SSHの仕組み

サーバサイド班　甲本健太

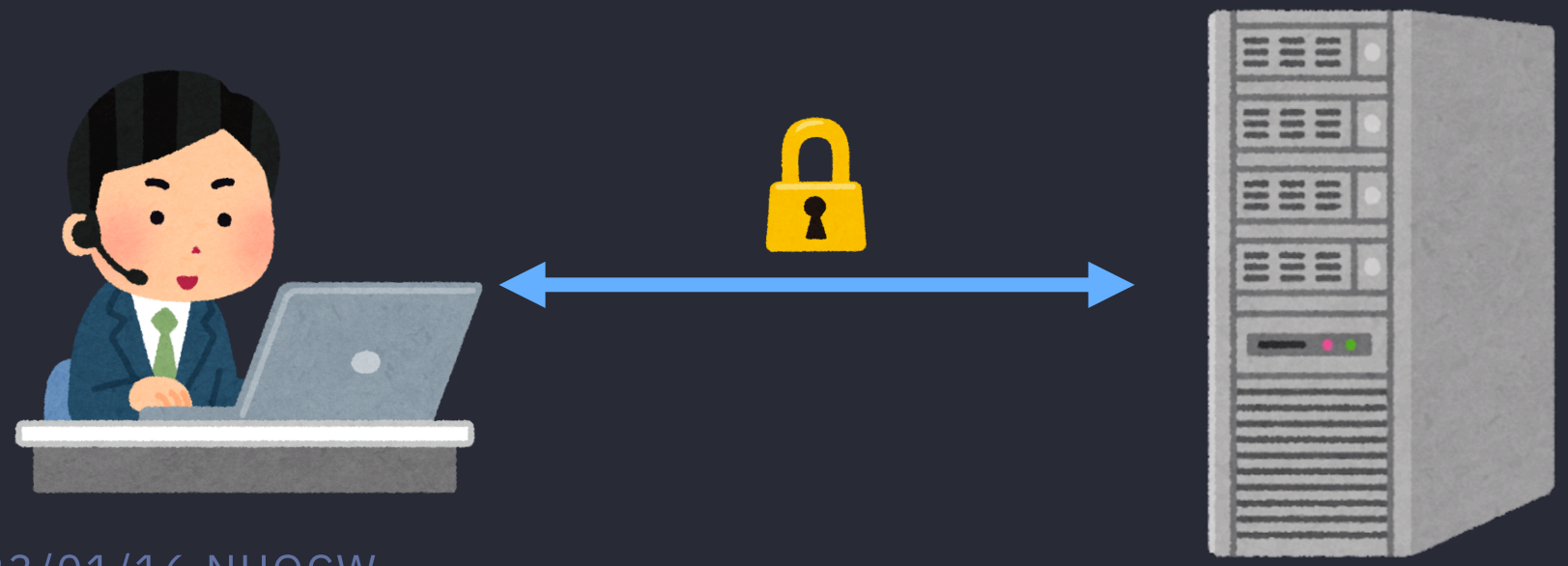
目次

- SSHとは
  - SSH接続の流れ
    1. クライアントによるサーバの認証
    2. セッションキーの生成
    3. サーバによるクライアントの認証
- SSHのログを見てみよう
- （おまけ）RSAとECDSA
- 参考

SSHとは

SSHとは**Secure Shell**（安全なシェル）の略称であり、安全にリモートコンピュータと接続するためのプロトコル。

- サーバに接続して作業するときなどに利用する



実演

コマンド

```
$ ssh -i {秘密鍵} {ユーザ名}@{接続先のIPアドレス}
```

ssh-configを使う

```
Host {登録名}
    HostName ocw.nagoya-u.jp  # 接続先
    User {ユーザ名}
    IdentityFile {秘密鍵のパス}
```

