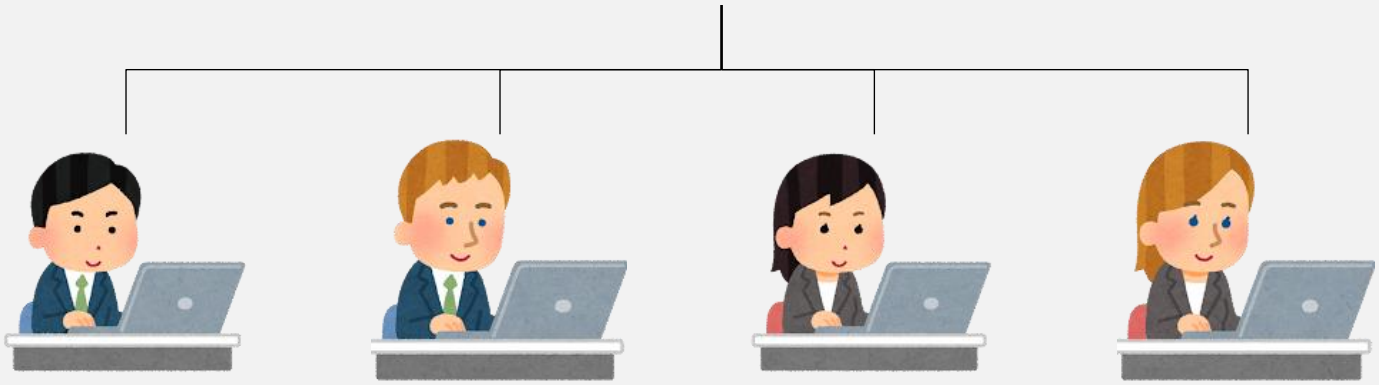
SSH公開鍵認証による接続

慶應義塾大学理工学部物理情報工学科
渡辺

端末とは

もともとは大型コンピュータ(ホスト)に接続され、利用者が命令を送るための
インタフェースのこと

ホストコンピュータ



端末(ユーザーインタフェース)

ホストコンピュータは高価であり、複数人で共有して利用するために
端末が必要だった

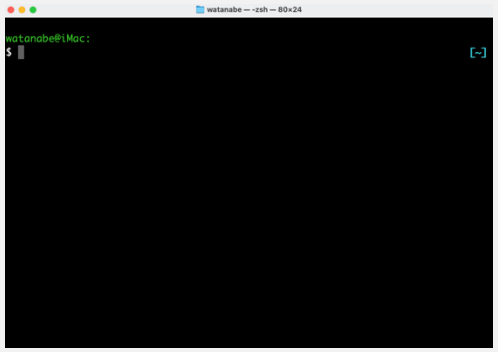
端末エミュレータ

- 昔はタイプライターが端末であった(テレタイプ端末, TTYの語源)
- その後ディスプレイを使ったビデオ端末が出現
- ホストコンピュータにはビデオ端末が付属していたが、DECのVT100が端末のスタンダードに

