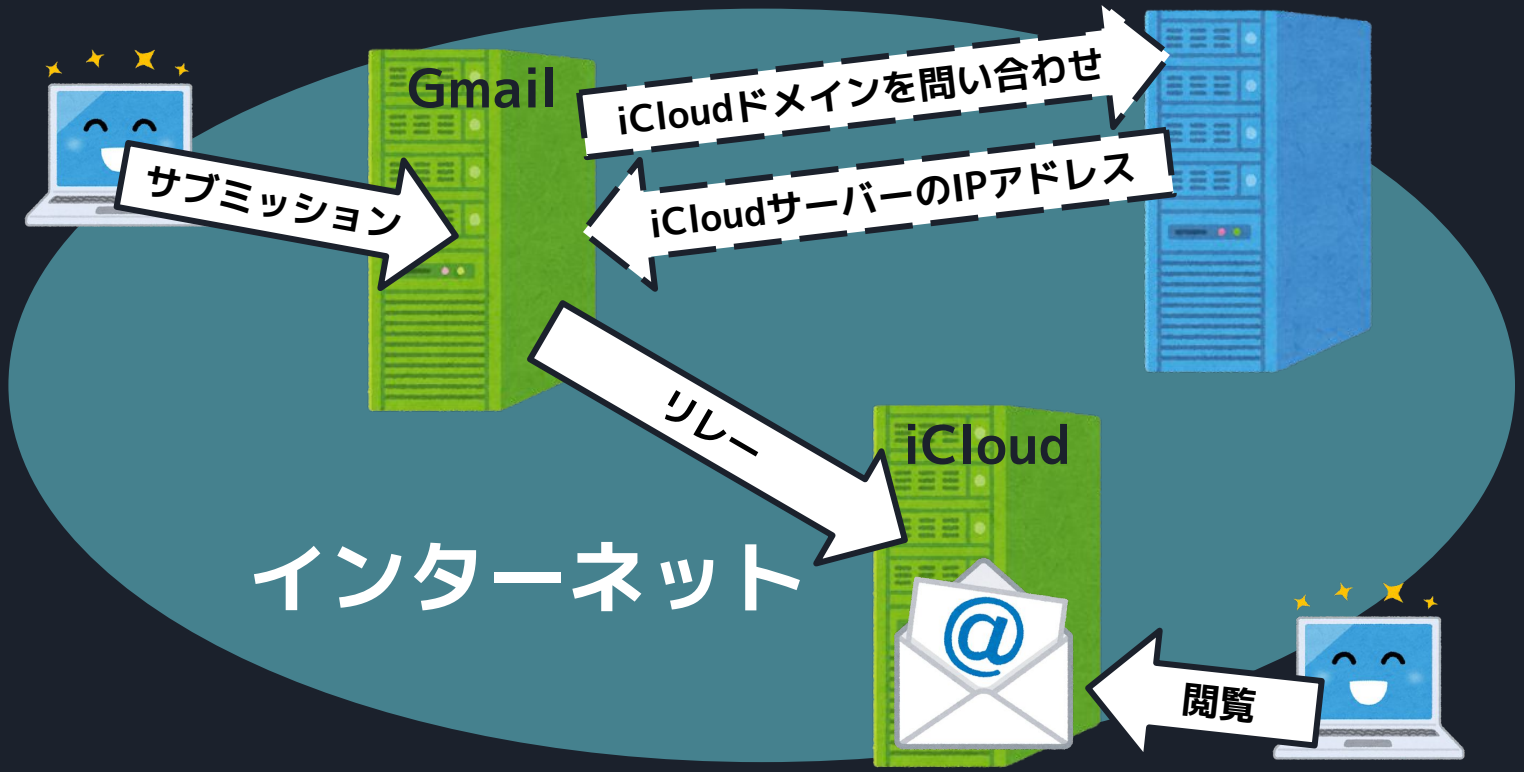


ユーザーが受信したメールを見る



受信したメールを見る部分
(POP / IMAP) は**詳細を割愛**

今一度全体像を俯瞰してみよう



アジェンダ

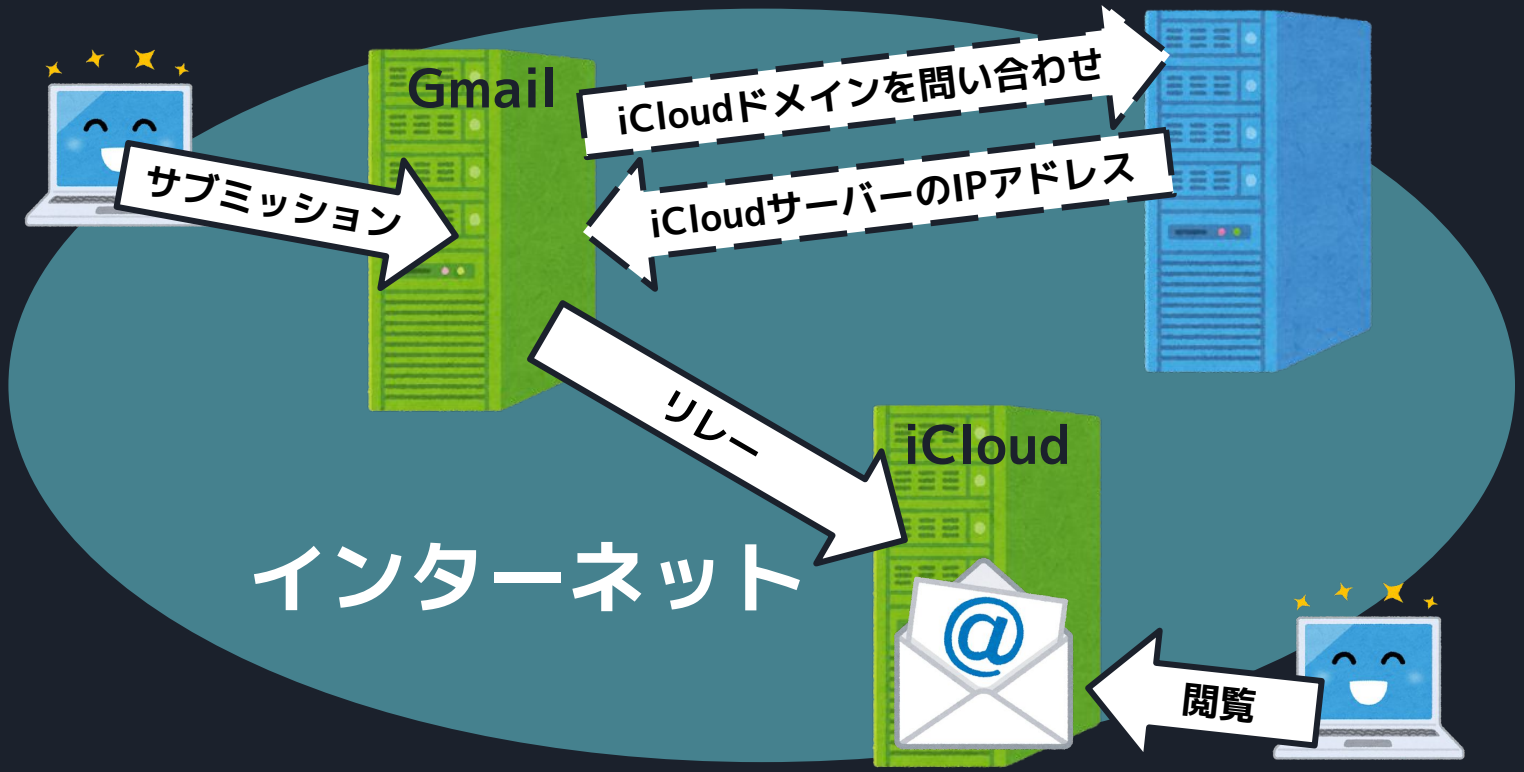
- なぜ、今メールなのか
- プロトコルスタックとRFC
- **メールが送られる仕組み**
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

アジェンダ

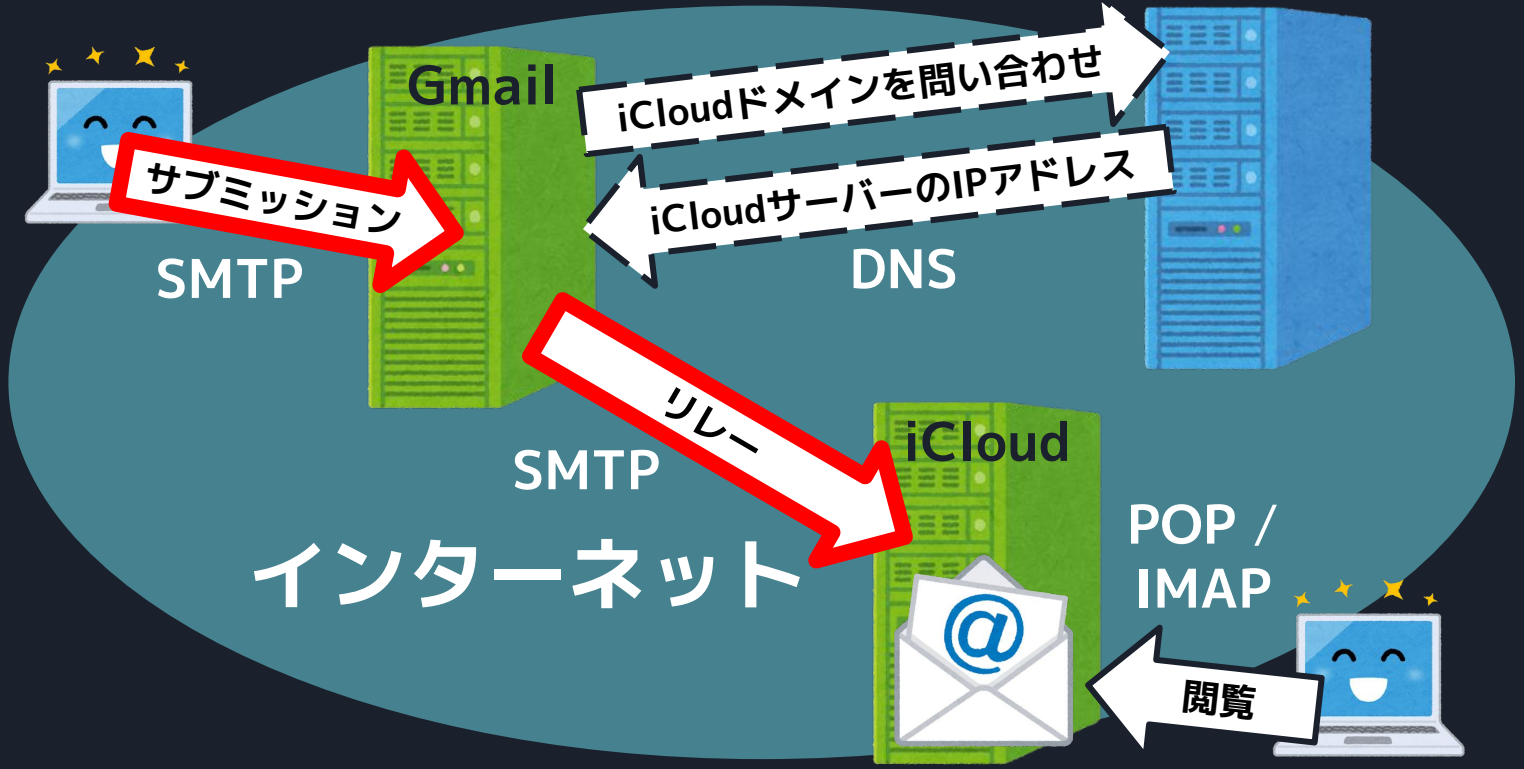
- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- **SMTPプロトコル**
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

SMTPプロトコル

メールを成り立たせている操作



一番大事な部分を支えるのが、SMTP！



SMTPとは

- **SMTP** (Simple Mail Transfer Protocol)
  - **メールの送信**に特化したプロトコル
  - 1982年に制定され、**大枠は今も変わっていない**
- **対話型**のプロトコル
  - httpは、行って帰ってきたら終わり
    - 「ページのデータ下さい」「はい」で表示
  - 自分の情報 / メールの情報を**一つずつ渡し**、それぞれに対してサーバーが応答を返す

プロトコルとしての立ち位置

- **OSI参照モデル**の中では以下の二つに位置する
  - 7: アプリケーション層
  - 6: プレゼンテーション層
  - 「メール」というデータの表現の仕方と、メールの送受信についてをどちらも定義している
- プロトコルとして
  - **TCP** (4: トランスポート層) の上のプロトコル
  - **関連プロトコル**がたくさんある (後述)

SMTPの大まかな流れ

送信側

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

送り主 (MAIL FROM)

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

送り主 (MAIL FROM)

OK

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

送り主 (MAIL FROM)

OK

宛先 (RCPT TO)

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

送り主 (MAIL FROM)

OK

宛先 (RCPT TO)

OK

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

送り主 (MAIL FROM)

OK

宛先 (RCPT TO)

OK

内容 (DATA)

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

送り主 (MAIL FROM)

OK

宛先 (RCPT TO)

OK

内容 (DATA)

OK

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

送り主 (MAIL FROM)

OK

宛先 (RCPT TO)

OK

内容 (DATA)

OK

終了 (QUIT)

受信側

SMTPの大まかな流れ

送信側

自己紹介 (HELO)

OK

送り主 (MAIL FROM)

OK

宛先 (RCPT TO)

OK

内容 (DATA)

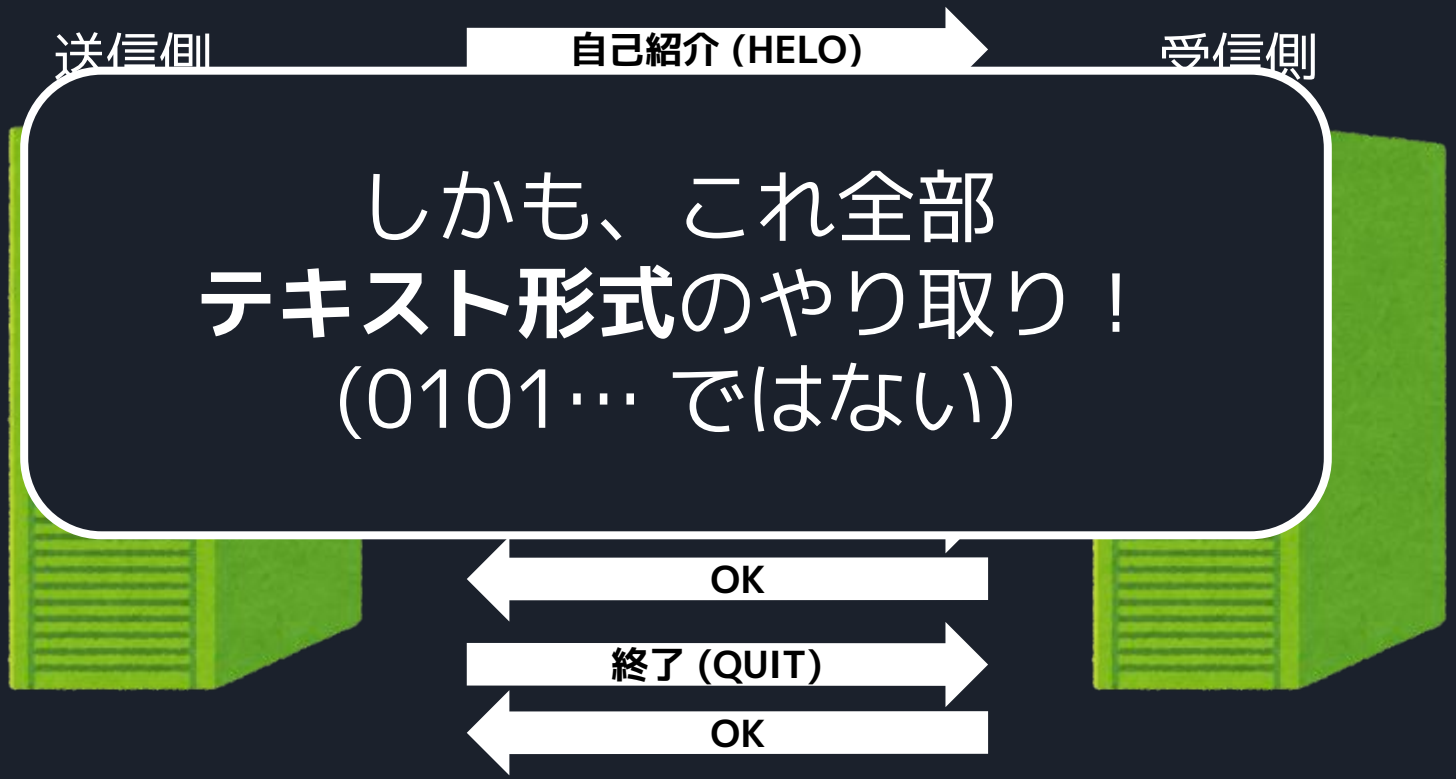
OK

終了 (QUIT)

OK

受信側

SMTPの大まかな流れ



自分でも書けそうな
気がしてきたでしょ？

出来ます。
やりましょう。

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- **SMTPプロトコル**
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- **ハンズオン - 「SMTPを手書きしてみよう！」**
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

ハンズオン
「SMTPを**手書き**してみよう！」

別で**ハンズオン資料を**
用意してありますので、
それに従って進めて下さい！

https://github.com/logica0419/seccamp-mini-ishikawa2025/blob/main/handson-1.md

- サーバーのIPアドレス
  - **163.43.70.121**
- ログインパスワード
  - **ishikawa2025**

https://github.com/logica0419/seccamp-mini-ishikawa2025/blob/main/handson-1.md

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- **ハンズオン - 「SMTPを手書きしてみよう！」**
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

アジェンダ

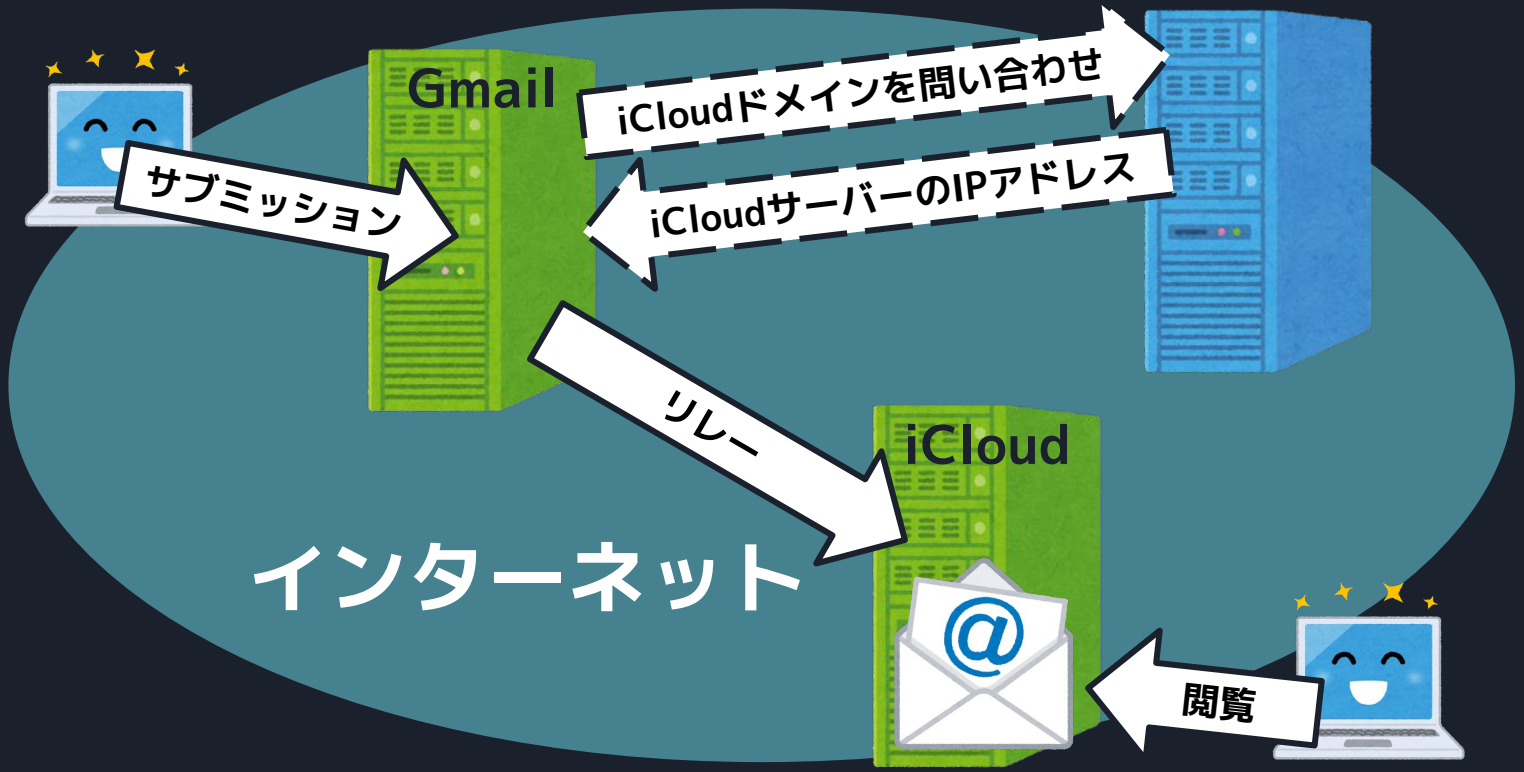
- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- **SMTPを安全にするための仕組み**
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

