Vol.2017-CSEC-77 No.8
Vol.2017-IOT-37 No.8
2017/5/26

パスワードレス認証方式を用いた認証連携に関する研究

森井理智[†1]　谷岡広樹[†2]　大平健司[†2]　佐野雅彦[†2]
松浦健二[†2]　関陽介[†1]　上田哲史[†2]

概要：現在，ID とパスワードを用いた認証方式は，様々な情報システム及びサービスにおいて幅広く利用され主流となっている．しかしながらパスワードは，使いまわし，フィッシング，漏洩等の様々な問題を抱えている．本研究では，パスワードを利用せずに様々なサービスを利用できる認証連携の仕組みを実現するため，認証連携のミドルウェアの一つである Shibboleth の外部認証に，Fast IDentity Online(FIDO)を利用することで，パスワードを用いない認証連携の仕組みを実現し，パスワードレスによる認証連携の仕組みの実現性を検証する．さらに，この仕組みによる認証方式のセキュリティ上の問題点を検討する．

キーワード：Shibboleth, FIDO, 認証, パスワード

Research on authentication cooperation using passwordless authentication method

MICHITOMO MORII[†1]　HIROKI TANIOKA[†2]　KENJI OHIRA[†2]
MASAHIKO SANO[†2]　KENJI MATSUURA[†2]　YOSUKE SEKI[†1]
TETSUSHI UETA[†2]

Abstract: Currently, authentication methods using ID and password are widely used in various information systems and services and become mainstream. However, passwords have various problems such as reuse, phishing, leakage etc. This research, In order to realize a mechanism of authentication collaboration that can use various services without using a password, by using Fast IDentity Online (FIDO) for external authentication of Shibboleth which is one of authentication middleware, We verify the feasibility of mechanism of authentication collaboration by passwordless realization of mechanism of authentication collaboration without password and consider security problems of authentication method by this mechanism.

Keywords: Shibboleth, FIDO, Authentication, Password

1. 序論

近年，多くの人がスマートフォンやタブレット端末を含む様々な端末から情報サービスを利用することが一般的になってきている．徳島大学を含む多くの教育機関においても，ICT 環境を整備することで，様々なデバイスから授業登録や電子メール，e-Learning 等の様々なシステムが利用できる Web サービスを提供しており，簡単なログイン操作と認証のセキュリティ強化が求められている．徳島大学においては，2008 年以降，Shibboleth[1]を用いた統合認証基盤を導入することによって，複数の Web サービスの認証を一度の認証で完了できるシングルサインオン(SSO)を実現しており，一組の ID とパスワードで複数のサービスを利用できる．

しかしながら，徳島大学では ID と初期パスワードは紙でユーザに配布され，安全性の観点から一定期間でパスワードを変更する規定がある．変更後のパスワードは，英数記号混じりの 10 文字以上といった複雑なものであるため，パスワードを忘れる利用者も少なくない．また，多くの利用者がパスワードの有効期間内のパスワード変更を行わず，パスワード再発行の申請をする利用者が多いことも問題となっている．

シングルサインオンを実現している統合認証基盤上では，パスワード自体にも課題がある．使いまわし，フィッシング，取り扱い不注意などを原因として，ひとたび ID とパスワードが漏洩すると，複数のサービスに対して不正利用の危険性がある．ID とパスワードを用いる以外に，カードを用いた認証や生体認証を組み合わせた多要素認証などの認証システムを導入している事例[2]もあるが，コストや利便性の観点から全面的に採用することは難しい．

そこで，パスワードの問題を解決するために Fast Identity Online (FIDO)を導入することを検討する．FIDO[3][4]は，認証の際に FIDO 準拠のデバイスで様々な認証方式のローカル認証に対応できる．FIDO による本人確認は，パスワードを用いずにユーザの署名を用いて行うことにより，パスワード等のクレデンシャル情報が，通信経路上に流れることがないため安全性が高い．また FIDO 対応端末であれば，スマートフォンやタブレット端末からも利用可能である．関連研究[5]によると，生体認証システムには偽装問題

