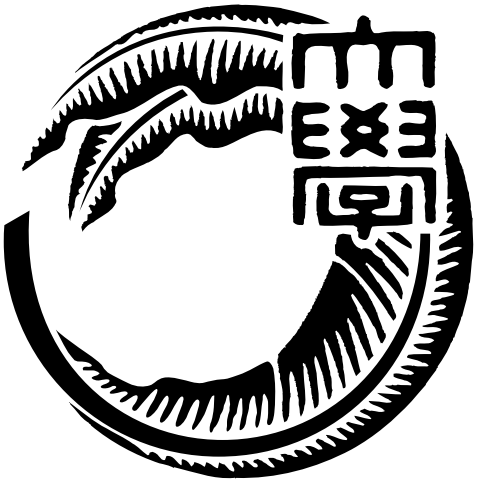
令和元年度 卒業論文

公開鍵暗号方式を用いたSSH認証による アカウント作成の簡略化

Simplify Creating Account by SSH Authentication using Public-key Cryptography



琉球大学工学部情報工学科

165714D 与那嶺東

指導教員 長田智和,谷口祐治

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　背景と目的

インターネット普及期には, ユーザーは限られているため, 性善説の発想で運用されており, できる限り制限なく自由に接続することを目指していた. しかし利便性が増すとともに, コンピュータウイルスによる被害, 企業情報や個人情報の漏洩, ネットワークを介した詐欺事件などのトラブルが増加してきた. そのような理由から, 現在では,「単に接続する」ということを超えて,「安全に接続する」ことが強く求められている [1]. 安全に接続するというセキュリティにおける基本に,IT システムが誰もしくは何であるかを正しく特定するために認証技術 [2] がある. 指紋認証や, 顔認証など, 認証技術にはユーザビリティの向上がこの頃顕著である. そのなかで, インターネットでの個人認証方式として,ID とパスワードを用いた認証が未だ一般的である [3]. 簡単なパスワードだと, アカウントの乗っ取りに繋がることがある. そのことから, 一般的にアカウントを作成する際, パスワードを強固にするために 8 ケタ以上 [4] で, 大小英数字の組み合わせを行わないと, アカウントが作成できないなどの制限がある. しかしながら, それだとパスワードの作成や管理の面倒さなどから, 覚えやすいパスワード (身近に関連したもの) の作成や, 複数の ID でのパスワードの使い回しが生じるきっかけになりうる. また, アカウント作成の面倒さから, 新規顧客開拓の損失につながる可能性がある. 上記の背景から, アカウント管理の面倒さを軽減することを目的とし, パスワード作成管理をなくすことを目標とする. そのための手段として, 公開鍵暗号方式を用いた SSH 認証をパスワードに代替できるものとして, 提案する.

1.2　論文の構成

本論文は, 以下の通りに構成されている.

第 1 章 はじめに

本研究の背景と目的について述べる.

第 2 章 基礎概念

認証と提案手法に関することについて述べる.

第 3 章 提案手法

WEB サービス登録・認証の実装に関することについて述べる.

第 4 章 検証

検証と考察について述べる.

第 5 章 今後の課題

提案手法と検証に関する今後の課題を述べる.

第2章　基礎概念

2.1　認証技術

「認証」とは、人やモノ、情報の正当性を証明することであり, 人やモノを対象に した認証を「主体認証」[5] という. 主体認証には,「主体の知識による認証」「主体の所有する者による認証」「主体の身体的な特性による認証」の 3 種類がある. 主体の知識による認証による代表例は, パスワード認証, パスフレーズ認証, および,PIN(Personal Identification Number) を用いたものである. これらは実装が安易な認証技術であり, 多くの IT システムで採用されている [6]. 所有するものによる認証では, 携行可能な物理的デバイスを利用する. 物理的デバイスとしては,IC カードや, ワンタイムパスワードでも利用されるようなハードウェアトークンなどがある [6]. 身体的な特性による認証には, 利用者の指紋, 色彩や生脈パターンなどを用いたものがある. これらをまとめて, バイオメトリクス認証 (生体認証) と呼ぶ [6].

2.2　公開鍵暗号方式

公開鍵暗号方式は, 暗号化鍵と複合鍵を別のものとする RSA のような暗号化方式 [7] である. 秘密鍵から公開鍵を求めることはできるが, 公開鍵から秘密鍵を求めることは困難 [7] である.

2.3　公開鍵暗号方式を用いた SSH 認証

秘密鍵はクライアントが持ち, 公開鍵はサーバ側が持つ. 公開鍵認証による SSH の流れを次に記述する. ①クライアントがサーバに ssh 認証をする. ②サーバは公開鍵を用いて, ランダムな値の暗号化を行い, クライアントに送信する. ③クライアントは, 送信された暗号化の値を解読し, サーバに送

る. ④サーバは,②で行った暗号化される前の値と,③で送られてきた,解読された値が一致しているか確認する.一致していたら認証成功にし,不一致なら認証失敗にする.

2.4 提案手法の実装で用いた技術

2.4.1 Ruby on Rails[8]

Ruby on Rails は Ruby 言語で記述された,Web 開発フレームワークである. Ruby のライブラリを補助してくれる gem があり,効率よく開発を進めることができる.

2.4.2 javascript[9]

javascript はクライアントサイドである,ブラウザで実行することができる非同期処理なプログラミング言語である. DOM(Document Object Model) を用いることにより,HTML と CSS を操作することができる.

2.4.3 node.js[10]

node.js はサーバサイトで javascript を実行できるプラットホームである.

ssh2(モジュール)[11]

ssh2 は javascript で記述された nodede.js 用モジュールである. node.js のコードで,OpenSSH(version 7.6) を用いることができる.

socket.io(モジュール)[12]

socket.io は,通常 WebSocket プロトコルを用いてクライアント,サーバ間でリアルタイム双方向通信を可能にするモジュールである. 現状次のプログラミング言語で用いることができる.「Node.js」「JavaScript」「Java」「C++」「Swift」「Dart」

2.4.4 OpenSSH[13]

OpenSSH は世界でもっとも使用されている SSH(安全に通信するためのプロトコル) 実装であり, オープンソースである. 主な認証方式としては,「パスワード認証」と「公開鍵認証」がある.

2.4.5 HTTP cookie[14]

HTTP cookie(Cookie) はウェブサーバとウェブブラウザ間で状態を管理する通信プロトコルである. Cookie により, ブラウザに小さなテキストデータを保存することができる. 本来ステートレスなプロトコルである HTTP に対して,Cookie で状態を管理することによりステートフルなサービスを実現することが可能である.

ステートレス [15]

ステートレスは,HTTP リクエストに対する HTTP レスポンスが変わらない.

ステートフル [15]

ステートフルは, セッションの状態を保持することにより, HTTP リクエストに対する HTTP レスポンスが変わる.

2.4.6 セッション [16]

セッションにより, コンピュータ間で半永久的な接続を設定できる. セッションを実装する手段として,Cookie を用いることが一般的である.

第3章　提案手法

3.1　公開鍵暗号暗号方式による SSH を用いた登録・認証

3.2　パスワード方式の登録・認証の実装

ここでは, パスワード方式の登録・認証は検証で比較対象として用いる. Railsチュートリアルのサイト [17] を参考に最低限の作成をした. 参考にした, 章・節・項を以下に列挙する.

- 第6章 ユーザモデルの作成
  - 6.1 User モデル
    - 6.1.1
    - 6.1.2
- 第7章 ユーザ登録
  - 7.1 ユーザを表示する
    - 7.1.2
  - 7.2 ユーザ登録フォーム
    - 7.2.1
    - 7.2.2
  - 7.3 ユーザ登録失敗
    - 7.3.1
    - 7.3.2
  - 7. 4ユーザ登録成功
    - 7.4.1

        * 7.4.2
- 第8章 ログイン, ログアウト
    - 8.1 セッション
        * 8.1.1
        * 8.1.2
        * 8.1.3
        * 8.1.4
    - 8.2 ログイン
        * 8.2.1
        * 8.2.2
        * 8.2.3
        * 8.2.5
- 第9章 発展的なログイン機構
    - 9.2 認可
        * 9.2.1
        * 9.2.2
-

3.3 ボツ案

3.3.1 ログイン状態のセッション橋渡し

実装したいこと
公開鍵暗号方式による ssh 認証を成功した際に，WEB サービスでログイン状態になる．ここでいう，ログイン状態は，セッションを用いて，ステートフルな通信をする (HTTP 通信はステートレスな通信).

