JavaScriptを用いたスマートフォン向け FIDO UAF Cloudの提案と評価

松本 悦宜[1]　中川 和弘[1]　渡辺 孝信[1]　岡田 満雄[1]　満永 拓邦[2]

概要：近年，パスワードを用いないユーザ認証方式として FIDO が注目されている．公開鍵方式を利用した FIDO 仕様である FIDO UAF は，クライアントからサーバにパスワードを送信する必要がなく，セキュリティ強度を高めることができる．しかしながら，FIDO UAF の導入においては，FIDO UAF 認証サーバの構築が必要となり，導入コストが高いという課題がある．本論文では，この課題を解決するため，JavaScript を用いて FIDO UAF をクラウドで実装する手法を提案する．

キーワード：認証，FIDO, UAF, JavaScript, スマートフォン

Proposal and Evaluation of FIDO UAF Cloud for Smart Phone Using JavaScript

Yoshinori Matsumoto[1]　Kazuhiro Nakagawa[1]　Takanobu Watanabe[1]　Mitsuo Okada[1]
Takuho Mitsunaga[2]

Abstract: Recently, FIDO has attracted attention as a user authentication without using a password. FIDO UAF does not need to send a password from the client to the server because FIDO UAF uses public-key cryptography. Therefore, FIDO is considered that it enables to increase the security level of authentication. When service providers install FIDO UAF, it is necessary for them to construct a FIDO UAF authentication server. However, there is a problem that the cost is too high. In this paper, we will propose a method to implement FIDO UAF in the cloud using JavaScript to solve the problem.

Keywords: Authentiation, FIDO, UAF, JavaScript, SmartPhone

1. はじめに

FIDO (Fast IDentity Online) は，次世代の標準化されたオンライン認証技術のひとつとして注目されている．規格の策定は FIDO Alliance *1 という業界団体により行われており，「U2F」と「UAF」という 2 種類の仕様が標準化されている [1] [2] .

U2F (Universal 2nd Factor) は，認証器を用いた二要素認証の仕組みである．ID とパスワードで認証を行った後で，ユーザが保有する認証器のボタンを押すことによって認証することができる．認証方法としては，秘密鍵で署名を作成して，認証サーバで署名検証することによって実現される．

一方，UAF (Universal Authentication Framework) は，UAF に準拠した認証器を用いて生体認証が行われ，パスワードを使用しないことが特徴として挙げられる．この際，認証はユーザが保有する認証器で行われるため，Web サーバに生体情報が送信されない．このため，仮に通信中に第三者の攻撃などを受けた場合でも，生体情報の漏えいリスクを軽減できるという利点がある．

Web セキュリティの観点からは，UAF ではパスワードが不要となるため，Web サービスのログイン画面に対する

[1] Capy 株式会社
Capy Japan Inc.
[2] 東京大学
The University of Tokyo
*1 FIDO Alliance https://fidoalliance.org/

パスワードリスト型攻撃やパスワードスプレー攻撃 [3] などの不正ログインを狙った攻撃に対して有効であると考えられる．また，ユーザ毎にハードウェアトークンを用意することやサービス提供者が生体認証のデバイスを用意せずに，スマートフォンに搭載された認証器で生体認証を行うことができるので利便性も向上すると考えられる．

一方，サービス事業者にとっては，FIDO UAF 認証サーバなどの専用環境を自社で構築しなければならないため，導入の際の負担は大きい．FIDO UAF 認証サーバは UAF に準拠したユーザの認証や登録など複雑な処理を行う必要があるので，多大な構築コストを要する．また，サーバの増加に伴い，運用コストの増加も懸念される．

本論文では，FIDO UAF 認証サーバをクラウドで提供し，各サービス事業者が，ログイン画面に設置した JavaScript を用いるとともに 2 つの API を実装することのみによって FIDO UAF を導入するシステムを提案する．このことにより，各サービス事業者側で FIDO UAF 認証サーバを構築する必要はなくなり，サーバ構築・運用に伴う費用を低減することができる．

