LinuC

# ゼロから学ぶLinux
# ～暗号化の基本を知る～

・LinuC 試験の出題範囲対応
「1.10.3暗号化によるデータの保護」
「2.09.2OpenSSLとHTTPSの設定」
など

担当：河原木忠司

LPI-JAPAN

# 自己紹介

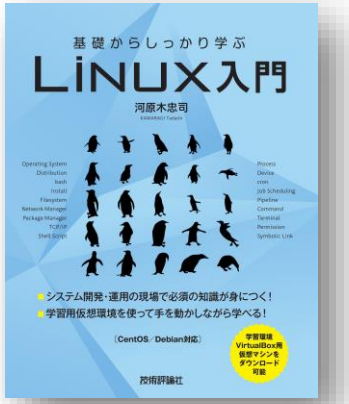
■河原木忠司（かわらぎただし）

- 20数年ほど、講師/エンジニアとして活動しております。
- 最近は講師、研修コンサルティング、執筆業に従事させていただいております。
  - サーバーインフラ系のコース、セキュリティ系のコースを中心に担当させていただいております。
  - LinuC Lv.1/Lv.2のver.10試験開発にも協力させていただきました。
    https://linuc.org/linuc/thanks.html

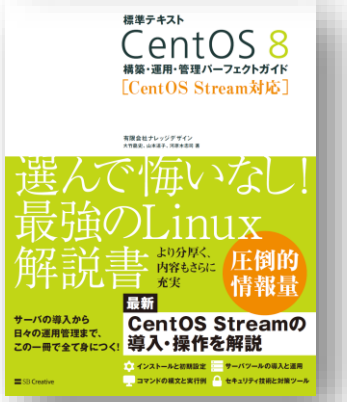
■最近執筆したもの


「最短突破 LinuCレベル1 合格教本 ver.10対応」（技術評論社）好評発売中です。

https://gihyo.jp/book/2020/978-4-297-11527-2




「基礎からしっかり学ぶ Linux 入門」（技術評論社）
https://gihyo.jp/book/2022/978-4-297-12545-5




「標準テキスト CentOS8 構築・運用・管理パーフェクトガイド[CentOS Stream対応]」（共著、SBクリエイティブ）
https://www.sbcr.jp/product/4815602567/

# セミナーの内容

- LinuCの位置づけ
- 暗号化とは
- 暗号化方式の違いと実装例
- デジタル証明書

【想定する受講対象】
Linuxについてゼロから学習したい人
Linuxを理解する前提として、暗号化の仕組みの基本についてゼロから学習したい人

■このセミナーでお話しする内容

暗号化の基本、各種暗号方式の基本について、ローカルファイルの暗号化のデモ操作を行いながらご紹介します。

お時間に余裕があれば、デジタル証明書、HTTPSの基本についてもご紹介をします。

※実装内容を把握するというより、「仕組み」「何か起こっているのか」を把握していただきたいです。

# LinuCの位置づけ

# LinuCとは

■クラウド時代の即戦力エンジニアであることを証明する
Linux技術者認定
https://linuc.org/

LPI-Japanで
運営

✓現場で「今」求められている新しい技術要素に対応
- オンプレミス／仮想化・コンテナを問わず様々な環境下でのサーバー構築
- 他社とのコラボレーションの前提となるオープンソースへの理解
- システムの多様化に対応できるアーキテクチャへの知見

✓全面的に見直した「今」身につけておくべき技術範囲を網羅
今となっては使わない技術やコマンドの削除、アップデート、新領域の取り込み

✓Linuxの範疇だけにとどまらない領域までカバー
セキュリティや監視など、ITエンジニアであれば必須の領域もカバー

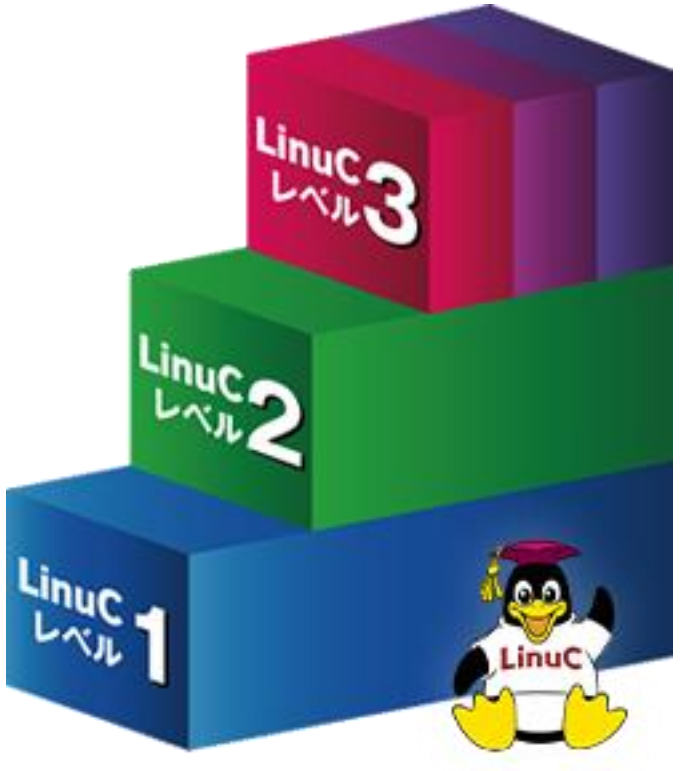
# LinuCとは

■**Lv.1～Lv.3までレベル分けをして、認定**

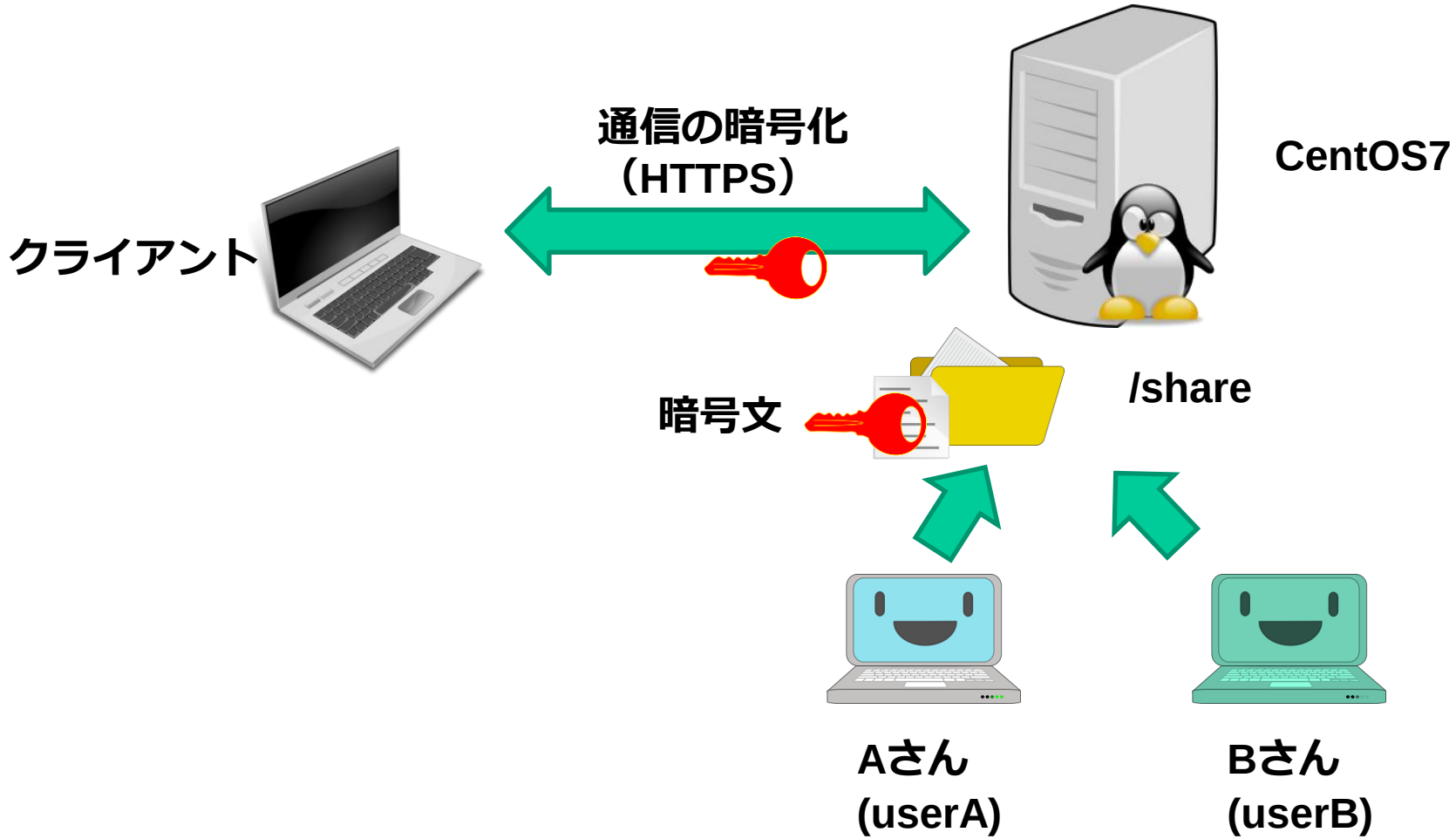
- **まずはLv.1の認定から**
- **Lv.1は101試験、102試験の両方に合格することで認定**
- **Lv.2は201試験、202試験の両方に合格することで認定**
- **Lv.3は300試験、303試験、304試験のいずれかに合格すること**

