ISSN 2355-3286

Analisis dan Implementasi Protokol Otentikasi FIDO U2F

Sunderi Pranata[1], Hargyo Tri Nugroho[1], Hirofumi Yamaki[2]
[1] Program Studi Sistem Komputer, Universitas Multimedia Nusantara, Tangerang, Indonesia
[2] School of Information Environment, Tokyo Denki University, Tokyo, Japan
sunderi.pranata@gmail.com[1]

Diterima 5 Mei 2017
Disetujui 16 Juni 2017

***Abstract*— It is known that password itself is not enough for formidable authentication method since it has a lot of vulnerabilities. Multi factor authentication (MFA) is introduced for the next generation for good authentication to address that issue. MFA combines two or more of three principles of good security, "something you know", "something you have", and "something you are". Most MFA mechanisms work as one time passwords (OTP). However, they can still be vulnerable to phishing and MiTM attack. On top of that, OTP can be hard to use as it requires user to input another password given by the device (SMS, token, authenticator). Implemented in small USB U2F device, FIDO U2F delivers easier yet stronger security on authentication process which implements public key cryptography, challenge-response protocol, and phishing and MitM protection.**

***Index Terms*— Authentication protocol, FIDO U2F, Multi factor authentication, OTP**

I. PENDAHULUAN

Kata sandi atau *password* umum digunakan sebagai mekanisme otentikasi di dunia maya. Namun, otentikasi dengan kata sandi sendiri memiliki banyak celah keamanan. Kata sandi yang dirancang dengan sembrono sangat rentan diretas. Bahkan masyarakat masih sering menggunakan kata sandi yang berisi data-data pribadi dan mudah ditebak seperti *username*-nya, nama binatang peliharaan, dan tanggal ulang tahun. Namun kata sandi yang kuat bukan berarti tanpa masalah karena kata sandi yang kuat adalah kata sandi yang umumnya sulit untuk diingat [1].

Masalah kata sandi ini sebenarnya sudah ada solusinya yaitu *Multi Factor Authentication* (MFA). *Multi Factor Authentication* adalah metode otentikasi dengan dua atau lebih dari tiga prinsip dasar otentikasi yaitu *something you know (SYK), something you have (SYH),* dan *something you are (SYA)* [2] [3]. Pada umumnya, bentuk MFA yang paling umum digunakan sampai saat makalah ini ditulis adalah SYK dan SYH yang secara berurutan berupa kata sandi dan *token* atau telepon genggam.

MFA bekerja dengan baik untuk menutupi kelemahan dari kata sandi tetapi masih berisiko terhadap serangan *phishing* dan MiTM. Selain itu, faktor SYH pada MFA biasanya berupa *One Time Password* (OTP). Sayangnya, bagi pengguna awam, OTP umumnya relatif sulit digunakan. Pengguna harus menggenerasi dan meng-*input* OTP sendiri hanya untuk melakukan satu kali proses otentikasi. Prosesnya relatif panjang dan tidak praktis. Hal inilah yang mendasari FIDO Alliance untuk membuat protokol otentikasi yang baru dan lebih aman serta lebih mudah digunakan yang kemudian dinamakan protokol FIDO dan diluncurkan pada Februari 2013 [4].

FIDO bekerja di atas konsep MFA dan *public key cryptography*. Salah satu keuntungan dari FIDO adalah bahwa pengguna tidak perlu kata sandi yang kompleks, pengguna bahkan dapat menggantikan kata sandinya dengan kata sandi yang sederhana seperti PIN 4 digit.

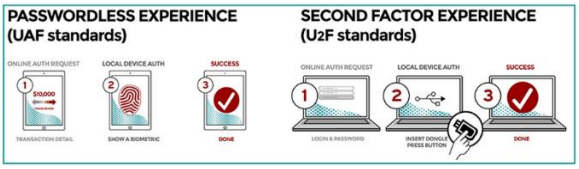
Pada penelitian ini FIDO diimplementasikan pada sebuah sistem yang bersifat *proof-of-concept* baik pada sisi server maupun klien untuk membuktikan kehandalan FIDO sebagai protokol otentikasi.