2. 先行研究・事例

Agrawal と Patidar はユーザのデバイスを用いて生体認証を行なった際のセキュリティ技術について調査した [4]．パスワード代替方法として，Everts らは，公開鍵暗号を使用して，Web サービスの認証を行うスマートフォン用のアプリケーションを開発している [5]．先行研究として，FIDO を使用した研究も進められており，Chifor らは FIDO UAF アプリケーションを開発している [6]．FIDO に関しては，U2F に準拠したサービスも多く，例えば Google，Dropbox など大手サービス事業者のサービスが挙げられる [7]．また，W3C (World Wide Web Consortium) は，FIDO2 を中核技術とした WebAuthn (Web Authentication) を策定している．このことにより，標準機能として実装され，ブラウザやスマートフォンアプリケーションなどでの展開が期待されている [8]．

3. 提案システム

3.1 提案システムのモデル

本システムの利用方法について説明する．登録時において，ユーザは，登録用のアプリケーションを起動する．アプリケーションは，Android OS で Android アプリケーションとして実装されている．ユーザが Android アプリケーションを起動するとログイン画面が表示される．ログイン画面のフォームにユーザ自身のユーザネームとパスワードを入力して送信する (図 1)．ここでは，登録時に既存システムへの導入を想定し，ユーザパスワードを使用しているが，認証時にはパスワードは使用しない．その後，Android アプリケーションの画面が切り替わり指紋認証を促す画面が表示される (図 3)．指紋認証に成功すると登録が完了となる．認証時において，ユーザは，ブラウザで認証画面にアクセスする (図 2)．認証画面は端末内に制限されずに，他の端末のブラウザで表示できる．ユーザは，認証画面で登録時に入力したユーザネームを入力する．このときにシステムからユーザの端末に Push 通知を送信する．ユーザの端末内の Android アプリケーションが起動され，指紋認証を促す画面が表示される (図 3)．端末内で指紋認証が行われ，FIDO UAF 認証サーバで認証を検証する．検証に成功すると，ログインに成功とみなされる．

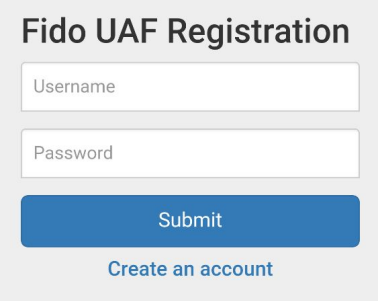

図 1　登録時のログイン画面

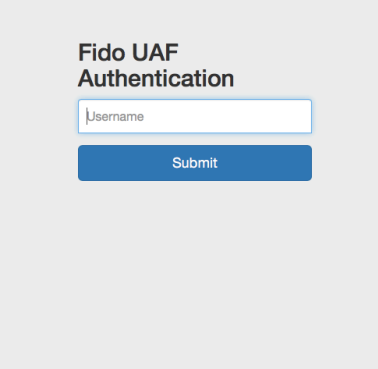

図 2　認証画面

図 3　指紋認証画面

3.2 提案システムの登録手順

まず初めに，FIDO UAF の仕様で規定されていない利害関係者の相互通信部分の登録手順について説明する．登録手順を図 4 に示す．提案システムの利害関係者としては，ユーザ，サービス事業者および UAF クラウドプロバイダの 3 名である．ユーザは，サービス事業者の利用者である．サービス事業者は，UAF クラウドプロバイダが提供する UAF 認証サーバ (UAF Auth Server) を用いて，UAF をユーザに提供する．この図において，ユーザは，Android アプリケーションを使用する．App Server はサービス事業者がサービスを運用するサーバである．UAF クラウドプロバイダーは，UAF Auth Server を提供する．

