

# JavaScript による **End-to-End** セキュリティ

## 入門編

栗原 淳

July 29, 2019

はじめに

# はじめに

この講義では

- End-to-End (E2E) セキュリティの原則
- Web サイトでの E2E セキュリティ実践のため、JavaScript での実装方法
  - ブラウザ側
  - サーバ側 (Node.js)

のさわりを学ぶ。

# モダン Web サイトと End-to-End セキュリティ

# Web サイトにおける昨今の情勢

- EU における General Data Protection Regulation (GDPR) の施行 (2018 年)
- GDPR に続いて、カリフォルニア、南米、オセアニアで類似の法律の制定の動き
- 日本においても、2020 年に個人情報保護法の改正法案提出の見通し



企業にとって、「正しく」「強固」に  
ユーザデータ、ユーザプライバシーを保護することは必須の事項

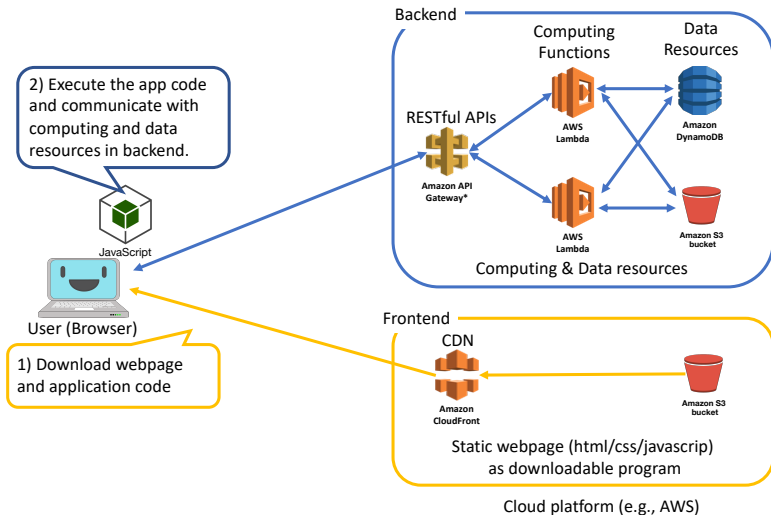
# 最近流行りの Web システム

- クラウドプラットフォーム上で構築
- 「サーバ」のない (サーバレス) 構成
- JavaScript (ReactJS など) を多用した、Single Page Application 構成