ホスト		端末
IBM System/360	⟷	IBM 2260
DEC PDP-11	⟷	VT100

VT100

ClickRick – CC-BY-SA 3.0
https://commons.wikimedia.org/w/index.php?curid=6693684



Macの「ターミナル」やWSLのUbuntuの画面は、「端末エミュレータ」と呼ばれ、VT100の動作をエミュレートするものがほとんど

リモート接続

パーソナルコンピュータから、別の強力なコンピュータに遠隔から接続したい
→リモート接続

自宅や居室

大学やサーバ室にあるサーバ

手元のPCの端末から、遠隔にあるマシンにリモートログインして作業
リモートにあるサーバに直接ログインしているかのよう作業できる

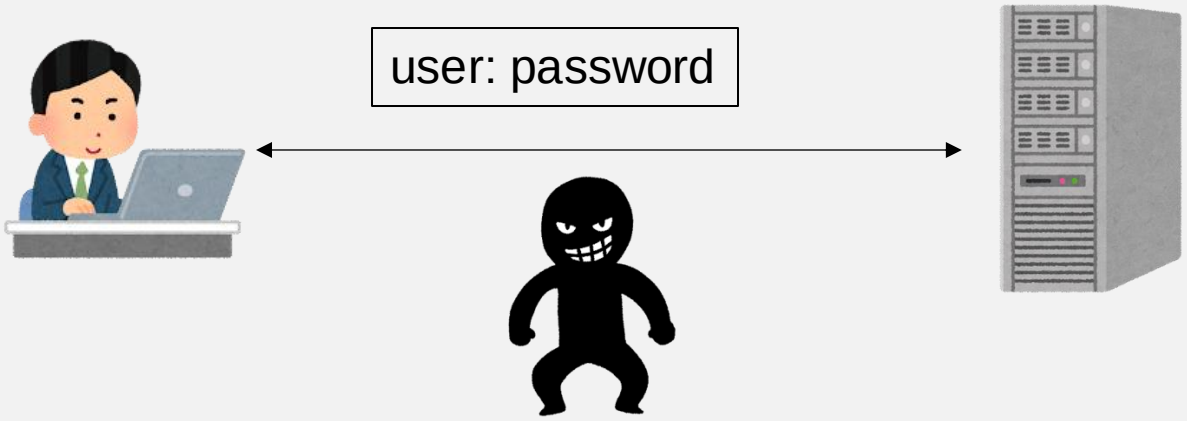
リモートログインに使われたのがtelnetやrlogin

リモート接続とセキュリティ

telnetやrloginは**通信を平文で送受信**する

自宅や居室

大学やサーバ室にあるサーバ



インターネットの通信は容易に傍受可能
通信路を傍受すると、接続先、ユーザ名、パスワードなどが全て見えてしまう

セキュアな通信手段が欲しい ➡ SSH

SSHとは

SSH (Secure Shell)

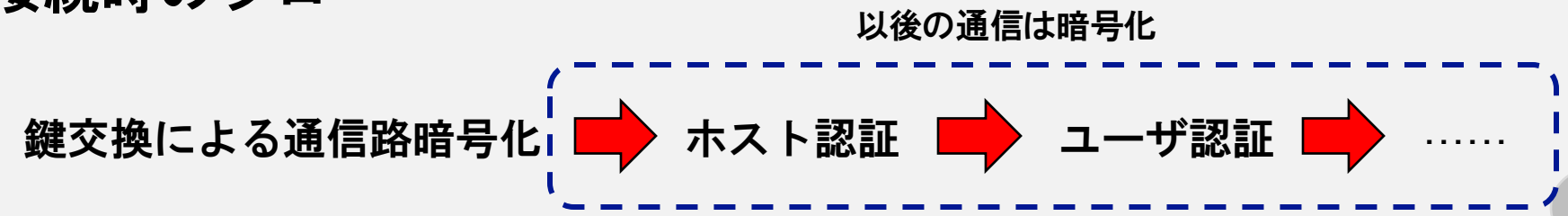
telnetやrlogin、rsh、ftpなどを代替するために生まれた
SSHは規格であり、その実装の一つがOpenSSH

```
$ ssh -V
OpenSSH_9.7p1, LibreSSL 3.3.6
```

SSHの役割

- 通信路の暗号化
  - 全ての通信を傍受されても、盗聴者が内容を復元できないようにする
- 認証(ホスト認証、ユーザ認証)
  - ホスト認証：接続しようとしているホストが正しいことを確認
  - ユーザ認証：接続しようとしているユーザが正しいことを確認

接続時のフロー



鍵交換

通信路が全て傍受されている前提で、秘密の情報を共有したい

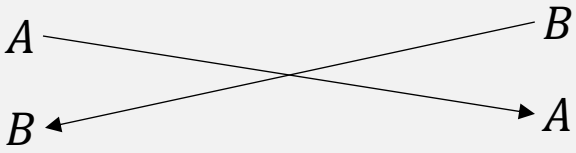
Alice

Bob

a, A

b, B

AliceとBobが鍵のペア(a, A)と(b, B)を生成

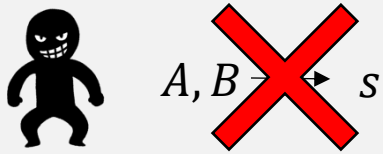


公開鍵A, Bを互いに伝える(傍受可能)