実装するための手段案

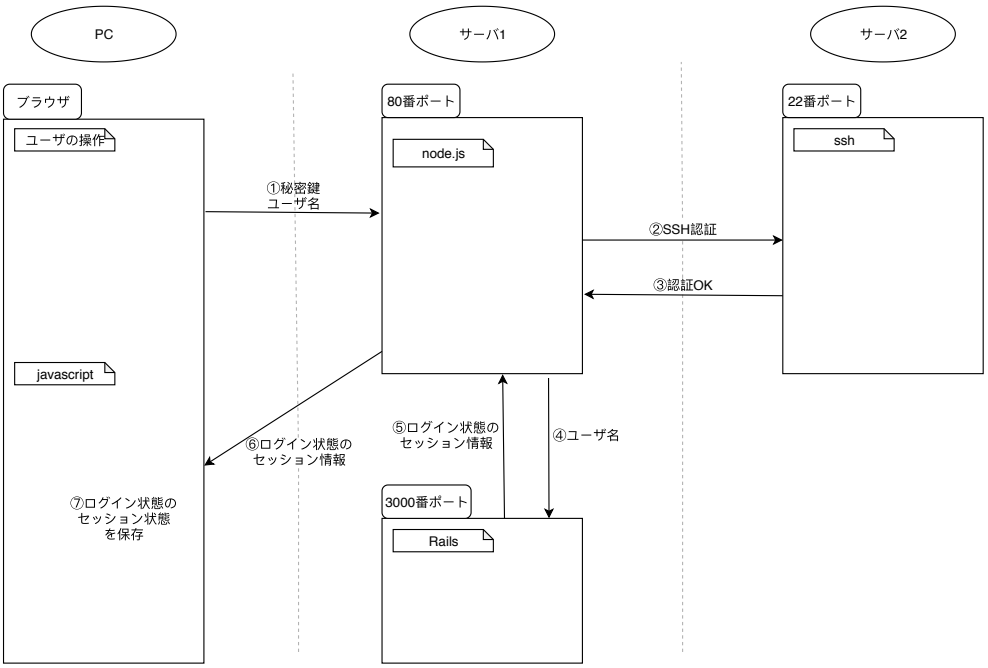


図 3.1: ログイン状態のセッション橋渡し案

ログイン状態のセッション情報を橋渡しすることにより，実現しようと考案した。以下の図 3.1 を用いながら説明する。
図 3.1 の① 〜 ③では，エンドユーザはブラウザを用いて，公開鍵暗号方式による ssh をしている。図 3.1 の④では，HTTP リクエストの POST をしている。図 3.1 の⑤では，HTTP レスポンスが来て，そのヘッダ情報に「セッション情報を Cookie に保存する」という情報が付与されている (のちのち, アクセス制限で,localhost からしかアクセスできないようにする). 図 3.1 の⑥では，図 3.1 の⑤の情報を, ブラウザの javascript 側に socket 通信で渡す. 最後に，図 3.1 の⑦で，ログイン状態のセッション情報を Cookie に保存する.

ブラウザを用いての検証
ログイン中のセッション情報を, ブラウザの Cookie に保存することで，ログインすることが可能かを検証する。
検証環境
　機器

MacBook Pro (Retina, 13-inch, Early 2015)

macOS High Sierra(バージョン 10.13.6)

サーバ側

Ruby on Rails(6.0.2.1)

localhost:3000

ユーザ側

2っのブラウザ

Google Chrome

Google Chrome Canary

Crome の開発環境 (デベロッパーツール) を用いて以下の検証を行った。

まず，ログインした際の動きとして，HTTP レスポンスのヘッダ情報に,set-cookie がある.Crome のデベロッパーツールでは，以下の図 3.2 と表示される.

図 3.2: ログイン成功した際の,HTTP レスポンスの抜粋

次に，図 3.2 の HTTP レスポンスを元に，ブラウザの Cookie にセッション情報を保存する。Crome のデベロッパーツールでは，以下の図 3.3 と表示される.

上記の 3.2,3.3 は,HTTP レスポンスのヘッダ情報で, ブラウザの Cookie に

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
_yes_password_session	TfiF9FGOjreVMXhw%2F366KAeIMzjd…	localhost	/	Session	415	✓		

TfiF9FGOjreVMXhw%2F366KAeIMzjdpBCBswAGpn12rrInXsgipqR0R4xMCF2R2tz2n%2FTqMvgAo568BLMYfc9Z2YpJFB%2BcmpXhAqPd6sUuTtmp5Nbna4FOykeF3itx4pw%
-yUt1PA8ewS%2BWMmd6--0j%2BA%2B45WSiVByarwRS%2F5nw%3D%3D

図 3.3: ブラウザの Cookie のセッション情報

値を保存している. その後,HTTP リクエストで,Cookie 情報を付与 [18] することにより, ステートレスなプロトコルである HTTP 上で，状態管理ができる [19](ログイン状態の維持).

現状で以下の図 3.4 のように，ログインしているブラウザ，ログインしていないブラウザがある.

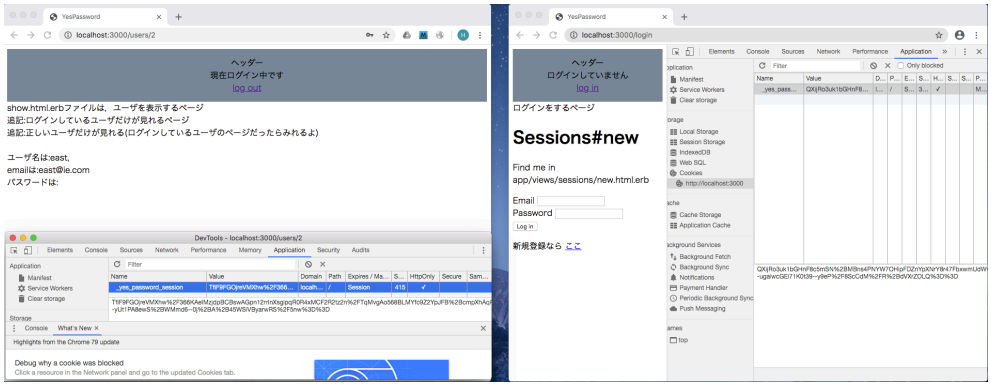


図 3.4: 2っのブラウザ (ログイン状態, ログインしていない状態)

上記の図 3.4 の状態から次の操作を行う. デベロッパーツールを用いて，ログインしているブラウザから，ログインしていないブラウザに，Cookie 情報の, コピーアンドペーストを行い. リロードする. その際の，状態が以下の図 3.5 になり, ログイン状態のセッションを橋渡し (Cookie に保存) することにより, ログインすることができることがわかる.

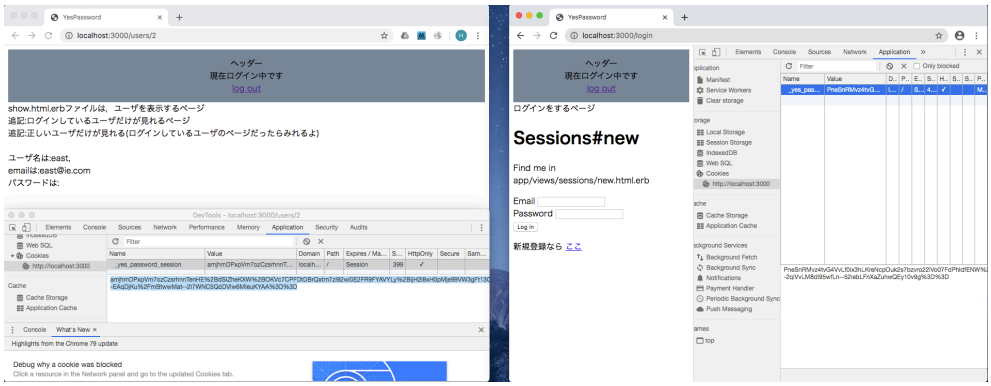


図 3.5: 2っのブラウザ (ログイン状態, ログイン状態)

図 3.5 の補足.
HTTP リクエストをする際に, 新しい Cookie 情報が付与されることが確認できた. そのことにより, ログイン中の 2っのブラウザが別のセッションになっている.

実際に実装
実際に実装を進めて，図 3.1 の① ～ ⑥までできた. しかしながら，図 3.1 の⑦ができなかった. その理由を以下の図 3.6 を元に説明する.