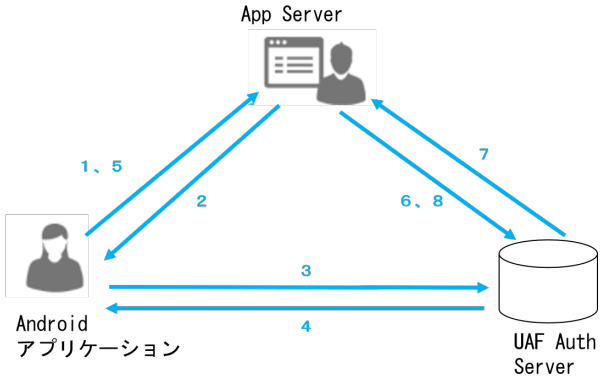

図 4　登録手順

(1) ユーザは，Android アプリケーションで登録用のログイン画面にアクセスする．
(2) App Server は，ログイン画面を表示する．ログイン画面は WebView を用いて Web ページとして表示される．このページには，FIDO UAF 用の JavaScript が含まれる．
(3) ユーザの Android アプリケーションが，UAF Auth Server に対して JavaScript の取得を要求する．
(4) UAF Auth Server が JavaScript を返却し，Android アプリケーションで JavaScript が実行される．
(5) ユーザがユーザネームとパスワードを入力すると，ユーザの Android アプリケーションが，ユーザネーム，パスワード，FirebaseToken（Push 通知用のトークン）およびハッシュ化されたユーザネームを App Server に送信する．ハッシュ化されたユーザネームは，JavaScript の処理によって生成される．
(6) App Server は，ハッシュ化されたユーザネーム，Service ID および FirebaseToken を送信する．Service ID はサービス事業者の識別子となる．
(7) UAF Auth Server では，事前に Service ID と App ID (FIDO UAF のエントリーを示す URI) とを紐づけを行っておく．UAF Auth Server は，Service ID から App ID を取得する．
(8) App Server は UAF Server から認証結果を返却する．

次に，FIDO UAF の仕様を含めた具体的な登録シーケンスを図 5 に示す．

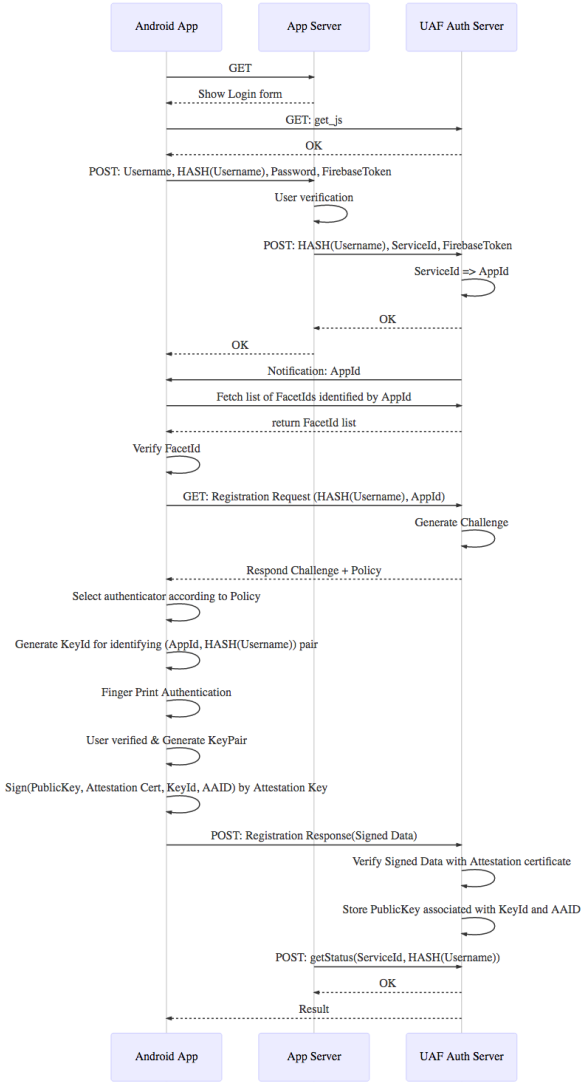

図 5　登録シーケンス

(1) ユーザは，Android アプリケーションで登録用のログイン画面にアクセスする．
(2) App Server は，ログイン画面を表示する．このページには，UAF 用の JavaScript が含まれる．
(3) Android アプリケーションは，UAF Auth Server に対して JavaScript の取得を要求する．
(4) UAF Auth Server が JavaScript を返却し，Android アプリケーションで JavaScript が実行される．
(5) ユーザがユーザネームとパスワードを入力すると，App Server はユーザネームとパスワードを検証する．その後，ユーザの Android アプリケーションが，ユーザネーム，パスワード，FirebaseToken およびハッシュ化されたユーザネームを App Server に送信する．
(6) App Server は，ハッシュ化されたユーザネーム，Service

ID および FirebaseToken を送信する．

(7) UAF Auth Server は，Service ID から App ID を取得して，Android アプリケーションに通知する．

(8) Android アプリケーション において UAF Auth Server から FacetId を取得して検証する．FacetID とは，クライアントプラットフォームを確認するために利用される．