SMTPを安全にするための仕組み

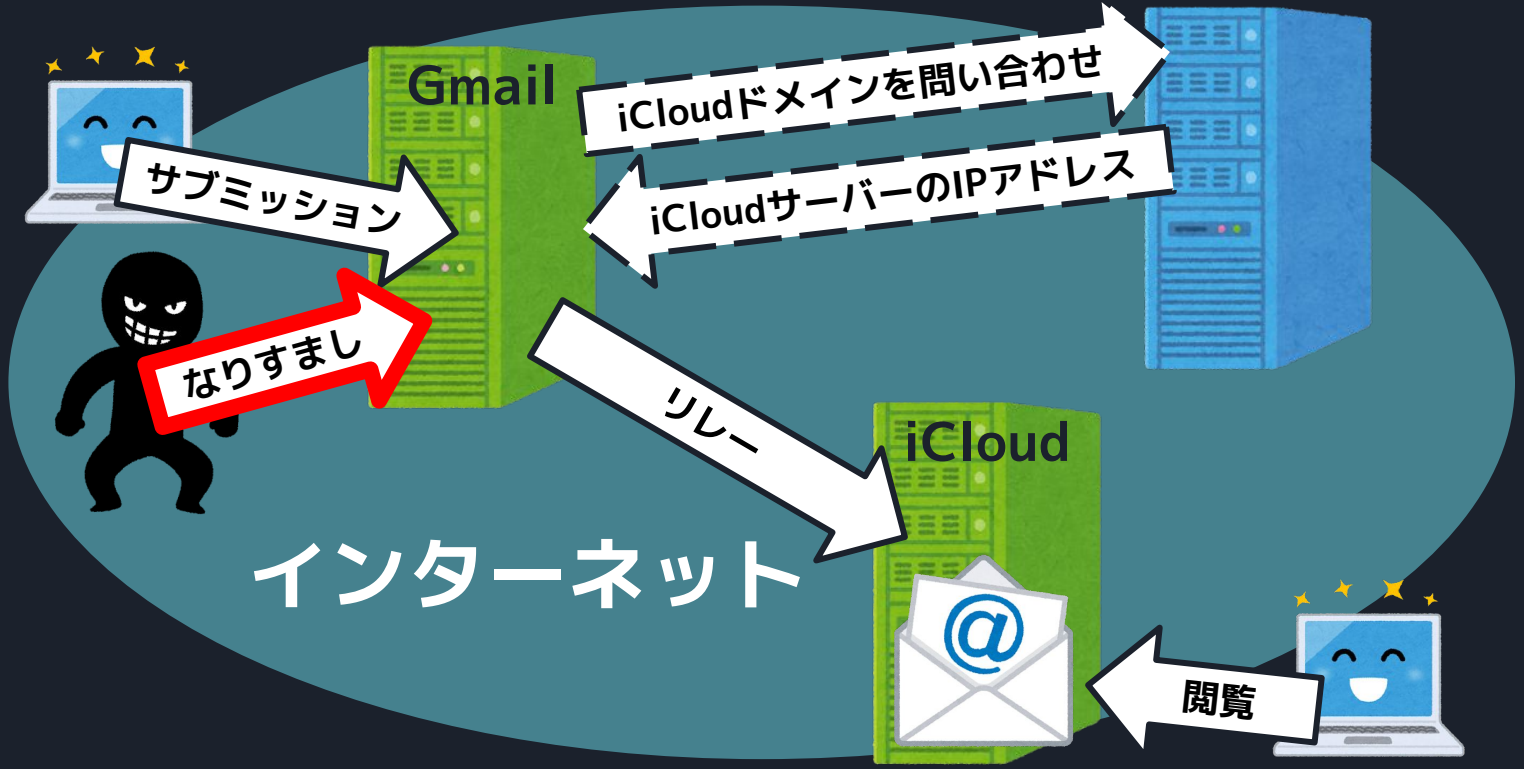
SMTPというプロトコル、
素の状態では大量の
危険にさらされている

ハンズオンで体感してもらえたはず

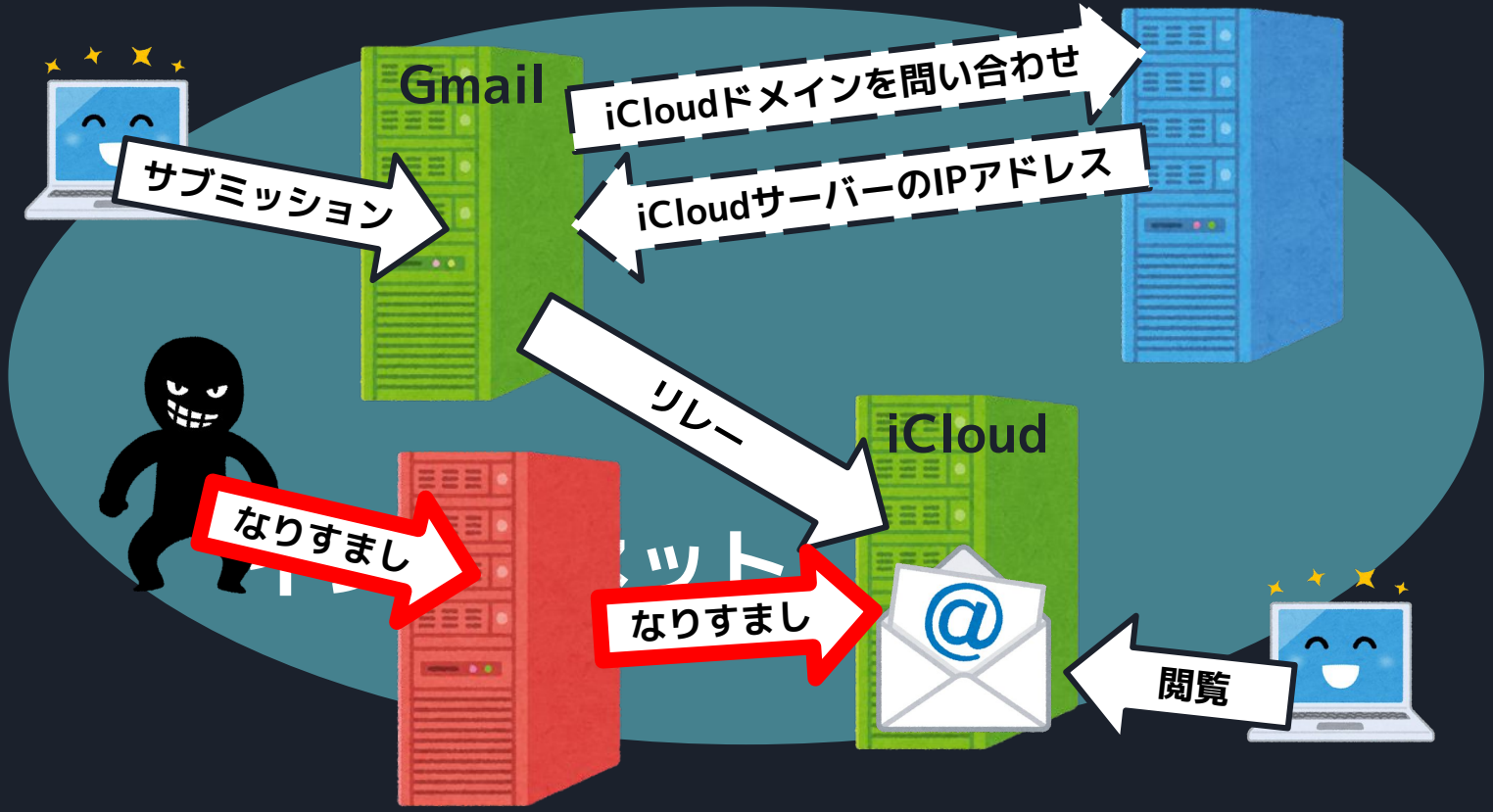
メールで起こり得る危険



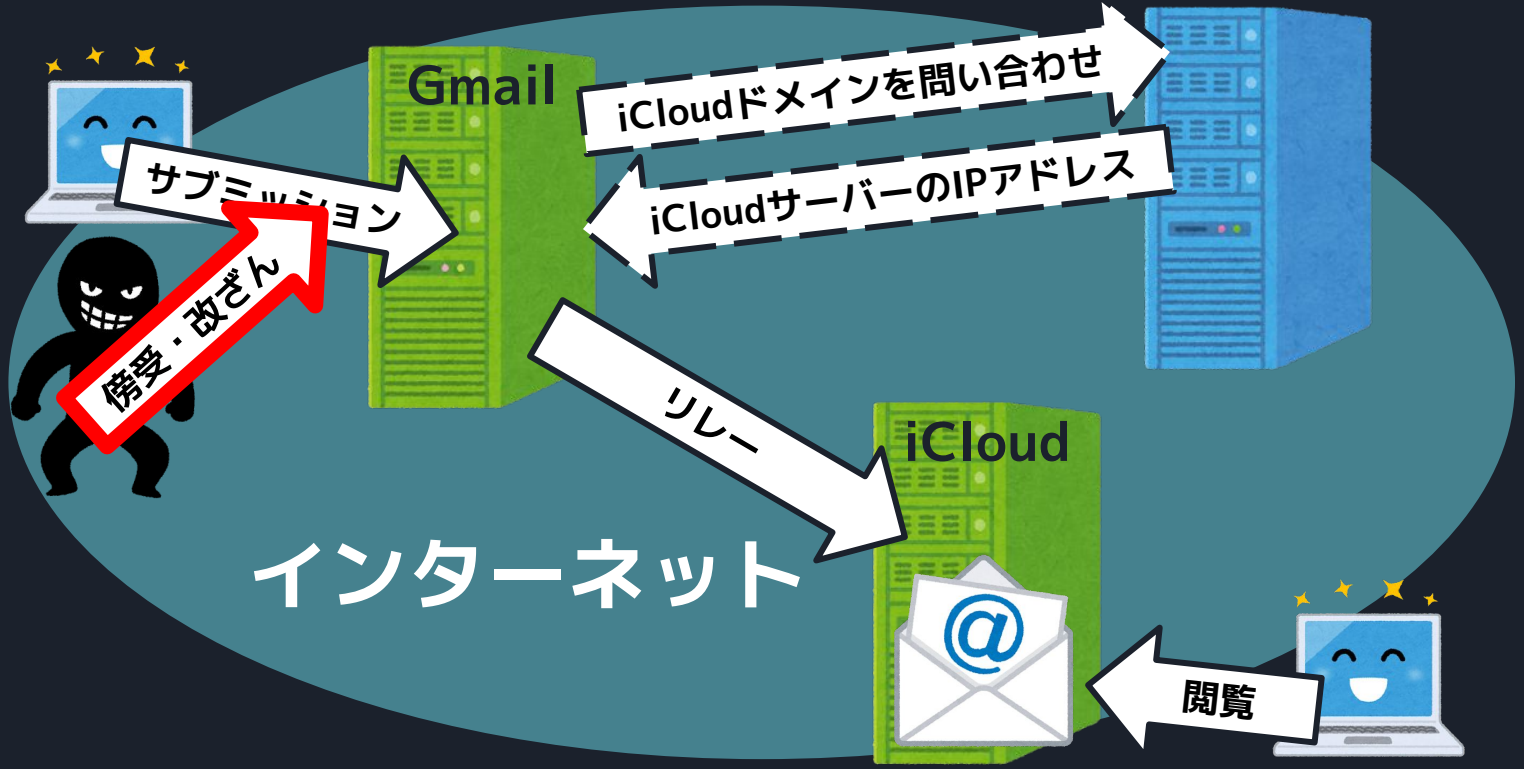
メールで起こり得る危険



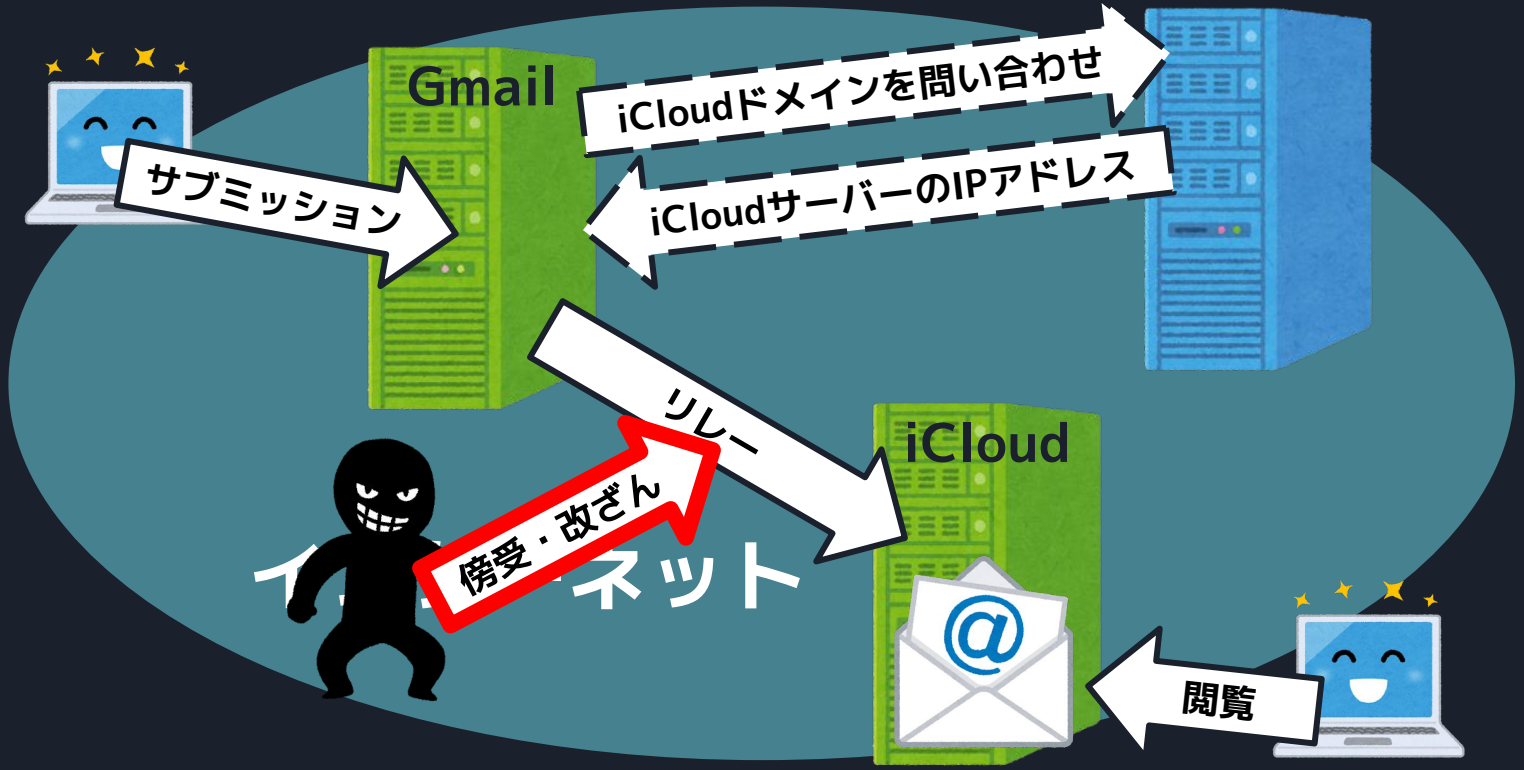
メールで起こり得る危険



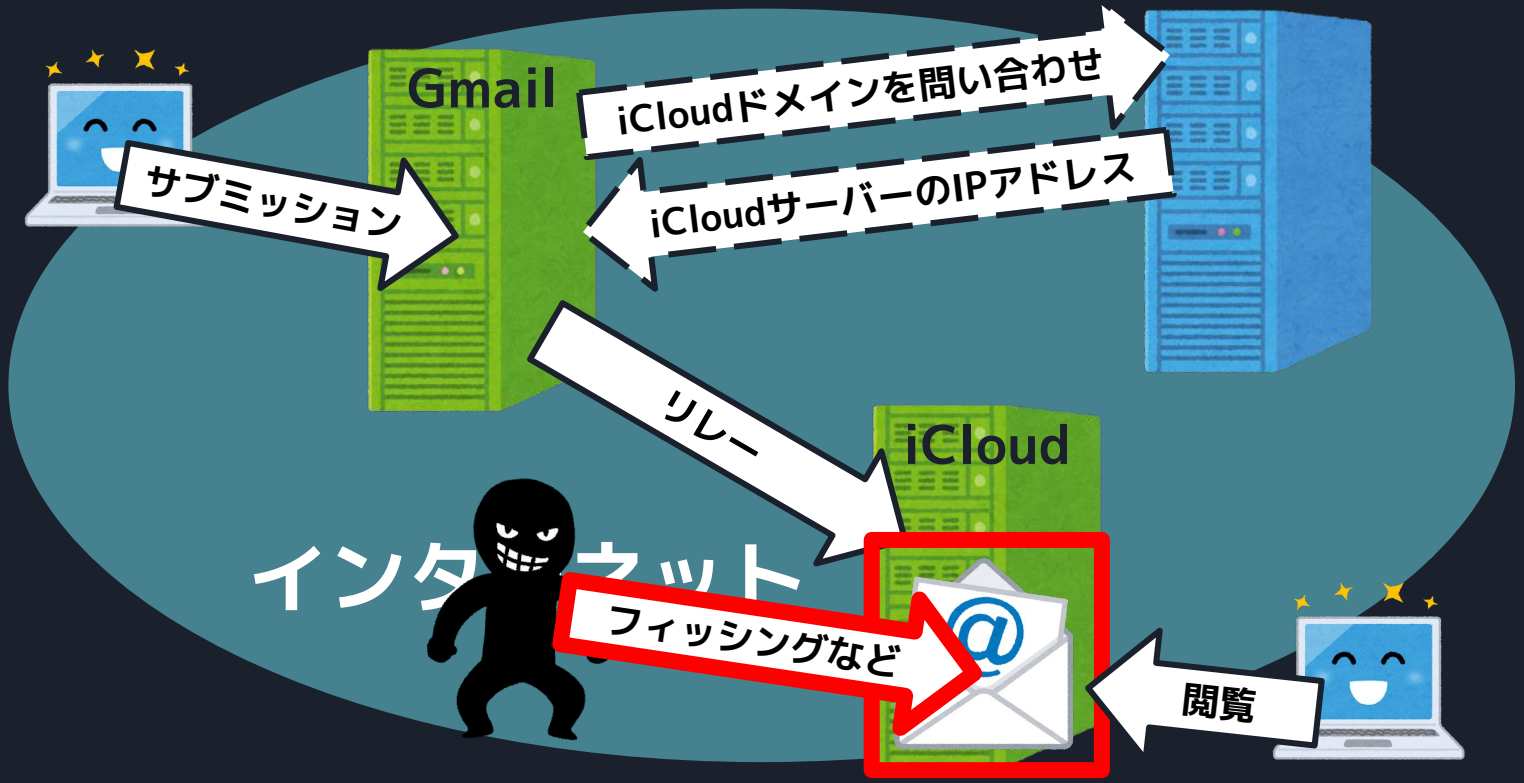
メールで起こり得る危険



メールで起こり得る危険



メールで起こり得る危険

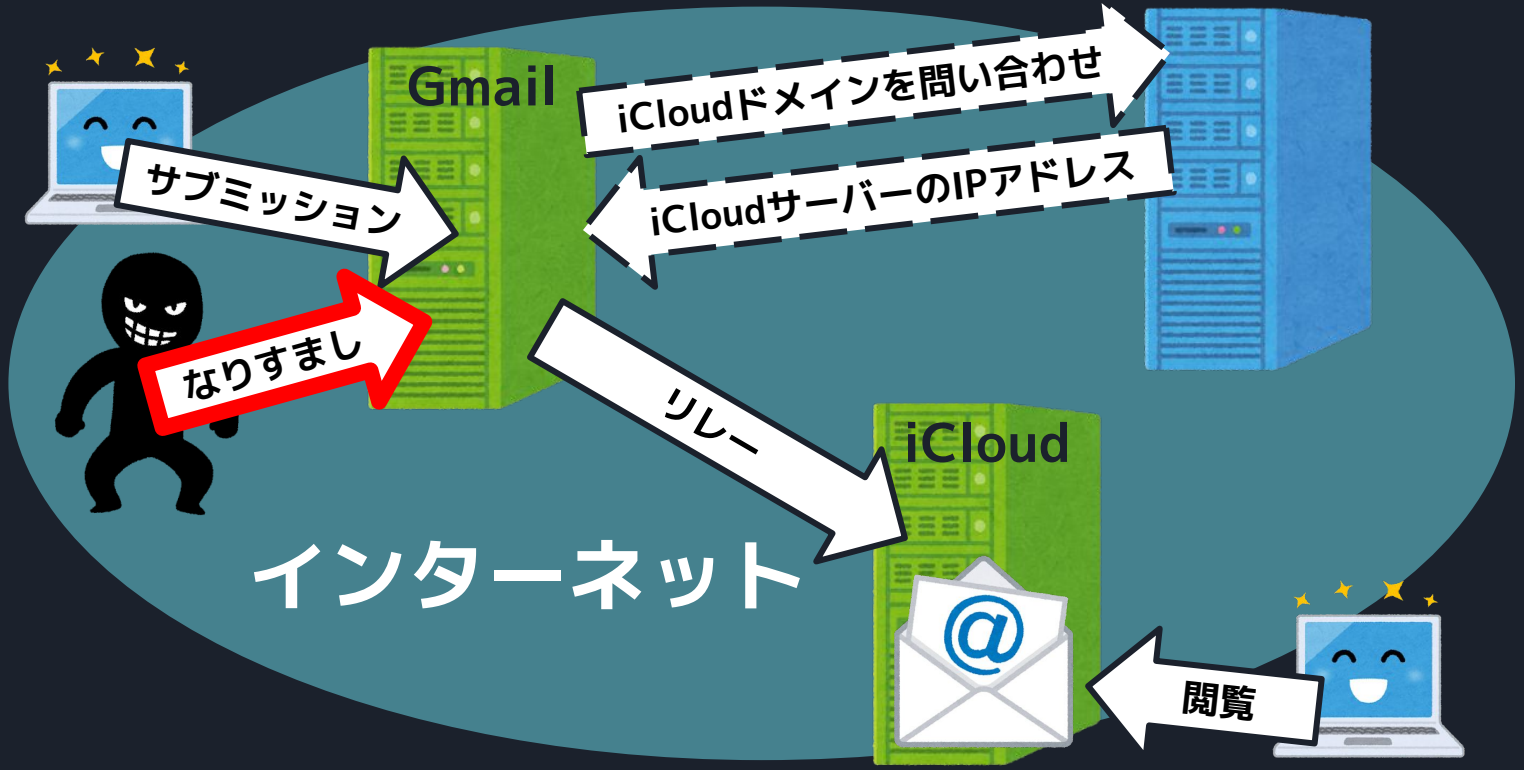


対策は2つ

- **プロトコル**
  - プロトコル (**仕様**) としてセキュリティ対策を定義
  - 広く**使ってもらいやすい**
  - **汎用的でなければいけない**ため、効力が薄いことも
- **メールサーバー独自**の対策
  - メールサーバーが独自に対策を実装
  - **目的に特化**させやすい
  - **実装依存**なため、統一化はしにくい