$s = aB$

$s = Ab$

$s = aB = Ab$を計算し、共通の秘密情報とする



傍受者は通信傍受で得たA, Bから秘密情報sを再現できない

以上のアルゴリズムをDiffie–Hellman (DH)鍵交換と呼ぶ

鍵交換

離散対数問題を使った実装例

gとpは既知とする

Alice: 秘密の整数aに対して$A = g^a \bmod p$を計算し、公開鍵とする

Bob: 秘密の整数bに対して$B = g^b \bmod p$を計算し、公開鍵とする

AliceはBobから受け取ったBを使って以下を計算する

$$s = B^a \bmod p = g^{ab} \bmod p$$

BobはAliceから受け取ったA使って以下を計算する

$$s = A^b \bmod p = g^{ab} \bmod p$$

傍受者はA, Bの情報からsを計算することができない(離散対数問題)
→AliceとBobは、傍受されている通信路を使って秘密の情報sを共有できた

楕円曲線上の演算を用いる Elliptic-curve Diffie-Hellman (ECDH)もよく使われている

認証

鍵交換により通信路が暗号化されたとしても、通信相手が正しい相手であるかどうかはわからない

➡ 公開鍵による認証

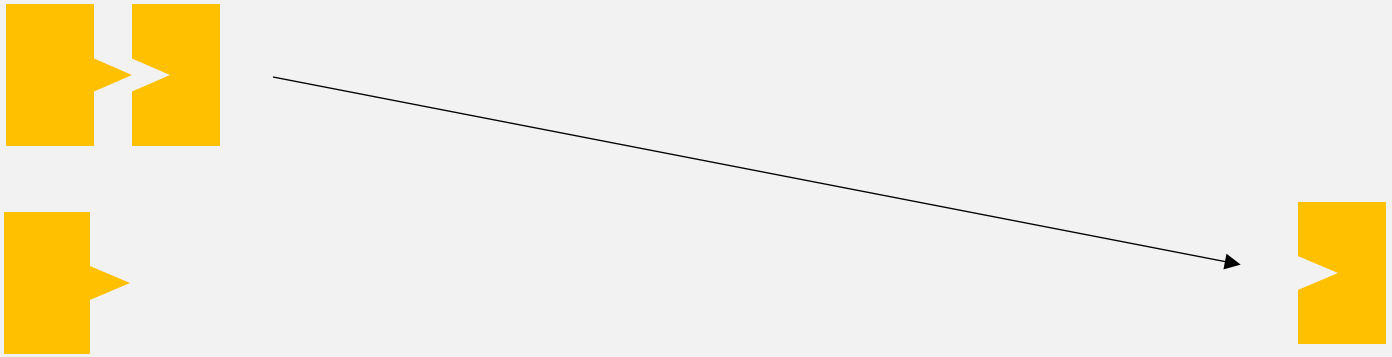
公開鍵認証

Aliceが通信を通じてBobに「いま通信しているのは自分である」と証明したい

Alice

Bob

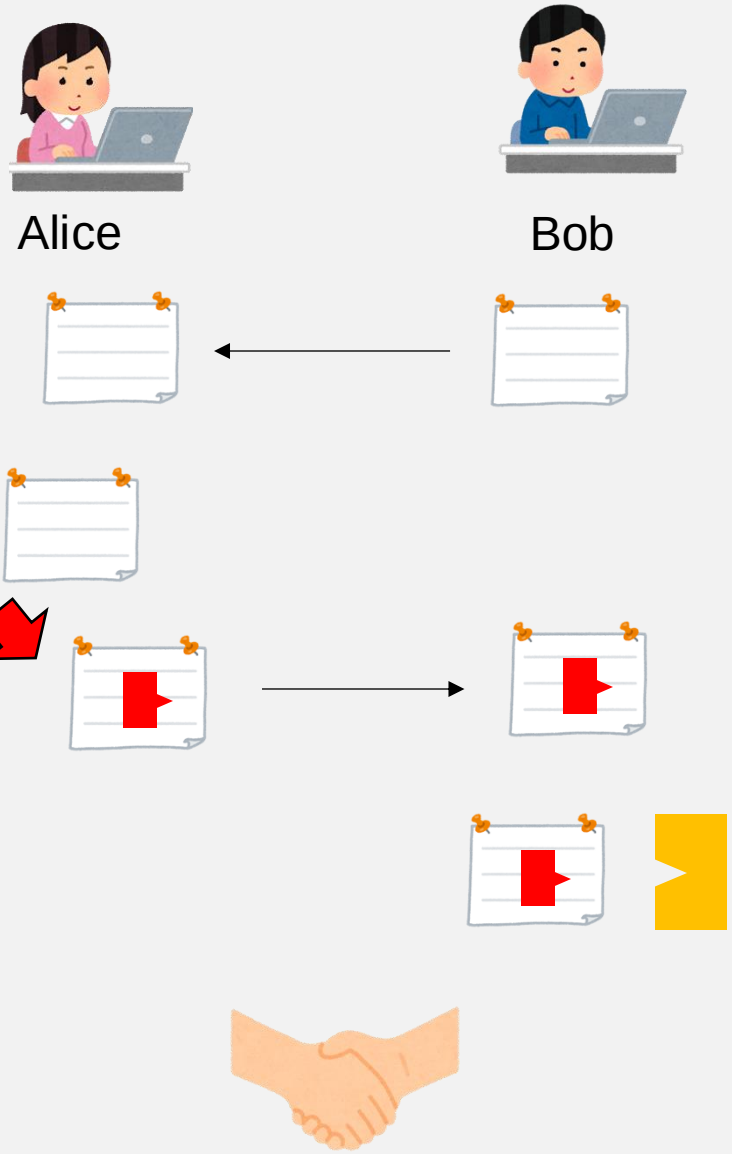
あらかじめAliceは鍵のペアを作っておき、公開鍵をBobに渡しておく



秘密鍵(署名鍵)

公開鍵(検証鍵)

