JavaScript による End-to-End セキュリティ

第 3 回 公開鍵暗号はどうやって使えばいいのか？ 編

栗原 淳

2019 年 10 月 17 日

はじめに

はじめに

第 1 回と第 2 回では

- End-to-End (E2E) セキュリティの原則と必要性
- Web サイトでの E2E セキュリティ実践のため、JavaScript で暗号 (AES) を正しく・安全に利用する方法

を勉強した。

ところで、AES(共通鍵暗号) とは別に、「公開鍵暗号」というのが存在する。

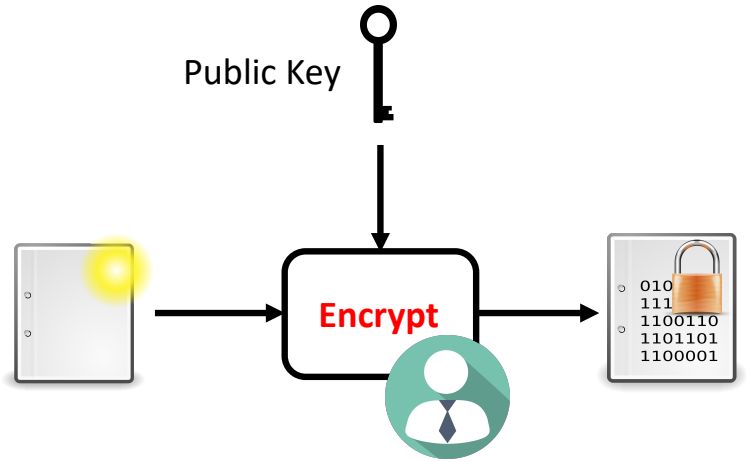
公開鍵暗号って？

既知だと思うが、まずざっと定義しておく。

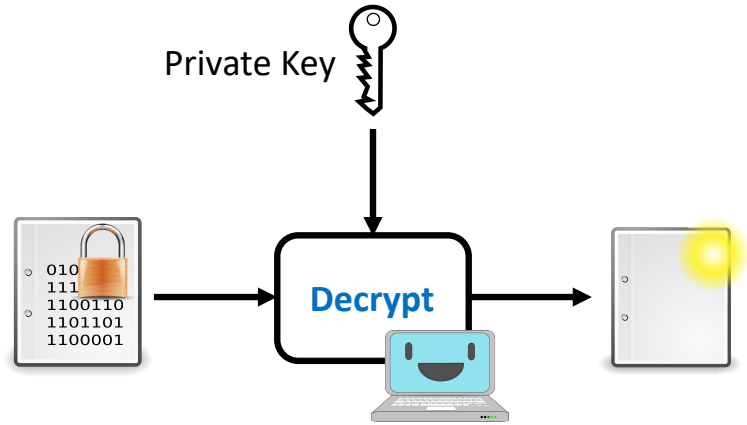
定義: 公開鍵暗号

以下のステップで暗号化・復号が行われる暗号方式のこと

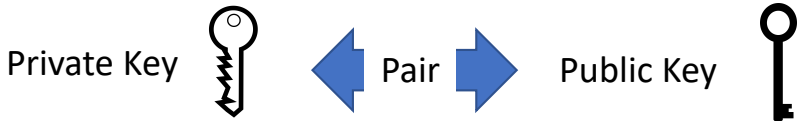
1. 特殊な数学的条件を満たす鍵ペア「公開鍵 PK と 秘密鍵 SK」を生成
2. PK は公開、SK は秘匿
3. データ D を PK によって暗号化して、暗号化データ X を生成
4. X は SK によってデータ D に復号される。



Anyone can encrypt information only with public key without knowing private key!



Only the private key owner can decrypt the information encrypted under its paired public key.