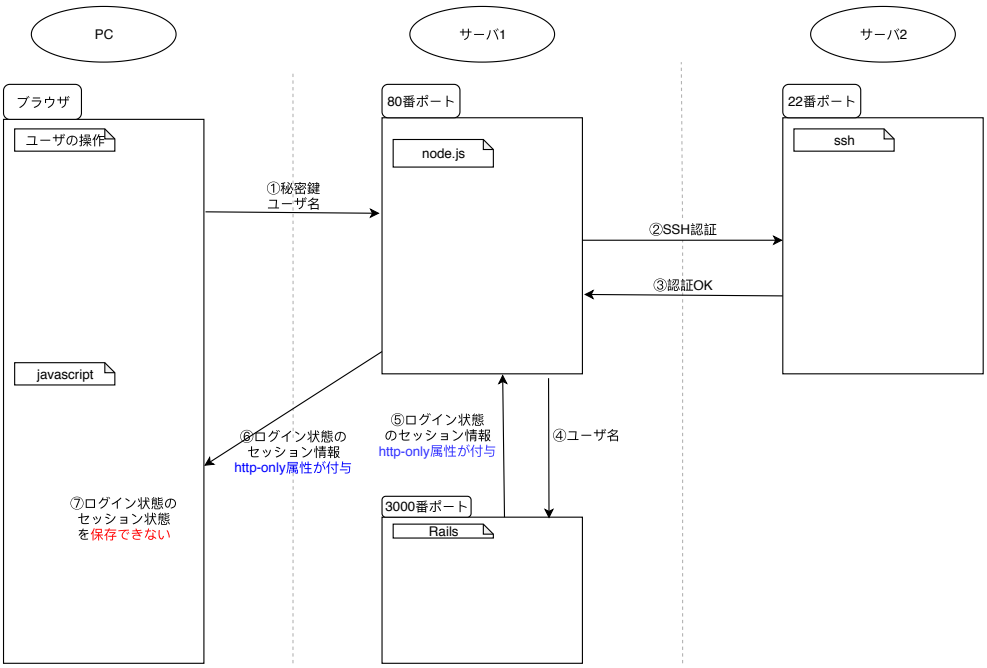


図 3.6: ログイン状態のセッション橋渡し案がボツになった

図 3.6 の⑤ で,rails ではセッションを発行する際に,http-only 属性を付与している [20]. そのことは，ログイン時の HTTP レスポンスである図 3.2 でも現れている．http-only 属性がついた Cookie は,javascript で扱うことができなくなり，セッションハイジャックの対策を行っている [21]．そのため, 図 3.6 の⑦のように, セッション状態を保存することができなかったため,「ログイン状態のセッション橋渡し」の案はボツになった.

第4章　検証

4.1　検証1

4.1.1　検証背景

検証目的

password 認証方式の，新規登録・認証の面倒さを解決するために，第3章の提案手法で password 認証方式に変わる，公開鍵暗号方式による ssh 認証を用いた,WEB サービス認証の提案を行った。
しかしながら，提案だけだと，新規登録・認証の面倒さを解決していることの根拠に乏しい。よって，検証を行い，第3章の提案手法は”面倒さの軽減”に効果的に繋がっているかの確認をする。

検証手段

検証の手段としては，実際に, 以下の2っの認証方式の登録・認証を被験者に体験してもらう。

- バグあり, パスワード方式認証 (以後 ”パスワード認証”と記述する)
- 公開鍵暗号暗号方式による ssh 認証 (以後 ”鍵認証 ”と記述する)

※バグは「認証の際に, どんなパスワードでも認証してしまう」
その後，2っの観点から，”面倒さ”を数値化する。1っ目の観点は「時間」である。”面倒さ”をアカウント登録・認証にかかる時間と推測し，計測化する。詳しい詳細については，検証環境のマニュアルに記述する。2っ目の観点は「アンケート」である。アンケートには，点数で答える方式, 文字で記入する欄 の2っがあり，点数で答える方式により”面倒さ ”を数値化する。アンケートの細かい内容は，4.1.3 の検証画面に記述する。

また，アンケートには”面倒さ”を数値化する以外にも，以下の2っの意味を込める。1っ目の意味は次の通りである。アカウント登録・認証にかかる時間 を，”面倒さ”と予想して検証しているが，その予想を確かめる必要がある。アンケートを取ることにより,「アカウント登録・認証にかかる時間」と,「アンケートによる面倒さ」が比例していることを確認することで，予想を確かめることができる。また，被験者の状態も確認することで，面倒と感じるのが，検証自体に対しての面倒さと関係があるのかを確認する。2っ目の意味は次のとおりである。記入欄で，改善点や感じたことの意見をもらうことで，今後の研究に生きるようなアンケートをもらう。

4.1.2 検証環境

検証場所

第3章で記述したとおり，検証場所は学科のVMを用いているため，学科のネットワーク内(有線LAN,wifiアクセスポイントie-ryukyu)から，アクセスして検証を行う。

検証の流れ

ここでは,被験者に行ってもらう，検証の流れを記述する。時間の観点で”面倒さ”を数値化する検証では,被験者にはパスワード方式,鍵方式の登録・認証をそれぞれ行ってもらう。その時，被験者は時間を測る。また，再現性を持って，検証を行うためにマニュアルを作成し,マニュアル通りに検証を行う。アンケートの観点で”面倒さ”を数値化する検証では，時間の観点で”面倒さ”を数値化する検証　が終わった直後に行うようにすることで,被験者の思った感情とアンケート結果の差異が少なくなるようにする。

検証画面

ここでは,被験者に行ったもらう検証画面をのせる。

以下の図4.1, 図4.2 図4.3 は,第3章の提案手法で実現したパスワード方式,登録・認証を,実際に被験者に行ってもらった時のブラウザ画面である。図4.1は，アカウント登録画面である。図4.2は，図4.1で登録したアカウ

ントに認証するための画面である。図 4.3 は，図 4.2 で認証成功した後の画面である。

図 4.1: 検証 1_アカウント作成 (パスワード方式)

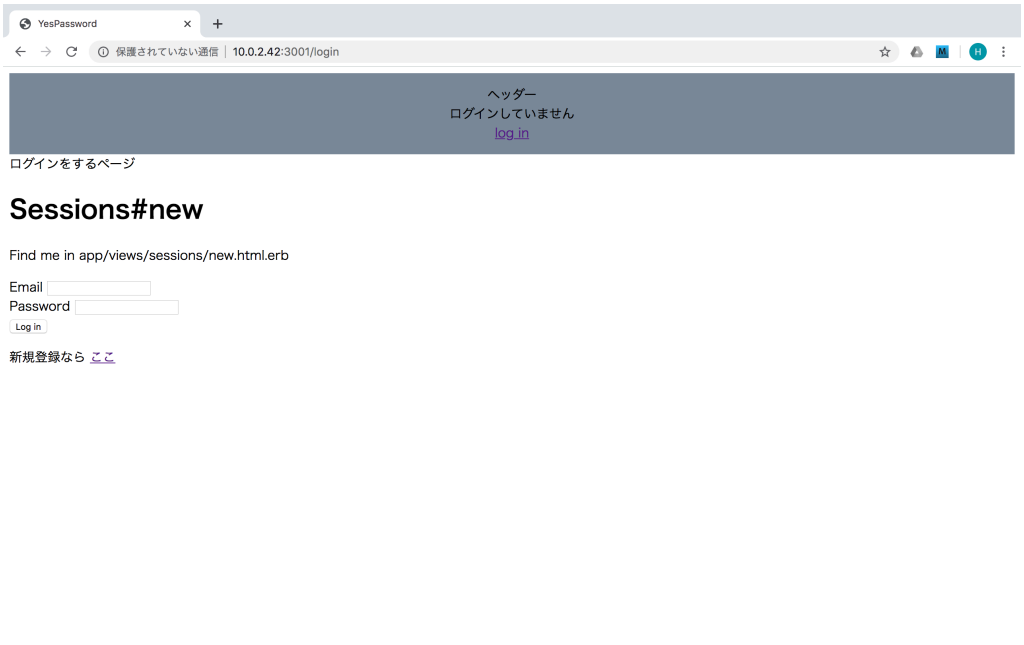


図 4.2: 検証 1_認証 (パスワード方式)

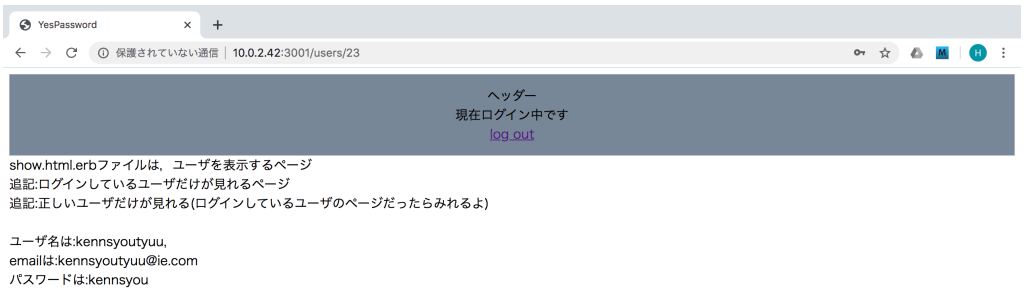


図 4.3: 検証 1_認証成功後 (パスワード方式)

以下の図 4.4, 図 4.5 図 4.6 は, 第 3 章の提案手法で実現した，鍵方式の登録・認証を, 実際に被験者に行ってもらった時のブラウザ画面である。図 4.4 は，アカウント登録画面である。図 4.5 は，図 4.4 で登録したアカウントに認証するための画面である。図 4.6 は，図 4.5 で認証成功した後の画面である。