公開鍵認証



1. BobはAliceにメッセージを送る

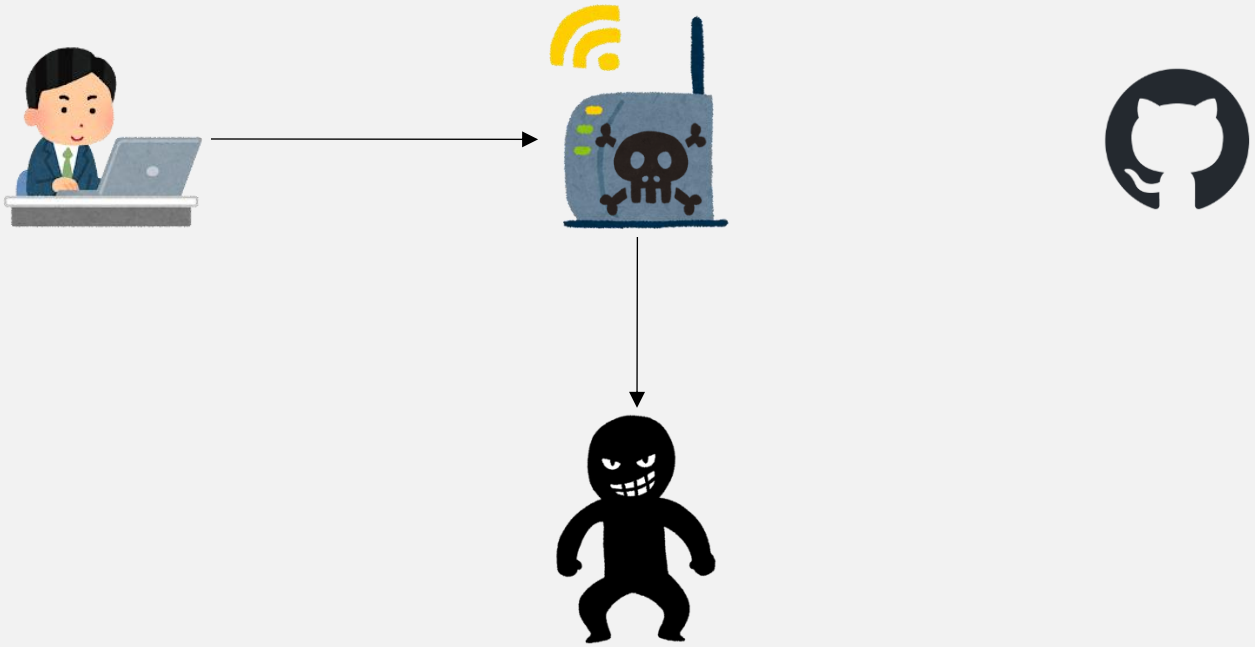
2. Aliceはメッセージに秘密鍵で
署名をしてBobに送り返す

3. BobはAliceの公開鍵で署名を検証する

4. 検証に成功したら認証完了

ホスト認証

インターネットは情報がリレーされるので、途中に悪意ある中継点があると通信路が書き換えられるおそれがある



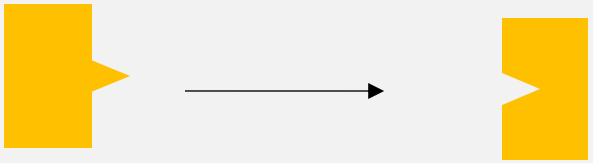
通信相手が正しいホストであると確認する→ホスト認証
ユーザ認証の前に行われる

ホスト認証(初回)