■**暗号化の仕組みについて問われる試験範囲**

- **101試験**
  - 1.01.1Linuxのインストール、起動、接続、切断と停止
- **102試験**
  - 1.10.3暗号化によるデータの保護
- **202試験**
  - 2.09.2OpenSSLとHTTPSの設定
  - 2.12.2OpenSSH サーバーの設定と管理
- **303試験**
  - **主題325：暗号化**



# 前提となる環境



# 【参考】前提となる環境の構築（GnuPG）

鍵ペアを生成

```
[userA@centos7 ~]$ gpg --gen-key
gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
                :
鍵の有効期間は? (0)
(null)は無期限です
これで正しいですか? (y/N) y
```

アルゴリズムや有効期限を既定のままにして進める

```
GnuPGはあなたの鍵を識別するためにユーザIDを構成する必要があります。

本名: userA
電子メール・アドレス: usera@example.com
コメント:
次のユーザIDを選択しました:
    "userA <usera@example.com>"
```

ユーザー情報の入力

```
名前(N)、コメント(C)、電子メール(E)の変更、またはOK(O)か終了(Q)? O
秘密鍵を保護するためにパスフレーズがいります。
                :
gpg: 鍵F6506C41を絶対的に信用するよう記録しました
公開鍵と秘密鍵を作成し、署名しました。
                :
uid                  userA <usera@example.com>
sub   2048R/3416FCC0 2023-01-19
```

パスフレーズの指定

鍵ペアの生成完了

# 【参考】前提となる環境の構築（Apache httpd: mod_ssl）

## ■httpd、mod_sslのインストール

```
[root@centos7 ~]# yum -y install httpd mod_ssl
```

## ■DNSサーバーにより、example.comドメインを構成

```
[root@centos7 ~]# host www.example.com
www.example.com has address 192.168.56.11
[root@centos7 ~]# host ns.example.com
ns.example.com has address 192.168.56.11
```

# 【参考】前提となる環境の構築（Apache httpd: mod_ssl）

## ■CA機能を構成

CA機能を新規構成

```
[root@centos7 ~]# /etc/pki/tls/misc/CA -newca
        :
writing new private key to '/etc/pki/CA/private/./cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
        :
Country Name (2 letter code) [XX]:JP
State or Province Name (full name) []:Tokyo
Locality Name (eg, city) [Default City]:Arakawa-ku
Organization Name (eg, company) [Default Company Ltd]:testCA
Organizational Unit Name (eg, section) []:test
Common Name (eg, your name or your server's hostname) []:ca.example.com
Email Address []:info@example.com
        :
Enter pass phrase for /etc/pki/CA/private/./cakey.pem:
        :
[root@centos7 ~]# scp /etc/pki/CA/cacert.pem 192.168.56.23:~
root@192.168.56.23's password:
```

パスフレーズの設定

地域情報などの設定

設定したパスフレーズを入力

ルート証明書（CAの証明書）をコピー

# 【参考】前提となる環境の構築（Apache httpd: mod_ssl）

## ■HTTPSの構成：秘密鍵の生成、CSRの構成

```
[root@centos7 ~]# openssl genrsa 2048 > server.key
Generating RSA private key, 2048 bit long modulus
          :
[root@centos7 ~]# openssl req -new -key server.key > server.csr
          :
Country Name (2 letter code) [XX]:JP
State or Province Name (full name) []:Tokyo
Locality Name (eg, city) [Default City]:Arakawa-ku
Organization Name (eg, company) [Default Company Ltd]:test Co.
Organizational Unit Name (eg, section) []:test
Common Name (eg, your name or your server's hostname) []:www.example.com
Email Address []:info@example.com
          :
[root@centos7 CA]# vim san.txt
subjectAltName = DNS:*.example.com, IP:192.168.56.11
          :
```

秘密鍵の生成

公開鍵を含むCSRの生成

地域情報などを設定

Webサーバーのホスト名を指定

Webサーバーのホスト名、IPアドレスを指定

# 【参考】前提となる環境の構築（Apache httpd: mod_ssl）

## ■HTTPSの構成：サーバー証明書の発行（署名）、設定ファイルの構成

```
[root@centos7 CA]# cd /etc/pki/CA
[root@centos7 CA]# openssl x509 -CA cacert.pem -CAkey private/cakey.pem -req -in ~/server.csr -out
~/server.crt -extfile san.txt -days 365 –CAcreateserial
[root@centos7 CA]# cp ~/server.{crt,key} /etc/httpd/
[root@centos7 CA]# vim /etc/httpd/conf.d/ssl.conf
        :
SSLCertificateFile /etc/httpd/server.crt
        :
SSLCertificateKeyFile /etc/httpd/server.key
        :
[root@centos7 CA]# systemctl restart httpd
```

サーバー証明書の発行（署名）

サーバー証明書と秘密鍵をhttpdのServerRootにコピー

サーバー証明書と秘密鍵のパスを設定ファイルに構成

httpdを再起動し、記述した設定内容を反映

# 暗号化とは

# 暗号化とは

■「暗号化とは、データを正規の権限のない人が読めないように、一定の計算手順に基いて元の状態が容易に推定できない形に変換すること。データを保存したり誰かに伝送する際に、意図せず第三者に盗み見られたり改竄されたりしないために行われる。」

- 暗号化（エンクリプション）とは - IT用語辞典 e-Words
  - https://e-words.jp/w/%E6%9A%97%E5%8F%B7%E5%8C%96.html