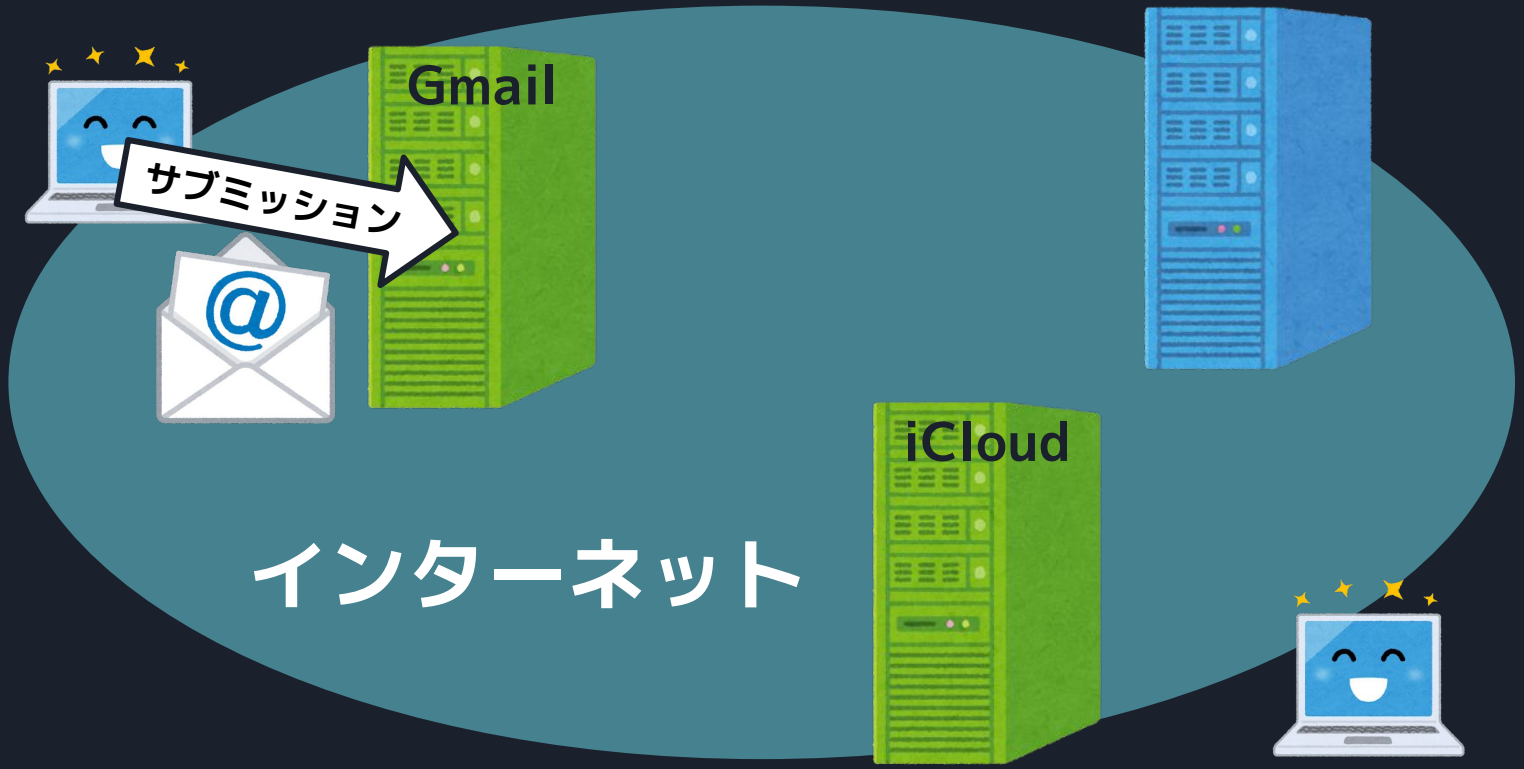
脅威への対策**プロトコル**
～ユーザーなりすまし編

このタイプのなりすまし



POP before SMTP

ユーザーが送信する (サブミッション)



…の前に

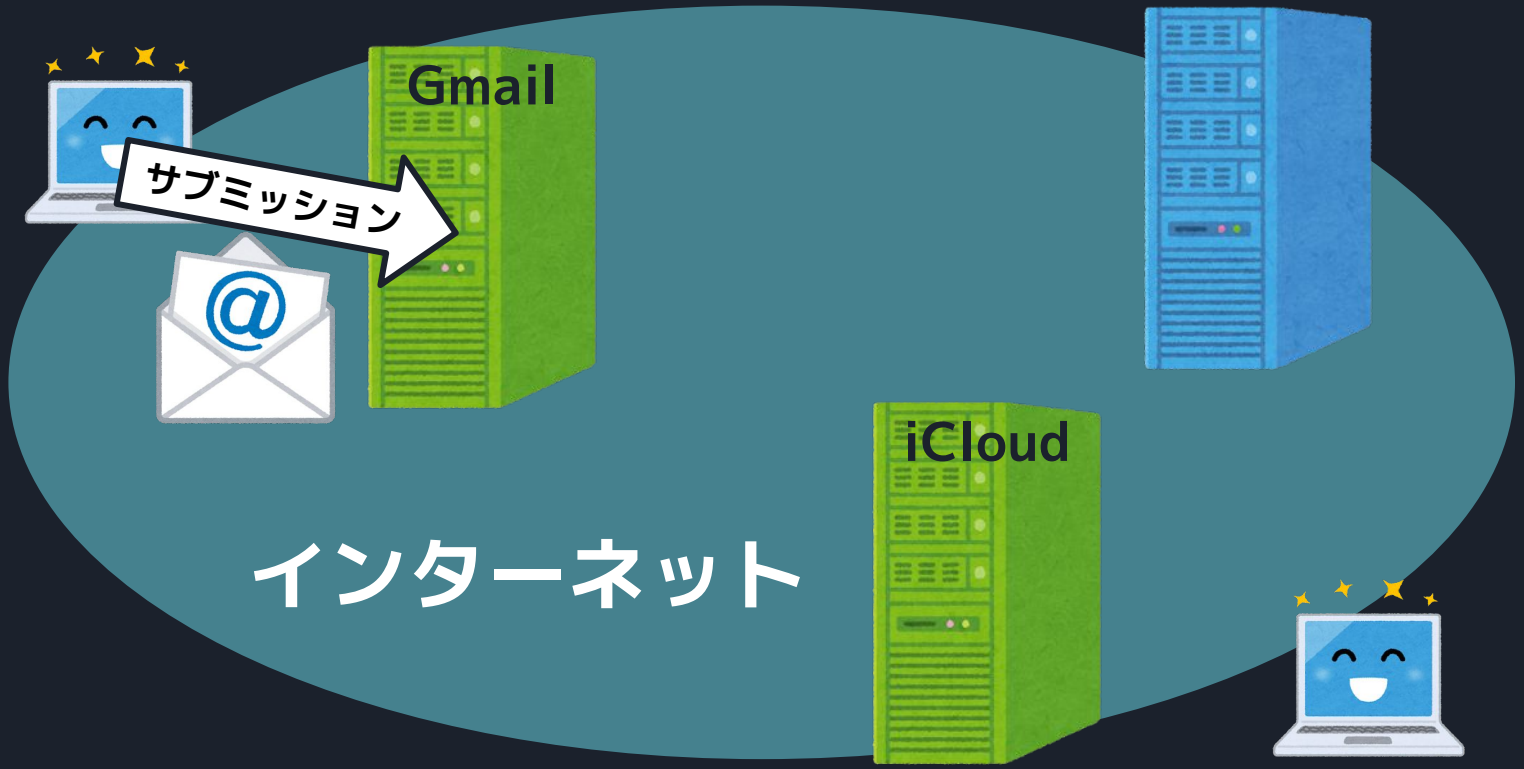
からの

Pop before SMTP

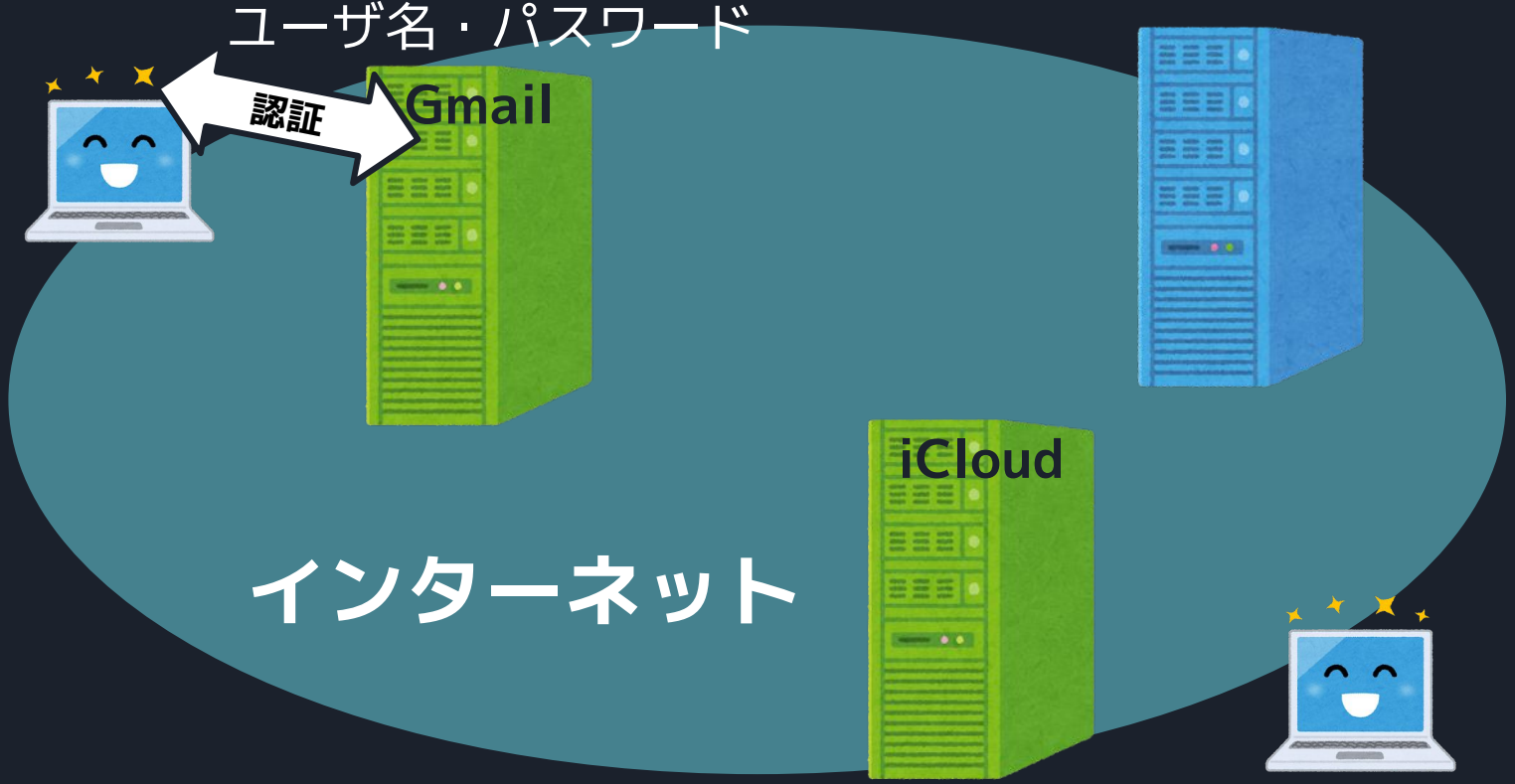
- 「**受信できる人間はこのメールサーバーの利用者**だね」問う考え方に基づく
  - **自分が受信できるメールアドレス**を持っていれば、メールサーバを利用できる
- Fromの書き換えを**検知する仕組みではない**ので、なりすましの強固な予防策とはならない
- この後の**SMTP AUTHの方がよく使われている**のでマイナーなプロトコル

SMTP AUTH (SASL認証)

ユーザーが送信する (サブミッション)



…の前に



からの

ユーザ名・パスワード

SMTP AUTH

- サブミッション前に、何らかの形で**ユーザー名とパスワード**をメールサーバーに送って認証
  - いわゆるアカウントの認証
- 認証情報の送り方によってタイプがある
  - SMTP AUTH **PLAIN** - まとめる、弱い
  - SMTP AUTH **CRAM-MD5** - まとめる、強いが面倒
  - SMTP AUTH **LOGIN** - 2つをバラバラに、弱い

SMTP AUTH = SASL認証

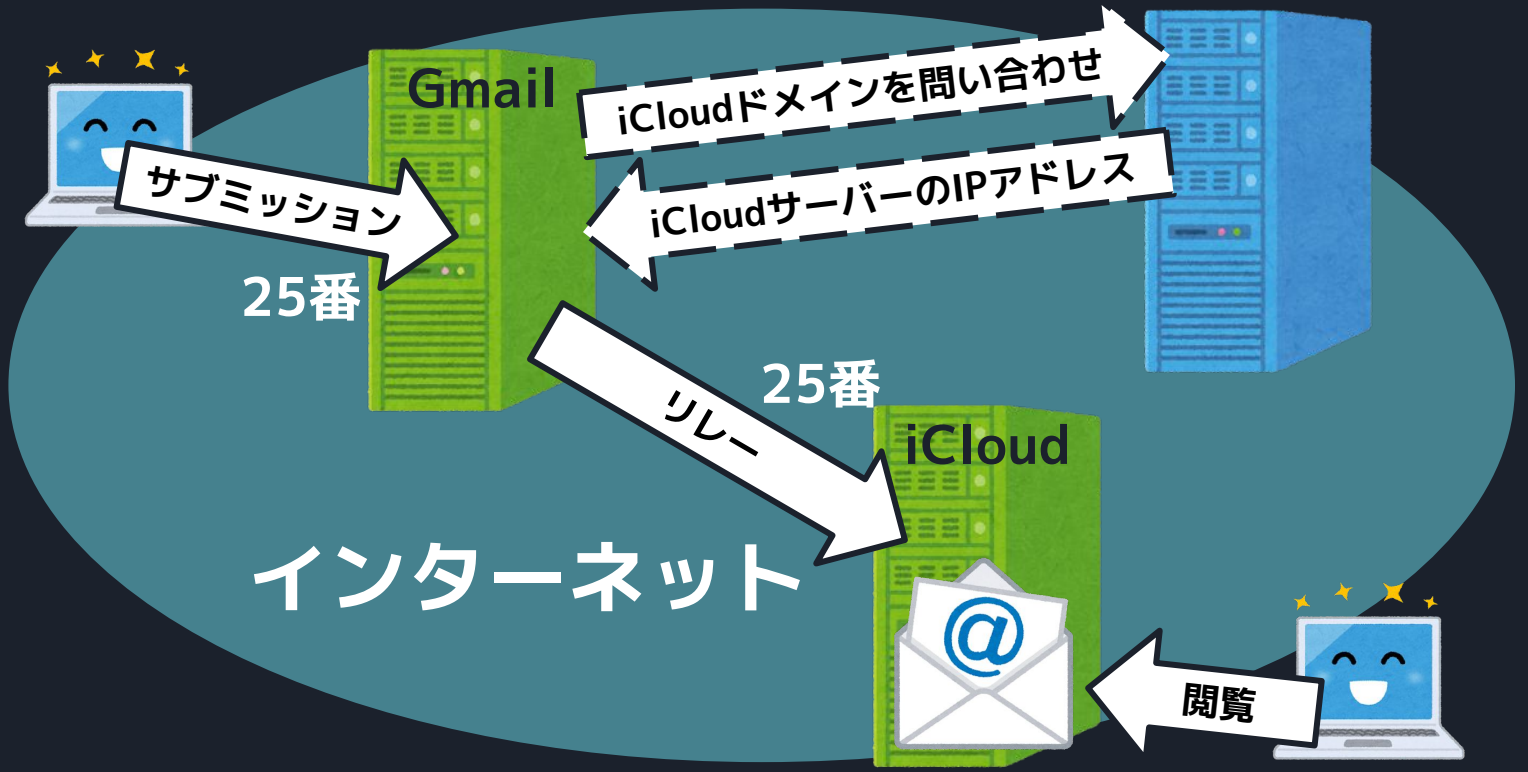
- **SASL** (Simple Authentication and Security Layer)
  - インターネットにおける、**認証とセキュリティのためのフレームワーク**
  - 基本的に**どんなプロトコルとも**組み合わせて使うことができる
- 紹介したもの以外にも、ワンタイムパスワードなどがサポートされている

OP25B

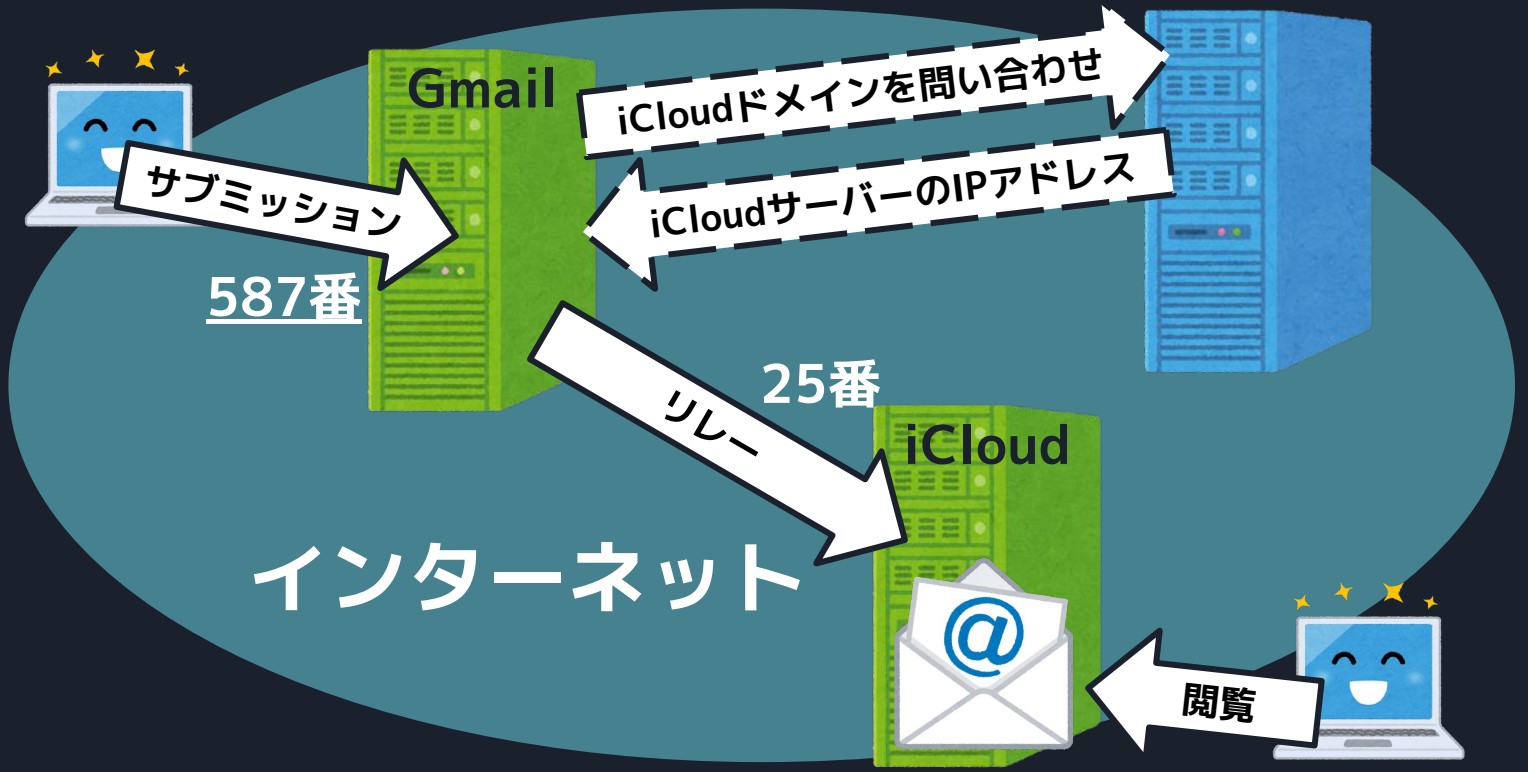
OP25B

- **リレー用ポートとサブミッション用ポートを分けよう！**という取り組み
- リレーでいちいち**認証は出来ない**
  - 全てのメールサーバーが、**他の全てのサーバーへの認証情報**を持たなくてはいけなくなる
- サブミッションは**認証を必ずさせたい**
  - セキュリティを強制させたい

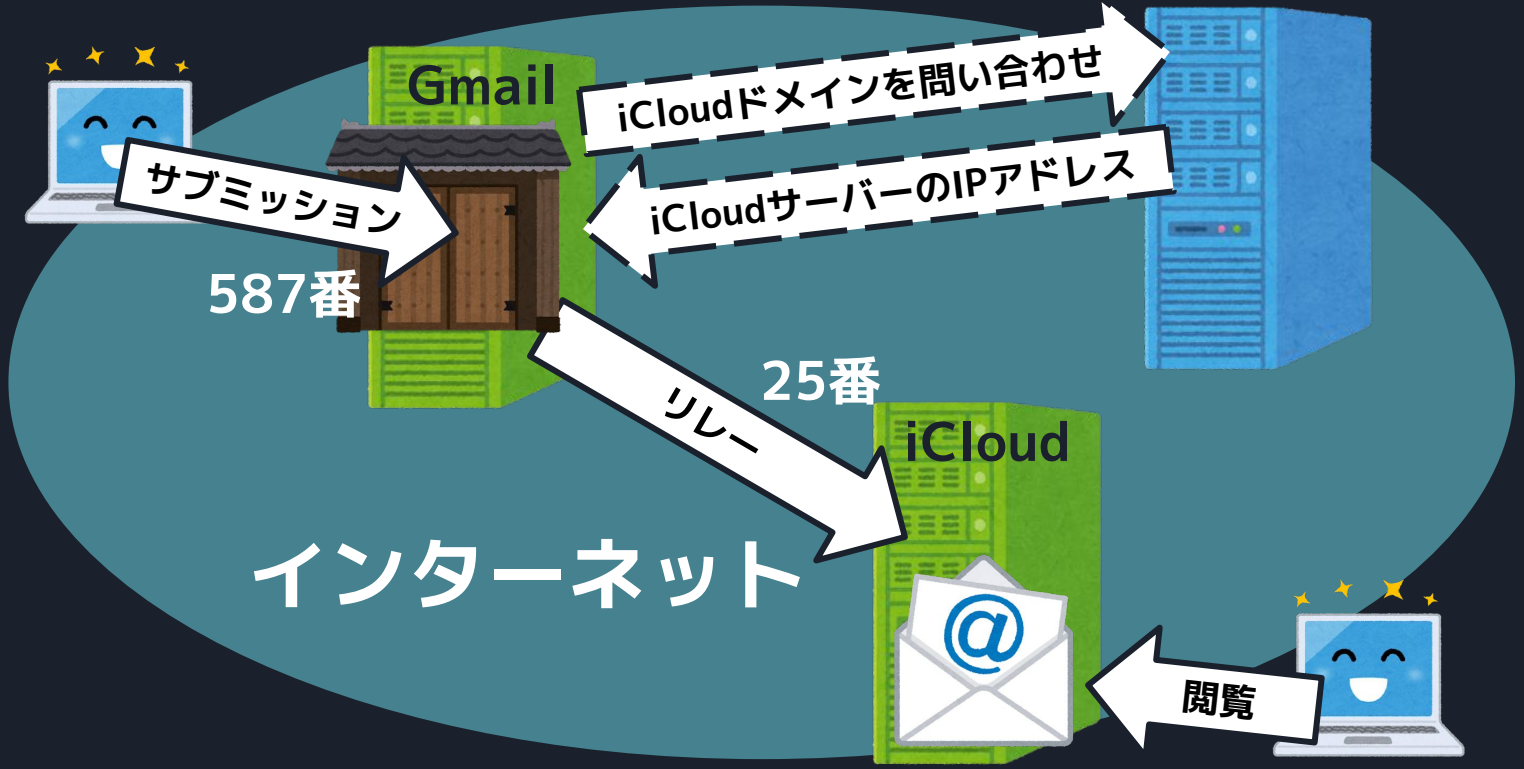
こうだったのを


Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
25番
25番
リレー
iCloud
インターネット
閲覧