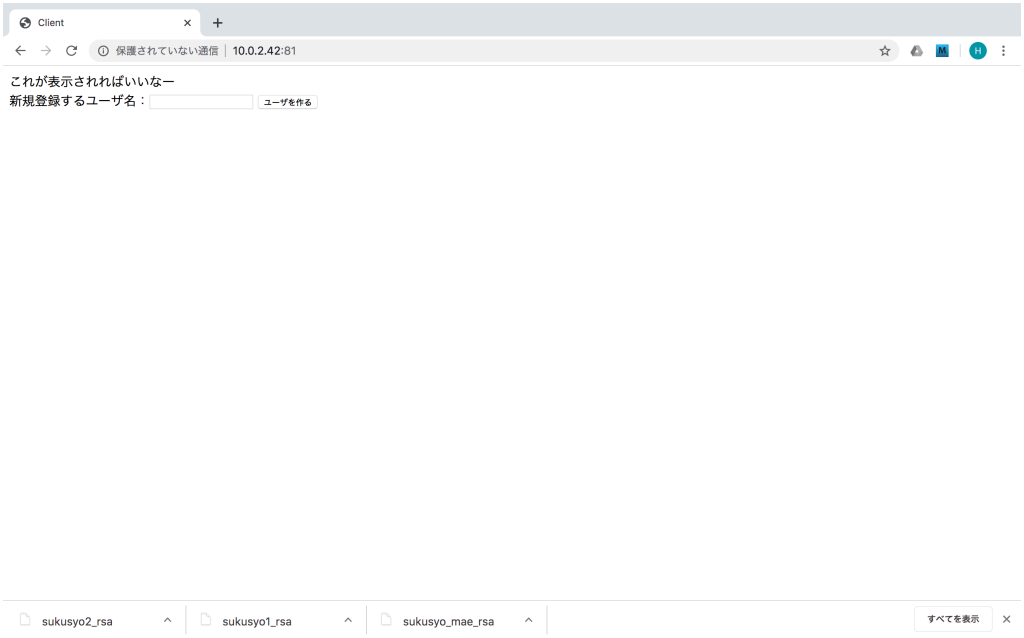


図 4.4: 検証 1_アカウント作成 (鍵方式)

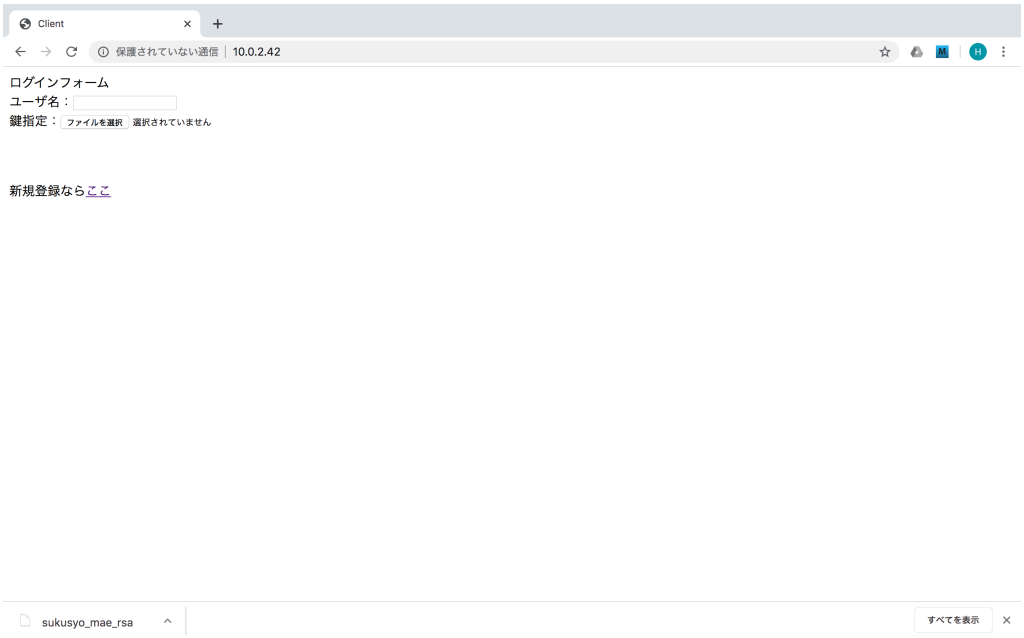


図 4.5: 検証 1_認証 (鍵方式)

図 4.6: 検証 1_認証成功後 (鍵方式)

以下の図 4.7, 図 4.8, 図 4.9, 図 4.10 は図 4.1 〜 図 4.6 までの, 時間計測が終わった直後に行う，アンケートの画面である。図 4.7 では,”検証自体の面倒さ”をアンケートで聞いている。その意図として,”面倒”と感じた感情は, 検証役になること自体が面倒だったことに起因していないかを確かめるためである。図 4.8 では, パスワード方式の登録・認証それぞれについて, どのくらい面倒かを質問している。図 4.9 では, 鍵方式の登録・認証それぞれについて, どのくらい面倒かを質問している。図 4.8, 図 4.9 の質問の意図としては, アンケートの観点から”面倒さ”を数値化することである。図 4.10 では，鍵方式の登録・認証について，被験者の技術視点と利用視点から，自由形式で意見を求めている。図 4.10 の意図としては, よりよい物を作るために, 被験者意見を収集し，今後の方針を決めるためである。

図 4.7: アンケート 1

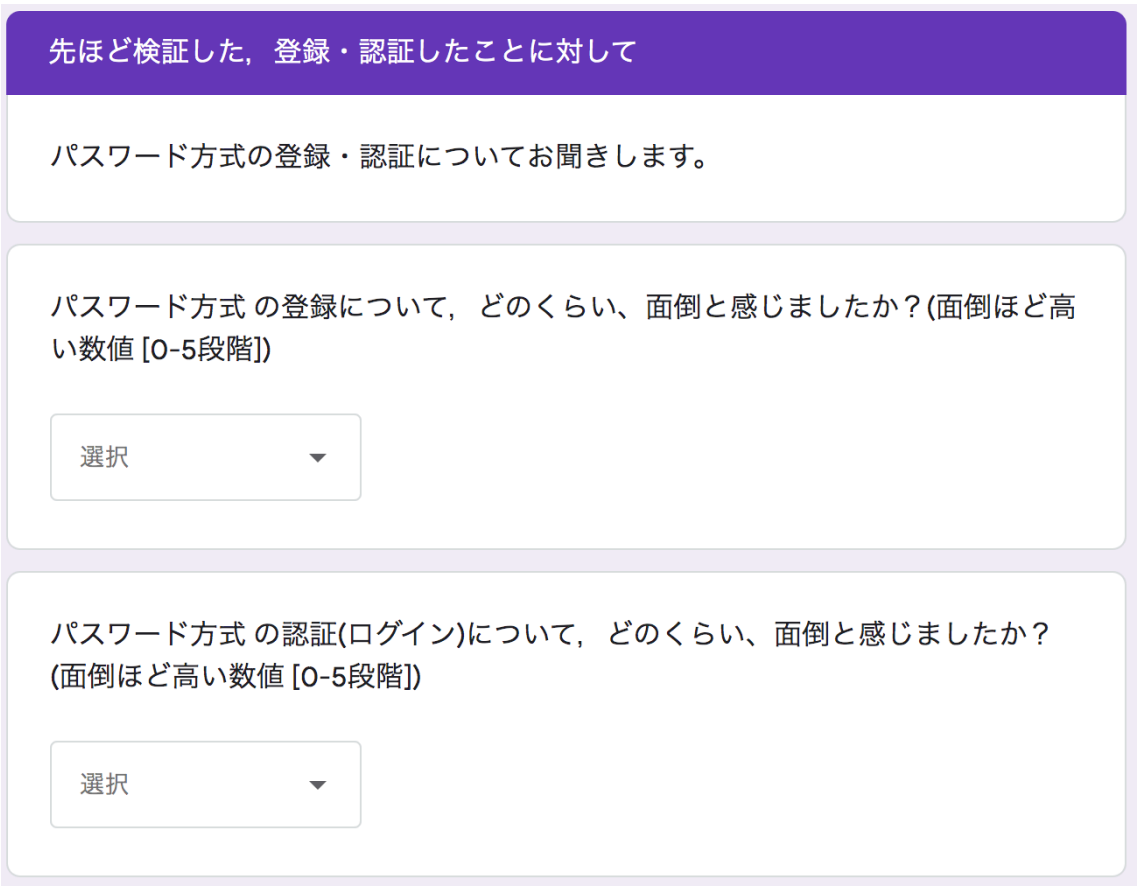

先ほど検証した，登録・認証したことに対して
パスワード方式の登録・認証についてお聞きします。
パスワード方式 の登録について，どのくらい、面倒と感じましたか？(面倒ほど高い数値 [0-5段階])
選択
パスワード方式 の認証(ログイン)について，どのくらい、面倒と感じましたか？(面倒ほど高い数値 [0-5段階])
選択