元の状態＝平文
推測できない状態＝暗号文

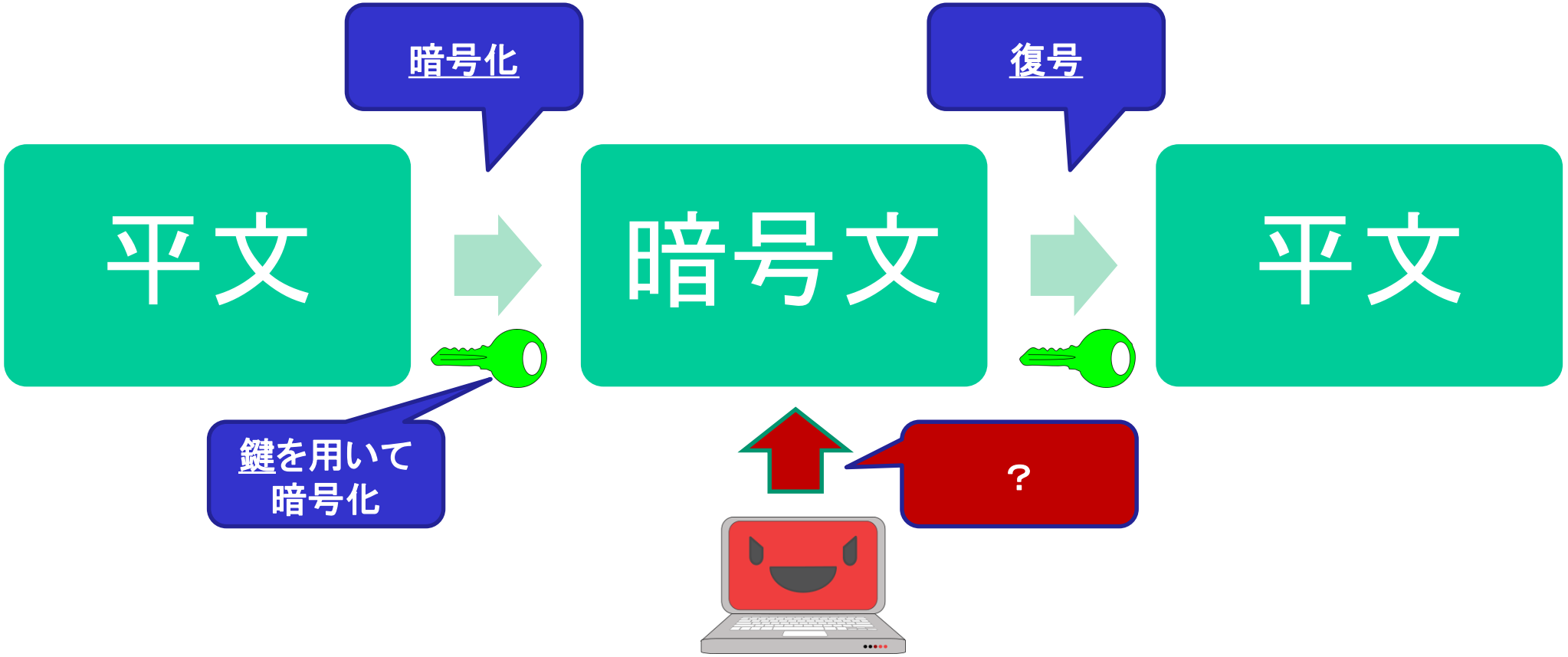
暗号文を生成するための
鍵（＝値）を利用

アルゴリズム

■同じ平文を同じアルゴリズムで暗号化しても、同じ暗号文にはならない

- point!
  - 暗号化には「アルゴリズム」と「鍵」が必要

# 暗号化と復号




- point!
  - 暗号文を生成⇒暗号化
  - 暗号文を平文に戻す⇒復号
  - 暗号化と復号には、同じアルゴリズム、そして対応する鍵が必要
  - 復号に利用する鍵が漏洩すると、第三者に読み取られる可能性

# 「強い」暗号化

## 脆弱な暗号化

- 鍵の内容がわからなくても、暗号化した内容が読み取られる可能性がある
- 古くて、脆弱になったアルゴリズムによる暗号化など。
  - 脆弱なアルゴリズムを利用した操作は拒否されることもある。

## 「強い」暗号化

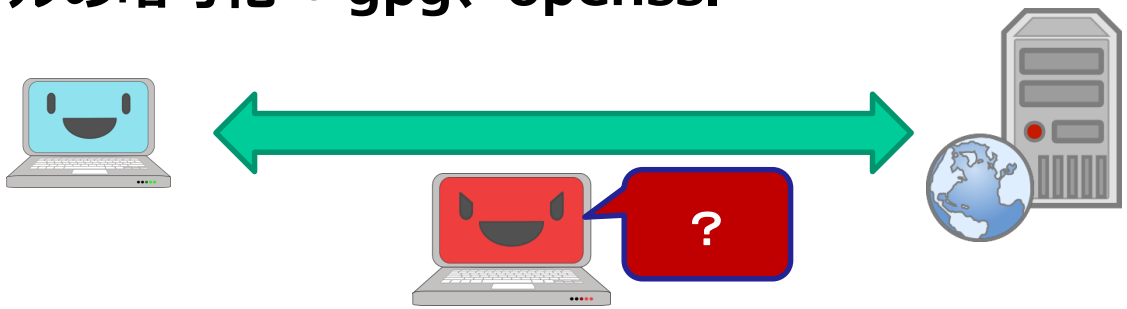
- 強力なアルゴリズム
  - 複雑な計算処理による暗号化
  - 以前は強力だったアルゴリズムも時を経て、脆弱なアルゴリズムになることがあり得る。
- 長い鍵
  - 大きい数値を用いた計算処理

# 暗号化の利用

## 機密性の確保

■通信内容やファイルの内容が許可したユーザーのみ読み取れるようにする

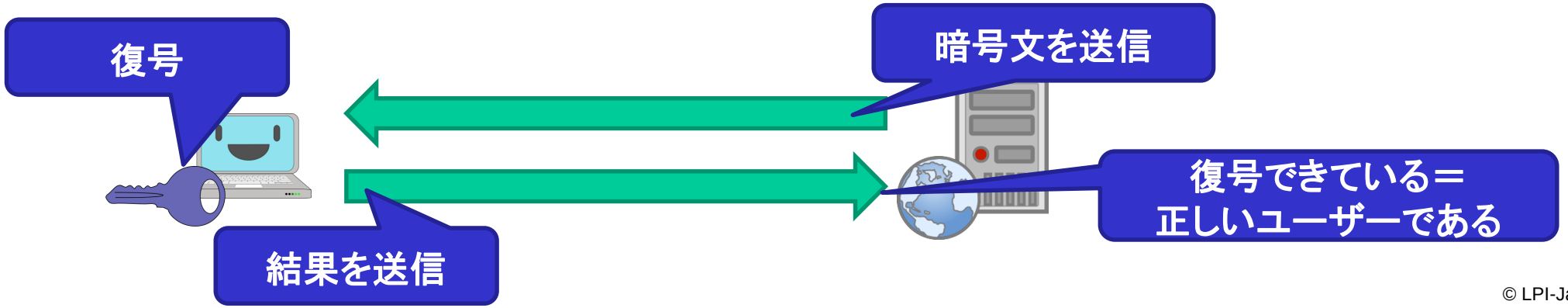
- 通信内容の暗号化⇒ SSH、HTTPS
- ファイルの暗号化⇒ gpg、openssl