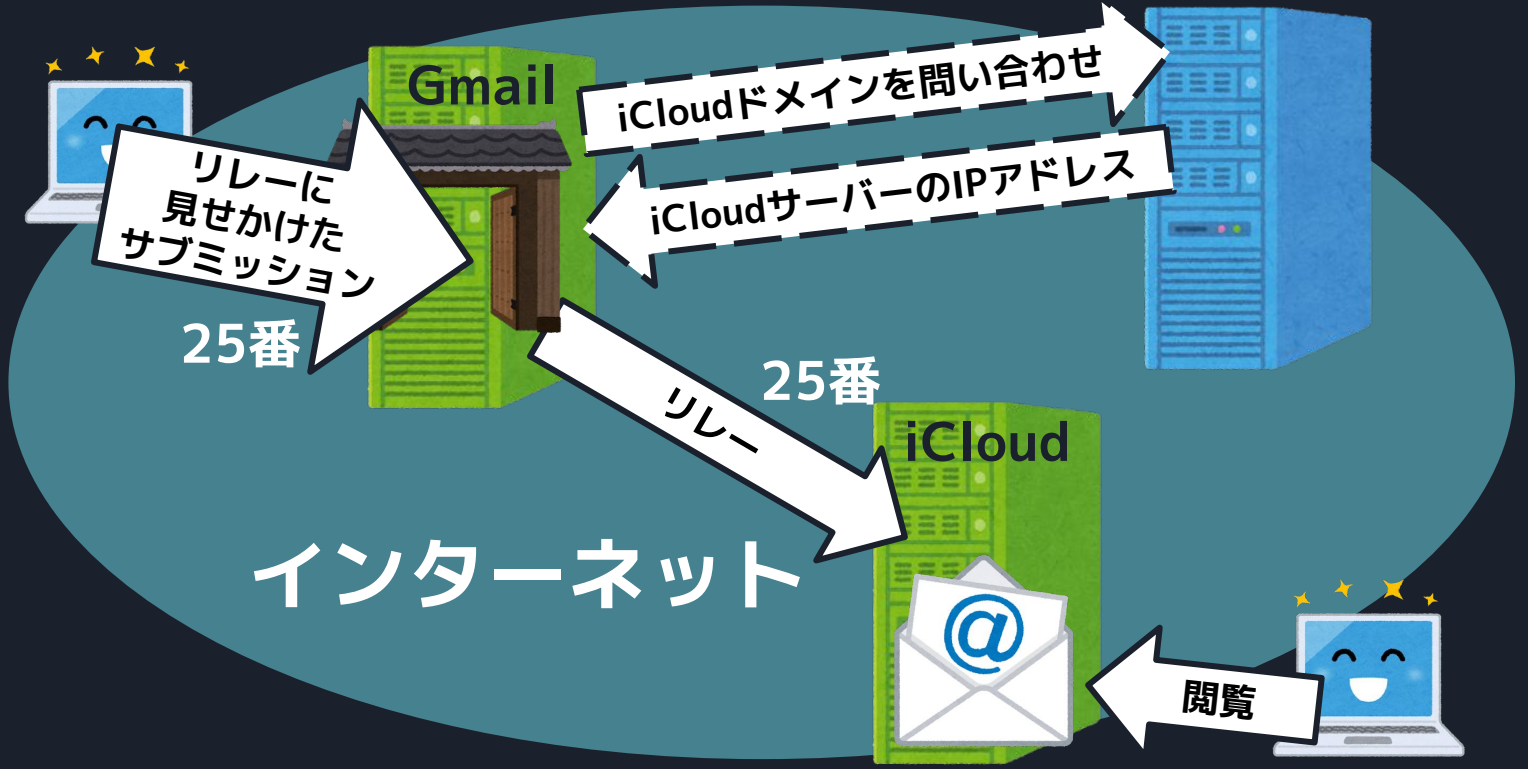
ちゃんと分けた


Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
587番
25番
リレー
iCloud
インターネット
閲覧

サブミッションポートは堅牢にする



でもアクセスするポートさえ変えれば…

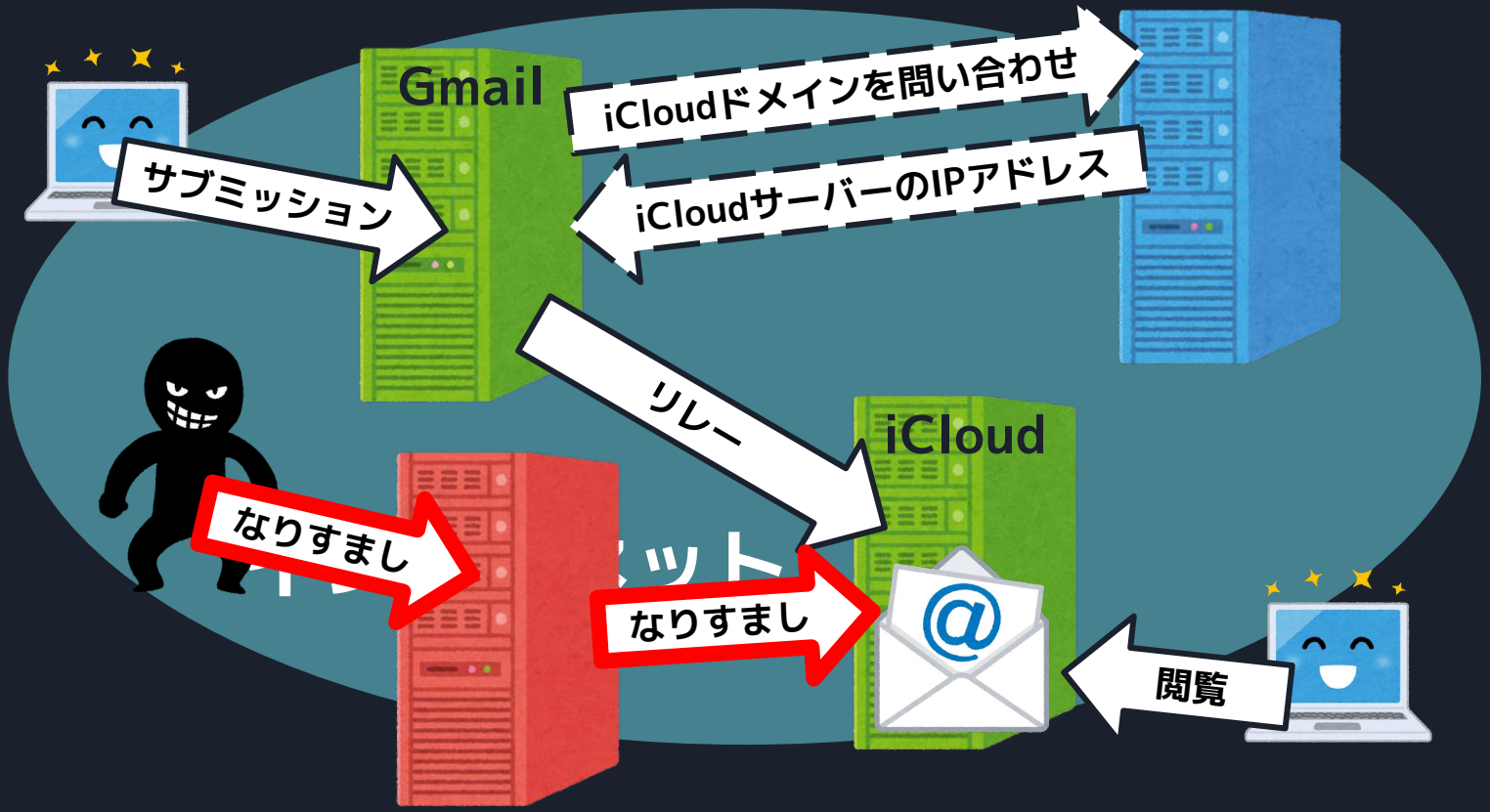


OP25B

- 一般ユーザーに**587番ポートしか使わせない**ために最終的に**ISP (通信事業者)** がインターネット経由で**25番ポート**にアクセスする行為を**封じた**
- 「OP25B」と調べると「外部の25番ポートへの接続を出来ないようにする」という内容しか出てこない
  - 背景情報をきちんと知ることで、初めてきちんと理解できるタイプの取り組み

脅威への対策**プロトコル**
～サーバー / ドメイン
なりすまし編

このタイプのなりすまし



SPF (Sender Policy Framework)

下準備

IPアドレス
x.x.x.x

インターネット

下準備

Gmail

IPアドレス

x.x.x.x

Gmailのサーバーは
x.x.x.xにあります！

iCloud

インターネット

おさらい: DNSサーバー

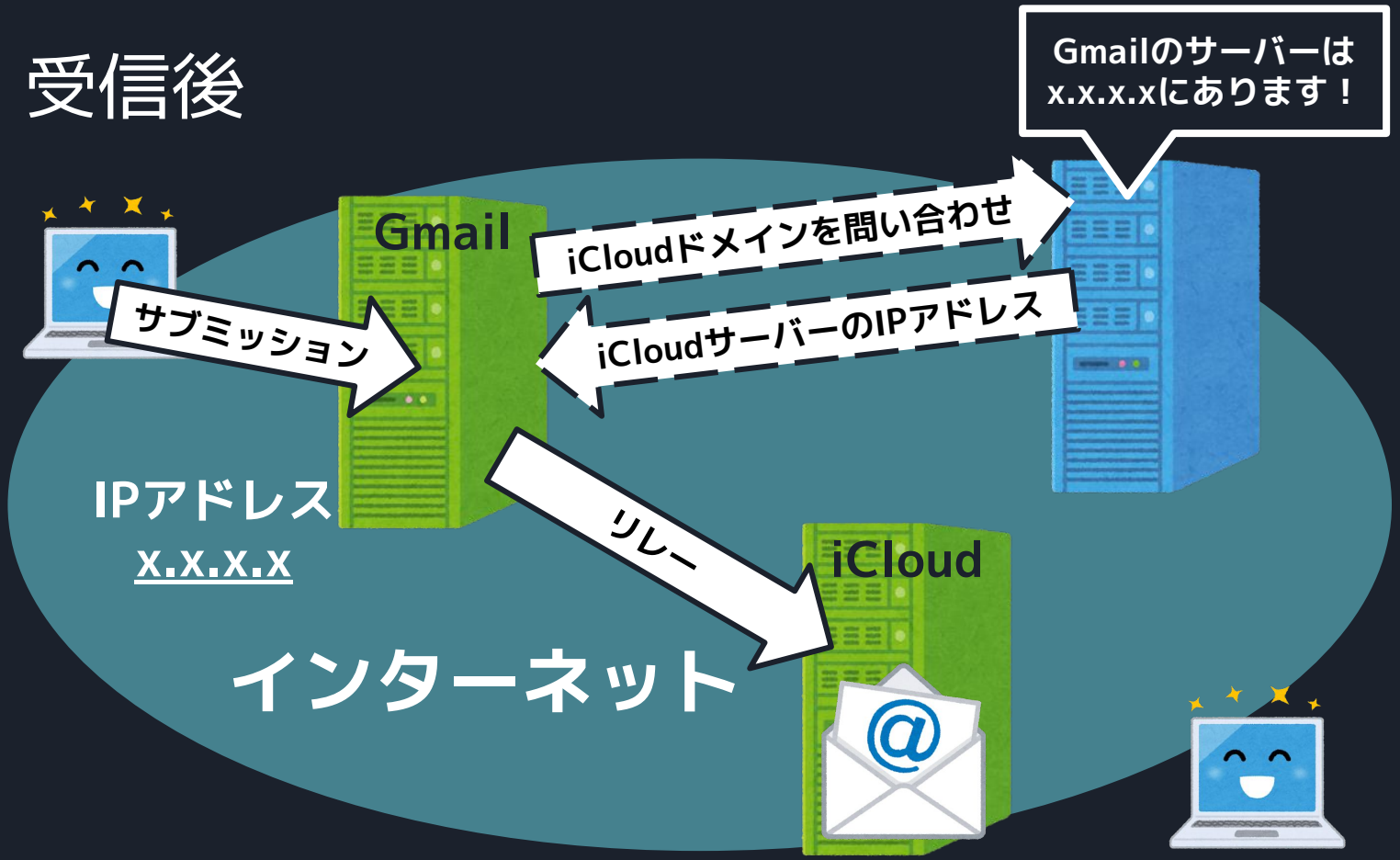
- **ドメイン名**と**IPアドレス**を紐付けるための仕組みを提供する
- おまけ機能として、**ドメイン名**と**任意のテキストデータ**を紐づける仕組みも提供している
  - これが、**メールを安全にするための**
    プロトコルで大活躍します

おさらい: DNSサーバー

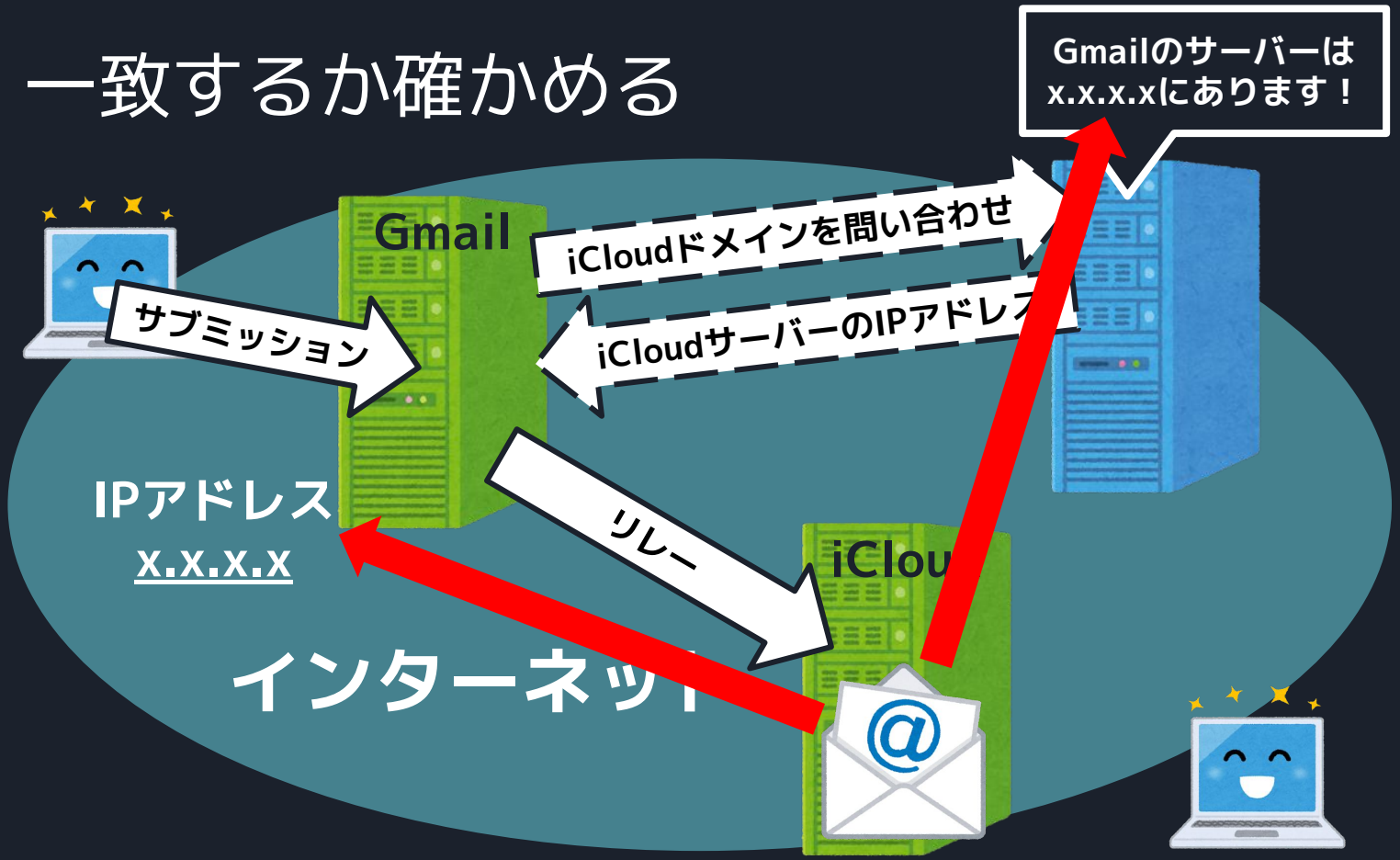
- **ドメイン名とIPアドレス**を紐付けるための仕組みを
  - これが、**メールを安全にするための**
    プロトコルで大活躍します

SPFにおいては、サーバーのIPを
この「**任意テキスト**」として
DNSに保存しておく

受信後


Gmailのサーバーは
x.x.x.xにあります！
Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
IPアドレス
x.x.x.x
リレー
iCloud
インターネット

一致するか確かめる


Gmailのサーバーは
x.x.x.xにあります！
Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
リレー
IPアドレス
x.x.x.x
iClou
インターネット

SPF (Sender Policy Framework)

- DNSサーバーに**正しいメールサーバーのアドレスを保存**しておくことで、受信後**メールを送ってきたサーバー**と**正しいメールサーバー**のアドレスが一致するか確認できるようにするプロトコル
  - 不審なメールサーバーを検知できる
- サーバー運用者の**設定が面倒**
- 間にサーバーが挟まってしまうと**意味を成さない**

DKIM (DomainKeys Identified Mail)

前提知識: 公開鍵暗号

- 公開鍵 / 秘密鍵という、みんなに**知られても良い鍵**と**知られてはいけない鍵**の2種類を用意する暗号化
  - cf. 共通鍵暗号
- 鍵 = 一定のルールで作られたランダムな**テキスト**
- 特徴
  - 公開鍵で暗号化した文書は、秘密鍵で**しか**復号できない

特徴の部分をより分かりやすく図解

秘密鍵 (知られてはいけない)　公開鍵 (知られても良い)

公開鍵暗号の使われ方

下準備

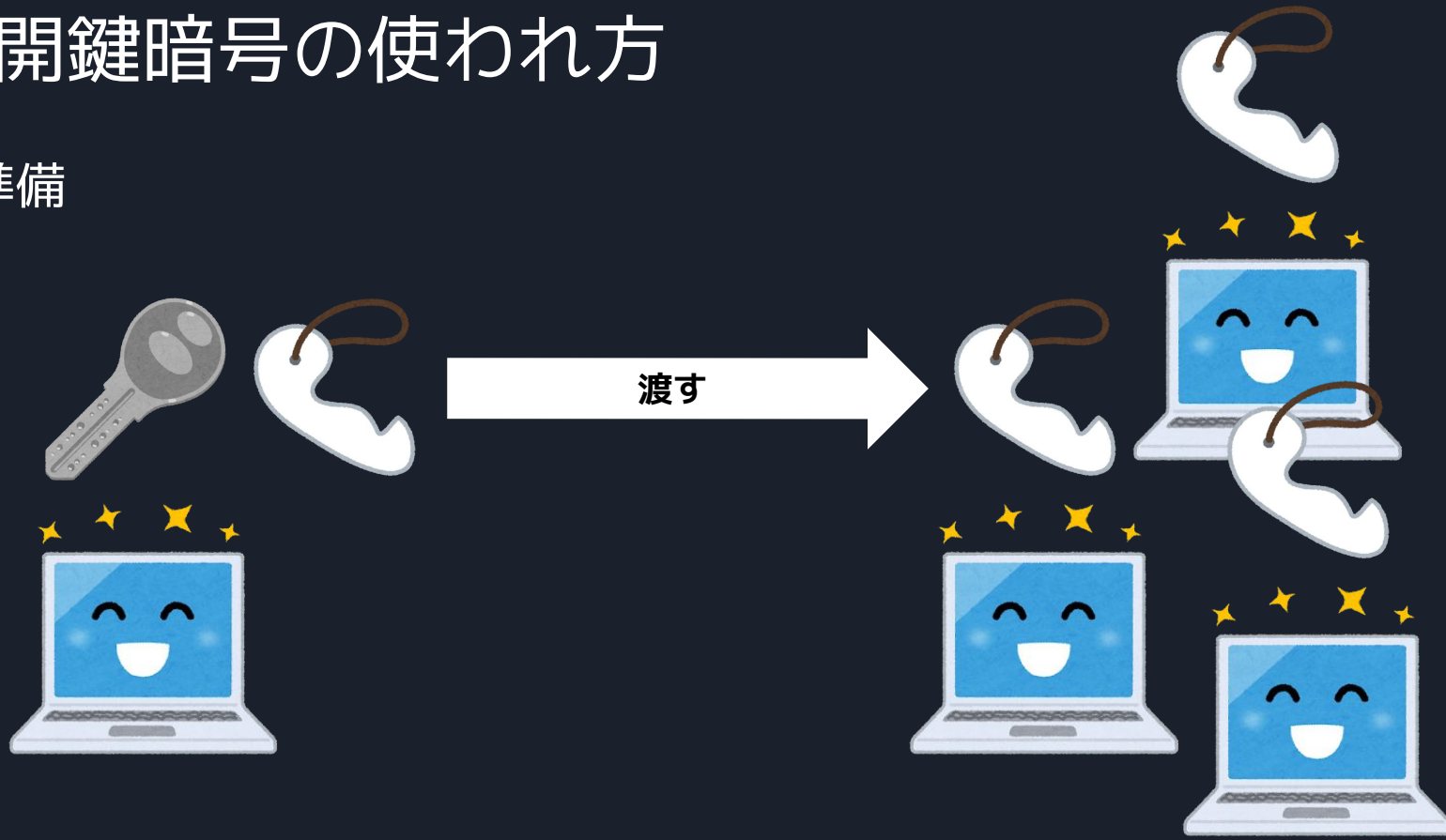
← 必ず**セットで作られる**

公開鍵暗号の使われ方

下準備

公開鍵暗号の使われ方

下準備



公開鍵暗号の使われ方

秘密鍵を持つ人しかわからないデータを作る (保護)

公開鍵暗号の使われ方

秘密鍵を持つ人しかわからないデータを作る (保護)

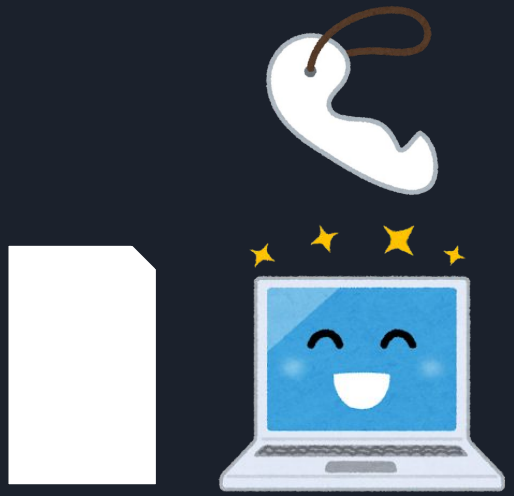
受け手

送り手

公開鍵暗号の使われ方

秘密鍵を持つ人しかわからないデータを作る (保護)