SSH接続の流れ

認証段階

1. クライアントによるサーバの認証
2. セッションキーの生成
3. サーバによるクライアントの認証

通信段階

4. データを暗号化して通信

1. クライアントによるサーバの認証

クライアント側が、サーバが怪しいものでないかを調べる

初めてサーバにアクセスするとき

サーバの公開鍵を検証し、**手動で**検証を行う
→ `Are you sure you want to continue connecting (yes/no)?`
ローカルの `known_hosts` ファイルにその情報を保存する。

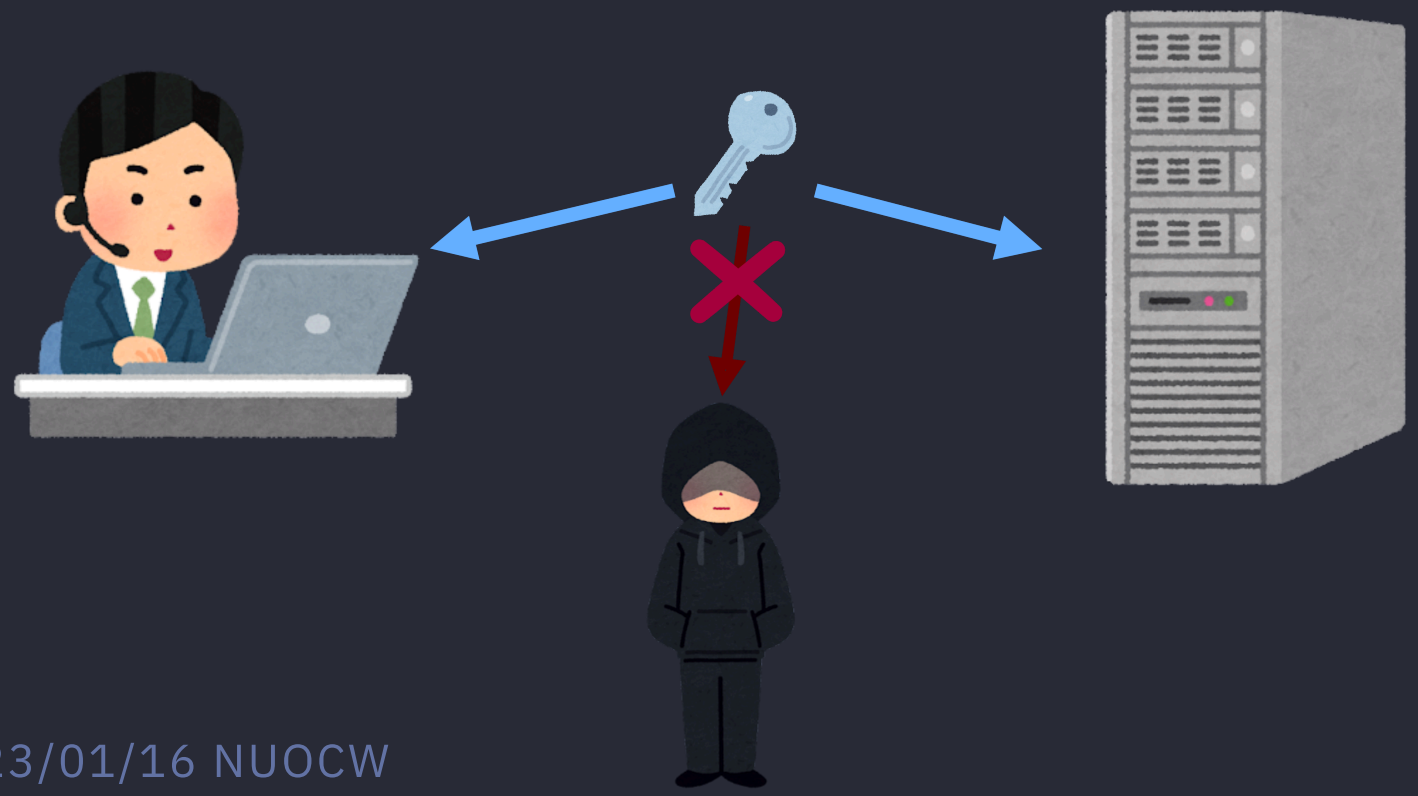
（他にも `ssh-keyscan` コマンドを用いるなどの方法もある）

2回目以降

ローカルの `known_hosts` にある公開鍵情報からサーバの身元を確認

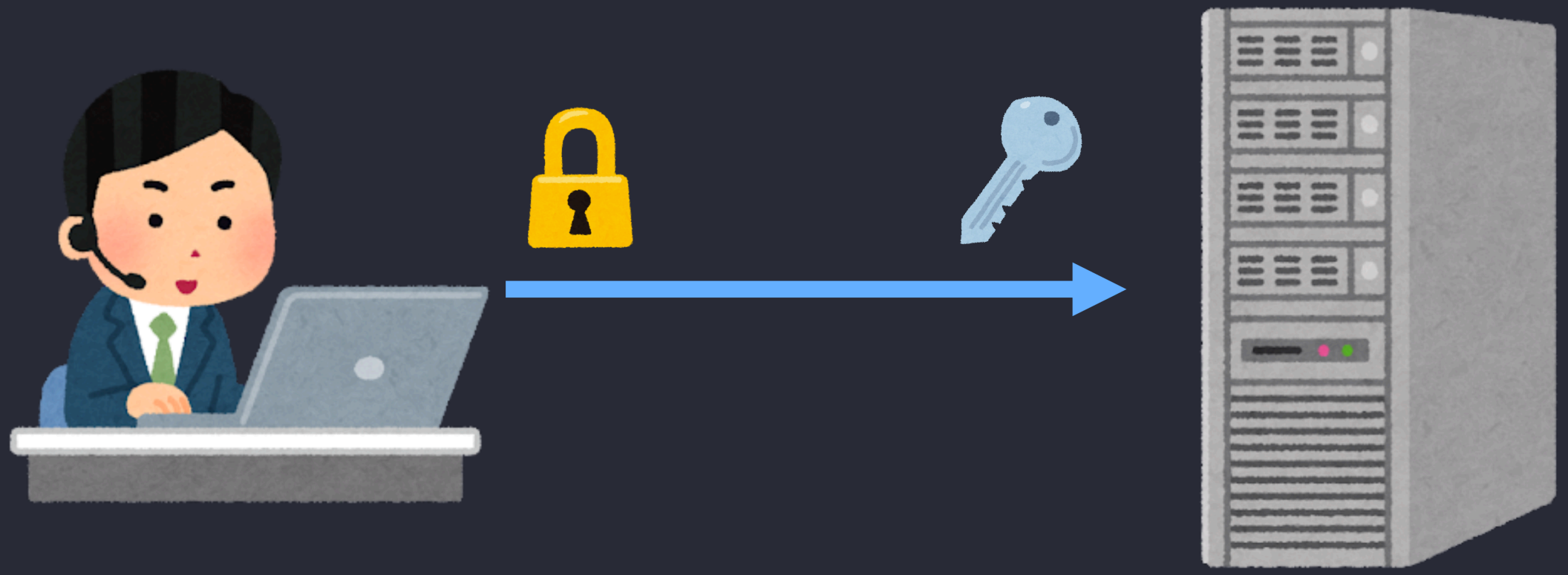
2. セッションキーの生成

通信の暗号化をするための鍵を生成する。
ディフィー・ヘルマンの鍵交換を用いて、中間者攻撃に対処