図 4.8: アンケート 2

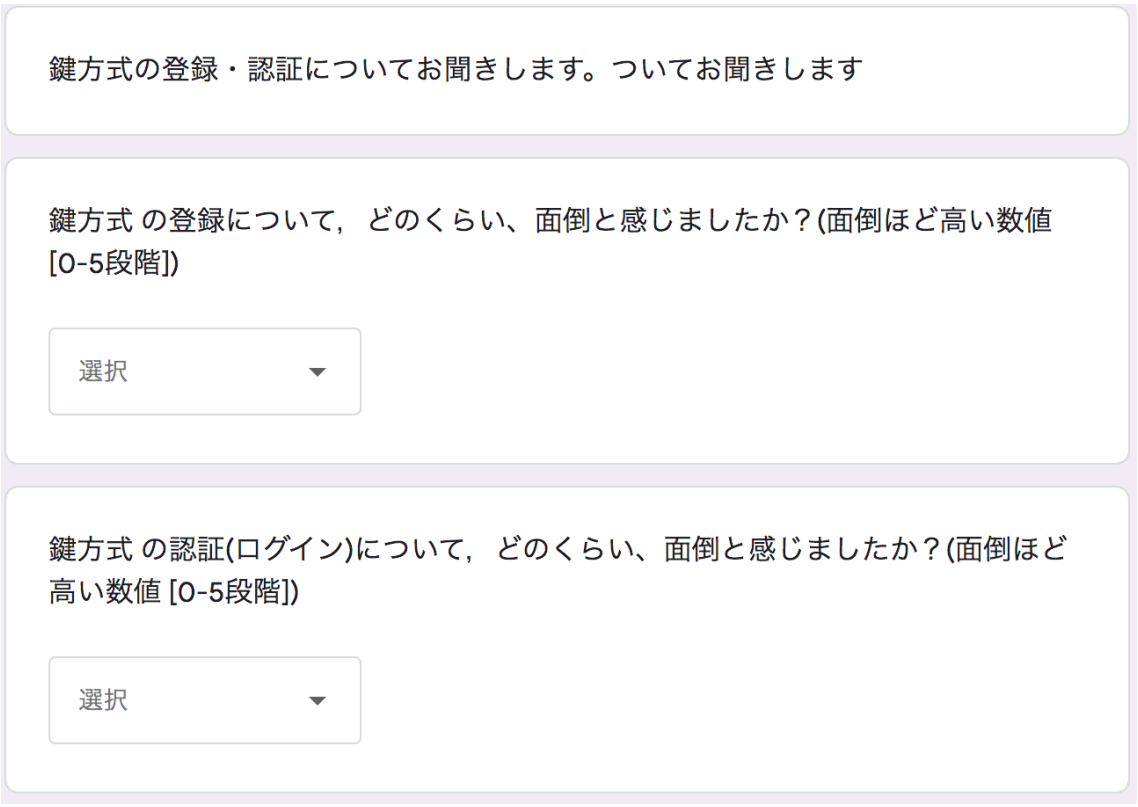


図 4.9: アンケート 3

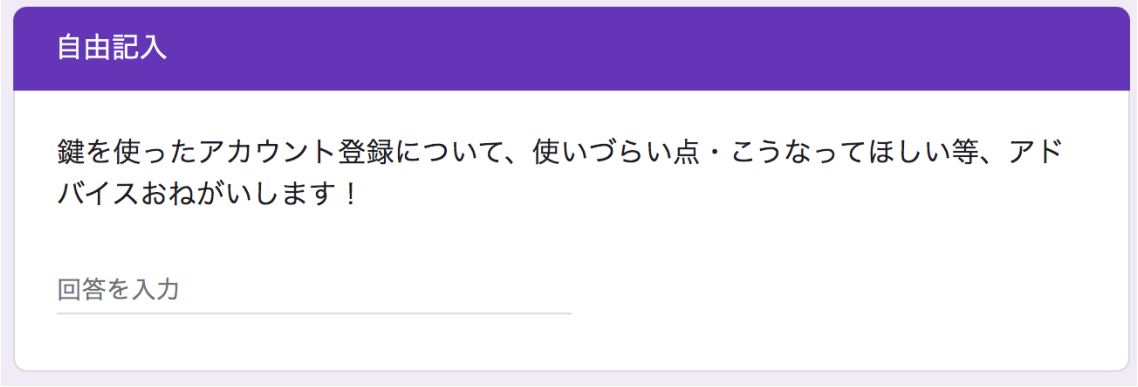


図 4.10: アンケート 4

検証マニュアル

以下の図 4.11, 図 4.12 は, 上記の図 4.1　～　図 4.10 の検証を行うためのマニュアルである. マニュアルを作成して, 検証を行った意図としては, 再現性を持って検証を行うためである.

検証方法 (検証目的)，兼マニュアル

時間計測

- 検証環境を準備する。
    - PCを用意する。
        - 被験者のPC もしくは，MacBook Pro (Retina, 13-inch, Early 2015)を用いる
    - ie-ryukyu のアクセスポイントに繋ぐ
    - ブラウザで,以下のURLが開けるか確認する
        - http://10.0.2.42:3001/login
        - http://10.0.2.42:3000/login
- 事前に,検証内容を伝える。

> 今回の実験の被験者になっていただきありがとうございます。被験者に行ってもらう検証次の通りになります。WEBサービスにおける，アカウント登録,認証をしてもらいます。2つの方式の，アカウント登録,認証を行い，時間を計測します。
> 最後に，アンケートを取らせていただくので，協力お願いします。

- 計測方法
    - アカウント登録
        - 被験者は，PCで **サインアップページ** を開いた状態にする

> パスワード方式
>
> > http://10.0.2.42:3001/signup
>
> 鍵方式
>
> > http://10.0.2.42:81/

        - 計測するための確認事項を述べる

> - 「この,webページで，アカウントの登録をお願いします。」
> - 「アカウント登録できたら，報告お願いします。」

        - ストップウォッチで時間を計測する
        - 被験者から報告を聞いたら，ストップウォッチの計測を止める
    - アカウント認証
        - 被験者は，PCで ログインページを開いた状態にする
        - ログアウト状態になっていることを確認する。

図 4.11: 検証マニュアル 1

- ログインんページに行く

  > パスワード方式
  >
  > > http://10.0.2.42:3001/login
  >
  > 鍵方式
  > http://10.0.2.42/

- 計測するための確認事項を述べる
  - 「この,webページで，登録をしたアカウントでログインをお願いします。」
  - 「ログインできたら，報告お願いします。」
- ストップウォッチで時間を計測する
- 被験者から報告を聞いたら，ストップウォッチの計測を止める

アンケート

[アンケート編集場所] ※URL省く
アンケート答える場所

- アカウント登録・認証の検証ありがとうございます。最後に，アンケートを取りたいと思います。
  検証に必要なため，ぜひお願いします。
- アンケート答える場所 で，アンケートに答えてもらう。

被験者から質問来た時に

- 今回の検証受けるに当たって、正直面倒とどのくらいかんじましたか？(0-10段階)

  > (面倒と、思ったことが，検証への意欲と関係あるか確認)

- 登録・認証について，面倒と感じた度合い(10段階)を確認する
  - アンケートにより，「パスワード形式」 と 「公開鍵暗号方式によるSSH 形式」の面倒さを数値化し，比較する。
  - アカウント登録・認証の「面倒さ」と，アカウント登録・認証の「かかる時間」の関係があるかの確認。(アカウント登録・認証にかかる時間 を，面倒さの関係性)
- アンケート

  > 今後の研究に生きるようなアンケートをもらう

図 4.12: 検証マニュアル 2

4.1.3 検証結果

被験者について

- 琉球大学情報工学科の学生 (4 人)

時間の観点からの面倒さ

ここでは被験者に行ってもらった，登録・認証にかかる時間を以下の表 4.1 に示す. 表 4.1 は パスワード方式, 鍵方式の登録・認証それぞれについて，個の時間, 平均時間を記述した表である.

表 4.1: 検証 1_認証・登録の計測時間

	被験者 1	被験者 2	被験者 3	被験者 4	平均
登録 (パスワード方式)	24.88	38.99	28.21	18.04	27.53
認証 (パスワード方式)	16.57	13.24	21.27	12.23	15.83
登録 (鍵方式)	19.96	19.14	21.73	7.81	17.16
認証 (鍵方式)	14.86	18.63	18.35	11.47	15.83

（単位：秒）

アンケートの観点からの面倒さ

ここでは，被験者に対して,2っの方式の登録・認証が終わった直後に, 記入してもらったアンケートについての数値のまとめを以下の表 4.2 に示す. 表 4.2 は 被験者による パスワード方式, 鍵方式の登録・認証それぞれについて０ ～ ５ 段階評価, さらに, 検証自体の面倒さについての評価を加えてまとめた表である.