(9) Android アプリケーションが登録リクエストを送る．

(10)UAF Auth Server が Challenge を生成して，認証器マッチングポリシーと Challenge を Android アプリケーションに送信する．

(11)Android アプリケーション は 認証器マッチングポリシーに基づいて適合する認証器を選択して生体認証を行い，認証器ベンダーの秘密鍵を使用して署名して UAF Auth Server に送る．

(12)UAF Auth Server が認証器ベンダーの公開鍵を用いて署名を検証し，認証用公開鍵を保存する．

(13)App Server は，UAF Auth Server から検証結果を取得する． 検証に成功した場合には登録に成功とみなされる．

3.3 認証

認証の具体的なシーケンスを図 6 に示す．

(1) ユーザは，ブラウザで認証用のログイン画面にアクセスする．

(2) App Server は，ログイン画面を表示する．このページには，UAF 用の JavaScript が含まれる．

(3) ユーザのブラウザが，UAF Auth Server に対して JavaScript の取得を要求する．

(4) UAF Auth Server が JavaScript を返却し，Android アプリケーションで JavaScript が実行される．

(5) ユーザがユーザネームを入力すると，ユーザのブラウザが，ハッシュ化されたユーザネームを App Server に送信する．

(6) App Server は，ハッシュ化されたユーザネームと Service ID を送信する．

(7) UAF Auth Server は，Service ID から App ID を取得して，Android アプリケーションに通知する．

(8) Android アプリケーション において UAF Auth Server から FacetId を取得して検証する．

(9) Android アプリケーションが認証リクエストを送信する．

(10)UAF Auth Server が Challenge を生成して，認証器マッチングポリシーと Challenge を Android アプリケーションに送信する．

(11)Android アプリケーション は 認証器マッチングポリシーに基づいて適合する認証器を選択して生体認証を行い，認証用証明書の秘密鍵を使用して署名して UAF Auth Server に送る．

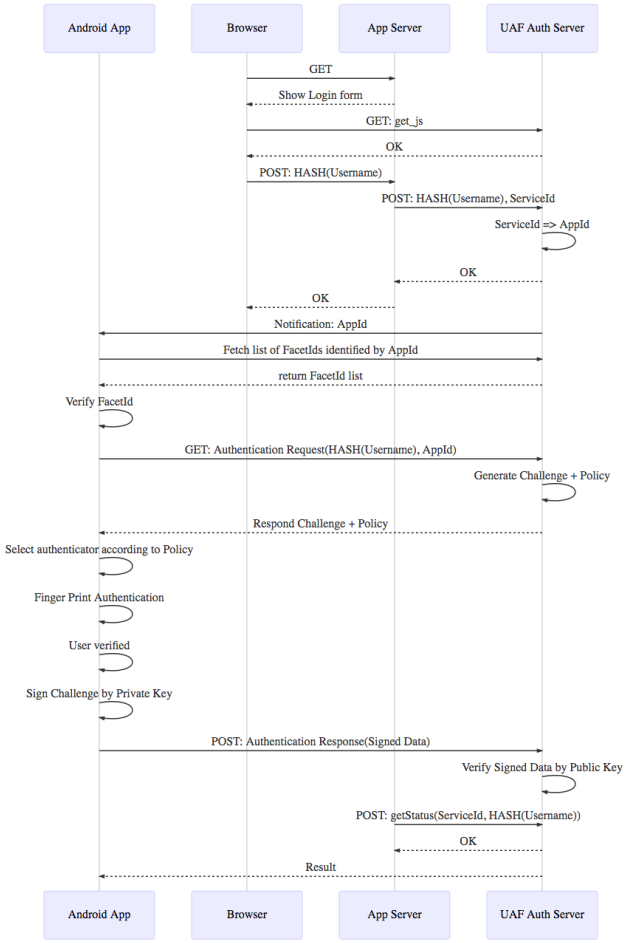

図 6 認証シーケンス

(12)UAF Auth Server が認証用公開鍵で署名を検証する．

(13)App Server が UAF Auth Server から検証結果を取得する．検証に成功した場合には認証に成功となる．

4. 評価

本論文で提案したシステムについて性能評価を行い，実用性を検証した．評価は表 1 に記載する環境で行った．評価には，Android アプリケーションの実行は Android 端末で行った．テストユーザ 2 名により，登録時と認証時の Android アプリケーションと UAF Auth Sever 間の通信速度を 50 回計測した．