受け手



送り手

公開鍵暗号の使われ方

秘密鍵を持つ人しかわからないデータを作る (保護)

受け手

送り手

公開鍵暗号の使われ方

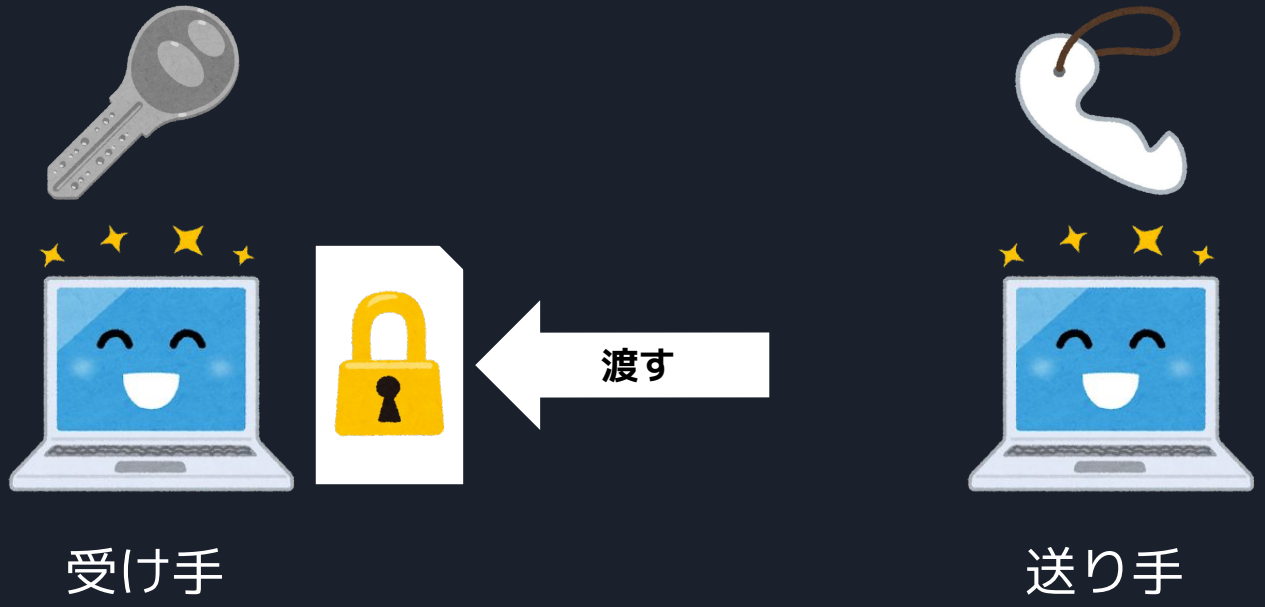
秘密鍵を持つ人しかわからないデータを作る (保護)

- 公開鍵で暗号化した文書は、
  秘密鍵でしか復号できない

受け手　　送り手

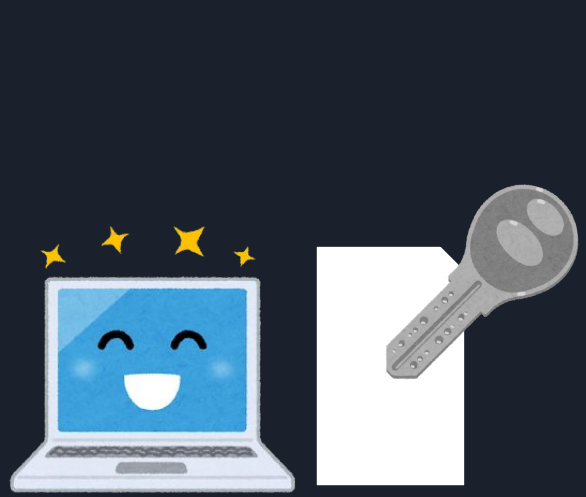
公開鍵暗号の使われ方

秘密鍵を持つ人しかわからないデータを作る (保護)



公開鍵暗号の使われ方

秘密鍵を持つ人しかわからないデータを作る (保護)



受け手

送り手

逆転の発想: 署名

- 署名鍵 / 検証鍵という、みんなに知られても良い鍵と知られてはいけない鍵の2種類を使って、**正しい人が作ったデータ**であることを確認する技術
- 特徴
  - 署名鍵で加工した文書 (署名) は、検証鍵で**必ず**復号できる
- 初期のアイデアは**公開鍵暗号の逆転**だったが、そこから独自に発展を遂げている

特徴の部分をより分かりやすく図解

署名鍵 (知られてはいけない)　検証鍵 (知られても良い)

署名の使われ方

送り主が**署名鍵を持っているか**確かめる (証明)

署名の使われ方

送り主が**署名鍵を持っているか**確かめる (証明)

送り手

受け手

署名の使われ方

送り主が**署名鍵を持っているか**確かめる (証明)

送り手

受け手

署名の使われ方

送り主が**署名鍵を持っているか**確かめる (証明)

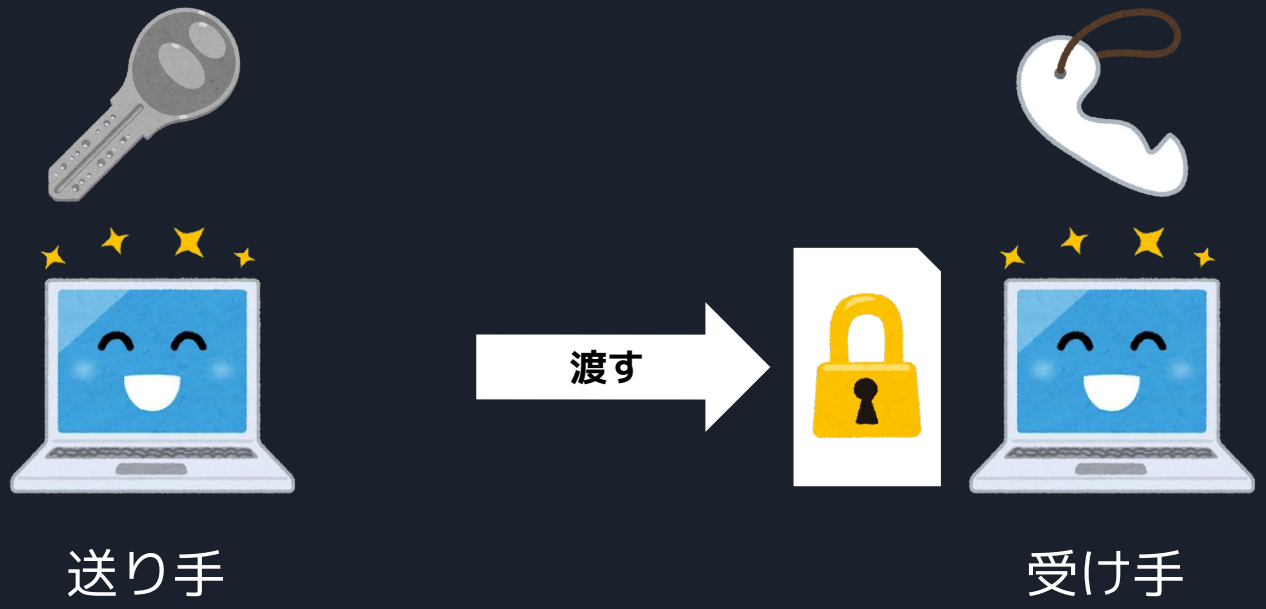
- 署名鍵で加工した文書 (署名) は検証鍵で必ず復号できる

送り手

受け手

署名の使われ方

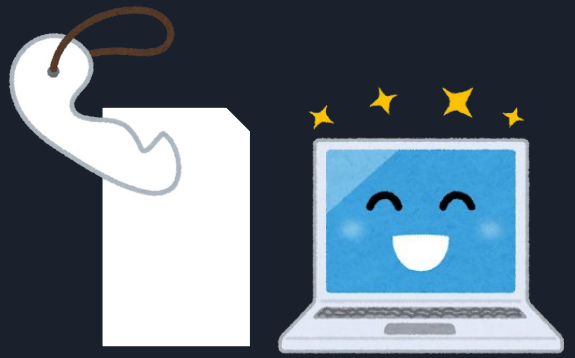
送り主が**署名鍵を持っているか**確かめる (証明)



署名の使われ方

送り主が**署名鍵を持っているか**確かめる (証明)