3. サーバによるクライアントの認証

サーバ側が、正しいクライアントに接続しているかを調べる



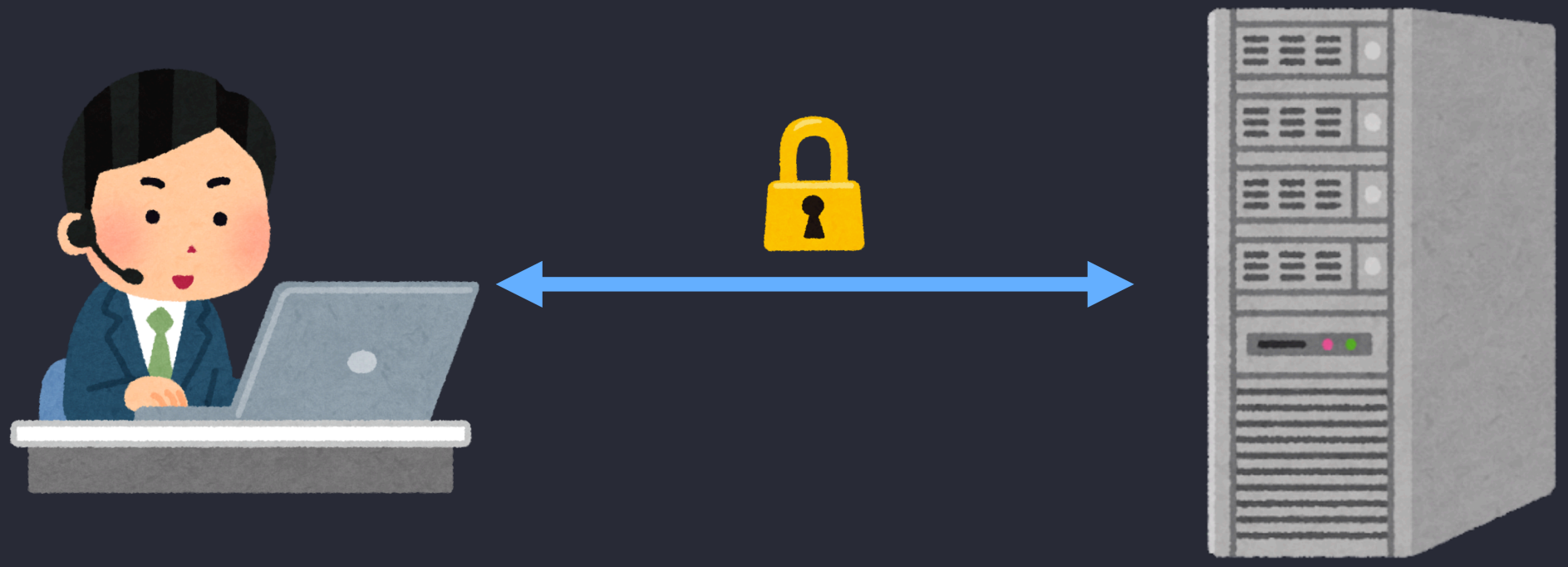
クライアント認証の流れ

1. 鍵ペア（公開鍵、秘密鍵）を生成する
2. サーバ側に公開鍵を渡す
3. サーバがトークン（乱数）を生成しクライアントに送る
4. 自分の秘密鍵を用いてトークンを暗号化し
   暗号化したトークンをサーバに送り返す
5. サーバ側で公開鍵を用いて5を復号し、
   その結果が3と一致すれば「**認証成功**」