†1 徳島大学大学院先端技術科学教育部
Graduate School of Advanced Technology and Science
†2 徳島大学情報センター
Tokushima University Information center

があると報告されているが，五味ら[6]は，FIDO は有望な技術であると述べている．しかしながら，FIDO に関する具体的な事例は報告されていないため，本研究では，FIDO の実現可能性の確認と通信経路を含めた検証を行う．

本稿では, Shibboleth と FIDO を連携した認証システムを実装することで，パスワードレスの統合認証基盤の実用可能性を検討する．また，FIDO を用いたパスワードレス統合認証基盤の利点と問題点を明らかにする．

2. 関連技術

2.1 Shibboleth

Shibboleth は，Internet2 の MACE（Middleware Architecture Committee for Education）プロジェクトによって開発された SAML（Security Assertion Markup Language）を利用して，シングルサインオン及び属性共有を実現したオープンソースソフトウェアである．SAML は，情報交換用技術標準を作成する非営利国際コンソーシアムである「OASIS」によって策定された XML（Extensible Markup Language）をベースにした言語であり，ユーザ認証に必要な情報を異なるドメイン間で安全に交換するための言語仕様になっている．

Shibboleth は Identity Provider (IdP), Service Provider (SP), Discovery Service (DS)の３つで構成されている．DS は IdP サーバを選択するために必要だが，今回は IdP サーバを指定するため省略する．IdP と SP については，以下に詳細を述べる．

(1) **Identity Provider (IdP)**

IdP サーバは，SP サーバからの要求に応えて属性を返信し，フェデレーション内で共有するサーバである．フェデレーションとは，定められた運用ポリシーに従い信頼しあうことで，認証連携を実現する連合体のことである．IdP サーバ自身には情報を持っておらず，LDAP（Lightweight Directory Access Protocol）サーバや Microsoft 社の Active Directory （AD）といったディレクトリサービスから特定のデータのみを抽出してデータを外部へ公開する．このとき，IdP サーバは ID とパスワードの組み合わせや，証明書認証等の方法でユーザ認証を行う．

(2) **Service Provider (SP)**

SP サーバは，サービスを提供する Web サーバである．SP サーバはサービスの利用に必要な利用者の属性を IdP サーバに要求し，IdP サーバから得た属性に基づいて，アクセス制御や Web サービスの利用を許可する．

Shibboleth の動作について説明する．図 1 は Shibboleth の動作の流れを示したものである．

(1) **ユーザは，ブラウザを利用して SP サーバにアクセスを行う．有効な SP のセッションを所有しているならば，SP サーバはアクセスを許可する．**

(2) **SP サーバは，IdP サーバにリダイレクトを行う．**

(3) **IdP サーバが提供している認証方式でユーザの認証を行う．このとき，有効なセッション情報を所有している場合は，認証を省略できる．**

(4) **IdP サーバは，ディレクトリサービスから属性情報を取得し，属性情報を付与して SP サーバにリダイレクトを行う．**

(5) **SP サーバは，IdP から送られた属性情報を基に，アクセスを許可する．**

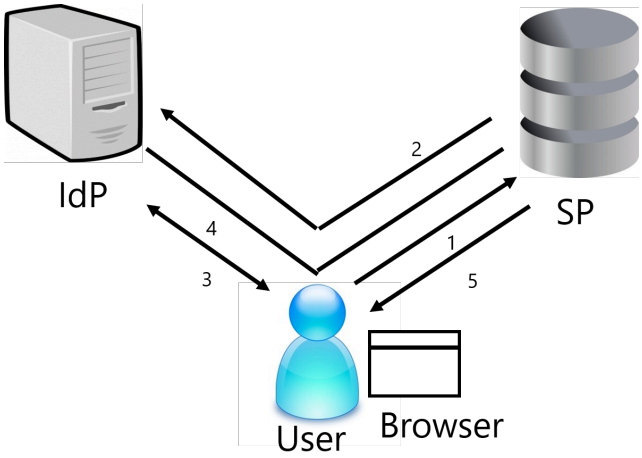

図 1　Shibboleth の動作

2.2 FIDO

FIDO は，オンライン認証に生体認証等を利用するための認証手順を定めた仕様である．主な特徴としては，以下の３つがあげられる．

- Client で，ユーザの署名等により認証を行うため，生体情報やパスワード等のクレデンシャル情報が通信経路上に流れない．
- サーバに，認証に必要な情報を登録する必要がない．
- 生体認証以外にも，PIN コードや NFC 等，様々な認証をサポートしている．