表 4.2: 検証 1_アンケートによる 6 段階評価

	被験者 1	被験者 2	被験者 3	被験者 4	平均
登録 (パスワード方式) の面倒さ	1	1	2	0	1
認証 (パスワード方式) の面倒さ	1	1	2	1	1.25
登録 (鍵方式) の面倒さ	0	2	0	1	0.75
認証 (鍵方式) の面倒さ	0	2	0	2	1
検証自体の面倒さ	2	1	1	1	1.25

（０ ～ ５段階）

4.1.4 考察

時間の観点での”面倒さ”では，時間がかかる方”面倒”と感じると予測していたため，パスワードの観点と, アンケートの観点の面倒さの数値は比例すると考えていた。しかし, 表 4.1, 表 4.2 被験者 2 のパスワード方式と鍵方式の「アカウント登録・認証にかかる時間の大きさ」と,「アンケートによる面倒の数値」が，半比例関係だった. 上記のような結果になった理由としては 2 通り考えられる. 1っ目は，普段登録, 認証で使っている，パスワード方式の方が，慣れているため,「面倒」と感じにくかったと考察することができる. 2 つ目は，アンケートに「PC から秘密鍵を選択するのが少し面倒だと感じた。」とあり，鍵の管理を面倒だと感じたことが，影響したと思われる.

鍵方式についてのアンケートの結果に「なんかわかった」というアンケートの記述があった. 被験者に確認したところ, 鍵方式の登録・認証の流れがよく分からなかったとのことだった. 実際に，鍵方式で登録する際に，登録できたのか戸惑っていた。この原因としては, 鍵方式の登録・認証の流れが分からなかったためだと考えられる.

表 4.2 の検証自体の面倒さは 0 数値にする人が無く, 検証自体に, 少しながら面倒と感じていることが分かる. その原因としては, 検証するための WEB ページ (図 4.1 〜 図 4.6) がデバッグページのままで, 登録・認証が作業的になってしまったためだと考えられる.

検証後の聞き取りで, パスワード方式の登録で以下のような日常的に, 使用しないと思われるパスワードの確認ができた.

- パスワードの使い回し
- パスワード ”password”を設定する

そのことから, 検証方法のマニュアルに, 普段使うような WEB サービスのように実施してもらうような注意事項を用意する必要がある.

4.2　検証2

4.2.1　検証背景

検証目的

password 認証方式の，新規登録・認証の面倒さを解決するために，第 3 章の提案手法で password 認証方式に変わる，公開鍵暗号方式による ssh 認証を用いた,WEB サービス認証の提案を行った。
しかしながら，提案だけだと，新規登録・認証の面倒さを解決していることの根拠に乏しい。よって，検証を行い，第 3 章の提案手法は”面倒さの軽減”に効果的に繋がっているかの確認をする。

検証手段

検証の手段としては，実際に, 以下の 2っの認証方式の登録・認証を被験者に体験してもらう。

- パスワード方式認証 (以後 ”パスワード認証”と記述する)
- 公開鍵暗号暗号方式による ssh 認証 (以後 ”鍵認証 ”と記述する)

その後，2っの観点から，”面倒さ”を数値化する。1っ目の観点は「時間」である。”面倒さ”をアカウント登録・認証にかかる時間と推測し，計測化する。詳しい詳細については，検証環境のマニュアルに記述する。2っ目の観点は「アンケート」である。アンケートには，点数で答える方式, 文字で記入する欄 の 2っがあり，点数で答える方式により”面倒さ ”を数値化する。アンケートの細かい内容は，4.1.3 の検証画面に記述する。
また，アンケートには”面倒さ”を数値化する以外にも，以下の 2っの意味を込める。1っ目の意味は次の通りである。アカウント登録・認証にかかる時間 を，”面倒さ”と予想して検証しているが，その予想を確かめる必要がある。アンケートを取ることにより,「アカウント登録・認証にかかる時間」と,「アンケートによる面倒さ」が比例していることを確認することで，予想を確かめることができる。また，被験者の状態も確認することで，面倒と感じるのが，検証自体に対しての面倒さと関係があるのかを確認する。

2っ目の意味は次のとおりである。記入欄で，改善点や感じたことの意見をもらうことで，今後の研究に生きるようなアンケートをもらう。

検証1より効果的な検証方法にするため変更して検証をおこなう.変更点とその理由についてはそれぞれの項で述べる.

4.2.2 検証環境

検証場所

第 3 章で記述したとおり，検証場所は学科の VM を用いているため，学科のネットワーク内 (有線 LAN,wifi アクセスポイント ie-ryukyu) から，アクセスして検証を行う。

検証の流れ

ここでは, 被験者に行ってもらう，検証の流れを記述する。時間の観点で”面倒さ”を数値化する検証では, 被験者にはパスワード方式, 鍵方式の登録・認証をそれぞれ行ってもらう。その時，被験者は時間を測る。また，再現性を持って，検証を行うためにマニュアルを作成し, マニュアル通りに検証を行う。アンケートの観点で”面倒さ”を数値化する検証では，時間の観点で”面倒さ”を数値化する検証　が終わった直後に行うようにすることで，被験者の思った感情とアンケート結果の差異が少なくなるようにする。

検証画面

ここでは,検証 1 より効果的に検証結果目的を達成するために，修正した検証 2 の検証画面とその意図について述べる。

まずは検証画面図の説明をする。

パスワード方式について

- 図 4.13 は，ホーム画面である.
- 図 4.14 は，アカウント登録画面である。
- 図 4.15 は，図 4.14 で登録したアカウントに認証するための画面である。
- 図 4.16 は，図 4.15 で認証成功した後の画面である。

鍵方式について

- 図 4.17 は，ホーム画面である.
- 図 4.18 は，アカウント登録画面である。
- 図 4.19 は，図 4.18 で登録したアカウントに認証するための画面である。
- 図 4.20 は，図 4.19 で認証成功した後の画面である。

次に,検証 1　との変更点と，その意図について述べる。検証 1 のアンケート (図 4.18) で，”検証自体が面倒” と答える被験者の平均が,5 段階中の 1 であった。その原因として, デバッグ画面や, 登録フォーム・認証フォームだけの WEB 画面が，作業的な検証に繋がり,”検証自体が面倒” に繋がった大きな要因の一つと考えられる。よって, ホーム画面 (図 4.13, 図 4.17) の用意や，デバッグのための文字の消去 (図 4.14 〜 図 4.16 , 4.18 〜 4.18)，さらには,「幸せになれる WEB サイト」(図 4.13　〜 4.18) と認識してもらうことで，被験者にとって, 作業的な検証から，WEB サービスに登録する検証になることを狙って, 変更した。

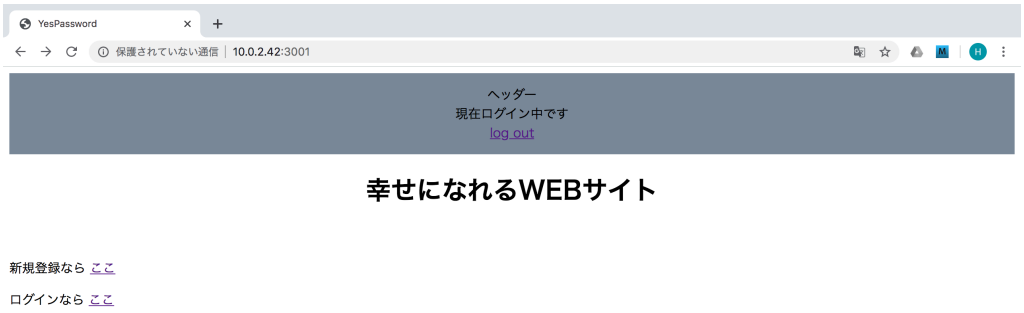


図 4.13: 検証 2_ホーム画面 (パスワード方式)

図 4.14: 検証 2_アカウント作成 (パスワード方式)

図 4.15: 検証 2_認証 (パスワード方式)

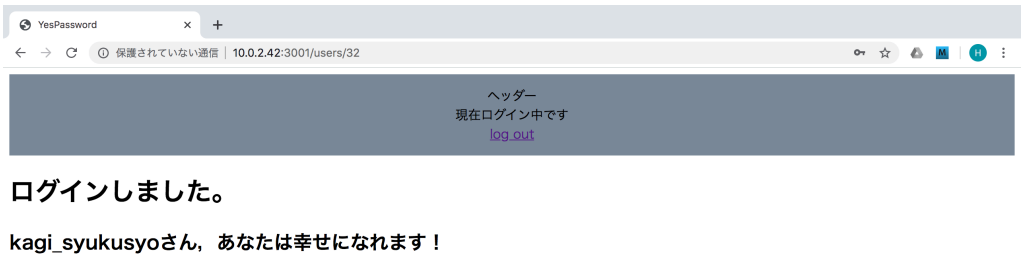


図 4.16: 検証 2_認証成功後 (パスワード方式)