## 認証

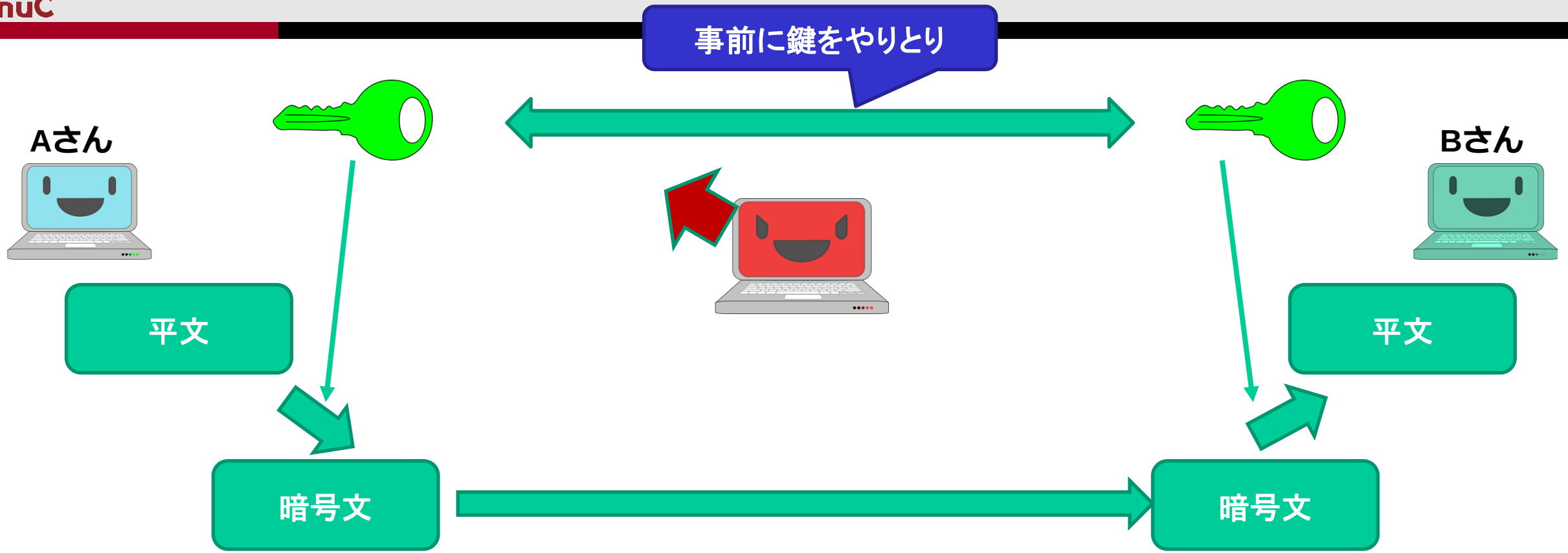
■正しい鍵を所有しているユーザー⇒認証成功

- SSHにおける公開鍵ペアによる認証



# 暗号化方式の違いと実装例

# 共通鍵暗号方式




- point!
  - 同じ鍵で暗号化、復号を行う
  - 鍵を盗聴されずにやりとりすることが課題

# 共通鍵暗号方式によるファイル暗号化の実装例

- パスフレーズによるファイルの暗号化

```
[userA@centos7 ~]$ echo test > userA.txt
[userA@centos7 ~]$ gpg -c userA.txt
          :
[userA@centos7 ~]$ ls user*
userA.txt  userA.txt.gpg
[userA@centos7 ~]$ mv userA.txt.gpg /share
```

「gpg -c ファイル名」で、パスフレーズによる暗号化

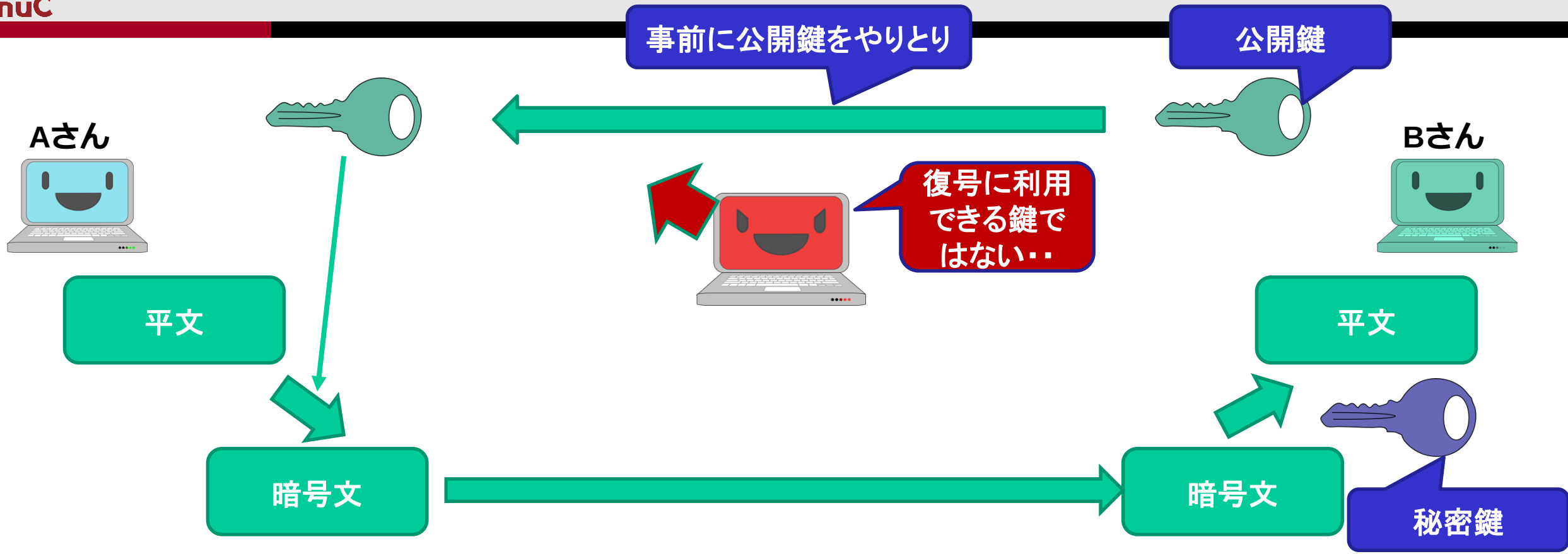
～.gpg: 暗号化されたファイル

- パスフレーズを指定したファイルの復号

```
[userB@centos7 ~]$ mv /share/userA.txt.gpg .
[userB@centos7 ~]$ gpg userA.txt.gpg
          :
[userB@centos7 ~]$ ls user*
userA.txt  userA.txt.gpg
[userB@centos7 ~]$ cat userA.txt
test
```

「gpg ファイル名」で、ファイルの復号
※暗号化したときに指定したパスフレーズを入力

# 公開鍵暗号方式




- point!
  - 受信者の公開鍵で暗号化、受信者の秘密鍵で復号を行う
  - 公開鍵は漏洩しても構わない⇒鍵のやりとりの課題が解消
  - 共通鍵暗号方式に比べて、処理に時間がかかる

# 公開鍵暗号方式によるファイル暗号化の実装例

- 公開鍵のエクスポート

```
[userB@centos7 ~]$ gpg -o /share/userB.pubkey -a --export
[userB@centos7 ~]$ ls /share/userB*
/share/userB.pubkey
```

受信者の公開鍵をエクスポート

- 公開鍵のインポート

送信者の環境で受信者の公開鍵をインポート

```
[userA@centos7 ~]$ gpg --import /share/userB.pubkey
gpg: 鍵2C321991: 公開鍵"userB <userb@example.com>"をインポートしました
gpg:         処理数の合計: 1
gpg:           インポート: 1  (RSA: 1)
[userA@centos7 ~]$ gpg --list-keys
/home/userA/.gnupg/pubring.gpg
------------------------------
          :
pub   2048R/2C321991 2023-01-19
uid                  userB <userb@example.com>
sub   2048R/4E523CD9 2023-01-19
```

インポートされていることを確認

# 公開鍵暗号方式によるファイル暗号化の実装例

- インポートした公開鍵に署名を追加

インポートした鍵を署名（鍵を信頼）

```
[userA@centos7 ~]$ gpg --sign-key userb@example.com
          :
本当にこの鍵にあなたの鍵"userA <usera@example.com>"で署名してよいですか
(F6506C41)
本当に署名しますか? (y/N) y
          :
```