FIDO 1.0 においては，二つの認証方式が考案された．一つは Universal Authentication Framework (以下 UAF)，もう一つは Universal Second Factor (以下 U2F) である．最新の FIDO 2.0 においては，この二つが統合された．

(1) **UAF**

UAF は，利用者が端末に生体情報を登録し，端末を認証サーバに登録しておけば，端末の認証のみでログインすることができる．FIDO 対応のデバイスで公開鍵と秘密鍵を生成し，その公開鍵を事前に認証サーバに登録する．

(2) **U2F**

U2F は, ID とパスワードを必要とする既存の認証方法に第 2 要素を加えた 2 段階認証の仕組みである．USB キーや NFC のようなデバイスを使用することが想定されている．パスワードと併用するため既存のシステムに組み込みやすく，パスワードの脆弱性に対して対策できる．

(3) **FIDO 2.0**

FIDO 2.0 は，UAF と U2F を統合したもので，WebAPI が提案されている．現在(2017/03/01)は，Microsoft 社開発の Web ブラウザである Microsoft Edge にのみ実装されている．Windows 10 の生体認証機能である Windows Hello や Microsoft Passport といった認証機能も FIDO 2.0 に準拠[7]している．

以下にフローを示す．Authenticator はユーザの検証を行うところであり，クレデンシャル情報が格納されている．Server は FIDO Server で, 認証の判断を行うところである．Account は利用者の情報が入っている．図 2 はユーザが端末を登録する際のシーケンス図である．

(1) **Client は Server に登録要求を行う．**
(2) **Server はアカウント情報を保持しているアカウントサーバ(Account)に対して, FIDO に対応したアカウントを作る，もしくはアップデートを行い，チャレンジ情報を Client に送る．**
(3) **Client は，チャレンジ情報に基づき，認証システム(Authenticator)を介して key pair を作成する．**
(4) **Client は，public key を含む認証情報に，登録する id を付与して Server に送る．**
(5) **Server は，Account に認証情報を登録する．**
(6) **Client は登録成功を受け取る．**