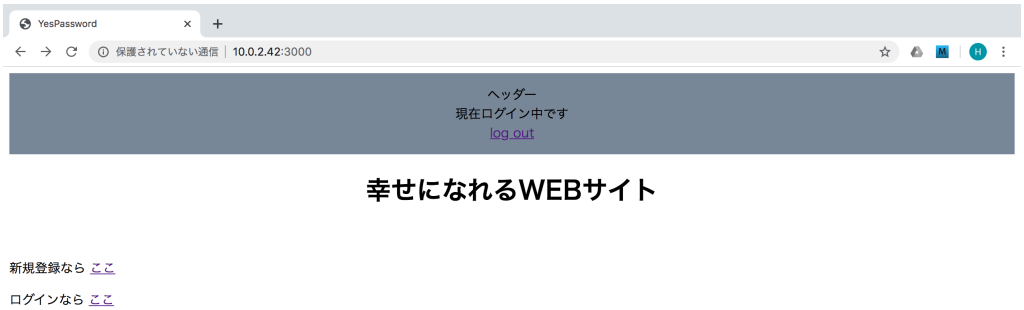


図 4.17: 検証 2_ホーム画面 (鍵方式)

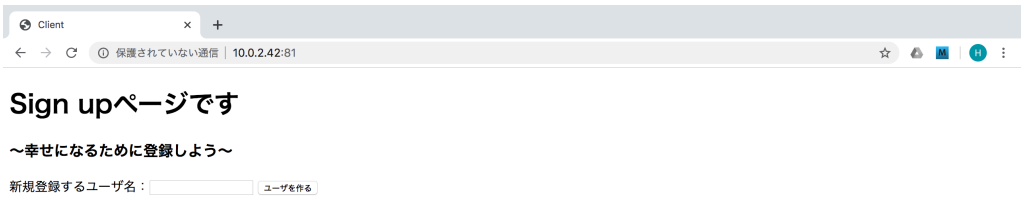


図 4.18: 検証 2_アカウント作成 (鍵方式)

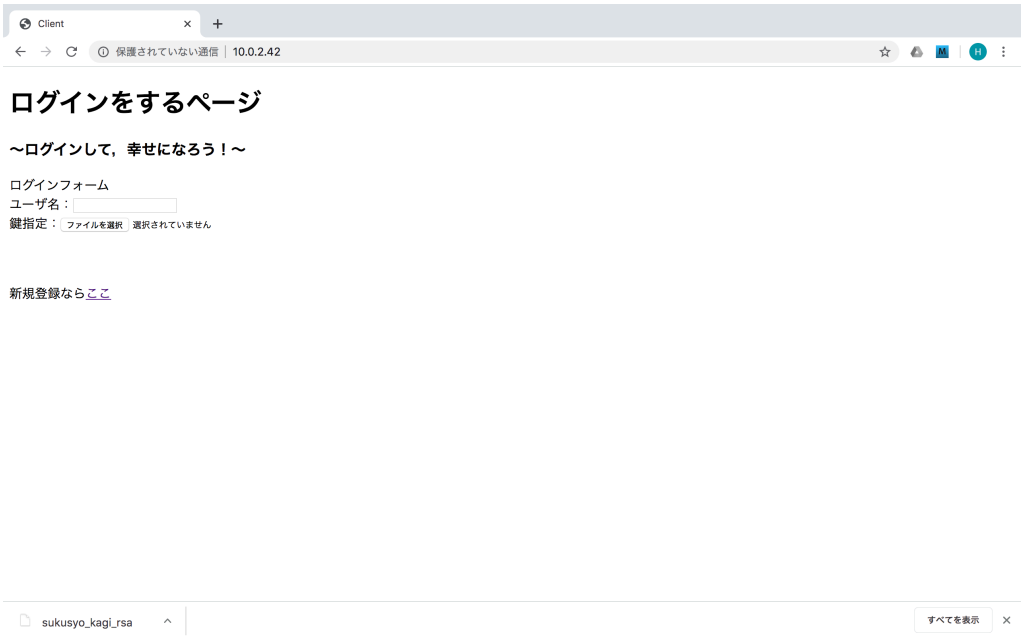


図 4.19: 検証 2_認証 (鍵方式)

図 4.20: 検証 2_認証成功後 (鍵方式)

検証マニュアル

以下の図 4.21, 図 4.22, 図 4.23 は, 上記の図 4.14 〜 図 4.18 , 図 4.7 〜 図 4.10 の検証を行うためのマニュアルである. マニュアルを作成して, 検証を行った意図としては, 再現性を持って検証を行うためである.

次に, 検証 1 のマニュアルと比べての相違点とその意図を以下に述べる. パスワードの使い回し，password というパスワードなど，日常的に設定しないと思われるパスワードの設定が見受けられた。そのため，マニュアル (図 4.21) に 「普段用いるように」，と被験者に注意を促す。また，パスワードの設定に関して，日常的に設定ような，より効果的な検証を行うためには，アカウントの登録で，パスワードに対して，バリデーション (パスワードの制限) をすることがあげられる。しかし，今回の検証では，あえてバリデーションをかけないことにする。理由は，鍵認証方式の, セキュリティは脆弱性が大きいのに対して，パスワード方式のだけセキュアにすると，検証比較として適していないと判断したためである。

鍵方式での，登録・認証に関して「何やっているかわからない」という意見があった。よって，マニュアル (図 4.21, 図 4.22) に,「”パスワード方式です” ”鍵方式です”」と軽く説明を加える。また，鍵方式の登録・認証に対しての，説明を詳しくすると，一般的に普及している，パスワード方式による，被験者の慣れ を検証に含めることができないので，軽く「”パスワード方式です” ”鍵方式です”」という説明を加えることにする。

検証方法 (検証目的)，兼マニュアル改

マニュアル

- 検証環境を準備する。
    - PCを用意する。
        - 実験である「与那嶺 東」のPCを用いる
            - MacBook Pro (Retina, 13-inch, Early 2015)
    - ie-ryukyu のアクセスポイントに繋ぐ
    - ブラウザで,以下のURLが開けるか確認する
        - http://10.0.2.42:3001/
        - http://10.0.2.42:3000/

- 事前に,検証内容を伝える。

> 今回の実験の被験者になっていただきありがとうございます。被験者に行ってもらう検証次の通りになります。WEBサービスにおける，アカウント登録,認証をしてもらいます。2つの方式の，アカウント登録,認証を行い，時間を計測します。

> (パスワードの使い回しは,気をつけてください)

> **ここでの注意事項ですが，`今後このWEBサービスを用いることがありうる`という意識のもと，普段のように，登録・認証をお願いします。**
> 最後に，アンケートを取らせていただくので，協力お願いします。

- 計測方法
    - アカウント登録
        - 被験者は，PCでホームページの，「幸せになれるWEBサイト」を開いた状態にする。

```
password認証方式なら
  http://10.0.2.42:3001/
鍵認証方式なら
  http://10.0.2.42:3000/
を開く
```

        - 計測するための確認事項を述べる

> - **このWEBサービスは，「幸せになれるWEBサイト」という名前の,WEBサービスです。**
>
> > password認証方式なら
> >
> > - まずは，`パスワード認証方式` での登録・認証をしてもらいます
> >
> > 鍵認証方式なら

図 4.21: 検証 2 マニュアル 1

        - 次に，鍵認証方式 での登録・認証をしてもらいます。
      - **このWEBサービスで，まずは，新規登録をしてもらいます。**
      - **新規登録ページに移動をお願いします。**
  - 新規登録ページに移動後の説明

    password 認証方式なら

      - これから，`password 認証方式`での登録を行います。

    鍵認証方式なら

      - これから,`鍵認証方式 `での登録を行います。

    「一斉のーせ」と行ったら，アカウント登録を開始してください。
    「アカウント登録できたら，終わったことの報告をお願いします。」
    「では始めます，いっせいのーせ 」
  - ストップウォッチで時間を計測する
  - 被験者から終わったことの報告を聞いたら，ストップウォッチの計測を止める
- アカウント認証
  - 被験者は，PCで ホームページを開いた状態にする
  - ログアウト状態になっていることを確認する。
  - 計測するための確認事項を述べる
    - パスワード認証方式

      先ほど,`パスワード認証方式`で登録したアカウントにログインしてもらいます。
    - 鍵認証方式

      先ほど,`鍵認証方式`で登録したアカウントにログインしてもらいます。

    ログインページに移動をお願いします
  - ログインページに移動後の説明
    - パスワード認証方式

      これから，`password 認証方式`での認証を行います。
    - 鍵認証方式

      これから，`鍵 認証方式`での認証を行います。

    「一斉のーせ」と行ったら，アカウントのログイン認証を開始してください。
    「アカウント認証できたら，終わったことの報告をお願いします。」
    「では始めます，いっせいのーせ 」
  - ストップウォッチで時間を計測する
  - 被験者から終わったことの報告を聞いたら，ストップウォッチの計測を止める

図 4.22: 検証 2 マニュアル 2

- アンケートをお願いする。

> はっい。ありがとうございます。最後に，アンケートを取らせていただきます。
> ぜひ，率直な意見でお願いします。

  - グーグルアンケート
  - cf.アンケート編集場所

図 4.23: 検証 2 マニュアル 3

4.2.3 検証結果

被験者について

- 琉球大学情報工学科の学生 (4 人)
- 琉球大学情報工学科の学生 (1 人)
- 琉球大学機械システム工学科の学生 (1 人)

時間の観点からの面倒さ

ここでは被験者に行ってもらった，登録・認証にかかる時間を以下の表 4.3 に示す. 表 4.3 は パスワード方式, 鍵方式の登録・認証それぞれについて，個の時間, 平均時間を記述した表である. 被験者 6 に関しては,password 方式の実装の 2っの問題点により, 計測ができなかったため, 省く.