II. KAJIAN PUSTAKA

A. FIDO

FIDO Alliance mengedepankan tiga aspek utama yaitu 1) kemudahan penggunaan, 2) privasi dan sekuriti, dan 3) standardisasi. Tujuan utama adalah untuk membuat layanan *online* dan *website* baik untuk korporasi atau terbuka, untuk mendukung fitur keamanan yang tertanam secara *native* untuk dukungan fitur keamanan yang baik dan mengurangi masalah yang muncul akibat proses pembuatan dan pengingatan kata sandi yang banyak dan berbeda setiap situs [4].

Protokol FIDO terbagi menjadi dua yang mengakomodasi dua pilihan bagi pengguna untuk berinteraksi dengan otentikasi dunia maya. Dua protokol tersebut memiliki banyak basis protokol yang sama tetapi dibedakan hanya untuk kasus penggunaan yang spesifik. Kedua protokol FIDO itu adalah FIDO U2F dan FIDO UAF [5].



Gambar 1. FIDO U2F dan UAF [6]

Gambar 2. Proses Otentikasi FIDO U2F [7]

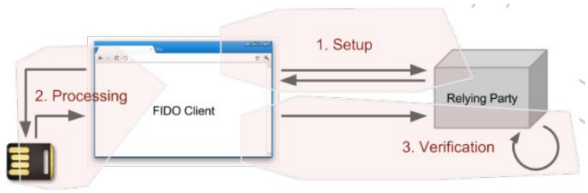
B. *FIDO U2F*

Protokol FIDO U2F memungkinkan layanan online untuk tetap menggunakan infrastruktur layanan kata sandi tetapi dengan menambahkan faktor keamanan kedua yang kuat untuk log in. Layanan online juga dapat meminta pengguna untuk mempresentasikan perangkat faktor kedua kapan pun dibutuhkan. Faktor kedua tersebut memungkinkan pengguna untuk menggunakan kata sandi yang sederhana seperti 4 digit PIN tanpa melemahkan aspek keamanan. Pada saat proses registrasi dan otentikasi, pengguna cukup mempresentasikan perangkat faktor kedua ke komputer hanya dengan gerakan menekan tombol pada perangkat USB atau *tap* NFC. Pengguna dapat menggunakan perangkat FIDO U2F terhadap banyak layanan online yang mendukung FIDO U2F tetapi harus disertai dengan web browser yang mendukung [5].

Dari sisi user, layanan online dapat melakukan proses otentikasi seperti biasa yaitu dengan memasukkan username dan kata sandi. Setelah itu, FIDO U2F akan digunakan sebagai faktor kedua yang memperkuat keamanan otentikasi. Server akan meminta pengguna untuk melakukan gerakan sederhana untuk membuktikan bahwa pengguna ada di tempat. Gerakan tersebut dapat berupa penekanan tombol pada perangkat FIDO U2F, tapping perangkat FIDO U2F, atau pun *plug in* perangkat FIDO U2F. Proses dari otentikasi faktor kedua dengan FIDO U2F jauh lebih sederhana dibanding dengan proses OTP yang mengharuskan proses generasi dan *input* manual kode ke sistem.

C. *FIDO UAF*

Protokol FIDO UAF memungkinkan layanan online untuk menawarkan layanan otentikasi tanpa kata sandi. Pengguna akan meregistrasikan perangkat otentikasi ke layanan online dengan pemilihan mekanisme otentikasi seperti gestur jari pada telepon genggam, melihat ke kamera, berbicara ke mic, memasukkan PIN, dll. Setelah teregistrasikan, pengguna cukup melakukan otentikasi tadi secara lokal kapan pun pengguna membutuhkan untuk otentikasi ke layanan tersebut. Pengguna tidak perlu lagi memasukkan kata sandi apa pun ketika ingin mengotentikasi dari perangkat tersebut. [5]



Gambar 3. Tiga Fase Dari Proses Registrasi Dan Otentikasi [8]