暗号化・復号の鍵を分けて、暗号化の鍵を公開してしまうことでパスワードなどの共有が不要になる。

⇒ AES などにはない、非常に強力な暗号化の概念。

今回は正しく・安全に公開鍵暗号を使っていくためのお話。

この講義で最終的に学びたいこと

- 公開鍵暗号はどういうものか。AES と比べた pros/cons。
- RSA 暗号と楕円曲線暗号[1] の違い。
- AES と公開鍵暗号を組み合わせてデータを暗号化するために。

細かい話もするが、なるべく数式を使わないで「イメージ」をつかめるようにする。

[1] 楕円曲線 Diffie-Hellman を取り上げる

この講義の対象と事前準備

対象:

- 暗号・セキュリティ技術に興味がある初学者
- Web に暗号技術を導入したい Web 系のエンジニア

必須ではないが触って楽しむのには必要な事前準備:

- Git が使えるようになっていること
- Node.js が使えるようになっていること
- Google Chrome 系ブラウザ and/or Firefox が利用可能なこと

公開鍵暗号の使い方 事始め

公開鍵暗号の種類

公開鍵暗号の定義「特殊な数学的条件を満たす鍵ペアを生成」

⇓

この「数学的条件」に複数の種類が存在。

JavaScript に限らず、各種環境で利用可能な代表的な公開鍵暗号：

- 素因数分解に関する条件
  → RSA 暗号
- 楕円曲線上の離散対数に関する条件
  → 楕円曲線暗号 (Elliptic Curve Diffie-Hellman; ECDH)

この２つの使い方、注意ポイントを今回は取り上げる。

RSA 暗号のさわり

RSA Cryptography

言わずもがな、公開鍵暗号の代表的な手法

- 1977 年、Rivest-Shamir-Edelman の 3 名により発明。2000 年に特許期間満了 (現在特許フリー)。暗号化以外に「署名」の手法への応用も有名。
- RFC 8017 (PKCS#1 v2.2)、ANSI X9.31、IEEE 1363、CRYPTREC 等、各所で標準に採用。
- 鍵長は 1024–4096bits が標準的に使われている。[2]
- 暗号化・署名の際には、元のデータにパディングが必要。パディング方法によりセキュリティが大きく左右される。[3]

[2] 原理的には無限に伸ばせる。

[3] RSA-OAEP(暗号化)、RSA-PSS(署名) が現状ベターな方法。これを話す。

楕円曲線暗号 **(ECDH)** のさわり

Elliptic-Curve Cryptography

AES と比べた公開鍵暗号の Pros/Cons

	Pros	Cons
AES	・安全性を担保する鍵長が短い (128bits〜) ・一般的に高速・SoC での最適化も望める [4]	・パスワードなどの事前共有が必要
公開鍵暗号	・パスワードなどの秘密情報の事前共有が不要	・安全性を担保する鍵長が長い (RSA: 2048bits〜) ・一般的に非常に遅い・重い

⇒ 使い所を考えて組み合わせて使うことがオススメ (後述)