初めて接続するホストで、クライアントが公開鍵を持っていない場合

ホスト

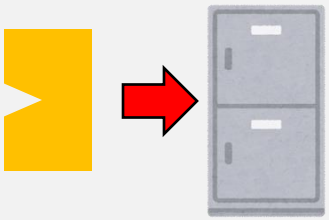
クライアント



1. ホストから公開鍵が送られてくる

2. この公開鍵を信じるか確認する

```
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```



3. 鍵を信じる場合(yes)、この鍵を「知っているホスト」に登録する

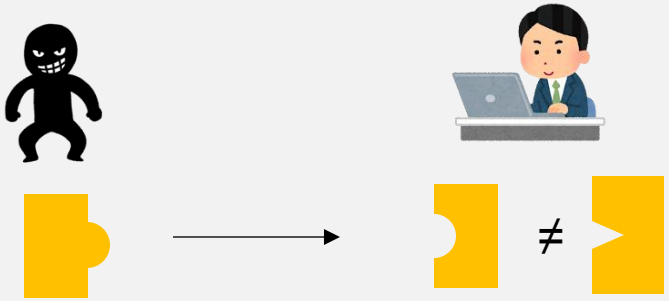
.ssh/known_hosts

ホスト認証(二回目以降)

ホスト

クライアント

ホストが秘密鍵で署名したメッセージを
すでに所持している公開鍵で検証



ホストがなりすましである場合、異なる
公開鍵を送ってくるのでわかる

公開鍵を偽装しても、秘密鍵を持っていな
いので公開鍵に対応した署名ができない

公開鍵認証のまとめ

ユーザ認証

- 公開鍵認証には、署名に使う秘密鍵と、検証に使う公開鍵のペアを使う
- 検証側は、あらかじめ相手の公開鍵を保持しておく
- 公開鍵で検証できる形で署名が可能なのは、秘密鍵の所持者のみ
- 正しく署名できた人を「公開鍵に対応する秘密鍵の所持者である」と認証できる

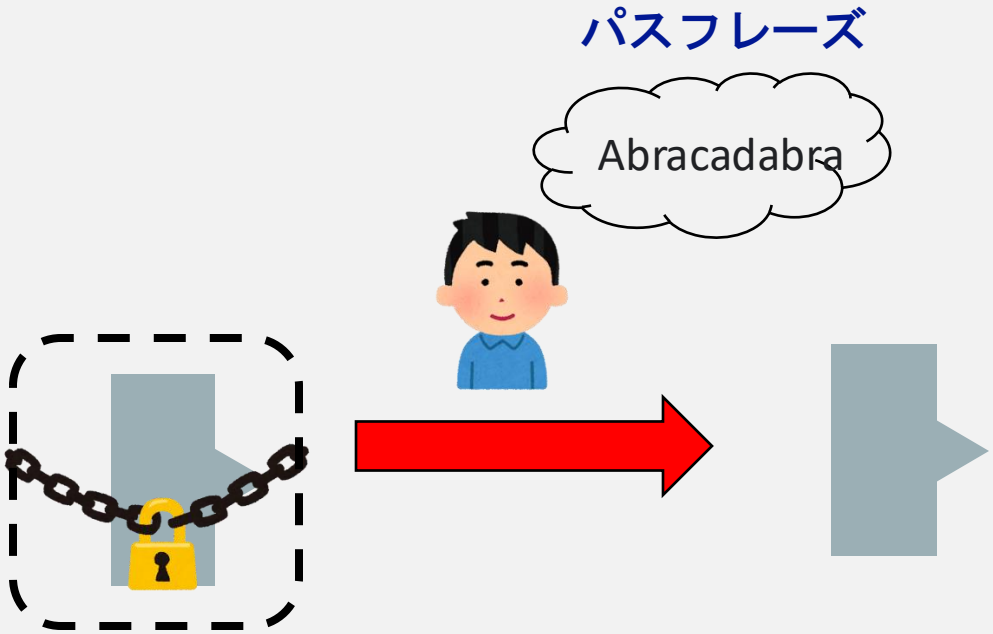
ホスト認証

- ホスト認証でも、あらかじめ相手の公開鍵を登録しておくことが望ましい