D. *Protokol FIDO U2F*

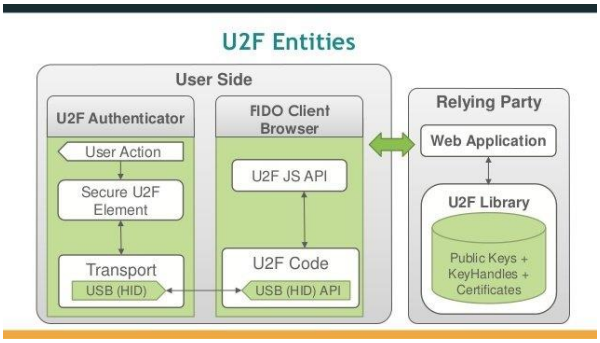
Token U2F menyediakan kriptografik *assertions* yang dapat diverifikasi oleh server dan kriptografi digunakan sebagai faktor kedua (setelah kata sandi) saat proses otentikasi. Token U2F umumnya kecil dan berupa perangkat dengan tujuan khusus yang tidak terkoneksi dengan internet tetapi dapat berkomunikasi dengan server. Maka dari itu, token FIDO U2F memerlukan FIDO *client* untuk me-*relay* pesan antara token dan server. Pada umumnya FIDO *client* berupa web browser.

Protokol FIDO U2F dibagi menjadi dua proses besar yaitu proses registrasi dan proses otentikasi. Proses registrasi akan mendaftarkan pasangan kunci (*key pair*) yang digenerasi atas kontrol token U2F sedangkan proses otentikasi akan membuktikan kepemilikan dari *key pair* yang telah diregistrasikan sebelumnya ke server. Kedua proses tersebut terdiri dari 3 fase yaitu [8]:

1. **Setup**: pada fase ini, FIDO *client* akan mendapatkan *challenge* dari server. Dengan *challenge* tersebut, FIDO *client* akan membuat *request message* kepada token U2F

2. **Processing**: pada fase ini, FIDO *client* akan mengirim *request message* ke token, dan token akan melakukan operasi kriptografi pada *message* tersebut dan membuat *response message*. *Response message* ini kemudian akan dikirim ke FIDO *client.*

3. **Verification:** pada fase ini, FIDO *client* mengirim *response message* dari token bersamaan dengan data-data lainnya yang diperlukan oleh server untuk melakukan verifikasi token *response* ke server. Server kemudian akan memroses *token response* ini untuk membuktikan kebenarannya. Jika benar pada proses registrasi maka server akan mendaftarkan *public key* yang baru untuk user tersebut sedangkan jika benar pada proses otentikasi maka server akan menerima bahwa klien memiliki *private key* yang dibutuhkan dan dapat melanjutkan ke proses selanjutnya.



Gambar 4. Entitas U2F [9]

E. Entitas FIDO U2F

Pada protokol U2F entitas yang penting terbagi menjadi tiga yaitu U2F *authenticator*, browser, dan server.

1. **Server**: server umumnya berupa aplikasi web online. Pada saat proses registrasi, entitas ini akan menggenerasi dan mengirimkan *challenge* ke browser yang meminta dan menerima *public key* yang digenerasikan oleh U2F authenticator, *key handles*, dan *certificates* dan menyimpannya ke dalam basis data. Pada saat proses otentikasi, browser akan membuat *challenge*, kemudian akan menerima *signed-challenge* dari browser dan melakukan *unsign-challenge*-nya dengan *public key* yang telah disimpan sebelumnya. Setelah proses otentikasi selesai, server dapat melanjutkan ke tahap selanjutnya seperti mengatur cookie atau session untuk browser.

2. **FIDO Client Browser**: FIDO *client browser* akan bertindak sebagai jembatan antara U2F *authenticator* dengan server. Dengan menggunakan library U2F javascript API yang telah dikembangkan, browser dapat berkomunikasi dengan U2F *authenticator*. Browser akan mengirimkan *challenge* untuk meminta proses pembuatan *key pairs* untuk proses registrasi maupun meminta proses *sign-the-challenge* untuk proses otentikasi.