[4] Intel AES-NI

安全性を担保する鍵長が大きく違うのはどういうこと？

AES と比べた RSA・楕円曲線暗号の鍵のビット長比較[5]。横 1 行がだいたい同じくらいの安全性と言われる。

AES	RSA	楕円曲線
128	3072	256–383
192	7680	384–511
256	15360	512–

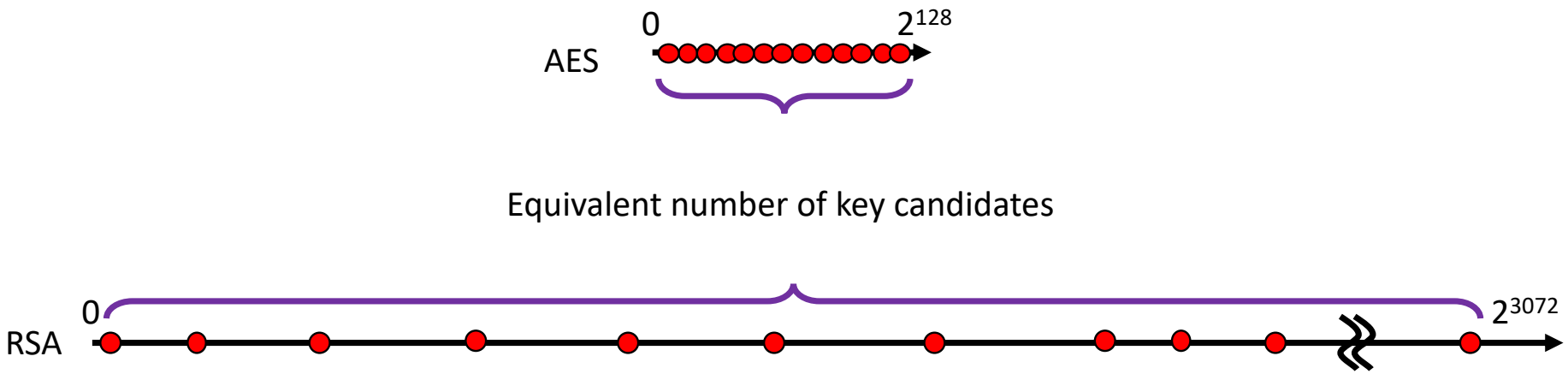
AES に比べて、楕円曲線で倍、RSA に至っては 24 倍以上の鍵長を使わないと、同じくらいの安全性を担保できない。

※ 鍵長は長ければ長いほど処理がどんどん重く・遅くなっていく…

[5] Recommendation for Key Management, Special Publication 800-57 Part 1 Rev. 4, NIST, 01/2016. https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-4/final

「AES-128 が、RSA-3072 と同じくらい」というイメージは、以下のように説明できる。

- AES: 数値 $= 0, 1, \ldots, 2^{128} - 1$ のうち、どれか 1 つが鍵。
- RSA: 特殊な条件を満たす数 = 素数の組を選んで、公開・秘密鍵を求める。



総当たりした時に「当たる」確率を揃えるには、RSA はその分巨大な素数まで候補にしないとならない。

RSA暗号を使ってみよう

RSA 暗号って何？

RSA 暗号を使うためのお作法

Padding:
RSA OAEP https://tools.ietf.org/html/rfc8017
PKCS1-v1.5 padding はやばい。Cryptrec からも落ちた。
https://www.cryptrec.go.jp/method.html
https://tools.ietf.org/html/rfc8017 既知の攻撃が知られており、
RSASSA, RSAES 共に PKCF1-v1.5 は非推奨@RFC
でも Node.js では OAEP 非サポートなので作った