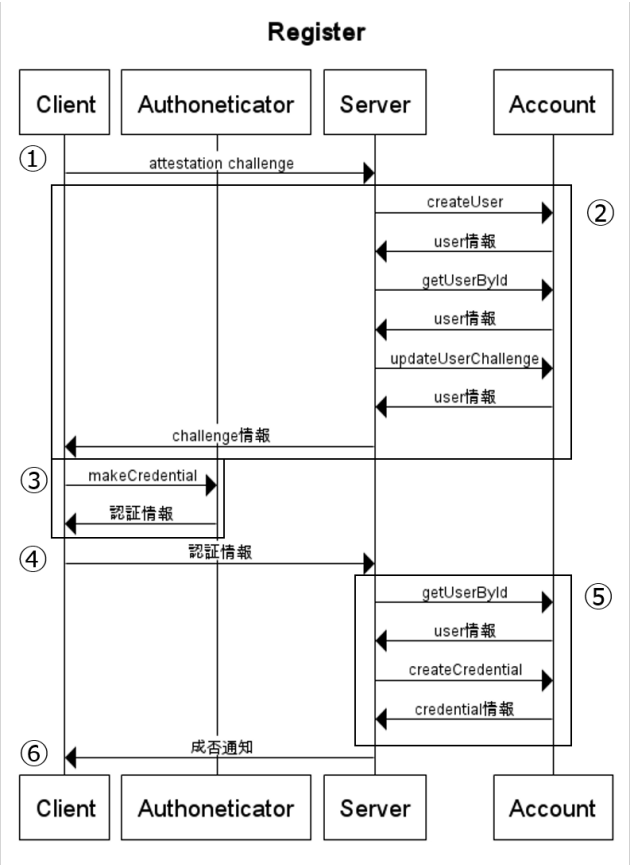

図 2　FIDO 2.0 の登録フロー

図 3 はユーザが認証する際のシーケンス図を書いたものである．

(1) **Client は Server にログイン要求を行う．**
(2) **Server はチャレンジ情報を作成しアカウントサーバ(Account)に登録し，ユーザ情報を取得して，チャレンジ情報を Client に送る．**
(3) **Client は，端末を用いて認証後，Server から送られてきたチャレンジ情報を private key で暗号化し，認証結果を合わせてレスポンス情報として，Server に送る．**
(4) **Server は，Account を参照してレスポンス情報が正しいか検証する．**
(5) **Client はログイン完了を受け取る．**

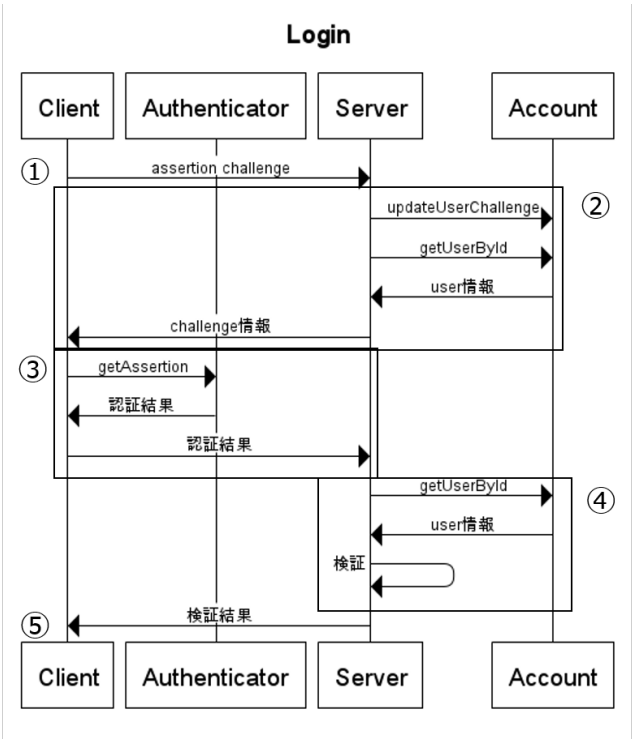

図 3　FIDO 2.0 の認証フロー

3. システム設計

3.1 システムの概要

Shibboleth は外部認証に対応しており，認証機能を外部システムに委任することが可能である．本システムでは，FIDO サーバ[8]による外部認証機能を利用して Shibboleth のユーザ認証を行う. FIDO サーバによる外部認証では，まず IdP サーバが FIDO サーバに認証を委任し，その認証が成功した状態を，セッションを IdP サーバと共有することで，サービスへのログインを実現する．FIDO サーバでの認証成功時，IdP サーバはユーザ ID に結びついている属性情報を LDAP サーバから取得し，セッション ID と紐づけて SP サーバと共有する．SP サーバが属性情報を取得できれば, FIDO サーバの認証による Shibboleth との認証連携は成功となる．

3.2 システム構成

本システムの流れについて説明する．図 4 は作成するシステムの構成図である．

(1) **Client は，SP サーバにアクセスする．SP サーバに有効なセッションが存在する場合，その情報を基にアクセスを許可する．**

(2) **SP サーバは，Client に認証を行ってもらうために IdP サーバにリダイレクトを行う．IdP サーバに有効なセッションが存在する場合，認証を省略する．**

(3) **IdP サーバは，FIDO サーバに認証を行うよう設定されているため，FIDO サーバにリダイレクトを行う．**

(4) **FIDO サーバは，Client に認証要求を行う．**

(5) **ユーザは，Client でローカル認証を行い，FIDO サーバは認証結果が正しいか検証する．正しい場合は，セッション ID を発行し，IdP サーバとセッション ID を共有しておく．**