3. **U2F Authenticator**: U2F authenticator adalah perangkat kecil USB yang dirancang secara khusus untuk otentikasi. Entitas ini akan menggunakan elemen U2F khusus yang dirancang secara aman untuk membuat *key pairs* dan *key handles* pada saat proses registrasi dan akan melakukan *sign-the-challenge* dengan *private key* yang diregenerasi (akan dijelaskan lebih lanjut pada bagian IV). Semua data tersebut akan dikirim ke browser yang kemudian akan diteruskan ke server.

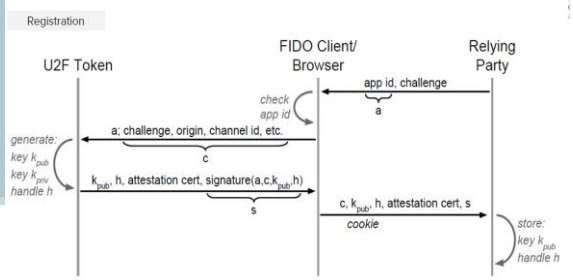
III. METODE

A. Perancangan Sistem

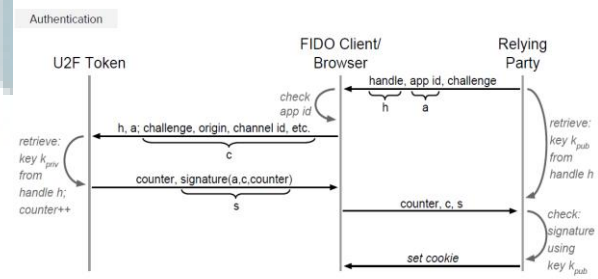
U2F mempunyai dua alir proses yaitu alir proses registrasi dan alir proses otentikasi. Untuk dapat melakukan proses otentikasi dengan U2F, pengguna harus lebih dulu meregistrasikan perangkat U2F authenticator ke server. Berikut alir proses registrasi dan otentikasi:

Kemudian, proses registrasi akan dimulai dengan server melakukan pengiriman *challenge* dan appID ke browser, kemudian browser akan mengecek appID dan mengirim data yang diperlukan ke token U2F. Token U2F akan menggenerasi *key pairs* dan *key handles* berdasarkan data yang dikirim oleh server, kemudian mengirimkan *signed challenge* ke browser beserta *attestation certificate* (optional). Browser kemudian akan mengirim *key handle*, dan *signed challenge* ke server. Server akan kemudian menyimpan *public key* dan *key handle* yang akan digunakan kembali pada proses otentikasi [10].

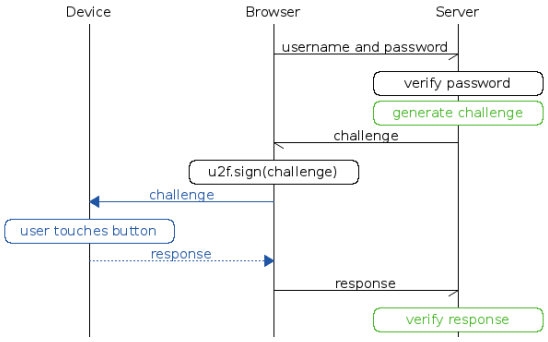
Setelah proses registrasi selesai, pengguna dapat memulai proses otentikasi. Proses otentikasi dimulai dengan pengiriman *key handle*, *appID*, dan *challenge* oleh server ke browser. Browser kemudian akan mengecek appID dan melanjutkan data yang diperlukan ke token U2F. Token U2F akan membuat ulang *private key* dari *key handle* dan appID, kemudian *sign challenge* tersebut dan mengirim hasilnya kembali ke server melalui browser. Server kemudian akan mengecek *signature* menggunakan *public key* yang telah disimpan pada saat proses registrasi sebelumnya.



Gambar 5. Proses registrasi U2F [11]



Gambar 6. Proses Otentikasi U2F [11]



Gambar 7. Diagram Sekuensial FIDO U2F [12]

IV. ANALISIS DAN IMPLEMENTASI

A. *Implementasi*

Pada makalah ini, penulis mengimplementasikan protokol FIDO U2F dengan membuat web server yang berjalan di atas HTTPS. Server dibuat dengan menggunakan bahasa pemrograman Go. Dikarenakan protokol FIDO U2F *web facet* harus berjalan di atas HTTPS, diimplementasikan juga protokol TLS dengan *self-signed certificate*. Pengimplementasian dari sisi klien menggunakan javascript API yang dibuat Google sebagai salah satu anggota dari FIDO Alliance untuk komunikasi antara perangkat U2F dengan browser.