送り手



受け手

署名の使われ方

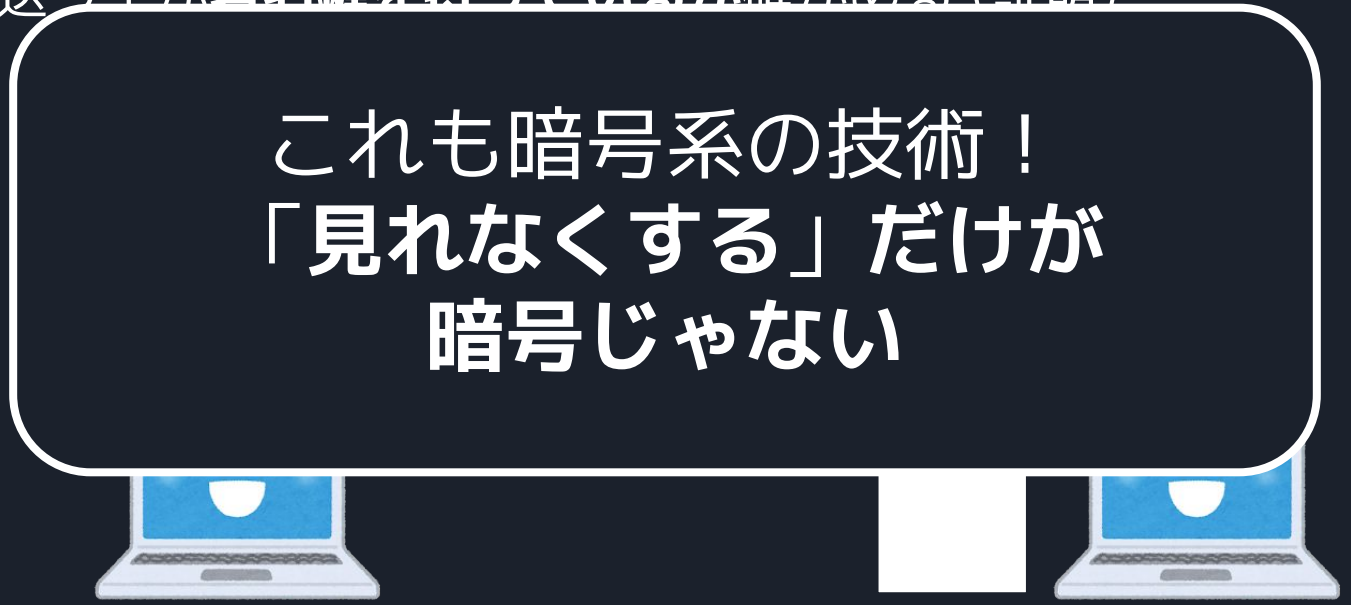
送り主が**署名鍵を持っているか**確かめる（証明）

正しく復号できた＝
正しい署名鍵で加工された
データだとわかる！

送り手　　受け手

署名の使われ方

送り主が**署名鍵を持っているか**確かめる（証明）



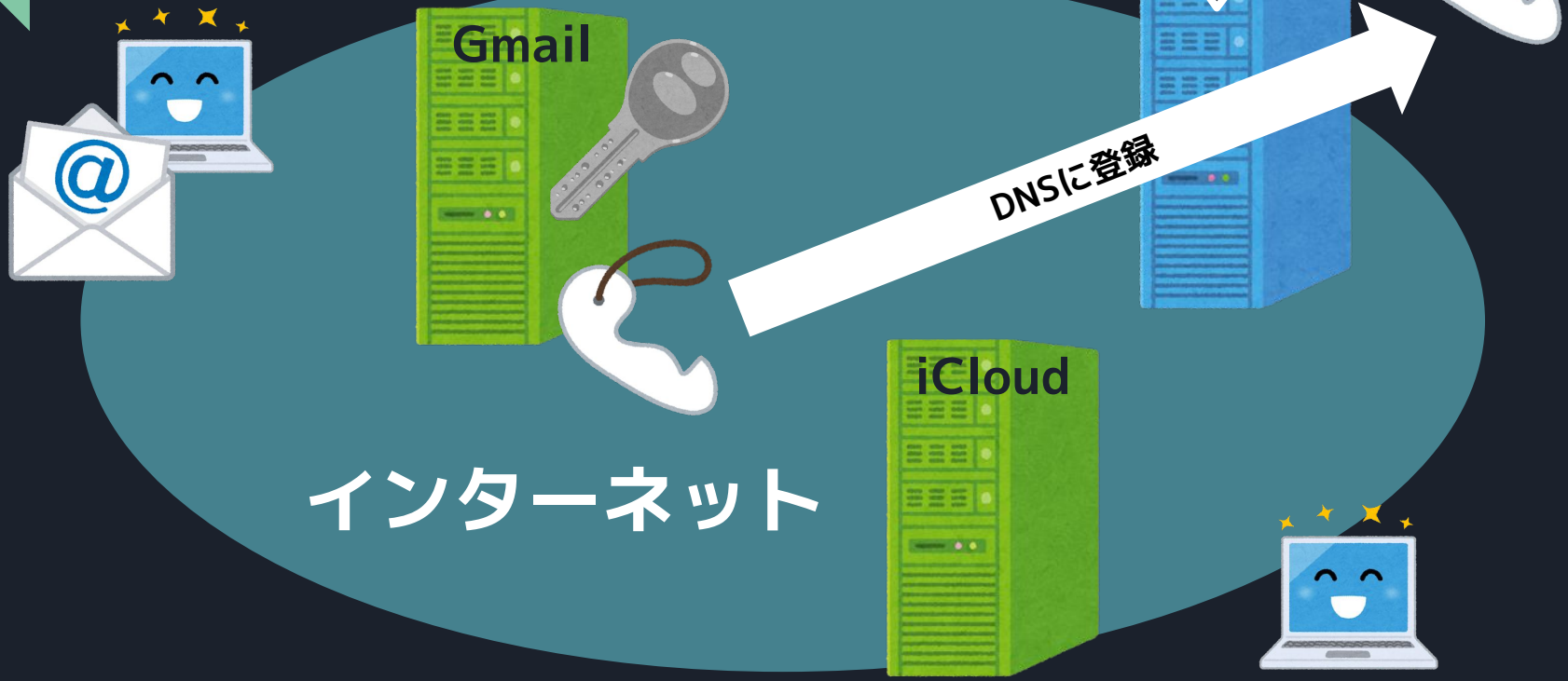
本題のDKIM

署名鍵　　検証鍵　　を用意

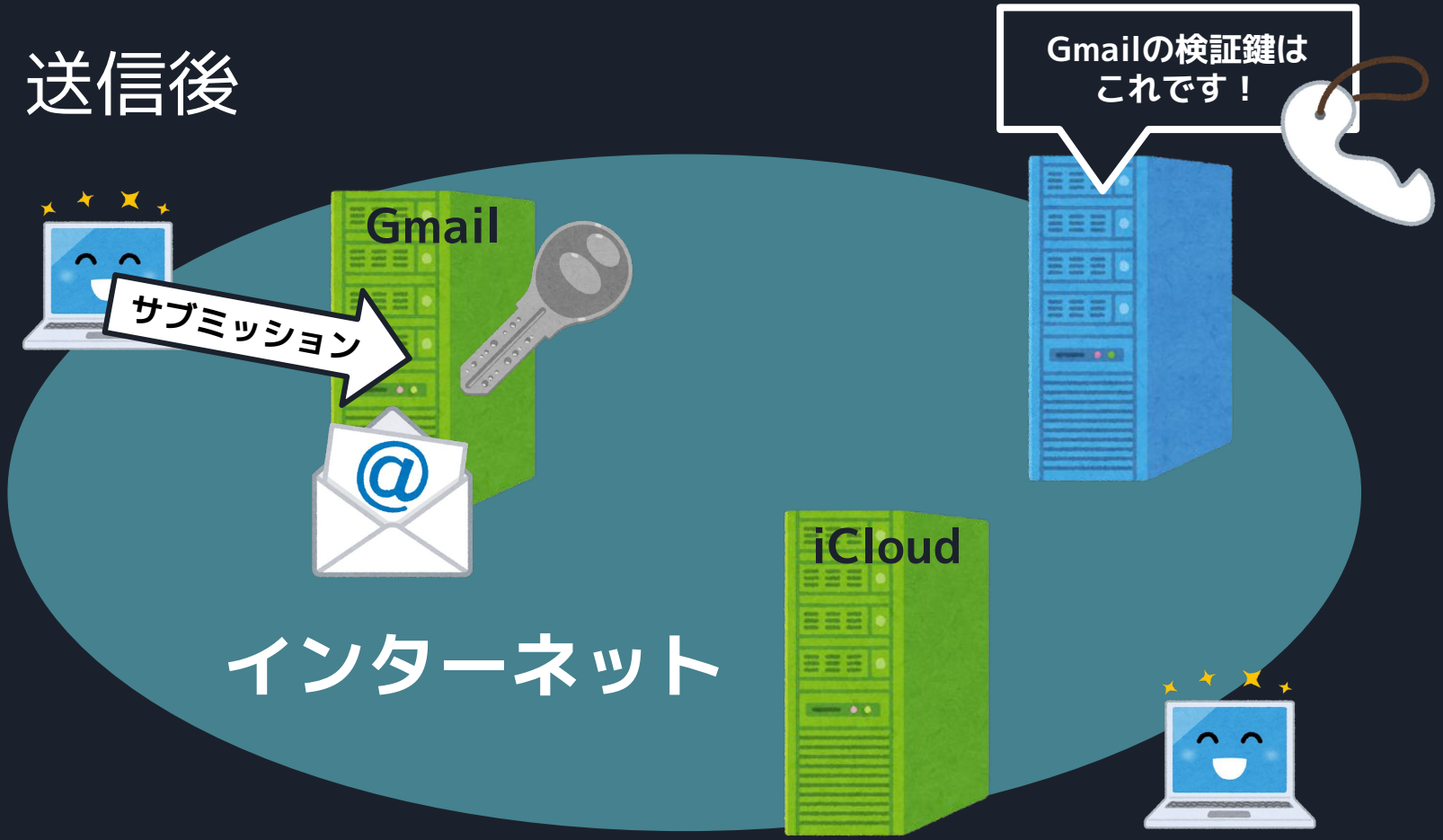
下準備

インターネット

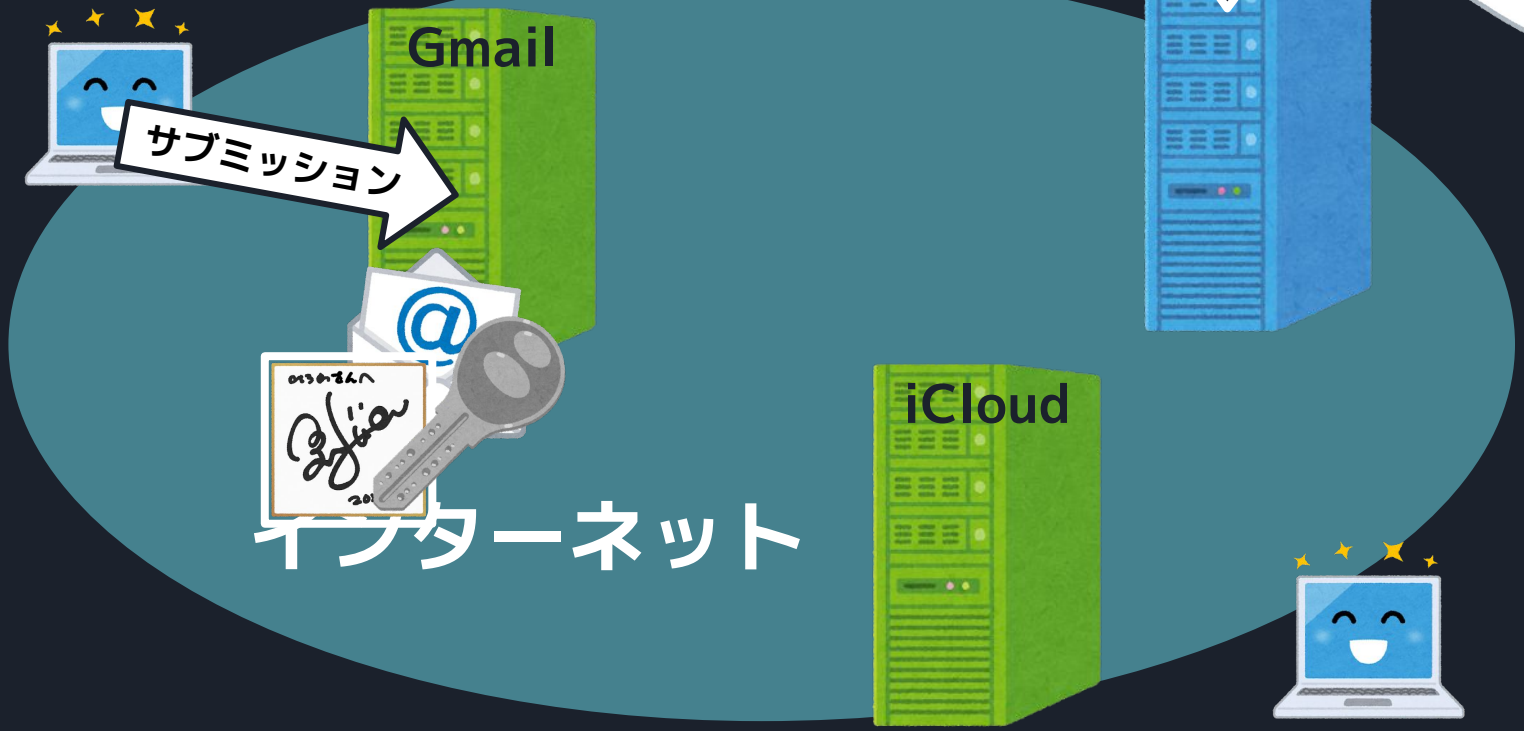
下準備



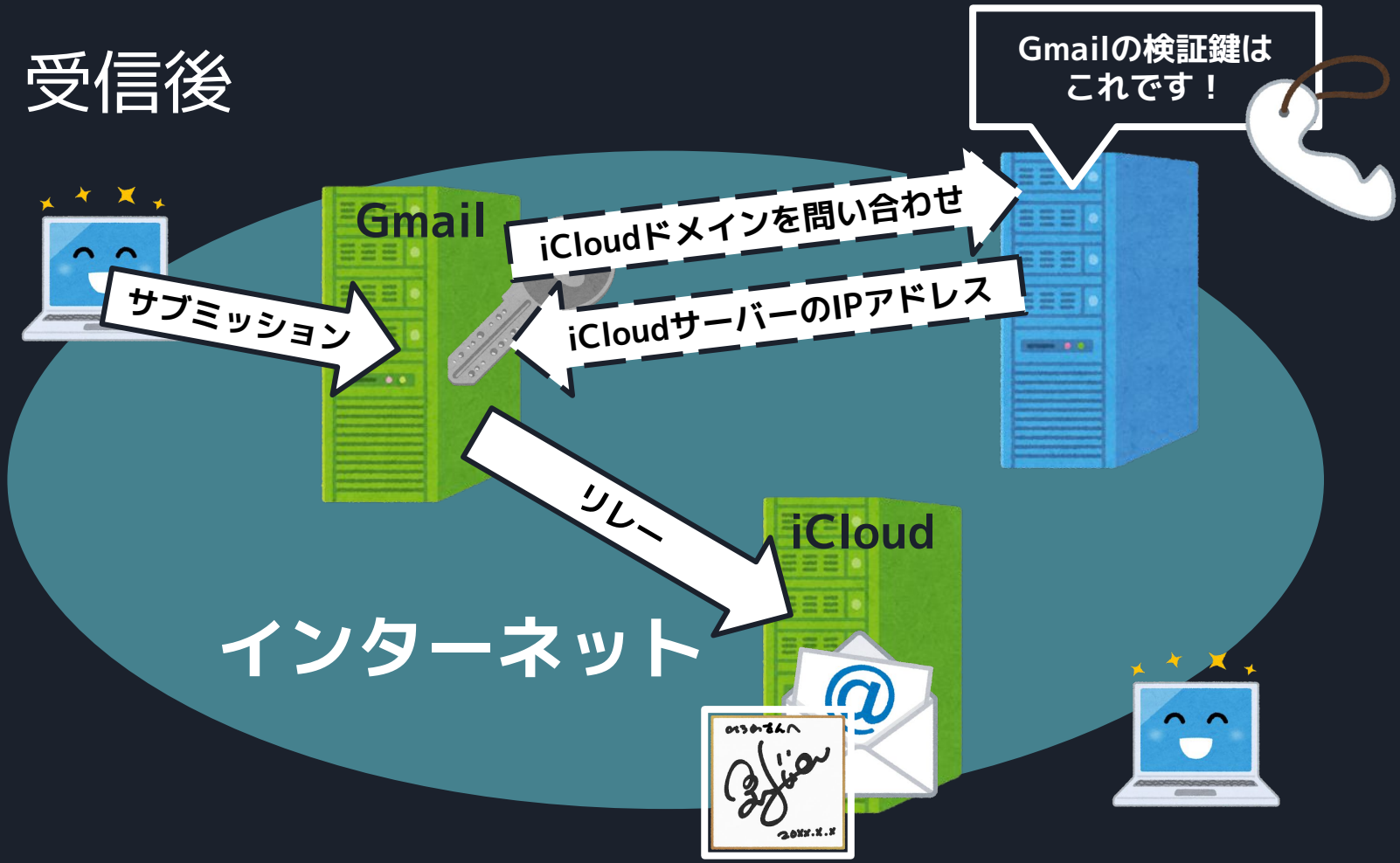
送信後



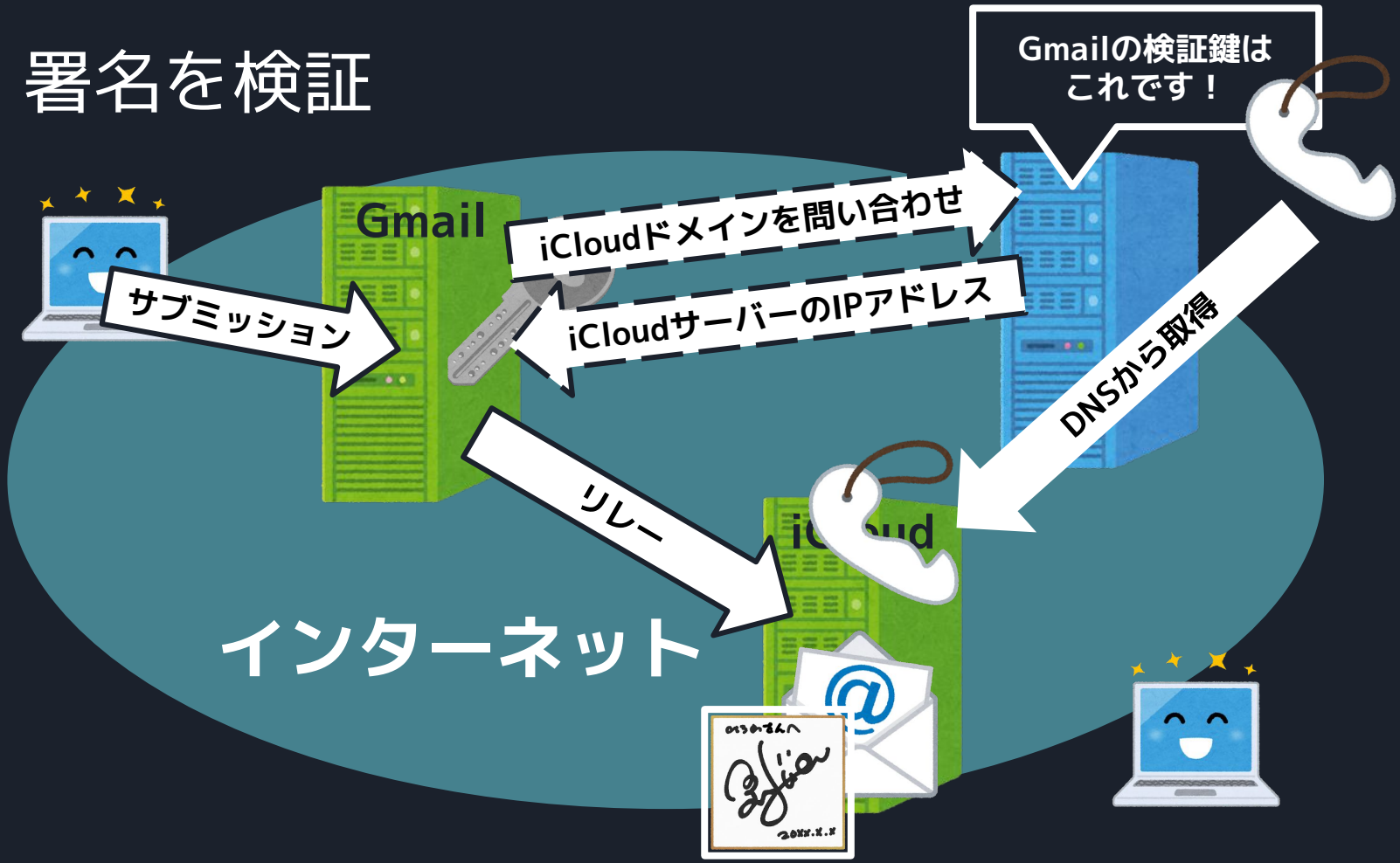
メッセージを加工し署名に



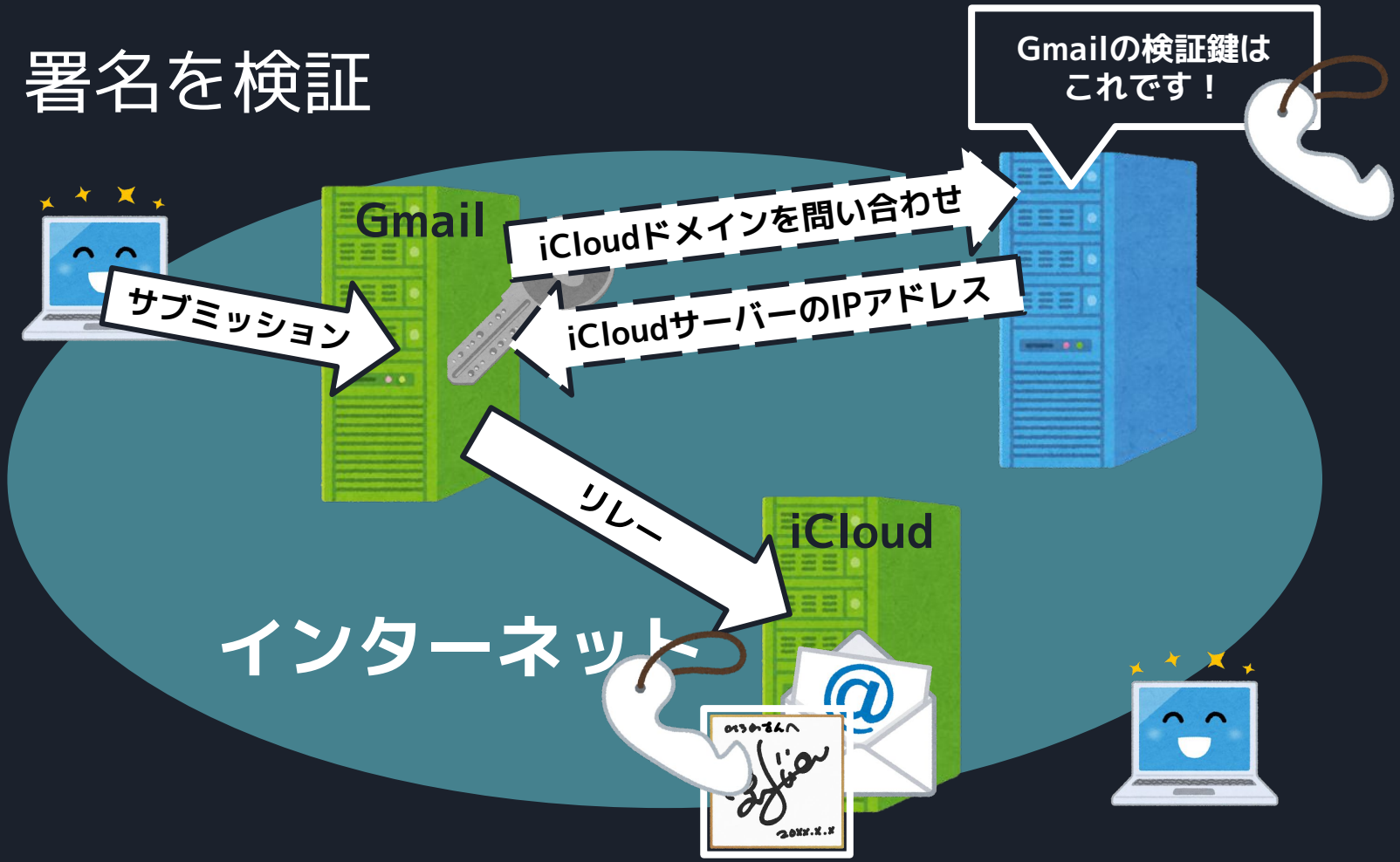
受信後


Gmailの検証鍵は
これです！
Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
リレー
iCloud
インターネット

署名を検証


Gmailの検証鍵は
これです！
Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
DNSから取得
リレー
インターネット

署名を検証


Gmailの検証鍵は
これです！
Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
リレー
iCloud
インターネット

DKIM

- 署名鍵をメールサーバーに持ち、検証鍵をDNSで公開しておく
- 署名鍵で**メール全体を加工したものを「署名」**としてメールに付けることで、正しいメールサーバーから送られてきたことを保証する仕組み
- サーバー運用者の**設定が面倒**
- サーバーを提供するサービス側の**実装が面倒**

DMARC
(Domain-based Message Authentication Reporting and Conformance)

DMARC

- **SPF**及び**DKIM**でメールサーバーが正しいと判定できなかった時「**どのようにメールを処理するか**」を決めることができるプロトコル
  - 同じく**DNS**にポリシーとして定める
- 「**自分のドメインの成りすまし**が出た」という場合のみ対処できる
  - それ以外のパターンは考慮していない
- サーバー運用者の**設定が面倒**

DMARCのポリシー

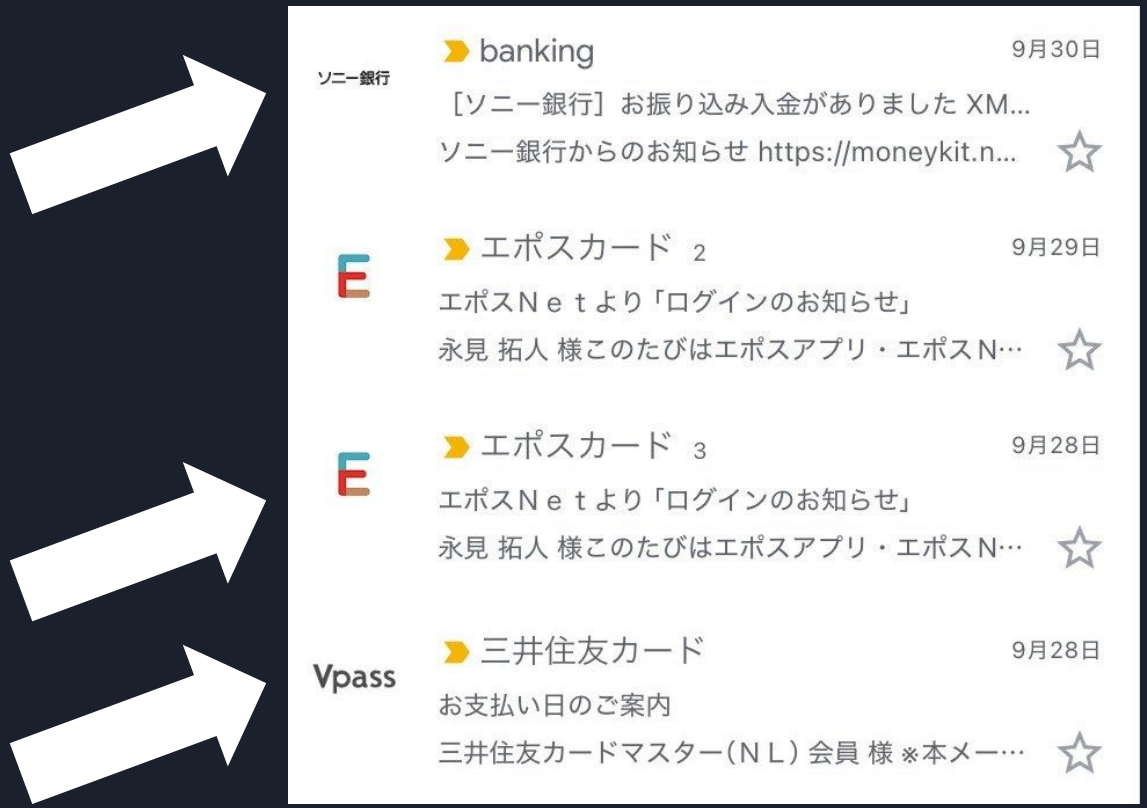
- **None**
  - メッセージは**通常通り送信**される
  - 「DMARCレポート」がドメインの所有者に来る
- **Quarantine**
  - メッセージが**別のフォルダに隔離**される
- **Reject**
  - メッセージは**その場で破棄**される

DMARCまでやると、現時点で「**そのドメインは安全**」と言えるライン

自分でサーバーを運用するときは、ここまで設定すべし

BIMI
(Brand Indicators for Message Identification)

要はこれのこと


ソニー銀行
banking
9月30日
［ソニー銀行］お振り込み入金がありました XM...
ソニー銀行からのお知らせ https://moneykit.n...
エポスカード 2
9月29日
エポスＮｅｔより「ログインのお知らせ」
永見 拓人 様このたびはエポスアプリ・エポスN…
エポスカード 3
9月28日
エポスＮｅｔより「ログインのお知らせ」
永見 拓人 様このたびはエポスアプリ・エポスN…
Vpass
三井住友カード
9月28日
お支払い日のご案内
三井住友カードマスター（ＮＬ）会員 様 ※本メー…

BIMI

- DNSで、そのドメインからのメールに**表示されるべきブランドロゴ**を設定できるプロトコル
  - どちらかというと企業とか向け
  - Gmail / Yahooメールなどは表示してくれる
- **DMARCの検証**が通って、初めて表示される
- サーバー運用者の**設定が面倒**
- 受信するサービス側の**実装が面倒**

ARC
(Authenticated Received Chain)

SPF・DKIM (・DMARC) の弱点

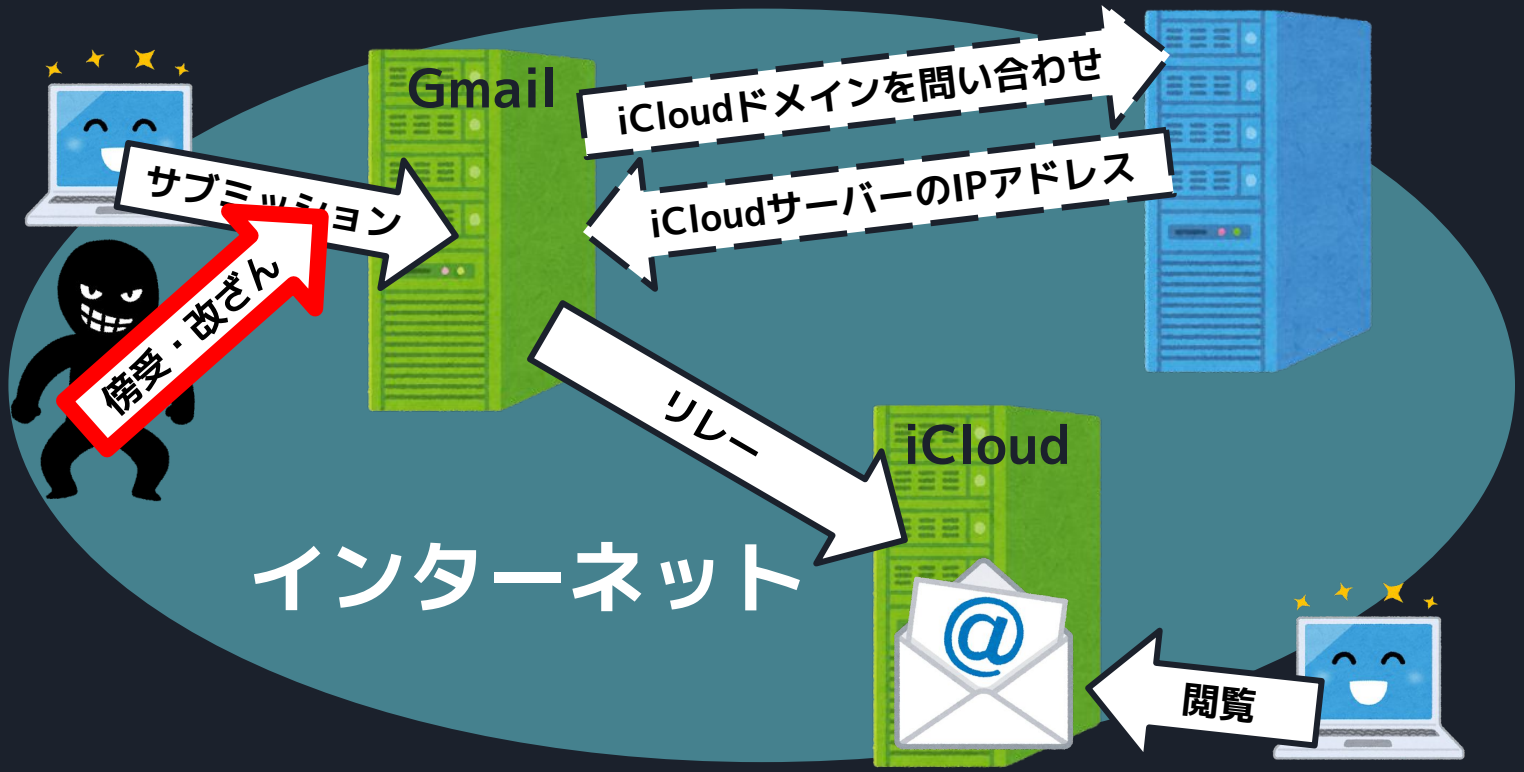
- **複数サーバーのリレー / メーリングリスト**などで検証がうまく行かなくなることがある
- SPF
  - 検証時、送り元のサーバーは**直前のサーバー**のため
- DKIM
  - リレーの過程で**ヘッダが変更**されることがある
  - メーリングリストでもヘッダが変更される

ARC

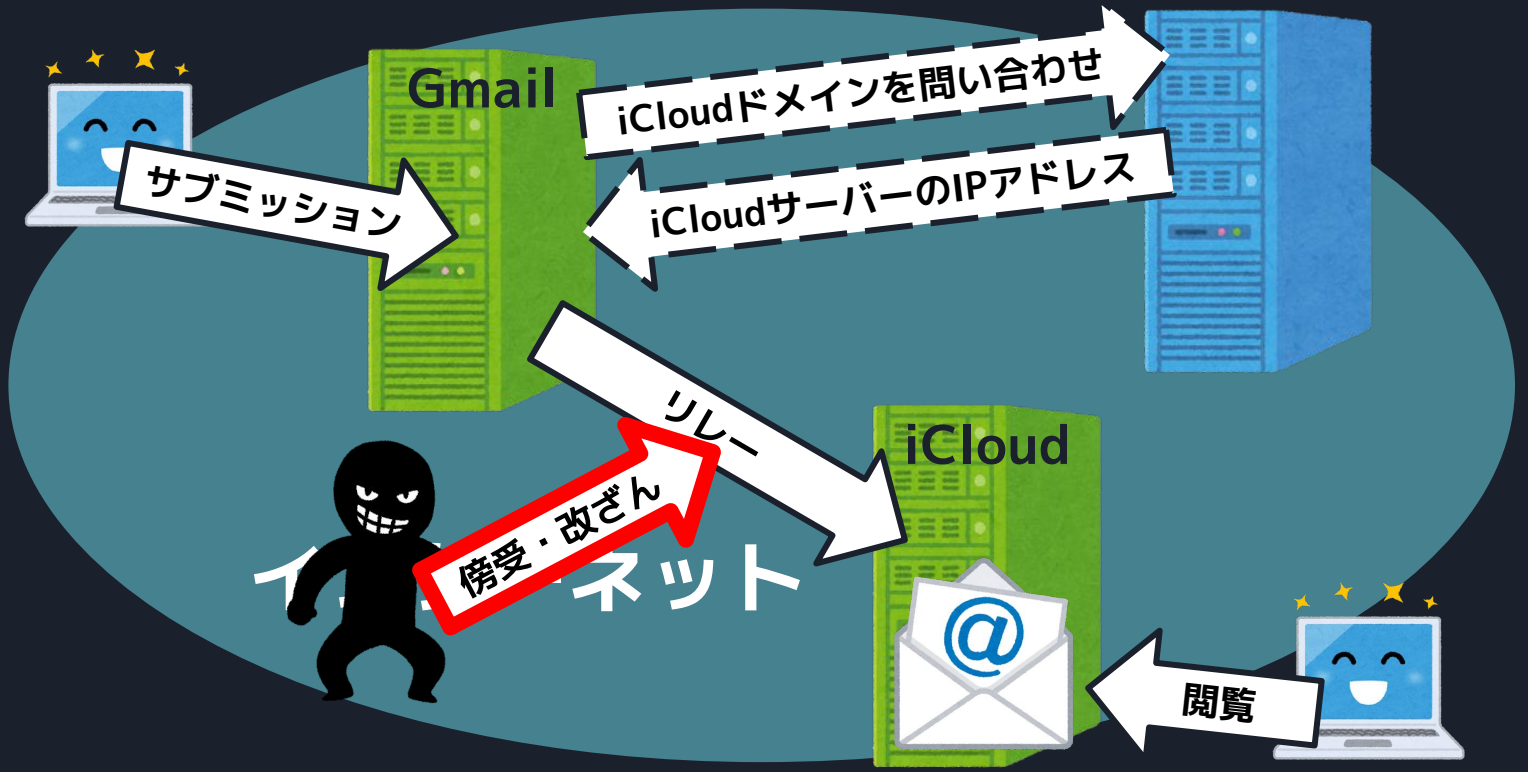
- リレーやメーリングリストに一斉転送する際、**中間のサーバーでSPF・DKIMを検証**して、その**検証結果をメールに付けて**次のサーバーに送る
  - SPF・DKIMがダメならARCのヘッダを見る
- 検証結果が偽造できないように、**署名を付ける**
  - 署名はDKIMと全く同じ仕組みなので**DNSに追加の設定をする必要ナシ**