- 公開鍵による暗号化

公開鍵を利用し、暗号化

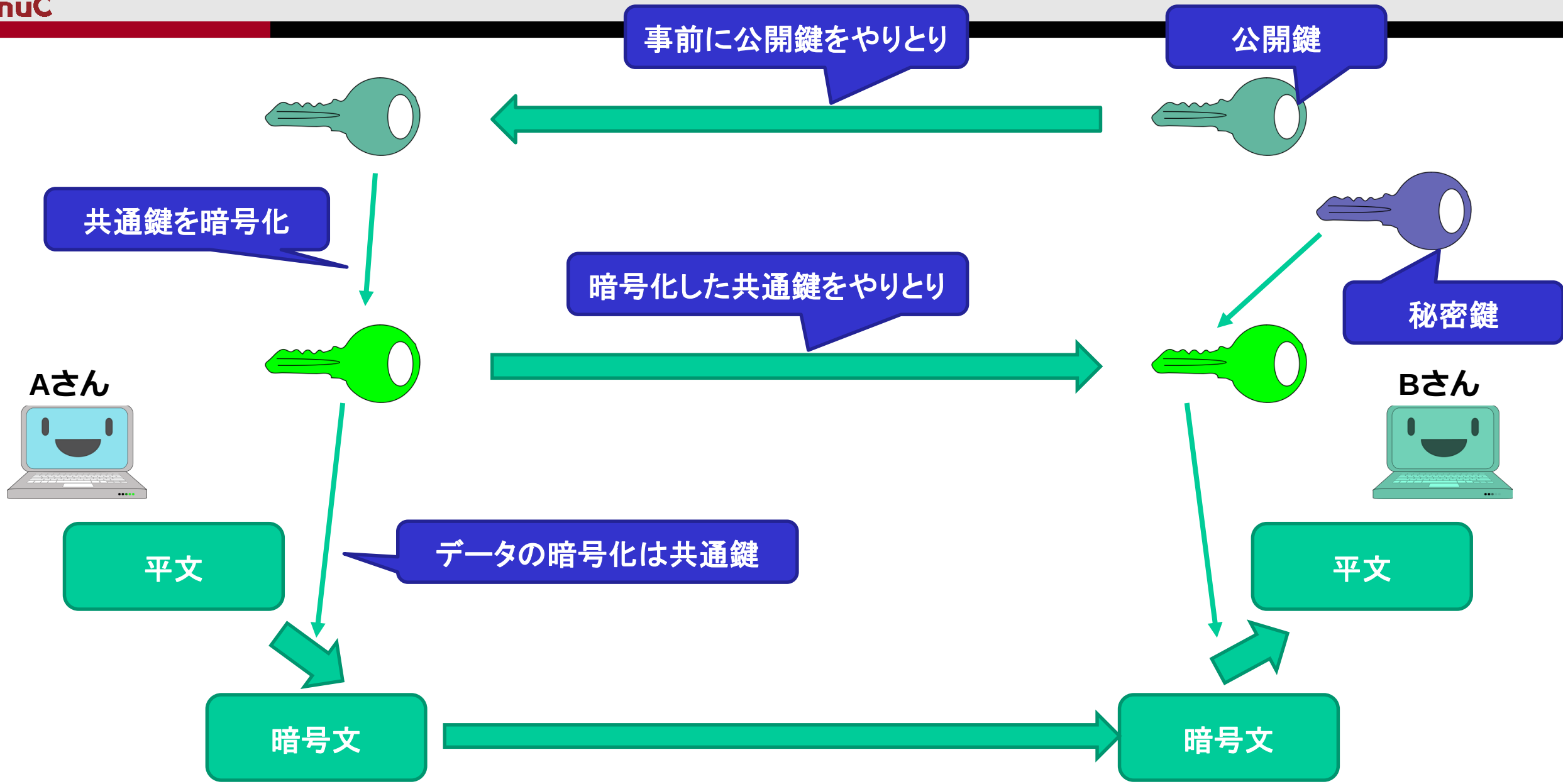
```
[userA@centos7 ~]$ echo test2 > userA2.txt
[userA@centos7 ~]$ gpg -e -a -r userB@example.com userA2.txt
          :
[userA@centos7 ~]$ ls userA*
userA.txt  userA2.txt  userA2.txt.asc
```

～.asc: 公開鍵により暗号化されたファイル

```
[userA@centos7 ~]$ mv userA2.txt.asc /share
```

共通鍵の時と同様、gpgコマンドで復号

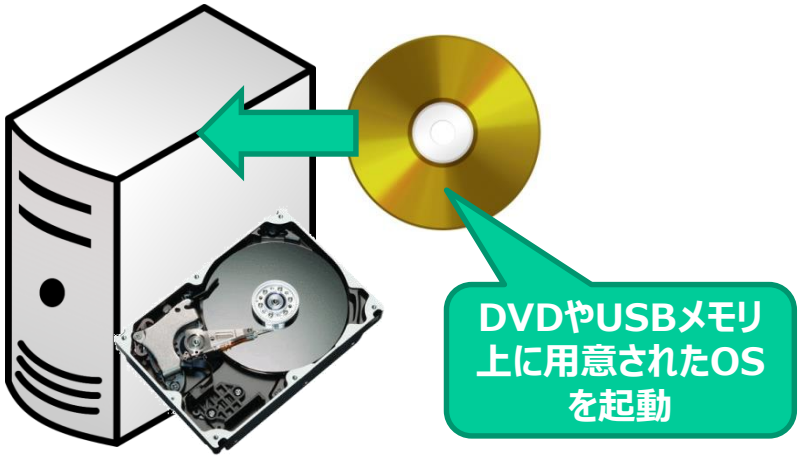
# ハイブリッド暗号方式



# ファイルの暗号化：パーミッション設定と何が違う？

■パーミッション設定は、OSが起動中のみ有効

- 暗号化していないファイル/ディスクは、別のOSで起動すると、パーミッション設定に関係なく、内容を参照できてしまう。
- 暗号化されているファイル/ディスクは、別のOSで起動しても、復号できなければ、内容を参照できない。



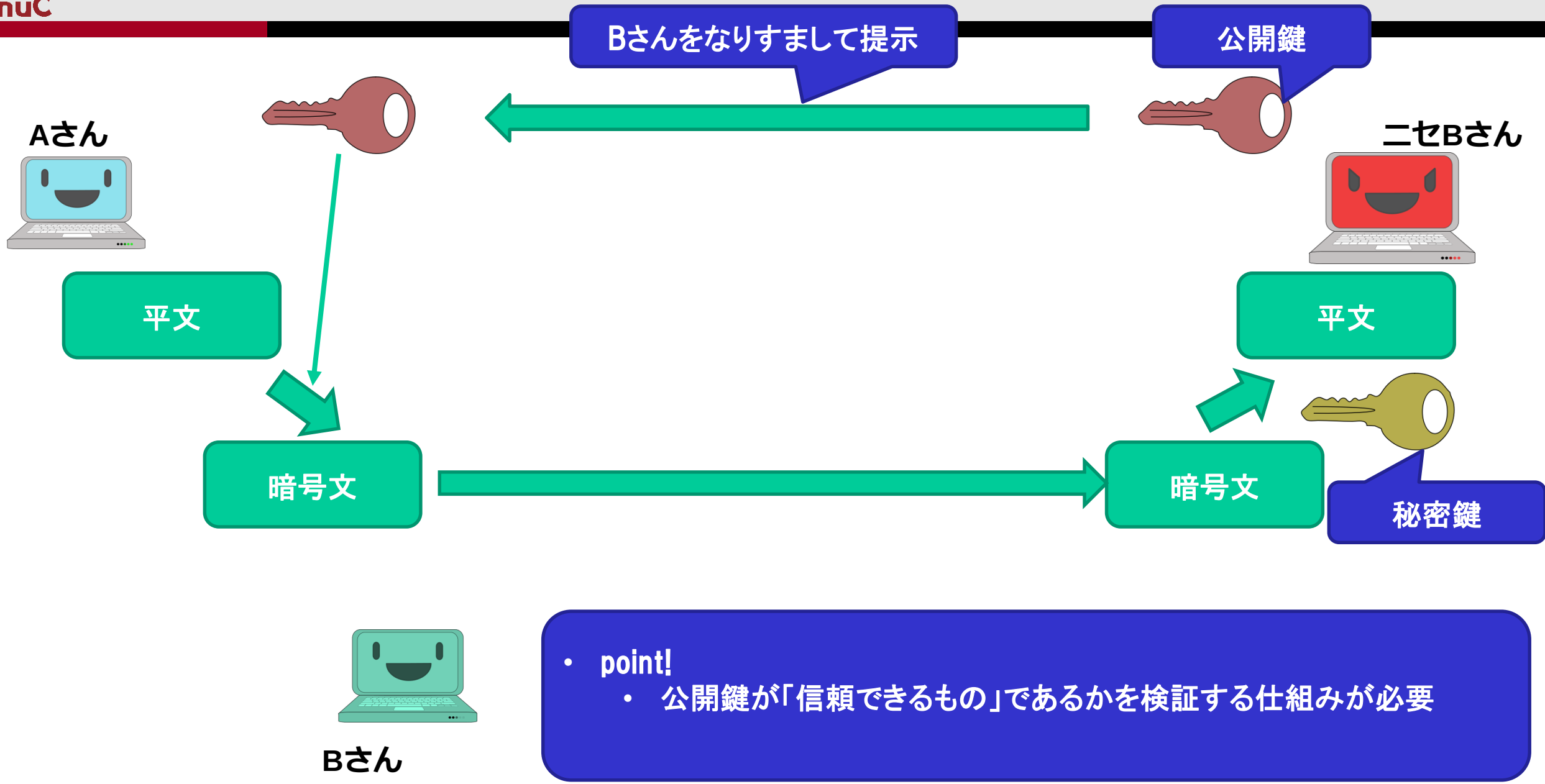

```
┌──(root㉿kali)-[~]
└─# cat /media/kali/ba395081-c459-42a1-8651-02764ca5294f/home/userA/userA2.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.22 (GNU/Linux)

hQEMA7HSXU9OUjzZAQf+Jk6TadhrmmR6CiGyAkrB+k8//USu5BIUK[illegible]QYVaaeKuMk
6Ei9Pzh/RdSGRLy7N7jt/HTU2LLqMEceNK+UmgSENTHi5cXOlDwXUCQn4T1j3fUD
cmsj5liPK9bV1PPFLDgQmYgzu0kv5/D817dIh1gRfoQGzEj2cYLCyeet0V6nhJSK
zeuWDXUtAzb0GgyFpm3Sx9fGs3nRLO56XqzD87g6tw58ziYeMSxSMlXl0dV5btUa
65VarlyBqPBM4HG2t+1FiDqw3G8y2ouDfHjfduWJT2h8NXFBYrM6780ORSl8CoRP
QGSl+k93hgSpPSae5yXiQWFPlnmdMizK0cY5eeavt9JLAUhanPF+svod+mv9C8xb
hRr0eCqHBOb+QFtrZkM9ArSbvPOw1DlcOgZV+17OL9k3wMpMNKbsYI7I5qSxFeTl
J1/aj1QqLKuXZUNo
```