- しかし、初回接続時に公開鍵のフィンガープリントを確認し、問題なければ登録する、という運用が多い

例：GitHubのSSHフィンガープリント https://docs.github.com/ja/authentication/keeping-your-account-and-data-secure/githubs-ssh-key-fingerprints

秘密鍵の暗号化

秘密鍵は電子ファイルなので、容易にコピーされてしまう
→ 暗号化により守る

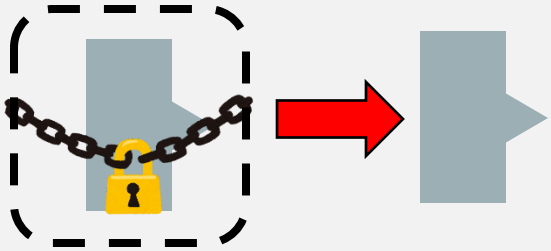


もし秘密鍵が流出したとしても、暗号化に用いたパスフレーズがわからなければ秘密鍵が使えない(**知識認証**)

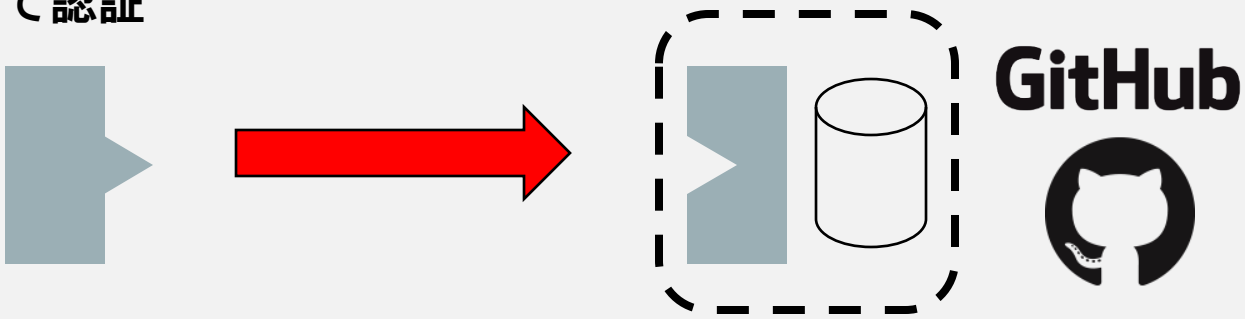
秘密鍵の暗号化

(1) ターミナルからGitHubにアクセスしようとする

(2) パスフレーズにより秘密鍵を復号



(3) 秘密鍵を使って認証

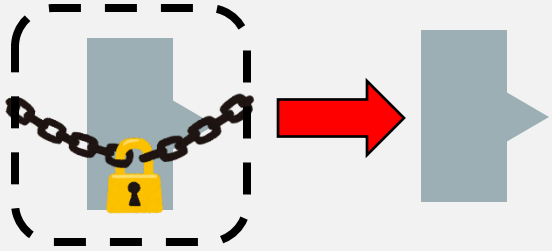


パスフレーズによる知識認証でGitHubにアクセスしている
わけはないことに注意

SSHエージェント

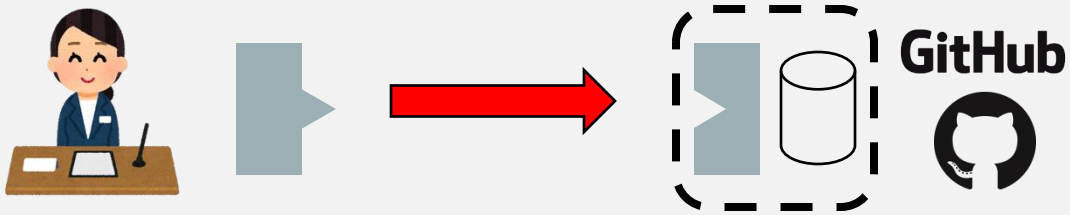
接続のたびにパスフレーズを入力するのは面倒
→ SSHエージェント

(1) **パスフレーズにより秘密鍵を復号**