Ada pun diagram sekuensial sederhana yang menggambarkan keseluruhan sistem ditunjukkan pada gambar 7.

Proses implementasi dibagi menjadi dua bagian besar yaitu implementasi sisi server dan implementasi sisi klien.

Pada **implementasi sisi server**, sebelum mengaplikasikan U2F, server harus berjalan di atas HTTPS. Berikut cuplikan kode yang ditulis dengan bahasa pemrograman Go:

```
s:=u2f.StdServer(&userDB{},"https://gou2f.com:8079")
http.ListenAndServeTLS(":8079", "cert.pem", "key.pem", nil)
```

Implementasi server U2F sendiri memiliki 4 fungsi utama yaitu *start registration*, *finish registration*, *start authentication*, *finish authentication*. Pada implementasi sisi server, digunakan library U2F yang mengikuti pseudocode untuk seperti in ditunjukkan pada gambar 8.

Implementasi sisi klien pada paper ini berupa web sebagai agen tengah antara server dan perangkat U2F. Perlu dicatat bahwa hingga saat makalah ini ditulis, tidak semua browser mendukung U2F secara *native*, browser yang mendukung U2F secara *native* salah satu contohnya adalah Chrome versi 38 ke atas. Untuk menggunakan U2F pada browser yang tidak didukung secara *native*, diperlukan instalasi plug in tambahan.

```
# handles HTTPS requests to /start_registration
def start_registration(username):
  challenge = u2f_lib.start_registration(APP_ID)
  challenge_store.set(username, challenge)
  return challenge

# handles HTTPS requests to /finish_registration
def finish_registration(username, device_response):
  challenge = challenge_store.pop(username)
  registered_device = u2f_lib.finish_registration(challenge, device_response)
  device_store.set(username, registered_device)
  return "Success!"

# handles HTTPS requests to /start_authentication
def start_authentication(username, password):
  verify_user_pass(username, password)
  registered_devices = device_store.get(username)
  challenge = u2f_lib.start_authentication(registered_devices, APP_ID)
  challenge_store.set(username, challenge)
  return challenge

# handles HTTPS requests to /finish_authentication
def finish_authentication(username, password, device_response):
  challenge = challenge_store.pop(username)
  u2f_lib.finish_authentication(challenge, device_response, registered_devices)
  return "Success!"
```

Gambar 8. Pseudocode Proses Registrasi Dan Otentikasi FIDO U2F [12]

FIDO Alliance telah membuat library pendukung untuk sisi klien dalam bentuk javascript API (u2f-api.js) yang pada intinya berisikan dua fungsi utama yaitu: **u2f.register**, yang digunakan pada saat meregistrasikan perangkat U2F; dan **u2f.sign** yang digunakan pada saat mengotentikasi perangkat U2F [13].