別のOSで起動しても
暗号化されたまま

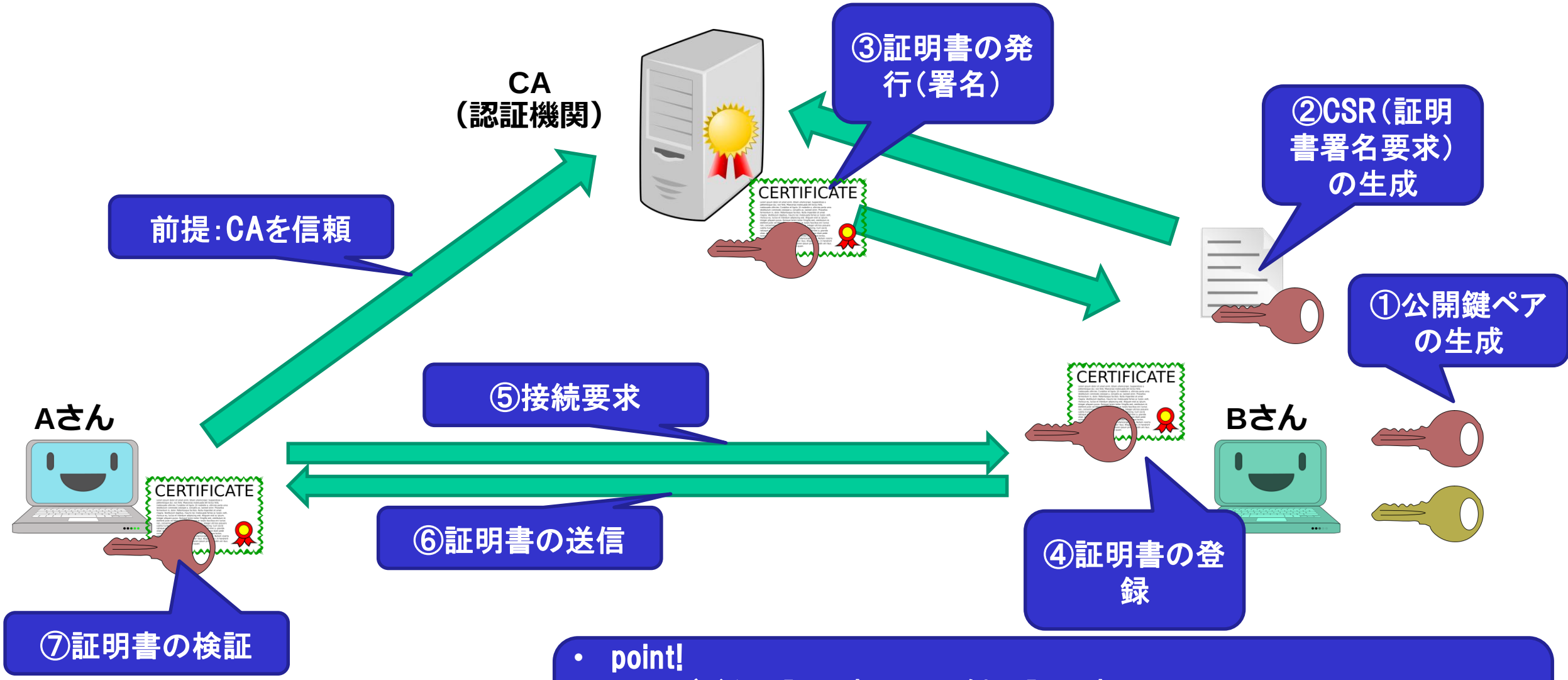
# デジタル証明書

# なりすましユーザーによる公開鍵の提示




- point!
  - 公開鍵が「信頼できるもの」であるかを検証する仕組みが必要

# デジタル証明書



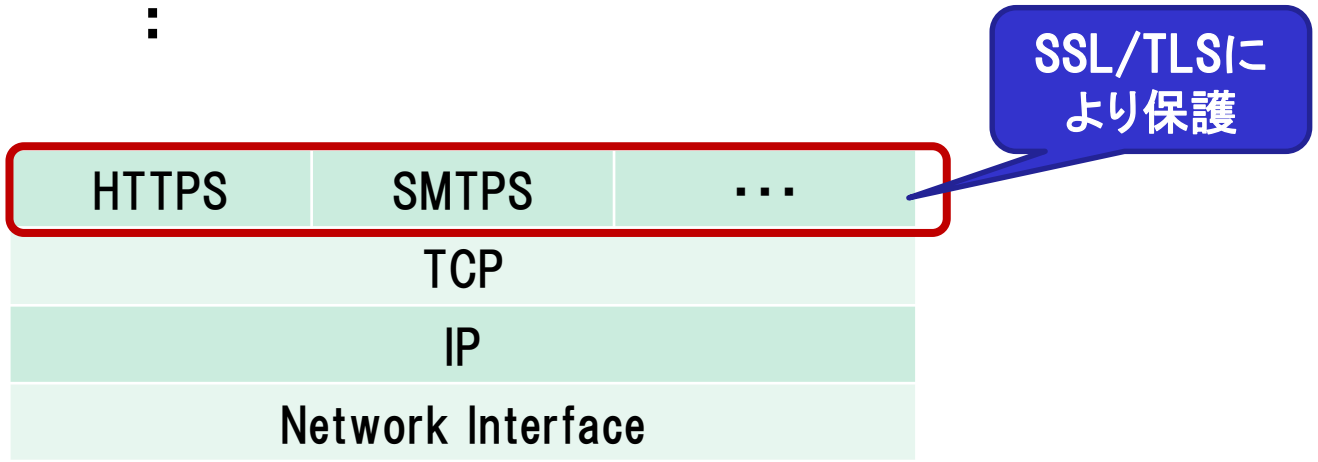

- point!
  - デジタル証明書＝公開鍵の証明書
  - 証明書を利用した信頼できるネットワーク基盤⇒PKI（公開鍵基盤）

# SSL/TLS

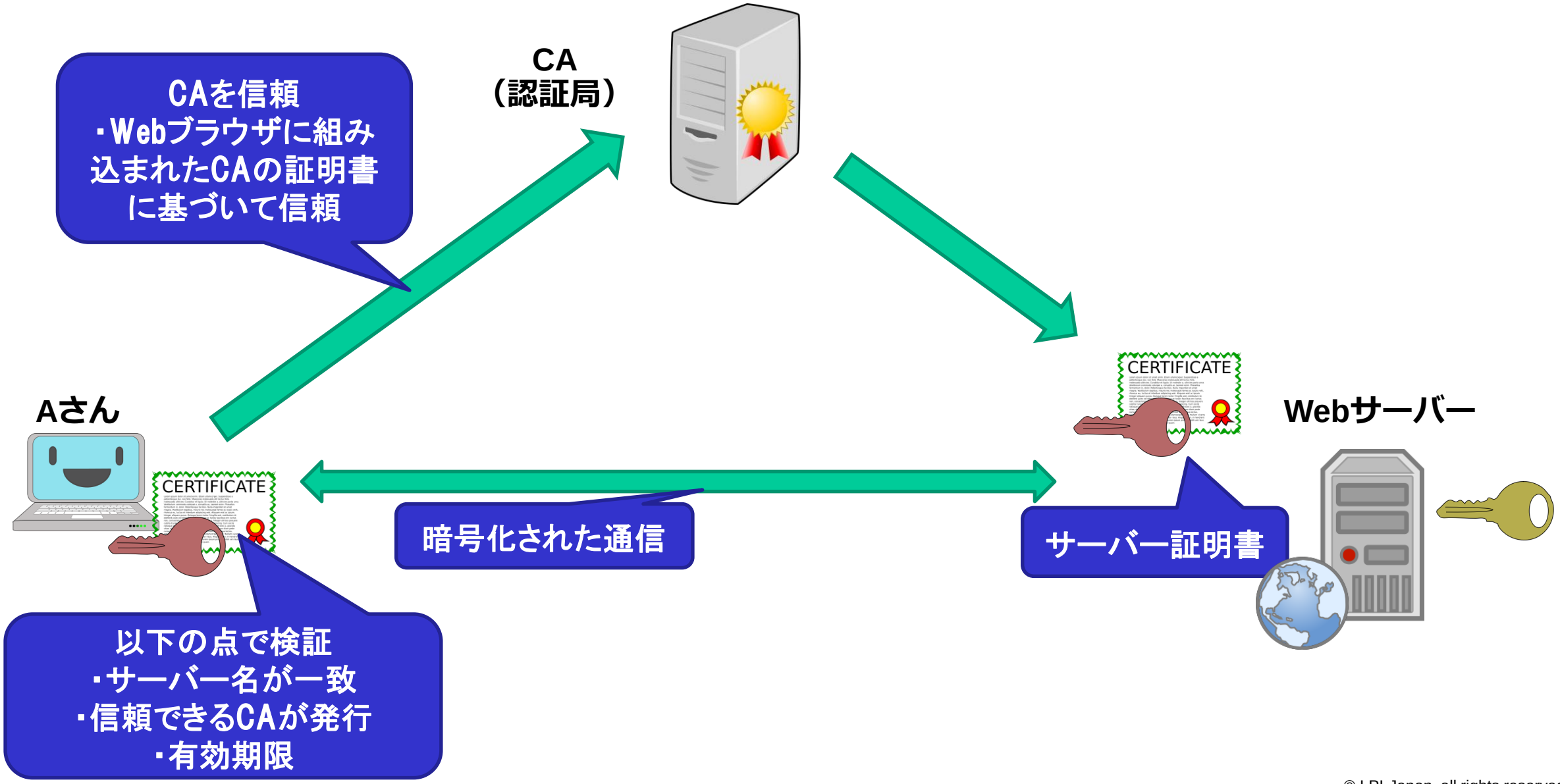
■アプリケーションプロトコルを保護する仕組み

- HTTP通信の保護⇒HTTPS
- SMTP通信の保護⇒SMTPS
- LDAP通信の保護⇒LDAPS

：

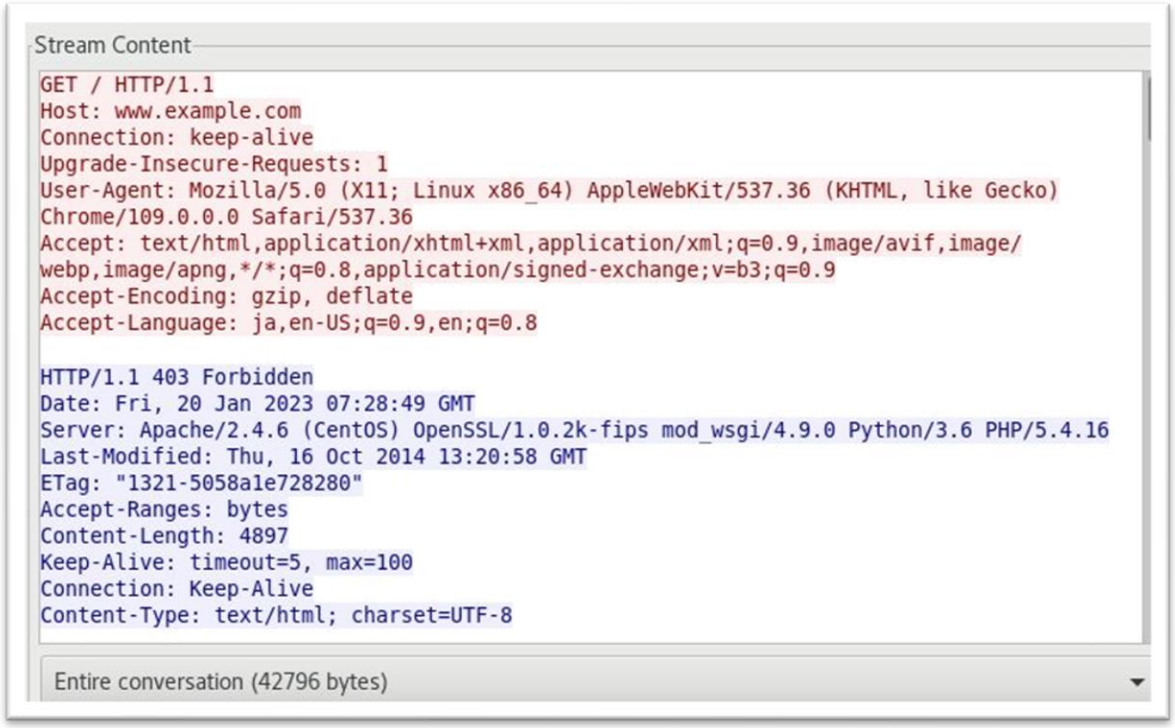