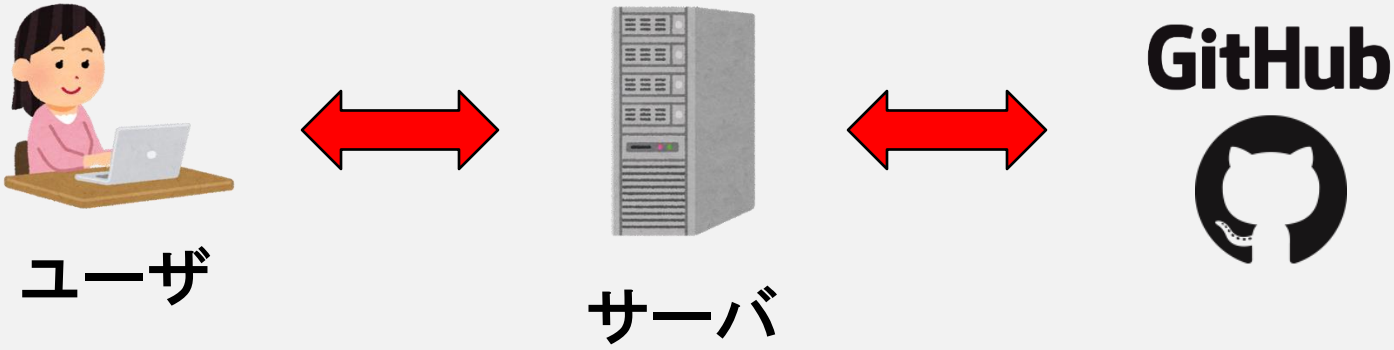
(2) **復号した秘密鍵をSSHエージェントに記憶してもらう**

(3) **次回接続から記憶した秘密鍵を使う(パスフレーズ入力を省略)**



多段SSH

まずサーバにログインし、そのサーバからGitHubにアクセスしたい



GitHubから見たらサーバが接続元なので、そこに秘密鍵が欲しい
しかし、サーバに秘密鍵を置きたくない→SSHエージェント転送

SSHエージェント転送

ユーザ

サーバ

GitHub

ユーザPCの
エージェント

署名リクエスト

サーバの
エージェント

署名リクエスト

署名応答

署名応答

認証成功

SSHエージェントが署名情報を転送することで、GitHubは接続したクライアントが登録済みの公開鍵の所持者であるとわかる

SSHエージェントのまとめ

SSHエージェントの役割は以下の2つ

1. **復号済みの秘密鍵を記憶する**
2. **別の接続先に署名を転送する**

SSHエージェントを使うためには、SSHエージェントが起動し、常駐している必要がある

MacはKeyChainがSSHエージェントを兼ねる
Windows (WSL)は別途SSHエージェントを起動する必要がある

SSHエージェントの使い方

復号した秘密鍵の記憶には ssh-add
SSHエージェント転送をする場合は ssh –Aオプション
ログアウトしたら秘密鍵の情報は消える