(6) **FIDO サーバは，IdP サーバにリダイレクトを行い，セッション ID を利用して IdP サーバから属性情報を取得する．**

(7) **IdP サーバは，許可されている属性情報を SP サーバに送信し，その情報を基にサービスを提供する．**

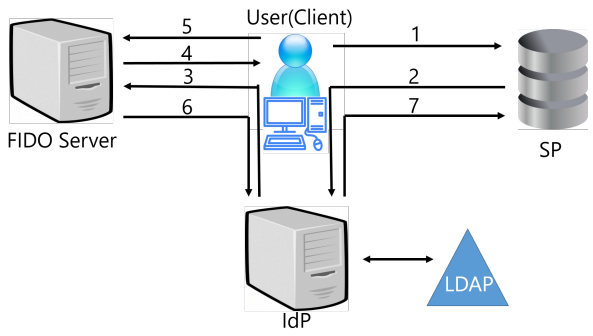

図 4　システム構成図

4. システム実装

4.1 実装環境

使用した主なソフトウェア・OS を表 1 に示す．本システムの開発は，仮想マシン上で行った．IdP サーバ，SP サーバ，FIDO サーバの他に，IdP サーバと FIDO サーバ間でセッションを共有するため memcached を採用した．また，ユーザ ID に関連する属性情報は，OpenLDAP により管理する．

なお, FIDO 2.0 に準拠した API が実装されているブラウザは Microsoft Edge のみであり，生体認証には Windows Hello を使用しているが，Windows 10 の OS ビルド 14393 以降を用いる必要がある．

表 1　使用した主なソフトウェア・OS

ソフトウェア	バージョン
Shibboleth-identity-provider (IdP)	3.3.0
Shibboleth (SP)	2.6.0
Node.js (FIDO Server)	4.7.2
memcached	1.4.4
OpenLDAP	2.4.40
Windows10(ホスト OS)	OS ビルド 14986
CentOS(ゲスト OS)	6.8

4.2 登録フェイズ

FIDO サーバは，ユーザが入力した ID でアカウントを作成またはアップデートを行い，ローカル認証時の署名に用いる鍵を作成するために必要な情報を Authenticator から Client に送信する．Client は，送られてきた情報を用いて鍵の作成を行う．その後，ローカル認証の ID や公開鍵といった登録が必要な情報に，FIDO サーバの ID を付与して送信する．FIDO サーバはその情報を保存し，登録の成否を送る．

4.3 認証フェイズ

図 5 は，実装したログイン画面である．PIN 番号及び設定した指紋認証でログインすることが確認できた. 図 6 は，ユーザが SP サーバに Shibboleth 認証の対象となるリソースに対して，アクセスを要求する際の流れを示したものである. ユーザが Web ブラウザを用いてリソースにアクセスを要求し，FIDO サーバによる認証を利用して，そのリソースにアクセスできるまでの過程を次に示す.

(1) Client－SP サーバ

Client は Shibboleth 認証の対象となるリソースにアクセスを要求する．この時，SP サーバから送信されたセッション情報が cookie に保存されている場合，アクセスを許可する．セッションが cookie に保存されていない場合や，有効期間を超過している場合は，IdP サーバにリダイレクトを行い，認証を促す.

(2) Client－IdP サーバ

IdP サーバは，IdP サーバに対する認証済みのセッション情報を Web ブラウザの cookie に保存している場合，Assertion を SP サーバに返す．セッション情報を持っていない場合や，有効期間を超過している場合は，外部認証ハンドラ（外部認証機能の呼び出し）を実行し，FIDO サーバにリダイレクトする．この際，疑似的にセッションを保持するためにセッション ID を作成し，conversation key として Web ブラウザの cookie に保存しておく．このとき，FIDO サーバ側でリダイレクトするための Path を FIDO サーバにしておく．