Berikut cuplikan kode yang ditulis dengan bahasa pemrograman Go:

```
<script src="https://demo.yubico.com/js/u2f-api.js"></script>
<script>
(function() {
  window.u2fRegister = u2fRegister;
  window.u2fSign = u2fSign;

  function ajax(url, args, cb) {
    var aj = new XMLHttpRequest();
    aj.onreadystatechange = function() {
      if (aj.readyState == 4) {
        if (aj.status == 200) {
          cb(JSON.parse(aj.responseText));
        } else {
          msg("failed: " + aj.responseText);
        }
      }
    }
    aj.open('POST', url, true);
    aj.setRequestHeader('Content-type', 'text/json')
    aj.send(JSON.stringify(args));
  }
  function u2fRegister() {
    ajax("/Register", {}, function(r) {
      msg("touch it to register");
      console.log('registerChallenge',r);
      u2f.register([r], [],function(response) {
        console.log('registerSigned',response);
        if (response.errorCode) {
          msg("failed to enroll:" + response.errorCode);
          return;
        }
        msg("finalizing/validating registration...");
        u2fRegisterFin(response);
      });
```

```
    });
  }
  function u2fRegisterFin(rr) {
    ajax("/RegisterFin", rr, function(r) {
      msg("device registered");
    });
  }
  function u2fSign() {
    ajax("/Sign", {}, function(r) {
      msg("touch it to sign in");
      console.log('authenticationChallenge', r);

    u2f.sign(r, function(response) {
      console.log('authenticationSigned',response);
      if (response.errorCode) {
          msg(response.errorCode);
          return;
      }
      msg("verifying");

      u2fSignFin(response);
    }, 5);
  });
}

function u2fSignFin(verify) {
  ajax("/SignFin", verify, function() {
    msg("logged in");
  });
}
function msg(m) {
            var e = document.getElementById("msg");
            e.innerHTML = 'HI: ';
            e.innerText = m;
}
}());
</script>
```

B. Analisis Protokol

Protokol U2F berbasis protokol *challenge-response* yang dibuat agar memiliki proteksi phishing dan MitM. Protokol U2F juga dibuat agar memiliki *key* yang unik terhadap situs (*application specific keys*), *device cloning detection*, dan *device attestation*.

1. ***Challenge-response****:* otentikasi berbasis *challenge-response* ini didasari oleh *public key cryptography*. Perangkat U2F memiliki *key* k_{priv} dan server memiliki *key* k_{pub}. *Key pair* ini digenerasi di dalam perangkat dengan operasi yang *tamper-resistant*, di mana $\boldsymbol{k_{priv}}$ **tidak dapat diekstrak dari perangkat** [10]**.** K_{priv} hanya berupa fungsi pada hardware dan didesain agar tidak dapat diekstrak dan hanya dapat digunakan langsung oleh proses yang telah ditentukan seperti proses *signing*.

2. **Proteksi dari phishing dan MitM**: konsep ini membutuhkan informasi dari koneksi HTTP yang sedang berlangsung (URI dan TLS channel ID). Informasi ini kemudian di-*sign* oleh perangkat U2F dan kemudian dikirim ke server, yang akan memverifikasi bahwa informasi tersebut adalah benar [10]. Proses ini menambahkan informasi origin (protocol, hostname, dan port number) yang akan menggagalkan phishing serta TLS channel ID yang akan menggagalkan MitM.

3. ***Application Specific Keys:*** *application specific keys* ditujukan agar server tidak dapat mengetahui bahwa digunakannya perangkat U2F yang sama pada dua akun yang berbeda. Sebagai contoh, example.com tidak akan tahu bahwa user1 dan user2 menggunakan perangkat yang sama. Perangkat U2F akan menggenerasi *key pair* dan *key handle* baru untuk setiap proses registrasi. *Key handle* akan disimpan oleh server dan dikirimkan kembali ke perangkat U2F saat proses otentikasi. Dengan ini, perangkat tahu *key* mana yang harus diotentikasi. Untuk mencapai tujuan ini, dibutuhkan *key handle* dan *appID* (untuk membantu menemukan key handle. Ada pun proses pembuatan *key handle* sebagai berikut (perangkat Yubikey karena hingga saat penulisan makalah, belum ada spesifikasi cara menyimpan *keys*).

Ketika pengguna meregistrasikan perangkat U2F, server akan menyediakan AppID (berupa URL situs dan mencegah phishing). Perangkat U2F akan kemudian menggenerasi nonce (N). Yubikey kemudian akan menggunakan AppID dan nonce untuk diproses melalui fungsi HMAC-SHA256, menggunakan *device-specific secret* sebagai *key*. *Device-specific secret key* ini digenerasi *on-chip* pada saat proses manufaktur. Keluaran dari proses tersebut adalah *private key*, dan nonce bersamaan dengan Message Authentication Code (MAC) akan menjadi *key handle*. MAC berfungsi untuk memastikan bahwa *key handle* tersebut hanya valid untuk kombinasi khusus antara perangkat U2F dan appID. Pada saat otentikasi, *key handle* akan dilewatkan pada yubikey lagi dan diverifikasi bahwa *key handle*-nya menggunakan MAC bahwa *key handle*-nya belum diubah dan bahwa *credential*-nya benar sesuai dengan appID. menggunakan MAC [14].