脅威への対策**プロトコル**
～傍受・改ざん編

傍受・改ざん


Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
傍受・改ざん
リレー
iCloud
インターネット
閲覧

傍受・改ざん


Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
リレー
iCloud
傍受・改ざん
ネット
閲覧

STARTTLS

STARTTLS

- **TLS** (Transport Layer Security) をメールでも扱えるようにしたもの
  - 「https」の「s」の技術
  - メール特有の技術ではない
- サブミッション / リレーなど**一つ一つの通信経路を暗号化**する
- **悪意のあるメールサーバーが間に入った**場合は対処できない

MTA-STS
(MTA Strict Transport Security)

MTA-STS

- 認証 (SMTP AUTH) やSTARTTLSを、メールサーバー**利用者に強制する**ことができるしくみ
  - より強固なメールサーバーを作るために必要
- Gmailなどが導入している
- **DNS**で「ポリシーがあるよ！」と**宣言**し、**HTTP**で**ポリシー本体**を公開する
  - 設定の**ハードルが高い**

OpenPGP・S/MIME
(Secure / Multipurpose Internet Mail Extensions)

OpenPGP・S/MIME

- **エンド・ツー・エンド**でメールを**暗号化**する仕組み
  - **送信者が暗号化**し、**受信者が複合**する
- **エンド・ツー・エンド**でメールを**署名**する仕組み
  - **送信者が署名**し、**受信者が検証**する

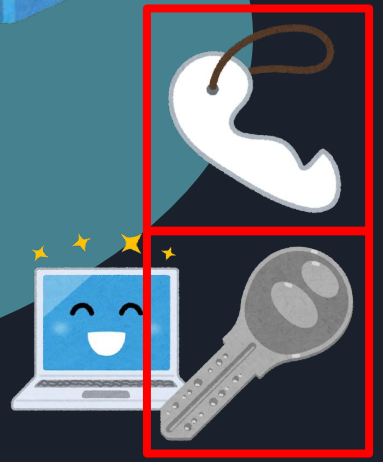
- 暗号化・署名で鍵を2ペア用意し、お互い鍵を持ち合う面白いプロトコル

下準備

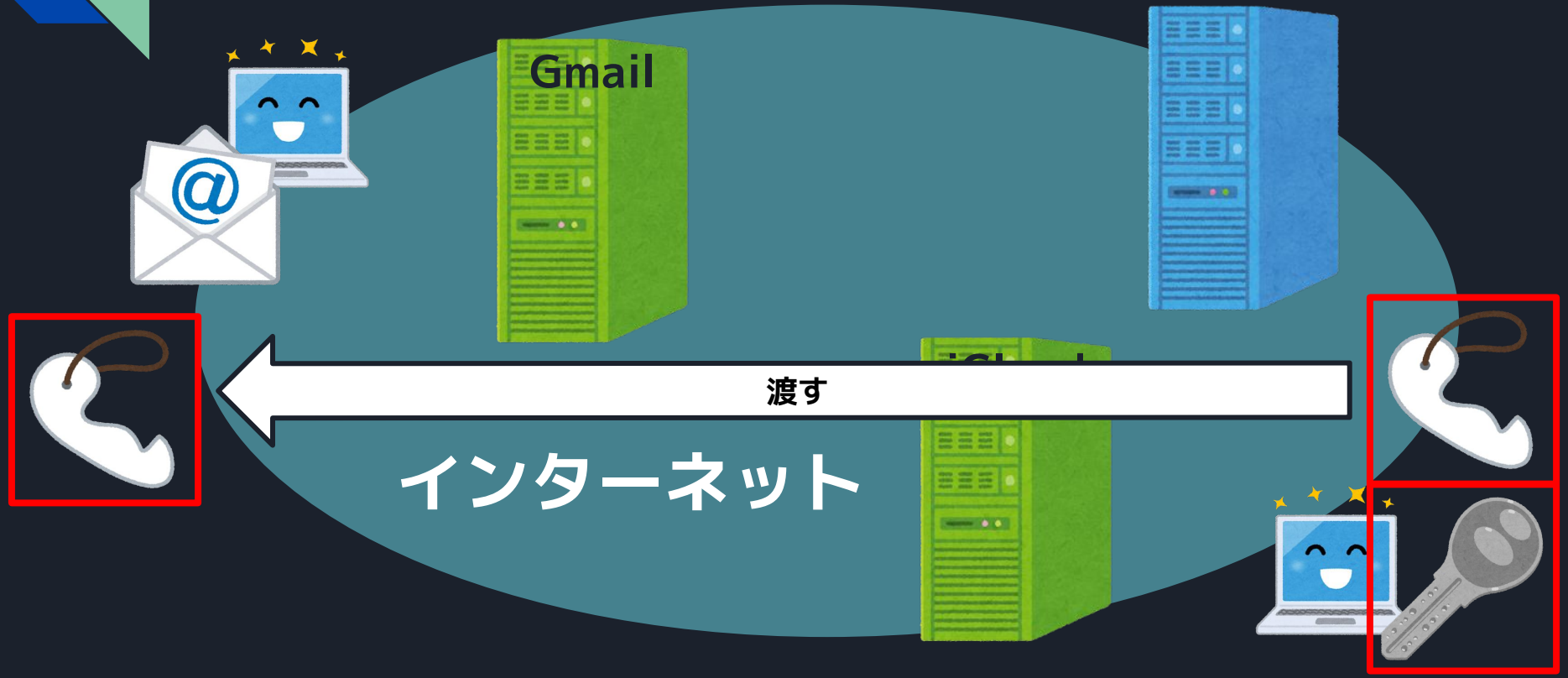
インターネット

鍵を準備 (暗号化サイド)

インターネット

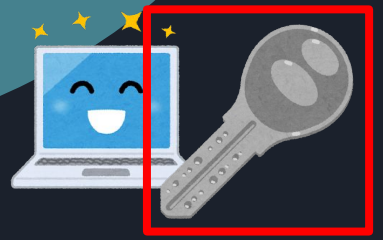


鍵を準備 (暗号化サイド)

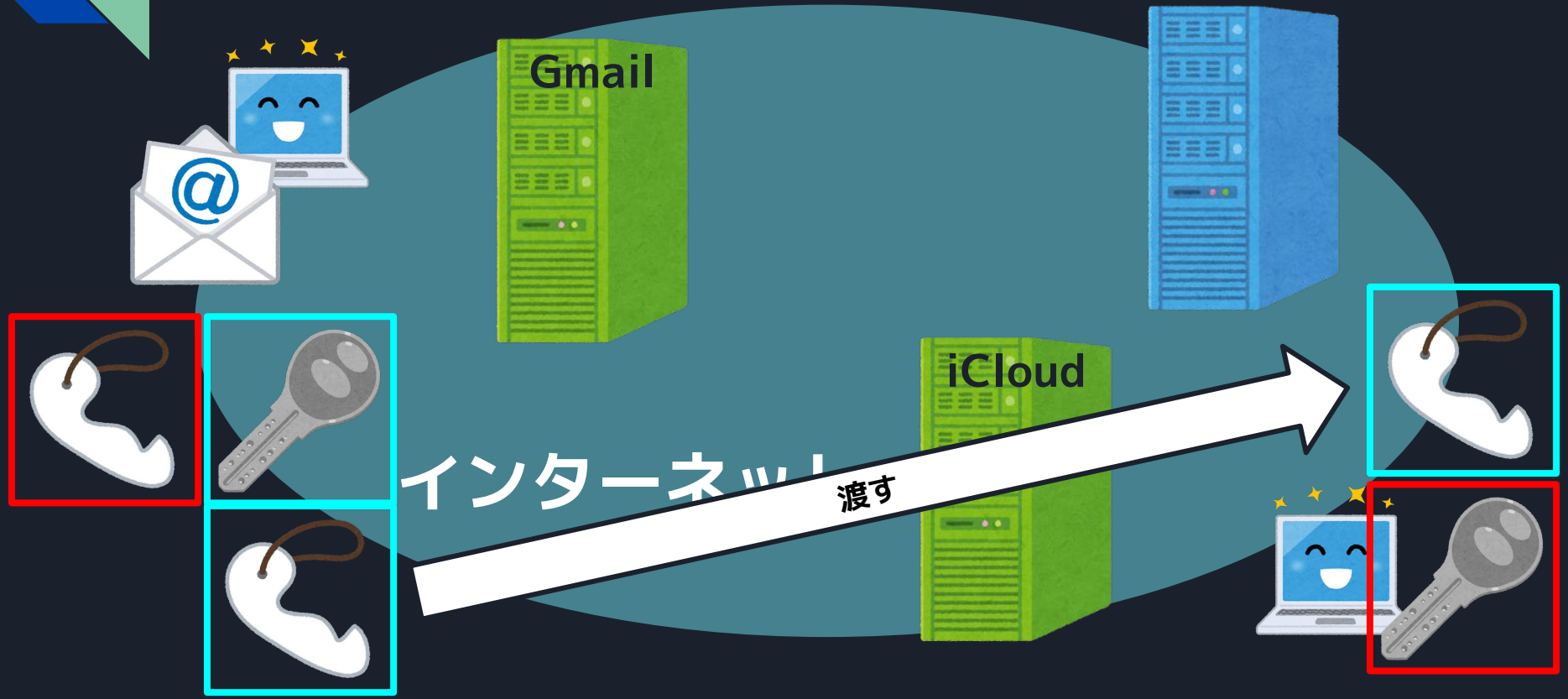


鍵を準備 (署名サイド)

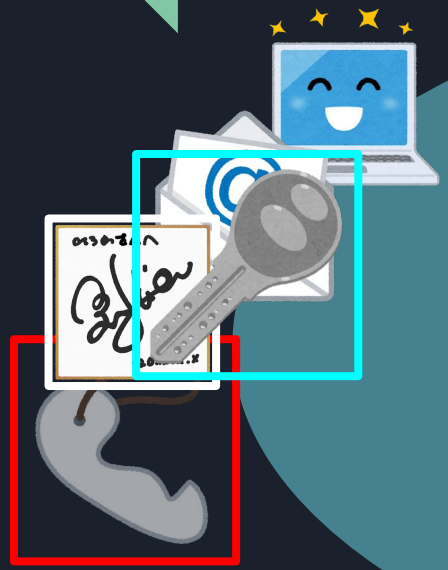
インターネット



鍵を準備 (署名サイド)