(3) **Client－FIDO サーバ**

FIDO サーバは，Client に対して認証画面を提示し，ID を入力としてローカル認証を行い，認証結果を求める．FIDO サーバは，送られてきた認証結果が正しいものかを検証し，セッション ID を発行する．セッション ID は session key という名前で cookie に保存する．FIDO サーバはセッション ID と認証した ID を IdP サーバと共有するために，memcached を用いてセッション ID をキーに ID を保存する．その後，②で保持していたセッション ID（conversation key）を用いてリダイレクトされた IdP サーバでセッションを再開する．

(4) **Client－IdP サーバ**

IdP サーバは，Client の cookie からセッション ID を取得する．さらにセッション ID を用いて memcached から ID を取得し，ID を用いて LDAP サーバから属性情報を取得する．属性情報の取得に成功したら，認証済みのセッション情報を作成して cookie に保存し，Assertion を SP サーバへ渡す．

(5) **Client－SP サーバ**

SP サーバは，cookie に保存されたセッション情報と，IdP サーバから受け取った属性情報を基にユーザに対してサービスを提供する．

図 5　ローカル認証のログイン画面

5. 評価・考察

本認証システムは，登録フェイズと認証フェイズに分けられる．FIDO を用いた認証方式によると，登録フェイズ及び認証フェイズにおいてパスワードを用いる必要がなく，通信経路上にもクレデンシャル情報が流れないことが確認できた．一方で，各フェイズにおける問題点について，不正アクセスと脆弱性の観点で考察する．

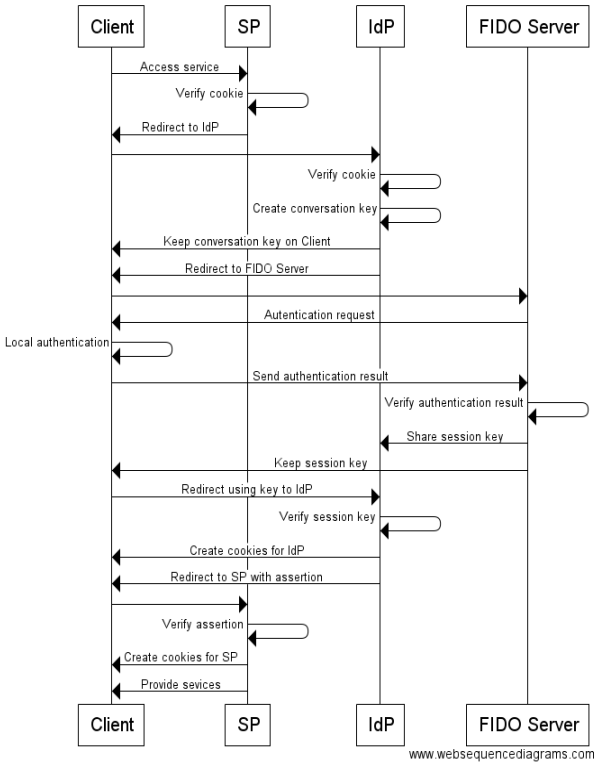

図 6　システム利用の流れ

5.1 不正アクセス

(1) **登録フェイズ**

認証システムに登録または更新するユーザについては，本人認証が正しく行われなくてはならない．なりすましや信頼できないデバイスによるスキミング等により，不正に登録された場合，アカウント乗っ取りが簡単に行われる．FIDO の仕組みを用いても，この問題は解消できない．

(2) **認証フェイズ**

通信経路への不正アクセスを考慮するために，すべての情報をパケットキャプチャし確認しところ，パスワードや生体情報といったクレデンシャル情報が流れていないことが確認できた．したがって，認証フェイズにおけるアカウント乗っ取りの可能性はないといえる．しかし，Web ブラウザの cookie や memcached によりセッションを共有しているため，セッションハイジャックの可能性は否定できない．

5.2 セキュリティの脆弱性

(3) **登録フェイズ**

登録フェイズにおける脆弱性は，署名確認するための鍵情報をクライアントから FIDO サーバへ送信する部分にある．FIDO サーバ側でユーザ ID と鍵を紐つけなければならないが，その鍵の信頼性が重要なポイントになる．盗聴されるだけで乗っ取られるパスワードと比べると安全性は高い．しかし，鍵を改ざんされたり，不正に登録されたりした場合，簡単にアカウント乗っ取りが行える．よって，鍵の改ざんの検知や，不正登録に対する対策が重要となる．