表 1 評価環境

環境	使用バージョン
Android アプリケーション	Android 8.0, 8.1 (API Version 26, 27)
クラウド環境	Google Kubernetes Engine
UAF Auth Server	Java 8，Spring BOOT 2.0.3

5. 結果

5.1 登録機能

登録機能に関する結果に関しては，以下の表のようになった．測定したリクエストとは，Android アプリケーションが登録リクエスト（図 5 における Registration Request 部分）して，UAF Auth Sever が Challenge を生成してチャレンジと認証器マッチングポリシーを Android アプリケーションに送信する部分となる．測定したレスポンスとは，署名したデータを Post して，UAF Auth Sever が署名を検証してその結果を Android アプリケーションに送信する部分となる．

表 2 登録機能のリクエスト時間

項目	時間
最小値	368 ms
最大値	644 ms
平均値	446 ms

表 3 登録機能のレスポンス時間

項目	時間
最小値	211 ms
最大値	494 ms
平均値	271 ms

5.2 認証機能

認証機能に関する結果に関しては，以下の表のようになった．測定したリクエストとは，Android アプリケーションが認証リクエストを送信（図 6 における Authentication Request 部分）して，UAF Auth Sever が Challenge を生成してチャレンジと認証器マッチングポリシーを Android アプリケーションに送信する部分となる．測定したレスポンスとは，署名したデータを Post して，UAF Auth Sever が署名を検証してその結果を Android アプリケーションに送信する部分となる．

表 4 認証機能のリクエスト時間

項目	時間
最小値	182 ms
最大値	411 ms
平均値	229 ms

表 5 認証機能のレスポンス時間

項目	時間
最小値	245 ms
最大値	564 ms
平均値	320 ms

6. 考察

性能評価の結果，通信上でエラーが発生しなかった．テストユーザ数が 2 名と制限している環境であるものの，リクエスト・レスポンス時間が 1 秒以内となっており，実用化の可能性があることが分かった．

7. 今後の課題

技術検証の観点では，本論文で提案するの UAF については，FIDO Alliance 認定を受けていないという課題がある．2018 年 11 月に予定されている FIDO Alliance 認定に申請する予定である．また，実用面ではユーザが端末を紛失するなどの例外的な事象が発生したときの対処方法が明確になっていないという課題がある．この課題については，例外的な事象が発生した際の手順を明確にすることによって解決を図りたいと考えている．

参考文献

[1] FIDO Alliance :
Alliance Universal Authentication Framework (UAF),
入手先 ⟨https://fidoalliance.org/specs/fido-uaf-v1.0-rd-20140209.zip⟩ (参照 2018.08.20).

[2] FIDO Alliance : Alliance Universal 2nd Factor (U2F),
入手先 ⟨https://fidoalliance.org/specs/fido-u2f-v1.0-rd-20140209.zip⟩ (参照 2018.08.20).

[3] JPCERT/CC :
適切なパスワードの設定・管理方法について
入手先 ⟨https://www.jpcert.or.jp/newsflash/2018040401.html⟩, (参照 2018.08.20).

[4] Agrawal Arpit, and Ashish Patidar: *Smart Authentication for Smart Phones*, International Journal of Computer Science and Information Technologies 5.4 (2014).

[5] Everts, Maarten, Jaap-Henk Hoepman, and Johanneke Siljee: *UbiKiMa: Ubiquitous authentication using a smartphone, migrating from passwords to strong cryptography*, Proceedings of the 2013 ACM workshop on Digital identity management (2013).

[6] Bogdan-Cosmin Chifor, Sorin Teican, Mihai Togan, George Gugulea: *A Flexible Authorization Mechanism for Enterprise Networks Using Smart-phone Devices*, International Journal of Advanced Computer Science and Applications (2017).

[7] Dropbox :
Introducing U2F support for secure authentication,
入手先 ⟨https://blogs.dropbox.com/dropbox/2015/08/u2f-security-keys/⟩ (参照 2018.08.20).

[8] W3C:
Web Authentication: An API for accessing Public Key Credentials Level 1
入手先 ⟨https://www.w3.org/TR/webauthn/⟩ (参照 2018.08.20).