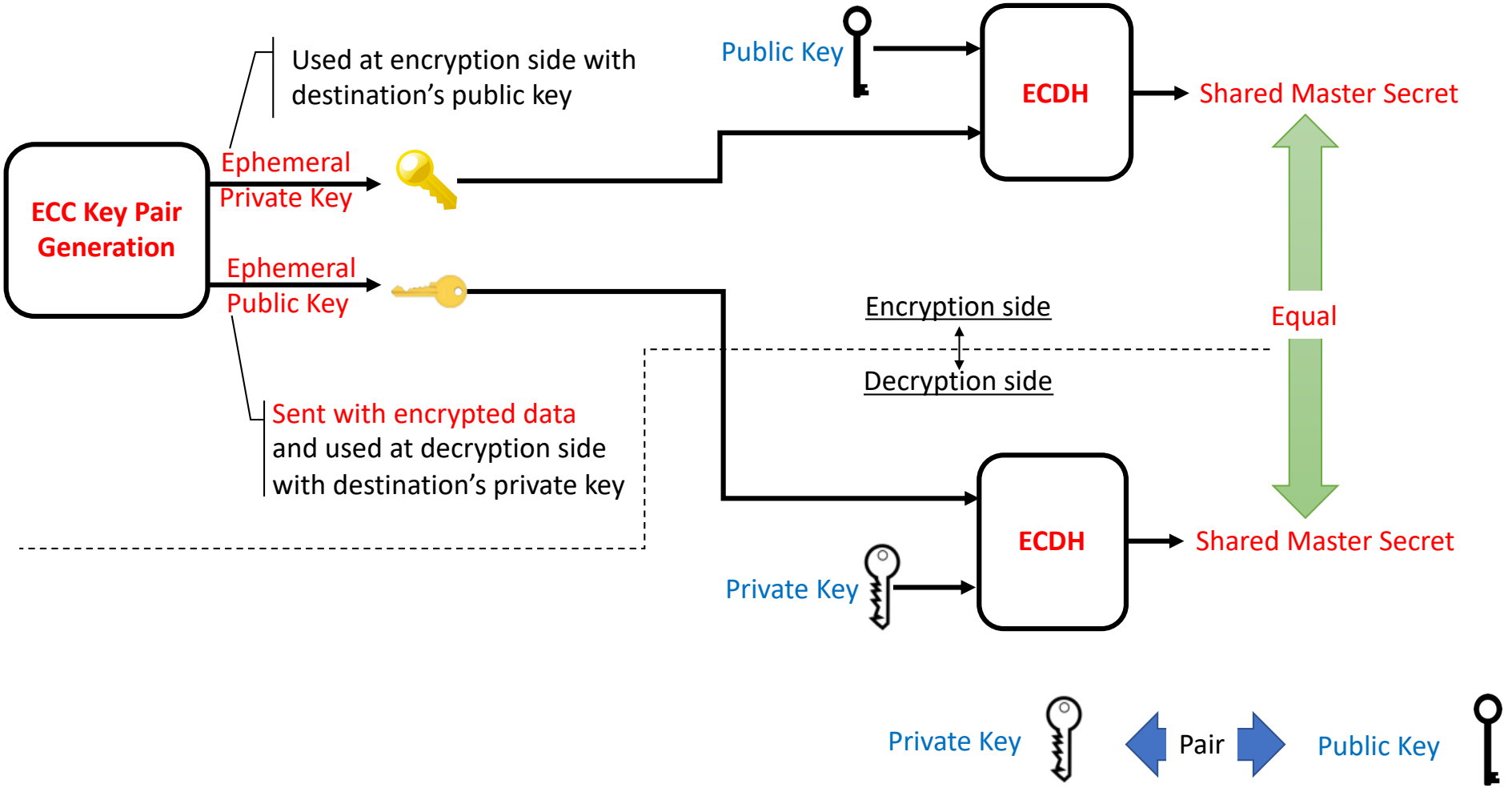
RSA-OAEP によるデータ暗号化をしてみよう

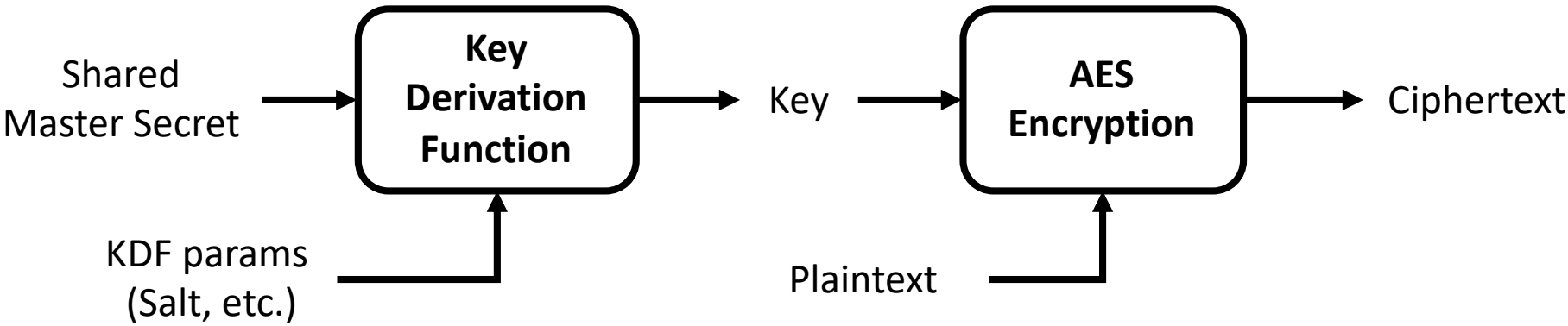
楕円曲線暗号 (ECDH) を使ってみよう

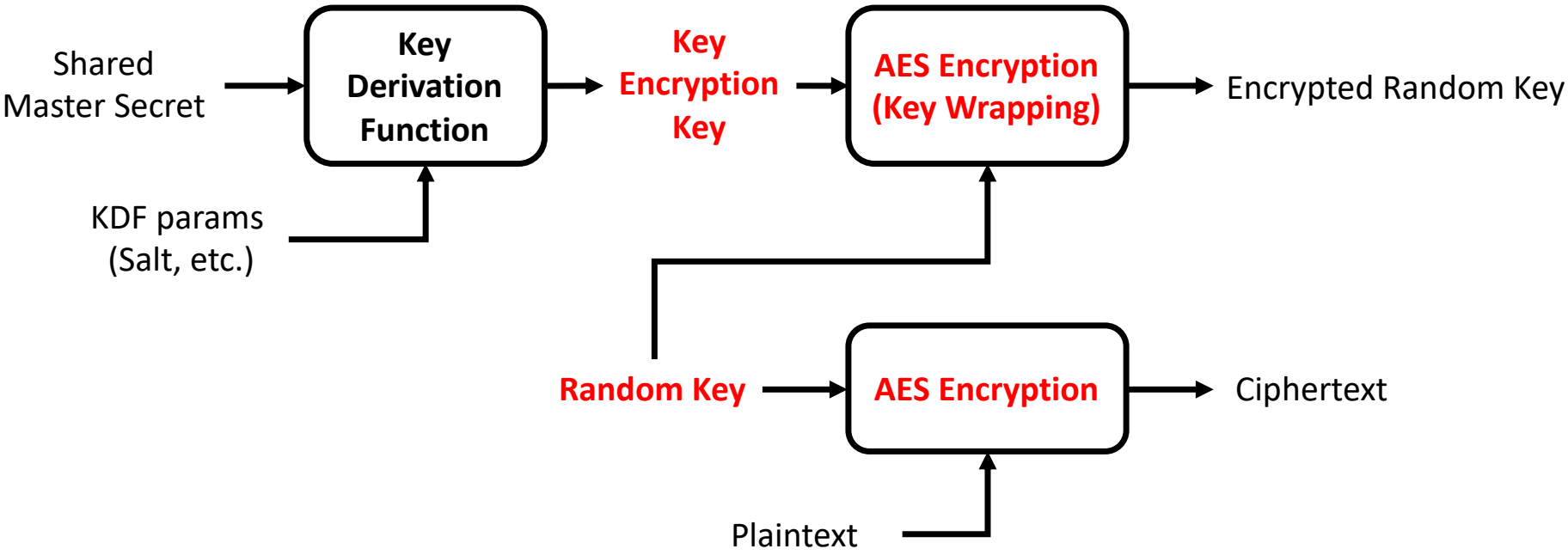
ECDHって何？

TLS https://tools.ietf.org/html/rfc8422
JOSE だと Concat KDF を使うだけ。
https://tools.ietf.org/html/rfc8037
直接 Concat KDF を暗号化の鍵にするか、あるいは Concat KDF の結果を AESKW の鍵として Content Encryption Key を暗号化するのに使う。

ECDH Ephemeral (ECDHE)




Shared
Master Secret
Key
Derivation
Function
KDF params
(Salt, etc.)
Key
AES
Encryption
Plaintext
Ciphertext


Shared
Master Secret
Key
Derivation
Function
KDF params
(Salt, etc.)
Key
Encryption
Key
AES Encryption
(Key Wrapping)
Encrypted Random Key
Random Key
AES Encryption
Ciphertext
Plaintext

ECDH によるデータ暗号化をしてみよう

今回は HKDF で暗号化してみる。

AESと公開鍵暗号のいいとこ取り

公開鍵暗号と **AES** の比較

ハイブリッド暗号化

AES の暗号化データをガンガン使い回せる！
msgpack-light を使ったコードを提供

まとめ

まとめ

お疲れ様でした。

- 公開鍵暗号を利用する際のお作法を学んだ。