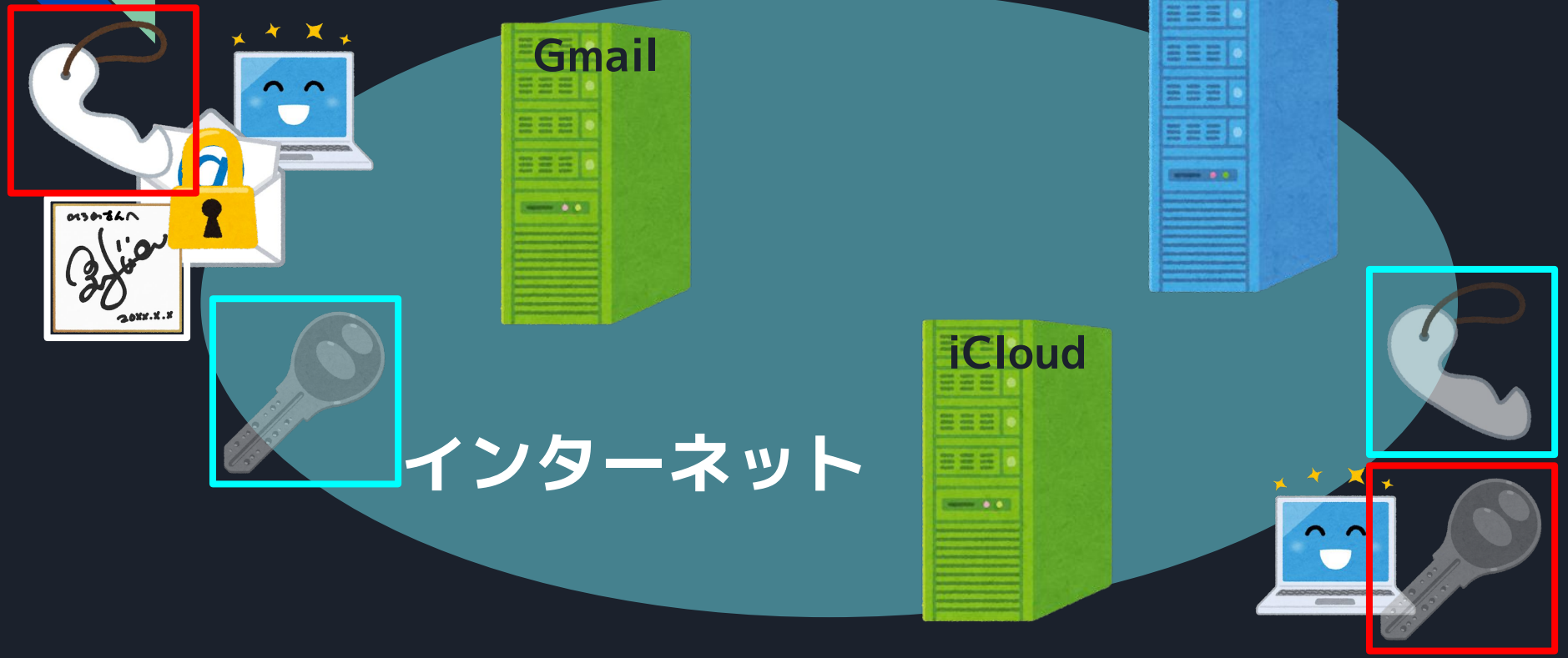


まず署名を作る

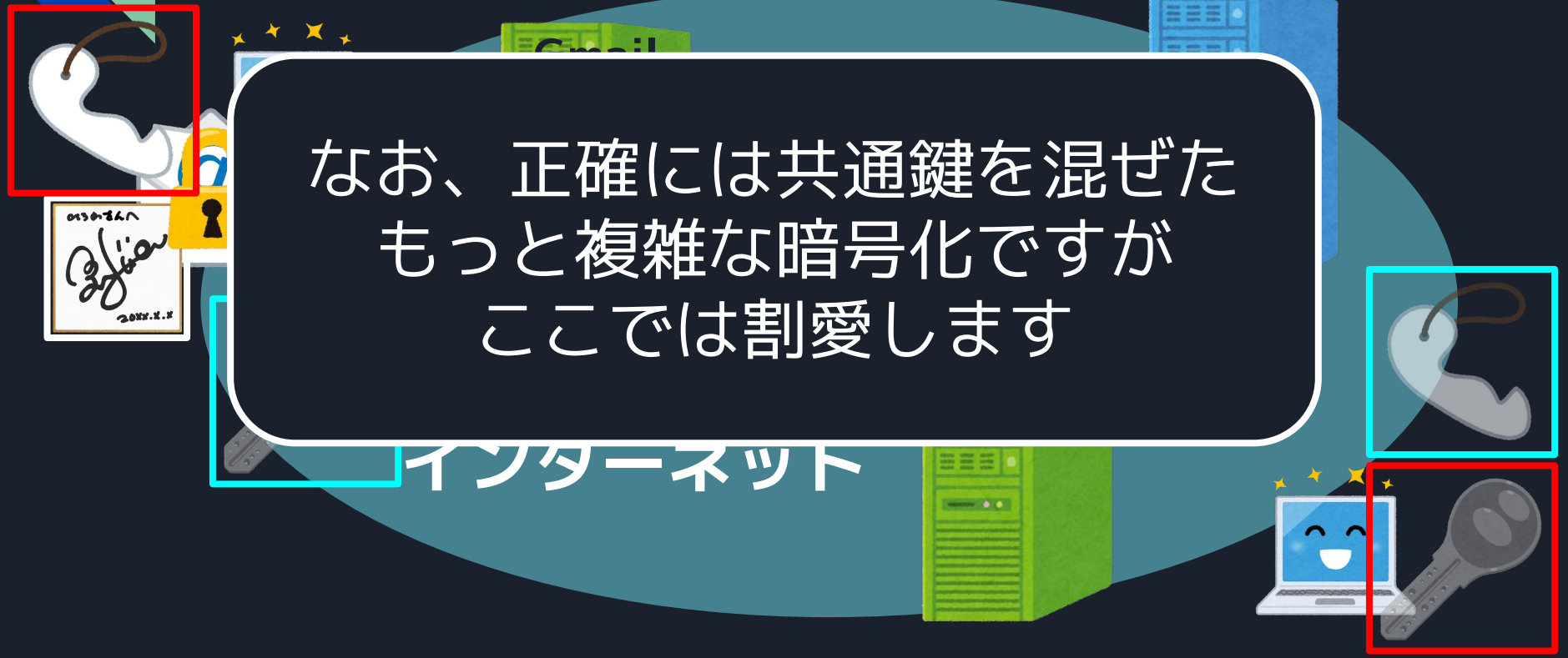


インターネット

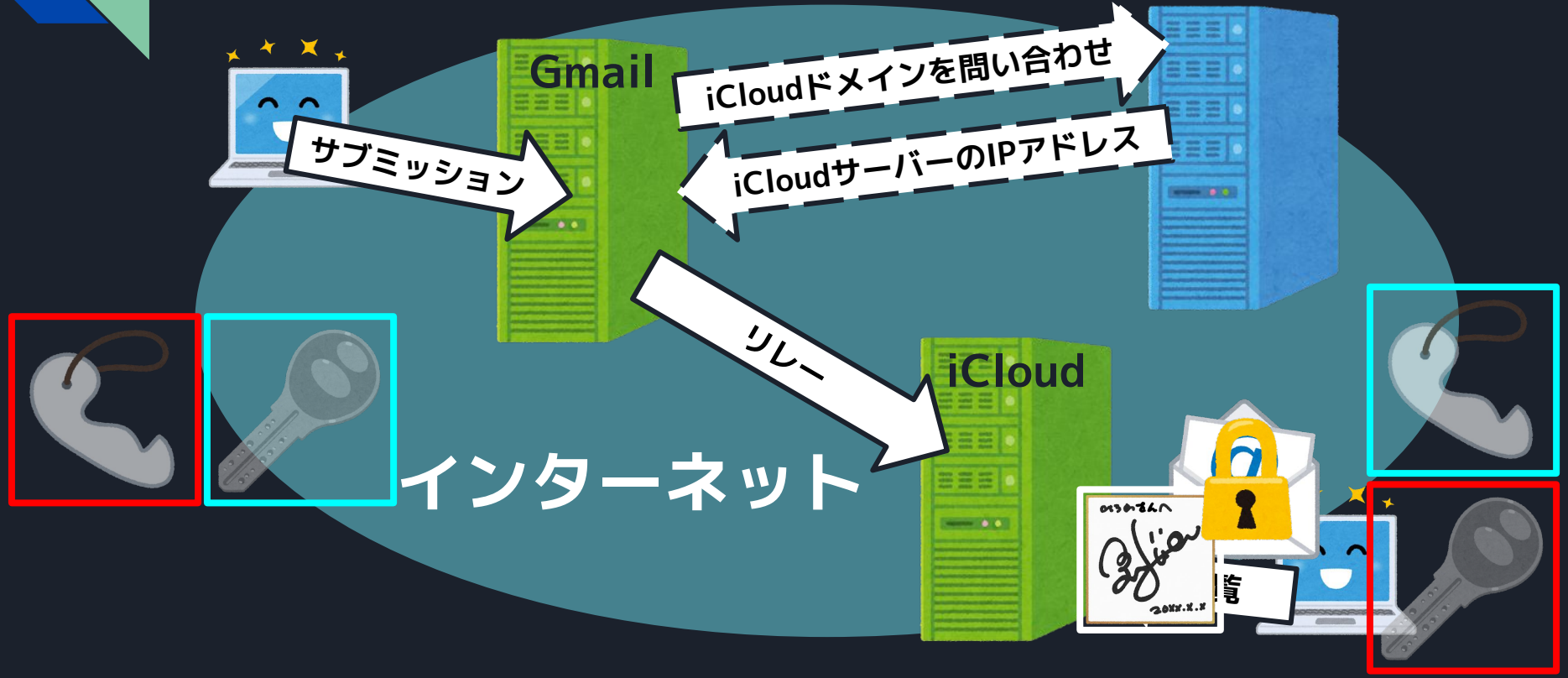
続いて本文を暗号化



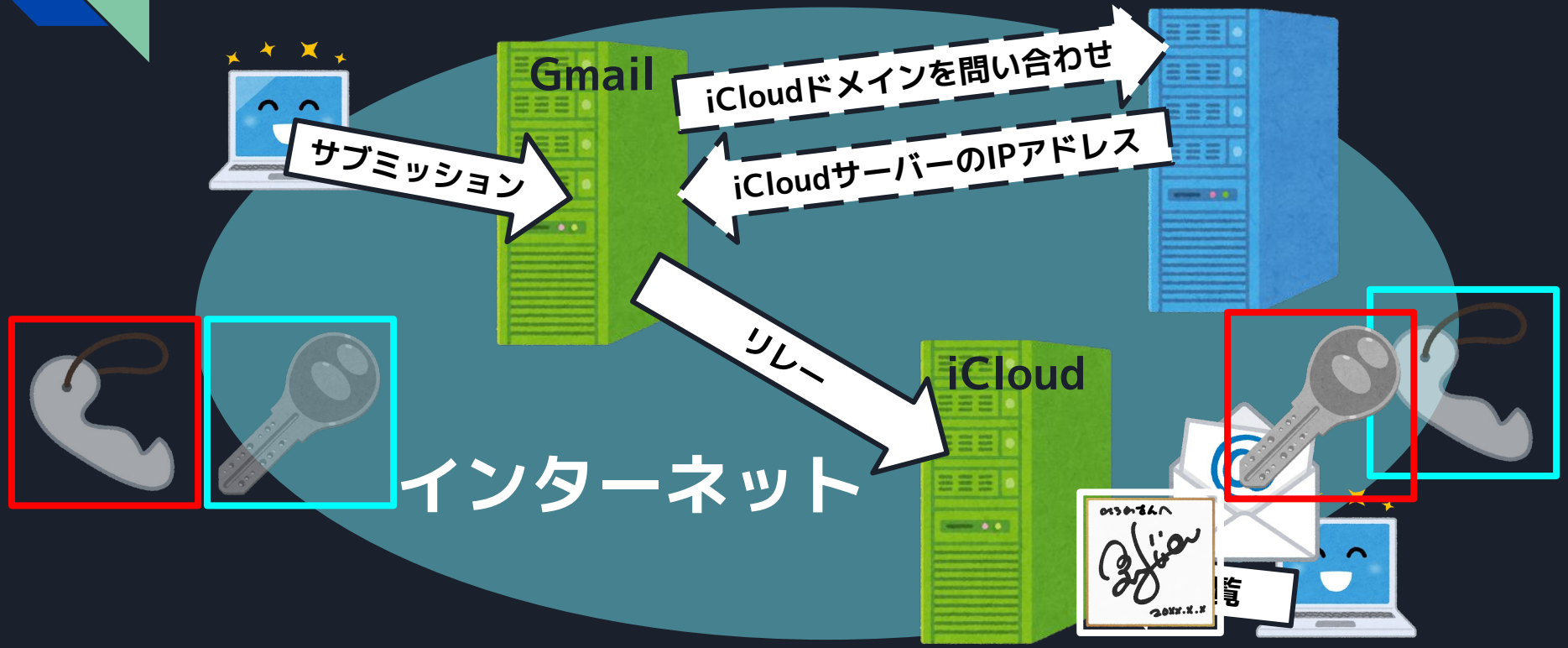
続いて本文を暗号化



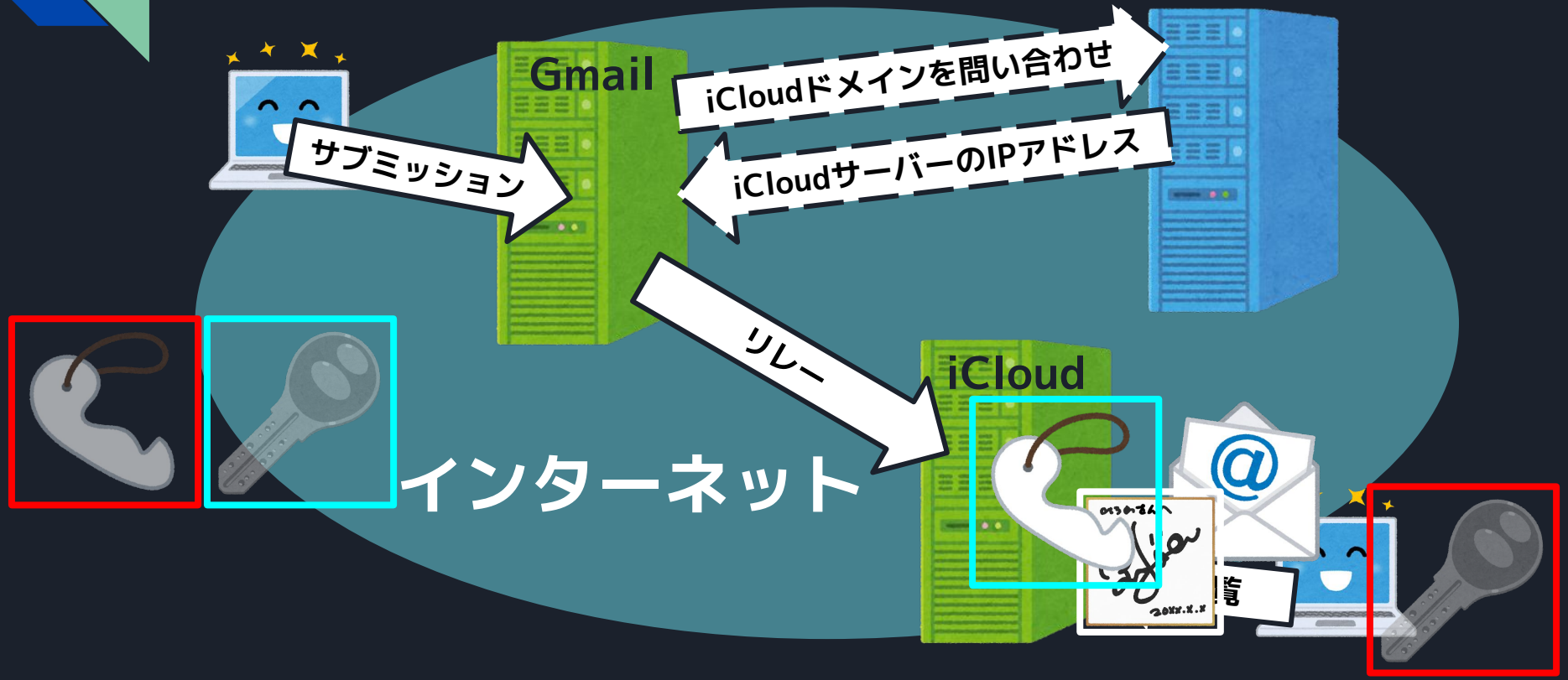
送信後、受信者の手元で


Gmail
サブミッション
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
リレー
iCloud
インターネット

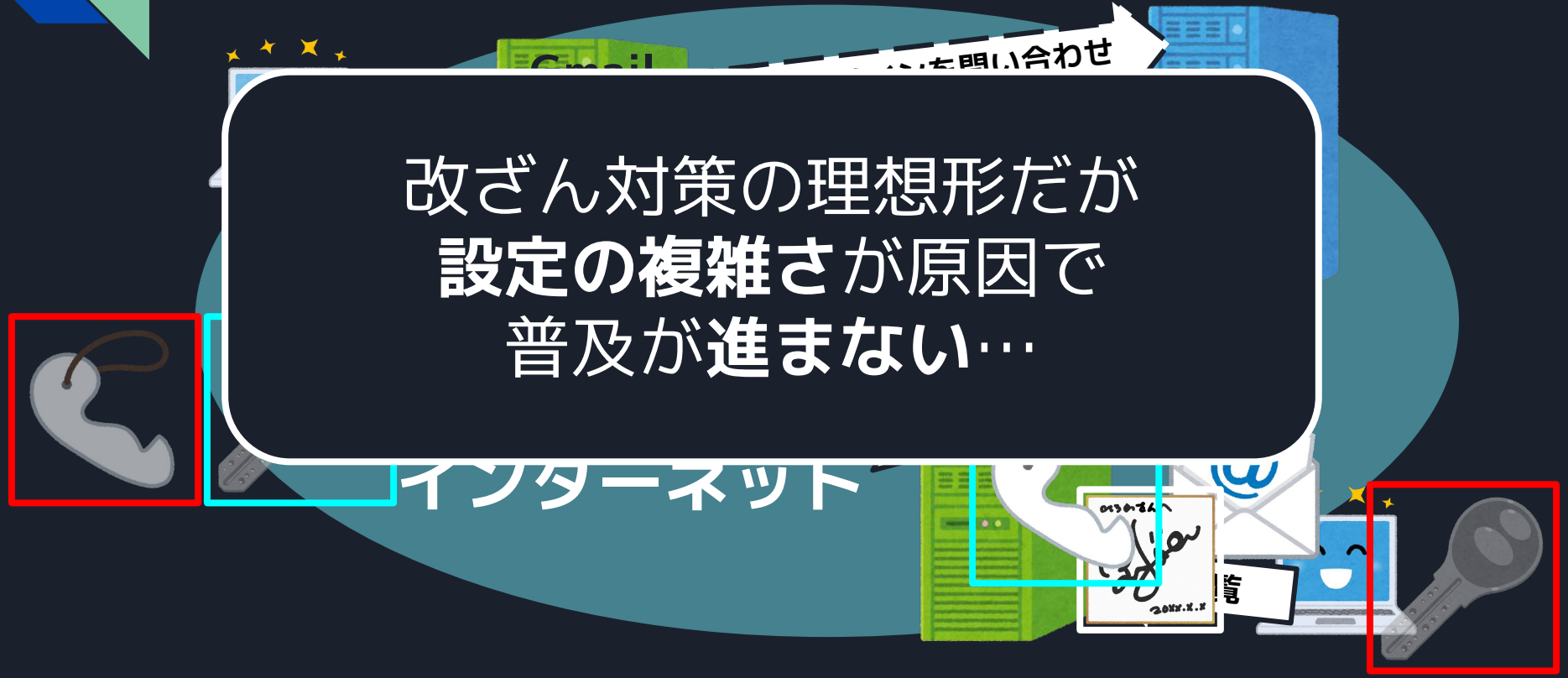
まず本文を復号


Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
リレー
iCloud
インターネット

最後に署名を検証


Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
リレー
iCloud
インターネット

最後に署名を検証



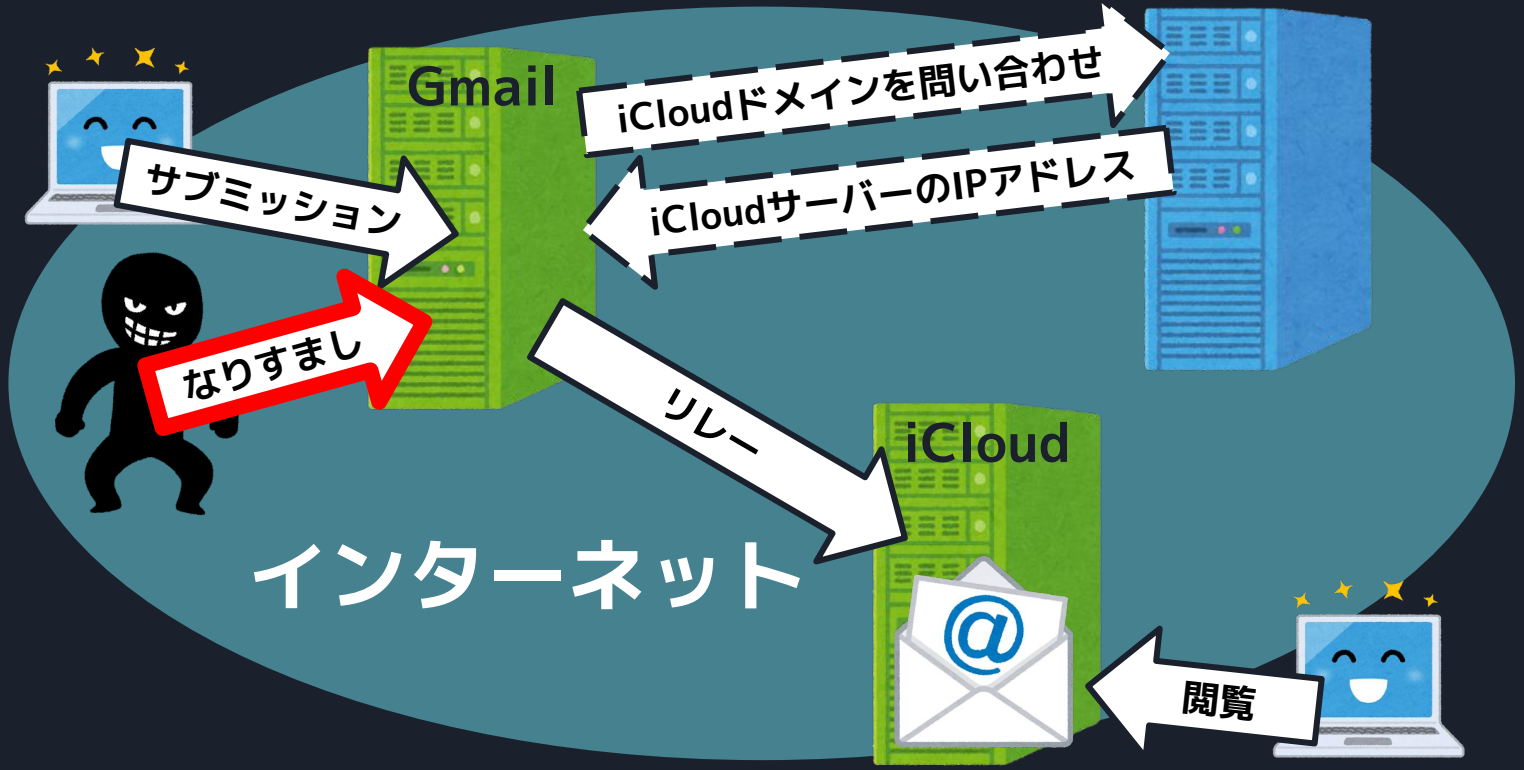
改ざん対策の理想形だが
設定の複雑さが原因で
普及が**進まない…**

OpenPGP・S/MIMEの問題と解決法

- **検証鍵の信頼性**が担保されない
  - 本人のでなかったら、**他人が署名してる**のでマズい
  - 暗号化は**間違ってたら復号できないだけ**なので良い
- OpenPGP
  - **多くの人に信頼されている**送信者は信頼
- S/MIME
  - **認証局**が信頼してるから信頼
  - 認証局にお金を払って、検証鍵を信頼してもらう

メールサービス独自の脅威対策

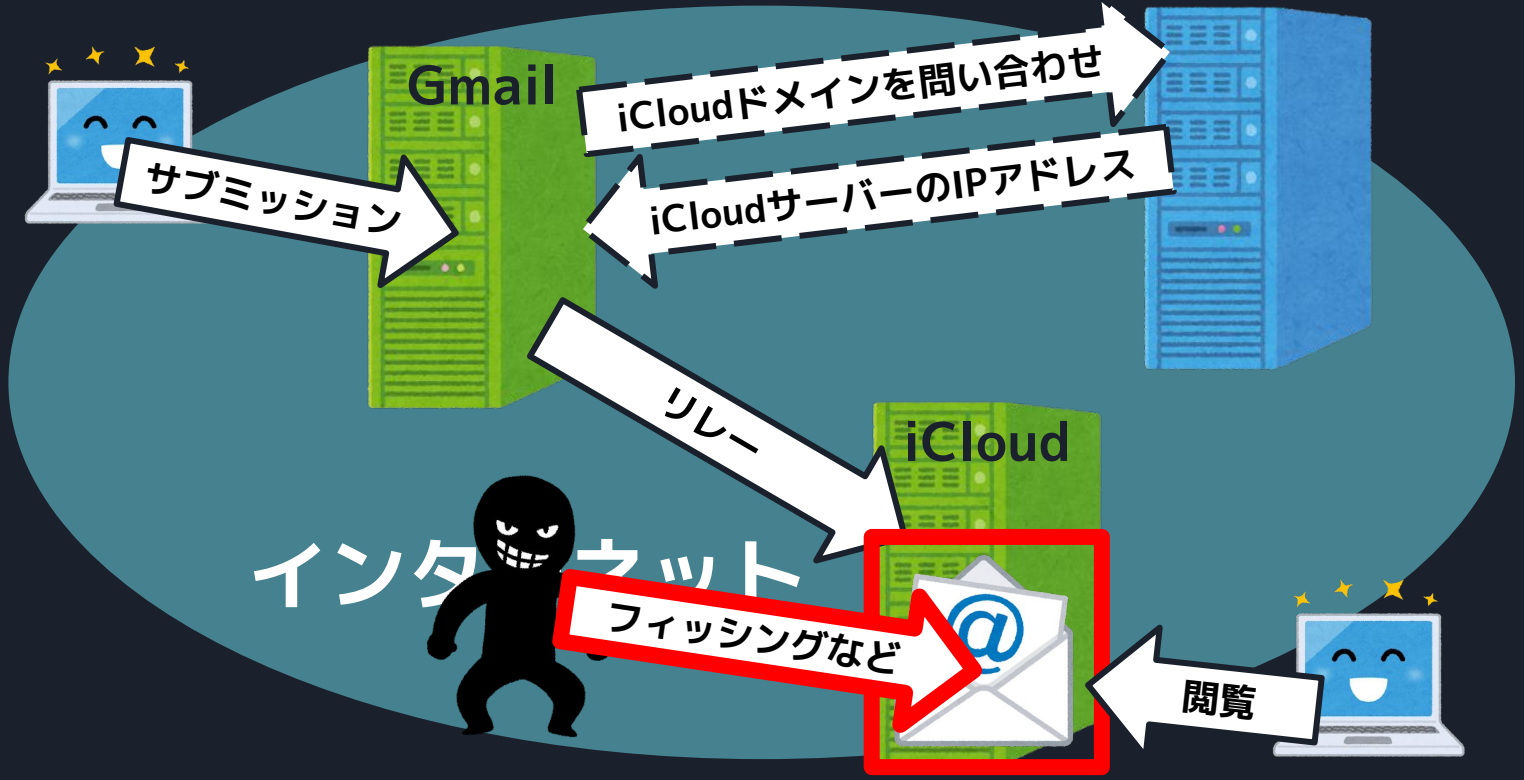
ユーザーなりすまし


Gmail
iCloudドメインを問い合わせ
iCloudサーバーのIPアドレス
サブミッション
なりすまし
リレー
iCloud
インターネット
閲覧

送信メールアドレス制限

- 「**自分が持っていること**」**が確認できる**アドレスから
  しか送信できないようにする
  - メールアドレス認証
  - ドメインを持っていることを証明する
    - DNSに指定されたテキストを登録するなど
- Gmailをはじめ、様々なメールサービスが導入している

メール内容の危険性判定



メールフィルタリング

- 機械学習などを用いて、「**怪しいメールのパターン**」を**特定**して危険なメールを判断する
  - かなり昔から研究が進んでいる分野
  - 機械学習の分野の先駆け
  - 「**ベイジアンフィルタ**」というアルゴリズムが有名
- Gmailなど、大手メールサービスを中心に導入
  - Gmailは最近流行りの**LLM**まで使っている、という噂がある

このように、様々な

セキュアにするための

取り組みがある

メールサーバーを管理するなら、知っておく必要アリ

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- **SMTPを安全にするための仕組み**
- ハンズオン - 「メールのソースを見てみよう！」
- メールとの向き合い方

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- **ハンズオン - 「メールのソースを見てみよう！」**
- メールとの向き合い方

ハンズオン
「メールの**ソース**を見てみよう！」

セキュリティプロトコルが
利用されている様子を
実際に見てみよう！

「生」のメールを見て体感してみよう

別で**ハンズオン資料を**
用意してありますので、
それに従って進めて下さい！

https://github.com/logica0419/seccamp-mini-ishikawa2025/blob/main/handson-2.md

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- **ハンズオン - 「メールのソースを見てみよう！」**
- メールとの向き合い方

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- **メールとの向き合い方**

メールとの**向き合い方**

なぜここまで**様々な**
対策が施されているのに、
電子メールは**未だ様々な**
サイバー犯罪に使われるのか？

ここからは
事実に基づいた憶測

人間が対策することなので
「正しさ」は保証できない

強固な対策が取りにくい

- 対策で、**メールを送れなくすること**は基本的にない
  - 正しい送信元か確かめたり、改ざん防止だったり…
- **プライベート**なやり取りであるので難しい
  - Webページの場合、特定のユーザーに**アクセス制限**をかける
    - 情報の提供者の階層が利用者と別れるため一元的な指標で不正行為などを定められる
  - メールでは、「**何を不正とするか**」が曖昧

共通化の難しさ

- Webページのセキュリティは**奇跡**に近いクオリティ
  - Chromiumという、Google Chromeの核になる部分が**ほぼ全てのブラウザ**で使われている
- 簡単さや社内政治のせいで、**メールサービスは乱立**
  - セキュリティ対策をしなくてもメールは送れてしまうので、**セキュリティ対策をしない**メールサービスが多い
- 何かしらの方法で**対策を強要する**必要がある

Gmailのセキュリティ基準強化

メール送信者のガイドライン

この記事のガイドラインに沿った対応を行うことで、個人用 Gmail アカウントにメールが正常に送信、配信されるようになります。2024 年以降は、Gmail 個人用アカウントにメールを送信する場合に、こちらに記載の要件を満たす必要があります。個人用 Gmail アカウントとは、末尾が @gmail.com または @googlemail.com のアカウントを指します。

メール認証の要件とガイドライン

ドメインに、以下のようにメール認証方式を設定する必要があります。

- すべての送信者: SPF または DKIM
- 一括送信者: SPF、DKIM、DMARC

Gmailのセキュリティ基準強化

- 「SPF / DKIMを利用していないと、**メールを弾く**可能性があるよ」と警告
  - 個人メールにおいて**圧倒的シェア**を持つ影響力
- さくらのメールサーバーをはじめとする、**様々なメールサービスが対応を迫られる**結果に
  - 国内のかなり**大手のサービスでもまだ仕組みに対応していない**ことが判明した
- セキュリティ面では、非常に大きな一歩

メールから逃れられない人類

- セキュリティを考えれば、メールは**排除すべき**存在
  - ここまで対策しても**完全にセキュア**とは言えない
- ここから20〜30年は**メールから逃れられない**であろう現状が存在する
  - **依存するもの**が多すぎる
  - 企業間のやり取り、アカウント、etc…
- 逃げられないのであれば「**まだマシ**」**な運用**をするよう関わる全員が心掛けなくてはならない

アジェンダ

- なぜ、今メールなのか
- プロトコルスタックとRFC
- メールが送られる仕組み
- SMTPプロトコル
- ハンズオン - 「SMTPを手書きしてみよう！」
- SMTPを安全にするための仕組み
- ハンズオン - 「メールのソースを見てみよう！」
- **メールとの向き合い方**

まとめ

今日学んだこと

- メールが様々な犯罪に用いられること
- RFC (プロトコル) がインターネットを支えていること
- メールがSMTPを使って送られていること
- SMTPは手書きで書けてしまうほど簡単で、なりすましが容易にできること
- メールを安全にするために様々な取り組みが施されていること
- メールを安全にするのはとても難しいこと

メールの仕組み**そのものが**
脆弱なのは否定できない
「**まだマシ**」なメールを運用する
ために知識を付けよう

ありがとう
ございました

怪しいメールには
気を付けよう！

参考ページたち

- https://www.irasutoya.com/
- https://www.npa.go.jp/hakusyo/r03/honbun/html/xf211000.html
- https://sendgrid.kke.co.jp/blog/?p=10300
- 東京工業大学 情報理工学院 情報工学系 講義
  「コンピューターネットワーク」
- https://atmarkit.itmedia.co.jp/ait/articles/1411/13/news019.html
- https://thinkit.co.jp/story/2015/04/28/5799
- https://www.gmo.jp/security/brandsecurity/dns/blog/dns-server/
- https://wa3.i-3-i.info/
- https://www.tohoho-web.com/
- https://qiita.com/angel_p_57/items/d7ffb9ec13b4dde3357d

参考ページたち

- https://ja.wikipedia.org/
- https://www.itmanage.co.jp/column/osi-reference-model/
- https://www.proofpoint.com/jp/threat-reference/dmarc
- https://www.naritai.jp/
- https://support.google.com/a/answer/9261504
- https://qiita.com/hirachan/items/d0028da3ebb80b138404
- https://jp.globalsign.com/managed-pki/about_smime.html
- https://www.prins.co.jp/knowledge/column/20181101-557/
- https://support.google.com/a/answer/81126
- https://www.sakura.ad.jp/corporate/information/announcements/2023/12/19/1968214527/