表 4.3: 検証 2_認証・登録の計測時間

	被験者 5	被験者 7	被験者 8	被験者 9	被験者 10	被験者 11	平均
登録 (パスワード方式)	29.68	20.56	16.20	42.06	51.85	17.88	29.71
認証 (パスワード方式)	17.29	14.95	10.40	17.95	19.62	13.33	15.59
登録 (鍵方式)	11.26	11.06	8.53	19.18	12.16	11.98	12.36
認証 (鍵方式)	13.63	16.23	8.65	43.83	12.3	15.51	18.36

（単位：秒）

アンケートの観点からの面倒さ

ここでは，被験者に対して,2っの方式の登録・認証が終わった直後に, 記入してもらったアンケートについての数値のまとめを以下の表 4.2 に示す. 表 4.2 は 被験者による パスワード方式, 鍵方式の登録・認証それぞれについて０ ～ ５ 段階評価, さらに, 検証自体の面倒さについての評価を加えてまとめた表である. 被験者 6 に関しては,password 方式の実装の 2っの問題点により, 時間の計測ができなかったため, 省く.

表 4.4: 検証 2_アンケートによる 6 段階評価

	被験者 5	被験者 7	被験者 8	被験者 9	被験者 10	被験者 11	平均
登録 (パスワード方式) の面倒さ	1	4	2	2	1	3	2.17
認証 (パスワード方式) の面倒さ	1	3	2	2	2	3	2.17
登録 (鍵方式) の面倒さ	0	2	0	2	0	2	1.00
認証 (鍵方式) の面倒さ	1	1	1	2	1	2	1.33
検証自体の面倒さ	0	2	0	3	0	0	0.83

（0 ～ 5 段階）

4.2.4 考察

検証者 6 の検証は失敗した. 鍵方式の登録の検証の際, 認証まで進んでしまい, データの取得ができなかったためである. そのことから, 次の考察ができる. 被験者 6 は, 唯一 情報工学科でない学生であった. 他の情報工学科の被験者と比較して，PC を専門にしていない学生である. そのことから, より利用者目線の被験者だと言える. 利用者目線では, 今回の鍵方式は使いづらい認証方式になっていると考察することができる.

被験者 7 の認証 (表 4.3) に注目すると，パスワード方式に比べて鍵方式の方が時間がかかっている. しかし, アンケートによる認証の面倒さ (表 4.4) に着目すると，鍵方式に比べてパスワード方式の方が面倒 という数値になっている. また, 被験者 7 の検証後に次のことを聞いた.「また，検証後に次のことを聞いた.「(公開鍵暗号方式方式) わかっているから，簡単だった」」被験者 6 と比較すると, 鍵認証方式について次のことが考察できる. 公開鍵暗号方式の知識を持っている人は，公開鍵暗号方式の知識を持っていない人に比べて, 簡単に感じる傾向があるということが考えられる.

第5章　今後の課題

この章では,鍵方式の実装・検証について，今後の課題を列挙し,説明をする.
以下に,今後の課題を列挙する.

- セキュリティの脆弱性の改善
  - http 通信を行なっていることの改善
  - 秘密鍵を通信していることの改善
  - CSRF トークン OFF[22] した上での実装
- 鍵方式の登録・認証がセキュアな上での検証を行う
- 利用者視点で使いやすいようにする
  - 鍵を紛失した時の再発行をできるようにする
  - 鍵の管理を行いやすいようにする
  - 鍵方式を知らない人でも使えるようにようにする

上記に列挙した今後の課題の説明を以下に記述する.
　「セキュリティの脆弱性の改善」と「鍵方式の登録・認証がセキュアな上での検証を行う」については, 現実世界に反映することができるようにするために,改善していく必要がある. 「利用者視点で使いやすいようにする」については, 検証 2 での検証, 考察で”筆者が構築した鍵方式は, 利用者目線だと, 使いづらい認証方式” ということが考えられるので改善する必要がある.

参考文献

[1] マスタリングTCP/IP 入門編 第5版 , 2016 , オーム社 , 序文,p11

[2] マスタリングTCP/IP 情報セキュリティ編 第1版 , 2014 , オーム社 , p18

[3] 7割以上のサービスが「IDとパスワードのみの認証」、多要素認証の採用に遅れ
https://www.is702.jp/news/3520/
最終閲覧日:2019/10/09

[4] 「8文字(8桁)のパスワード」は今では時代遅れでかなり危険。どうしたら安全？
https://keepmealive.jp/8letters-danger/
最終閲覧日:2019/10/17

[5] https://www.fom.fujitsu.com/goods/pdf/security/fpt1610-2.pdf
最終閲覧日:2019/10/10

[6] マスタリングTCP/IP 情報セキュリティ編 第1版 , 2014 , オーム社 , p68 〜 p75

[7] マスタリングTCP/IP 情報セキュリティ編 第1版 , 2014 , オーム社 , p34

[8] Ruby on Rails について https://railstutorial.jp/chapters/beginning?version=5.1#sec-introduction
https://tech-camp.in/note/technology/14322/
https://techplay.jp/column/529
最終閲覧日:2020/2/15

[9] javascript について
https://developer.mozilla.org/ja/docs/Learn/JavaScript
https://ja.wikipedia.org/wiki/JavaScript
最終閲覧日:2020/2/15

[10] node.js について
https://techacademy.jp/magazine/16248
最終閲覧日:2020/2/15

[11] ssh2 モジュールについて
https://github.com/mscdex/ssh2
最終閲覧日:2020/2/15

[12] socket.io について
https://qiita.com/ToshioAkaneya/items/
eadfd3897c60cfc3ff69
https://github.com/socketio/socket.io
最終閲覧日:2020/2/15

[13] OpenSSH について
https://www.ossnews.jp/oss_info/OpenSSH
https://ja.wikipedia.org/wiki/Secure_Shell
最終閲覧日:2020/2/15

[14] HTTP cookie について
https://ja.wikipedia.org/wiki/HTTP_cookie
https://railstutorial.jp/chapters/log_in_log_out?version=
4.2#sec-sessions_and_failed_login
最終閲覧日:2020/2/15

[15] ステートレス,ステートフルについて
https://qiita.com/wind-up-bird/items/b210e294ecb147d67e2b
最終閲覧日:2020/2/15

[16] セッションについて
https://railstutorial.jp/chapters/log_in_log_out?version=4.2#sec-sessions_and_failed_login
最終閲覧日:2020/2/15

[17] Railsチュートリアル
https://railstutorial.jp
最終閲覧日:2020/2/15

[18] https://tools.ietf.org/html/rfc6265#section-1
最終閲覧日:2020/1/20

[19] https://qiita.com/mogulla3/items/189c99c87a0fc827520e
最終閲覧日:2020/1/20

[20] https://qiita.com/yasu/items/8ae3077bdbee606681f6#cookiestore%E3%81%8C%E5%95%8F%E9%A1%8C%E3%81%AA%E3%81%AE%E3%81%8B
最終閲覧日:2020/1/21

[21] https://developer.mozilla.org/ja/docs/Web/HTTP/Cookies
最終閲覧日:2020/1/21

[22] https://qiita.com/nishina555/items/4ffaf5cc57a384b66230
最終閲覧日:2020/2/15

謝辞

本研究の遂行，また本論文の作成にあたり、御多忙にも関わらず終始懇切なる御指導と御教授を賜わりましたhoge助教授に深く感謝したします。

また、本研究の遂行及び本論文の作成にあたり、日頃より終始懇切なる御教授と御指導を賜わりましたhoge教授に心より深く感謝致します。

数々の貴重な御助言と細かな御配慮を戴いたhoge研究室のhoge氏に深く感謝致します。

また一年間共に研究を行い、暖かな気遣いと励ましをもって支えてくれたhoge研究室のhoge君、hoge君、hogeさん並びにhoge研究室のhoge、hoge君、hoge君、hoge君、hoge君に感謝致します。

最後に、有意義な時間を共に過ごした情報工学科の学友、並びに物心両面で支えてくれた両親に深く感謝致します。

2010年3月

hoge