4. Device cloning detection*:* Perangkat U2F memang *tamper-resistant* dan k_{priv} tidak dapat diekstrak. Namun, untuk menyediakan *cloning detection* pada perangkat U2F yang tidak memiliki perangkat *tamper-resistant* (contoh: implementasi dengan software), protokol ini dirancang dengan penambahan *counter*. Konsepnya sederhana, perangkat akan menambah *counter* setiap kali proses otentikasi dan server akan memverifikasi bahwa *counter* tersebut bernilai lebih tinggi dari yang sebelumnya [10].

5. Device attestation: proses pengesahan memungkinkan server untuk memverifikasi properti perangkat U2F, seperti nomor model perangkat U2F. Proses ini diimplementasikan melalui *attestation certificate*, di-*sign* oleh *device vendor*, dan perangkat mengirimkan data tersebut ke server pada saat registrasi. Proses ini tidak berpengaruh pada proses otentikasi dan bersifat optional [10].

V. SIMPULAN

Implementasi FIDO U2F pada makalah ini masih sebatas *proof of concept* dan masih jauh dari solusi

yang sempurna yang dapat diterapkan. Namun FIDO telah terbukti mudah dan handal untuk digunakan. Ada pun implementasi yang dapat ditambahkan pada masa yang akan datang antara lain penambahan fitur otentikasi pengguna dan kata sandi sebelum otentikasi U2F; penambahan fitur registrasi metode otentikasi pengguna; penambahan fitur penyimpanan data pengguna ke basis data permanen (pada implementasi makalah ini, digunakan memori untuk menyimpan data); maupun pengaturan *cookie* dan *session* untuk klien.

DAFTAR PUSTAKA

[1] K. Chanda, "Password Security: An Analysis of Password," *I. J. Computer Network and Information Security*, 2016.

[2] J. Anita, L. Kaspars dan I. Saudinis, "Multi factor authentications a necessary solution in the fight with information technology security threats," dalam *10th International Scientific and Practical Conference*, Rezekne, 2015.

[3] Wikimedia Foundation, "Multi-factor authentication," 8 December 2016. [Online]. Available: https://en.wikipedia.org/wiki/Multi-factor_authentication.

[4] Wikimedia Foundation, "FIDO Alliance," 10 12 2016. [Online]. Available: https://en.wikipedia.org/wiki/FIDO_Alliance.

[5] FIDO Alliance, "Universal 2nd Factor (U2F) Overview," 2014.

[6] FIDO Alliance, "Specifications Overview," 2014.

[7] Yubico, "FIDO U2F (UNIVERSAL 2ND FACTOR)," [Online]. Available: https://www.yubico.com/about/background/fido/. [Diakses 05 Mei 2017].

[8] FIDO Alliance, "FIDO U2F Raw Message Formats," 2014.

[9] FIDO Alliance, "FIDO U2F Specifications: Overview & Tutorial," 07 April 2016. [Online]. Available: https://www.slideshare.net/FIDOAlliance/fido-u2f-specifications-overview-tutorial. [Diakses 05 Mei 2017].

[10] Yubico, "U2F Technical Overview," [Online]. Available: https://developers.yubico.com/U2F/Protocol_details/Overview.html.

[11] FIDO Alliance, "FIDO U2F & UAF Tutorial," 24 Maret 2016. [Online]. Available: https://www.slideshare.net/FIDOAlliance/fido-u2f-uaf-tutorial. [Diakses 05 Mei 2017].

[12] Yubico, "Using a U2F library," [Online]. Available: https://developers.yubico.com/U2F/Libraries/Using_a_library.html.

[13] FIDO Alliance, "FIDO U2F JavaScript API," 2016.

[14] Yubico, "Key generation," [Online]. Available: https://developers.yubico.com/U2F/Protocol_details/Key_generation.html.