# HTTPS

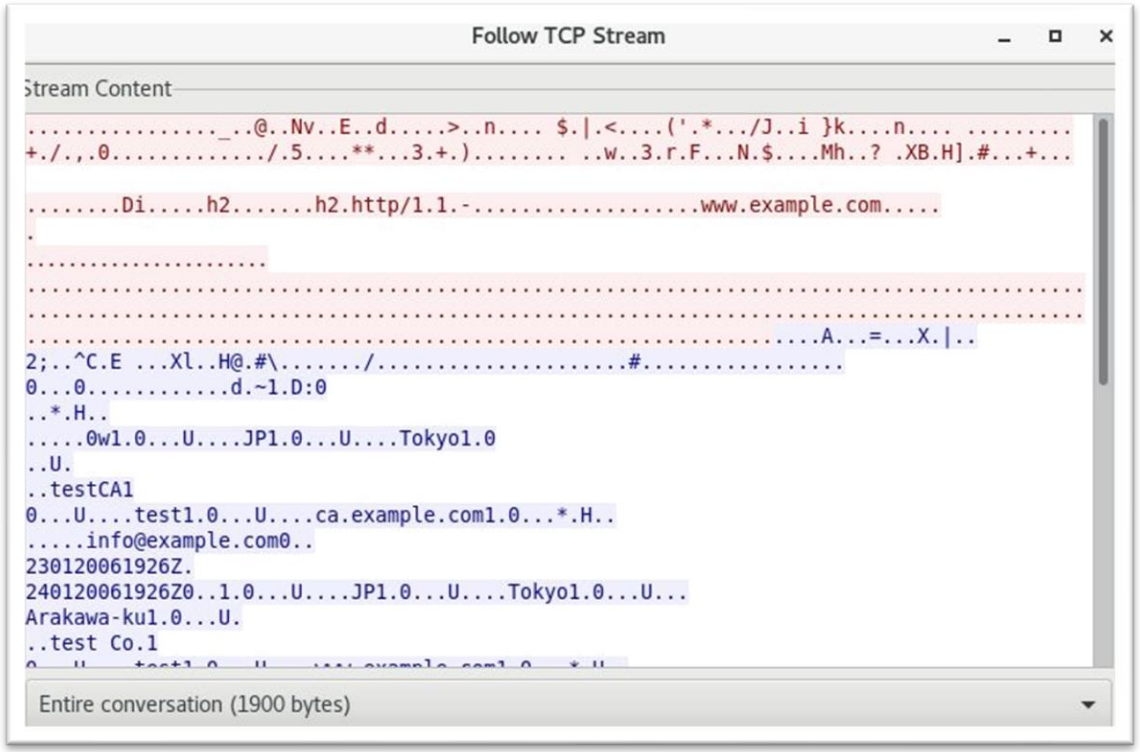


# HTTPS

## ■HTTP通信は非暗号化



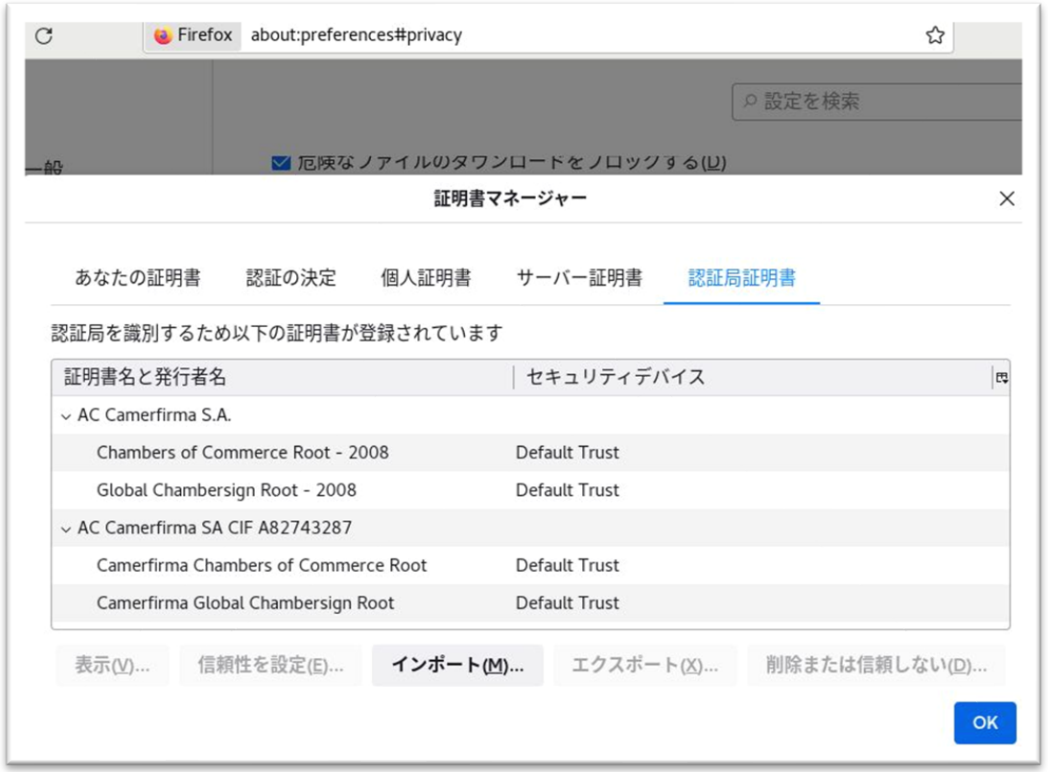

## ■HTTPS通信は既定で暗号化



# Webブラウザに組み込まれているCAの証明書

## Firefox

- 設定画面 – [プライバシーとセキュリティ] – [証明書を表示]



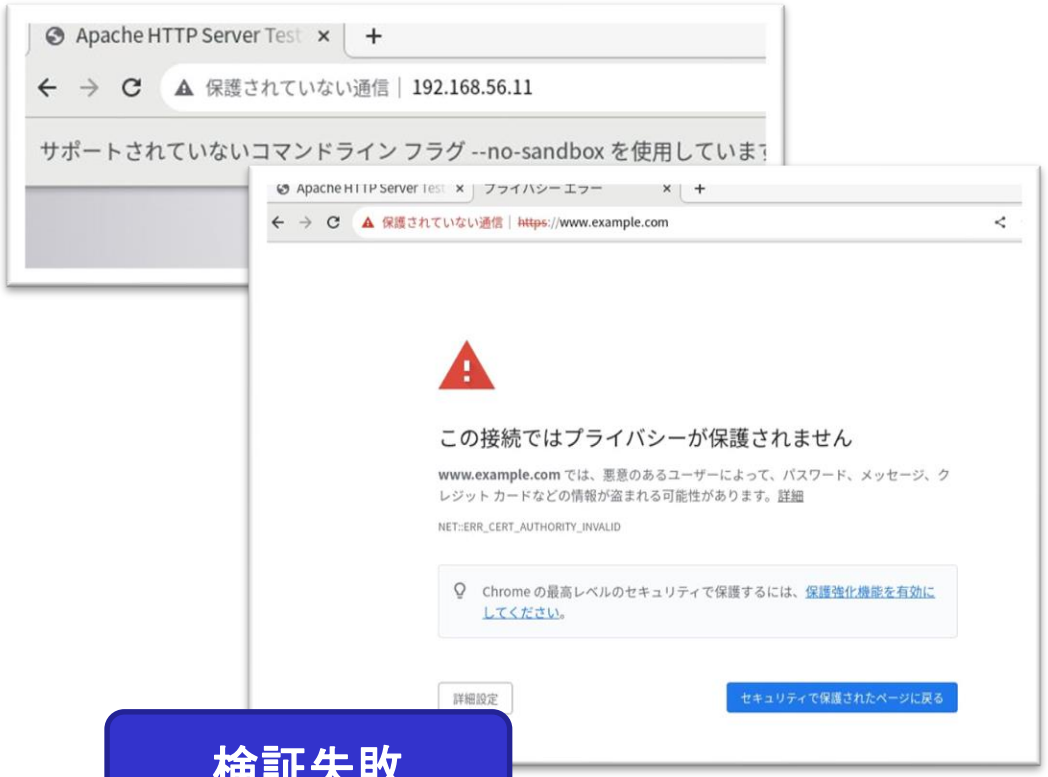

## Google Chrome

- 設定画面 - [プライバシーとセキュリティ] – [セキュリティ] – [証明書の管理]

# 証明書の検証

- クライアント（Webブラウザ）は、接続先サーバーが信頼できることを確認するため、以下の項目を検証
  - 信頼しているCAが発行した証明書
  - 有効期限が切れていない証明書
  - 証明書に登録されている名前と接続先が一致


検証成功




検証失敗

# Q&A / おわりに

■ご参加いただき、ありがとうございました。

■ご質問がありましたら、よろしくお願いいたします。

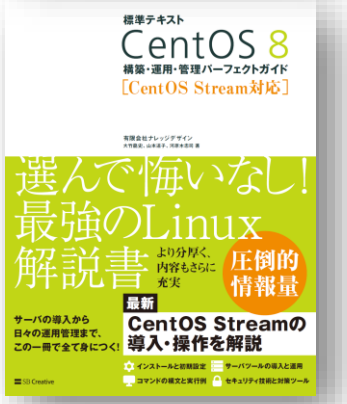
■参考文献


「最短突破 LinuCレベル1 合格教本 ver.10対応」（技術評論社）好評発売中です。

https://gihyo.jp/book/2020/978-4-297-11527-2


「Linux標準教科書」(LPI-Japan)
※オンラインでダウンロードして利用できます


「暗号技術のすべて」（ipusiron著、翔泳社）
https://www.shoeisha.co.jp/book/detail/9784798148816




「標準テキスト　CentOS8 構築・運用・管理パーフェクトガイド[CentOS Stream対応]」（共著、SBクリエイティブ）
https://www.sbcr.jp/product/4815602567/