(4) 認証フェイズ

デバイスや API の脆弱性について検討する．使い捨てのセッションを用いる方法は，既存の Shibboleth と同様であり，外部認証に FIDO サーバを用いたとしても，cookie や memcached などのセッションの管理方法に脆弱性がない限り，安全性の低下はないと考えられる．そのため，認証フェイズにおける脆弱性は，基本的に生体認証システムの脆弱性であるといえる．

本システムでは統合認証基盤としての Shibboleth に FIDO を組み合わせている．鈴木と宇根[5]は，「生体認証システムを適切に利用していくためには，生体認証に特有の脆弱性としてどのようなものが知られているかを把握し，適切な対策を講じていくことが重要である.」と述べているが，FIDO がローカル認証に用いている生体認証システムに脆弱性が存在した場合，その影響範囲は，Shibboleth が適用されているサービス全体に及ぶため，事前に十分な対策を講じておくことは必要不可欠である．ローカル認証に生体認証システムを用いない場合も，何らかの脆弱性が存在する可能性はあるため，FIDO を用いたとしても，常に認証フェイズが安全であると考えてはいけない．

6. 結論

本研究では, Shibboleth の外部認証に FIDO を用いた統合認証システムの実現可能性を検討し，パスワードレス認証システムを実現した．また，FIDO を用いた認証方式における利点と問題点を検討した．

従来のパスワード認証から FIDO によるローカル認証に変更することで，通信経路上にクレデンシャル情報が流れず，従来のパスワード認証よりも利便性を高めつつ，安全性を向上させられることが確認できた．パケットキャプチャにより，通信経路に流れているデータを調べ，不正ログインの困難性を確認した．また，登録フェイズと認証フェイズにおける脆弱性を検討し，どのような問題点があるかを整理し，特に登録フェイズにおけるなりすましや乗っ取りの危険性について指摘した．

本システムの構築においては，Shibboleth の外部認証機能を利用した. Shibboleth サーバ及び FIDO サーバを実装し，セッション ID を IdP サーバと共有することにより，Shibboleth による FIDO 認証を実現した．具体的には，memcached を用いてセッション管理機能を実装し，Shibboleth の外部認証機能でセッションを共有し，リダイレクトすることで FIDO サーバを連携した．

今後の課題としては，登録フェイズにおいて，鍵の改ざんの検知や，不正登録に対する対策，端末のセキュリティ強化が考えられる．また，実装に関しての課題としては，FIDO 非対応端末を用いたパスワードレスのシステムを実装する方法を検討したい．

参考文献

[1] Shibboleth, https://shibboleth.net/ (参照 2017-04-14)
[2] 河野圭太, 藤原崇起, 稗田隆, “岡山大学事務情報システムにおける Shibboleth との連携を考慮した多要素認証の導入,” 研究報告セキュリティ心理学とトラスト（SPT）, 一般社団法人情報処理学会, Vol.2014, No.5 pp.1-6, 2014-10-02.
[3] FIDO Alliance, https://support.office.com/ja-jp/ (参照 2017-04-14)
[4] Fido Alliance approach vision, https://fidoalliance.org/approach-vision/ (参照 2017-04-14)
[5] 鈴木雅貴, 宇根正志: 生体認証システムの脆弱性の分析と生体検知技術の研究動向, 日本銀行金融研究所, 金融研究, 2009.10.
[6] 井澤秀益, 五味秀仁: 次世代認証技術を金融機関が導入する際の留意点−FIDO を中心に−, Discussion Paper, No.2016-J-3, 日本銀行金融研究所.
[7] Windows10 で FIDO 認証技術をサポート, https://blogs.windows.com/japan/2015/02/19/microsoft-announces-fido-support-coming-to-windows-10 (参照 2017-04-14)
[8] GitHub – apowers313/fido2-Server: A FIDO 2.0 / W3C WebAuthn Server, https://github.com/apowers313/fido2-Server (参照 2017-02-13)