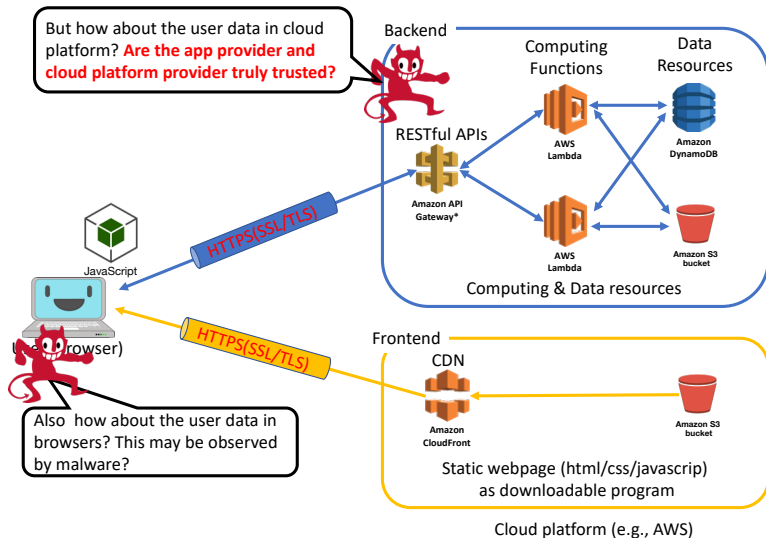


ユーザの手元で計算を実行する機会の増加

## AWS を例にした典型的な構成:



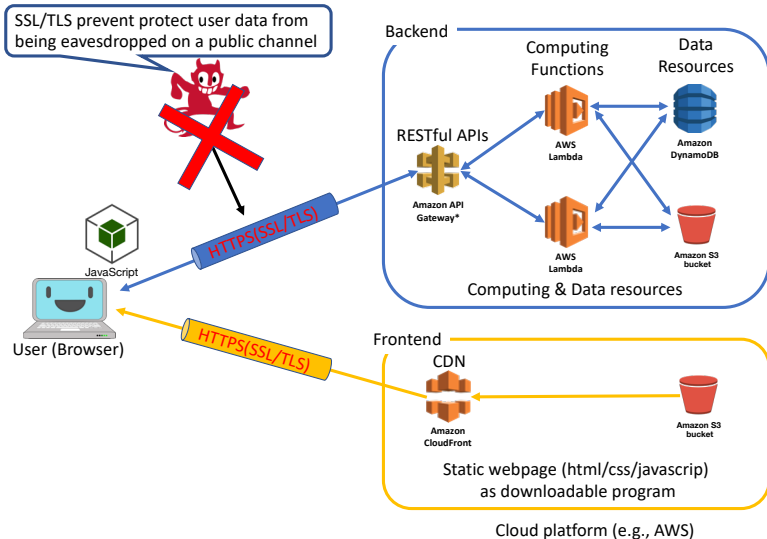
# AWS を例にした典型的な構成:



通常、ユーザ・クラウド間の HTTP 通信路は SSL/TLS で保護  
⇒ HTTP 通信路から外部の盗聴者へのユーザデータ漏洩を防止



# AWS を例にした典型的な構成:



しかし、クラウド PF 内・ブラウザ内のデータ保護は……？

# 「暗号化しているから安全です」という叙述トリック

## (Web とはちょっと違いますが…) 某クラウドストレージ事業者の例

従来の暗号化を凌駕

□ は、□ アプリとサーバー間で転送中のファイル、および保管中のファイルを保護します。各ファイルは不連続のブロックに分割され、強力な暗号を使用して暗号化されます。変更されたブロックのみが同期の対象になります。 [詳しくはこちら](#)

クラウドベースの (Web) サービスでよくある文言：

- (SSL/TLS で) 転送中のデータを暗号化して保護
- ストレージに保存されるデータは暗号化して保護

- (SSL/TLS で) 転送中のデータを暗号化して保護  
⇒ 公開通信路の盗聴からデータを保護
- ストレージに保存されるデータを暗号化して保護  
⇒ ストレージ自体が盗まれた時や、第三者のストレージを使っている場合のデータ漏洩を防止

いずれも事業者に対しての秘匿性を担保しているわけではない<sup>1</sup>



(望む・望まないにしろ) 事業者はユーザデータを不必要に取得

---

<sup>1</sup>事業者はデータを見放題ということ。

このようなクラウドサービス・Web App を作ることは：

- ユーザにとって：共有不要な相手とデータを共有している
- 事業者にとって：昨今のプライバシー・セキュリティ要求の高まりから、**無用なリスクを背負いこむ可能性が大**

今後、Web App を作っていくにあたって

「必要な相手とだけ」確実に・正しく、データを共有できるように、適切なデータ秘匿が必要

# 余談：秘匿性・データプライバシーをウリにしたサービス

## ■ Tresorit<sup>2</sup>:

事業者・サーバに情報が漏れないことを謳ったクラウドストレージサービス。Dropbox に近い。

## ■ KeyBase<sup>3</sup>:

事業者・サーバに情報を漏らさず、メッセージ・ファイル共有（クラウドストレージ）が可能な SNS。

## ■ Signal<sup>4</sup>:

事業者・サーバに情報を漏らさないメッセージング・通話アプリケーション。「最も安全な」チャットサービスと呼ばれており、各類似サービス (WhatsApp など) にプロトコルを提供。

北米・EU 共に、スノーデンの事件以降、事業者にも情報を与えない  
**End-to-End 暗号化**を謳ったサービスが強く注目を浴びている。

<sup>2</sup><https://tresorit.com/>

<sup>3</sup><https://keybase.io/>

<sup>4</sup><https://signal.org/>

# End-to-End セキュリティの原則とは

# Web システムにおける **End-to-End** セキュリティ

導入する意味

# JavaScriptで暗号を使ってみよう [基礎編] 今回

は AES を使ってみます。  
AES とは。



# ブラウザでの暗号化: WebCrypto API

# サーバでの暗号化: **Node.js Crypto**

# ブラウザ・サーバ間での相互接続性の確認

しかしサーバで復号しているのであんまり意味がない。

補足: **API** が違うのがめんどくさい…

手前味噌だが、統合 **API** を使って楽をすると良い

# ブラウザ同士での相互接続性の確認

API を通じて暗号化データをやり取りしてみる。  
E2E Security!

- 1 今回は共通鍵暗号
- 2 公開鍵暗号& Hybrid Encryption
- 3 ハッシュ・署名と HMAC
- 4 超マニアック講座：RFC とアルゴリズム・フォーマット

# 引用文献

# Appendix

This page is not counted.