c.f. 電子署名

4. データを暗号化して通信

- 2．セッションキーの生成 で生成された鍵を使って暗号化通信をする
- 高速な暗号化、復号が可能な**共通鍵暗号**を使う



SSHのログを見てみよう

`ssh` コマンドに `-v` オプションをつけると接続の際のログを確認できる

```
 Powell  ~
❯ ssh -v __ocw_pub3
OpenSSH_9.0p1, LibreSSL 3.3.6
debug1: Reading configuration data                    /.ssh/config
debug1:                    /.ssh/config line 19: Applying options for __ocw_pub3
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 21: include /etc/ssh/ssh_config.d/* matched no files
debug1: /etc/ssh/ssh_config line 54: Applying options for *
debug1: Authenticator provider $SSH_SK_PROVIDER did not resolve; disabling
debug1: Connecting to ocw.nagoya-u.jp port 22.
debug1: Connection established.
debug1: identity file                    /.ssh/id_rsa type 0
debug1: identity file                    /.ssh/id_rsa-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_9.0
debug1: Remote protocol version 2.0, remote software version OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
debug1: compat_banner: match: OpenSSH_8.2p1 Ubuntu-4ubuntu0.5 pat OpenSSH* compat 0x04000000
debug1: Authenticating to ocw.nagoya-u.jp:22 as '                    '
debug1: load_hostkeys: fopen                    /.ssh/known_hosts2: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts2: No such file or directory
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: curve25519-sha256
debug1: kex: host key algorithm: ssh-ed25519
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: SSH2_MSG_KEX_ECDH_REPLY received
debug1: Server host key: ssh-ed25519 SHA256:0Lp0ifjOhfQVM3EybIfedM1wqY/v8mIkeK6+rumfQzQ
debug1: load_hostkeys: fopen                    /.ssh/known_hosts2: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts2: No such file or directory
debug1: Host 'ocw.nagoya-u.jp' is known and matches the ED25519 host key.
debug1: Found key in                    /.ssh/known_hosts:13
debug1: rekey out after 134217728 blocks
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: rekey in after 134217728 blocks
debug1: get_agent_identities: bound agent to hostkey
debug1: get_agent_identities: ssh_fetch_identitylist: agent contains no identities
debug1: Will attempt key:                    /.ssh/id_rsa RSA SHA256:TkimB1rfsbEwWQSBe4e3Iczs2o78DyD3IsCq6Duj4co explicit
debug1: SSH2_MSG_EXT_INFO received
debug1: kex_input_ext_info: server-sig-algs=<ssh-ed25519,sk-ssh-ed25519@openssh.com,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,sk-ecdsa-sha2-nistp256@openssh.com>
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Offering public key:                    /.ssh/id_rsa RSA SHA256:TkimB1rfsbEwWQSBe4e3Iczs2o78DyD3IsCq6Duj4co explicit
debug1: Server accepts key:                    /.ssh/id_rsa RSA SHA256:TkimB1rfsbEwWQSBe4e3Iczs2o78DyD3IsCq6Duj4co explicit
Authenticated to ocw.nagoya-u.jp ([133.6.120.68]:22) using "publickey".
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: pledge: filesystem
debug1: client_input_global_request: rtype hostkeys-00@openssh.com want_reply 0
debug1: client_input_hostkeys: searching                    /.ssh/known_hosts for ocw.nagoya-u.jp / (none)
debug1: client_input_hostkeys: searching                    /.ssh/known_hosts2 for ocw.nagoya-u.jp / (none)
debug1: client_input_hostkeys: hostkeys file                    /.ssh/known_hosts2 does not exist
debug1: client_input_hostkeys: host key found matching a different name/address, skipping UserKnownHostsFile update
debug1: Remote:                    /.ssh/authorized_keys:1: key options: agent-forwarding port-forwarding pty user-rc x11-forwarding
debug1: Remote:                    /.ssh/authorized_keys:1: key options: agent-forwarding port-forwarding pty user-rc x11-forwarding
debug1: Sending environment.
debug1: channel 0: setting env LC_TERMINAL_VERSION = "3.4.19"
debug1: channel 0: setting env LANG = "ja_JP.UTF-8"
debug1: channel 0: setting env LC_TERMINAL = "iTerm2"
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-132-generic x86_64)
```

（おまけ）RSAとECDSA

- RSA
  - 「素因数分解の困難性」ベース
  - 公開鍵暗号の最初の実装
- ECDSA
  - 「楕円曲線上での離散対数問題」ベース
  - 比較的最近開発された方法

同じ長さの鍵では、ecdsaの方が強度が高い
→ 最近はecdsaが推奨されています！

まとめ

- SSHは公開鍵暗号、共通鍵暗号、電子署名、鍵交換アルゴリズムといった暗号技術をフルに使ったすごい技術！
- サーバにSSH接続をするときは、より安全な公開鍵認証を使おう！

参考

- Secure Shell (Wikipedia)
  https://ja.wikipedia.org/wiki/Secure_Shell
- OpenSSH (公式)
  https://www.openssh.com/
- Understanding SSH Workflow
  https://medium.com/@hellomudit/understanding-ssh-workflow-66a0e8d4bf65