Secure Two-Factor Authentication with SwissPass Crypto Card: A Case Study

Annett Laube*, Reto Koenig†
Bern University of Applied Sciences (BFH), Institute for ICT Based Management
Höheweg 80, CH-2502 Biel/Bienne
Email: *annett.laube@bfh.ch, †reto.koenig@bfh.ch

***Abstract*—Two-factor authentication requires two pieces of independent evidence, mostly one based on possession and the second based on knowledge. The major drawback of these methods are usability and the costs to procure and (re)place the hardware token. The SwissPass is a contactless crypto card mainly used to inspect travel tickets (GA and Half-Fare travel cards) by the Swiss federal railways. This paper presents an authentication protocol using the widely spread SwissPass that allows to log in into web and mobile applications in a secure and intuitive way via smart phone. The protocols are further developed to create the SwissPass Authenticator providing federated authentication on the smart phone.**

***Keywords*—Authentication; mobile applications; smart cards; privacy**

I. Introduction

More and more users, but also resource providers, are worried about insecure authentication protocols. Simple user name/password combinations are not longer sufficient to maintain an adequate level of security for internet business. Two-factor authentication should be considered the minimum acceptable level of access control.

The most commonly used two-factor authentication (2FA) protocols, due to their high usability and availability on all platforms, are based on using SMS verification. Most of them are proofed to be vulnerable. America's National Institute for Standards and Technology has advised abandonment of SMS-based two-factor authentication. In [1] is noted:

> Out-of-band authentication using the PSTN (SMS or voice) is discouraged and is being considered for removal in future editions of this guideline.

The alternatives, using hard tokens, have some drawbacks. The first issue is costs. It is expensive to procure and (re)distribute tokens to a large (enough) number of people. Also the costs for maintenance, support and training are not negligible. Besides the costs, usability, especially when using smart phones, is often a problem. The third point is availability: people tend to loose or to forget their tokens and are than unable to authenticate.

The SwissPass card [2] is a widely spread smart card in Switzerland, introduced in August 2015 and today used by estimated 2 to 3 million travelers [3][1]. The SwissPass' main function is the inspection of travel tickets (GA and Half-Fare travel cards) by the Swiss federal railways and other Swiss transport companies.

The wide distribution, its built-in cryptographic and RFID functions make the SwissPass an ideal possession based factor for a two-factor authentication. This paper shows, how the SwissPass can be used in a user friendly and secure 2FA to sign in into mobile and web applications. In a second step, the SwissPass Authenticator providing a federated authentication, comparable to Google Authenticator, is proposed.

The paper is structured as follows: After this introduction the related work is discussed in Section II. The system context and the main assumptions are presented in Section III. In Section IV, we describe the developed authentication protocols, incl. federated authentication. A security discussion follows in Section V. The implementation is shortly discussed in Section VI. The paper concludes in Sect. VII.

II. Related Work

Beside the (insecure) SMS based 2FA already mentioned, only a few 2FA protocols are suited for smart phones. Attacks on these protocols are described e.g. in [4]–[7].

The most prominent are *Google Authenticator* [8] and *Authy* [9] which offer both two-step verification services using the Time-based One-time Password Algorithm (TOTP) [10] and HMAC-based One-time Password Algorithm (HOTP) [11] after a SMS or voice based registration phase. The 2FA is also available with no internet connection and can be federated into other mobile and web applications. These apps store a generated shared secret on the smart phone, that is needed together with the device when the user authenticates. Known attacks include phishing and the extraction of the secret which is stored as plain text on the smart phone.

Yubico is currently piloting a mobile client, enabling Universal 2nd Factor (U2F) [12] crypto to be integrated directly into mobile applications. The NFC enabled *YubiKey NEO* (containing the secret key) can be used as a second authentication factor in online or offline scenarios. The advantages of a hardware token are 1) no manual typing of codes (only scanning the token), 2) no trust in token providers (which generate and distribute the secret key over a network) needed, 3) malware infection of the tokens is not possible. But the Yubikeys are not free of charge and some processes, like backup or replacement, are cumbersome.

Blockchain-based solutions [13] provide digital tokens over a blockchain, which contain personal information of the user.

[1]In 2016, 8.4 million people lived in Switzerland and roughly a third of them having a SwissPass.

Upon providing this token, by using it to digitally sign the login process, the user can authenticate to web and mobile applications. Companies like, SecureKey or BitID, are currently working on these decentralized identity solutions which promise to be highly usable, secure and privacy protecting.

III. System Context and Assumptions

Fig. 1 shows the four roles (following the definition in OAuthV2.0 [14, Section 1.1] participating in the federated authentication protocols. The end-user wants to access a protected resource hosted by the resource provider, in the example a *webshop*. The user is using a smart phone application, the *Webshop App (WA)*, to request the access. The *Webshop App* enables the end-user to register, to log in, to view and to order some products. The WA delegates the authentication to the *SwissPass Authenticator App (SAA)*. The SAA, which has to be also installed on the end-user's smart phone, provides the two-factor authentication based on the SwissPass card and a password. The SAA is communicating with the authorization server, the *SwissPass Authentication Service* (SAS), to identify and to authenticate the end-user.

The SwissPass card is a contactless RFID smart card hosting a ISO 14443A [15] and a ISO 15693 chip. The ISO 14443A chip contains a built-in private key and is able to sign a challenge using ECDSA with SHA-256. On request, it returns a compressed certificate $Cert_{SC}$, containing a unique identifier, called *MediumID*, the public key and a signature of the certification authority (CA), who issued the card. The ISO 144443A chip supports NFC and can be read with any NFC capable smart phone.

All communications between the smart phone apps and the backend services (Webshop, SwissPass Authentication Service) use transport layer security (TLS) to be protected against eavesdropping.

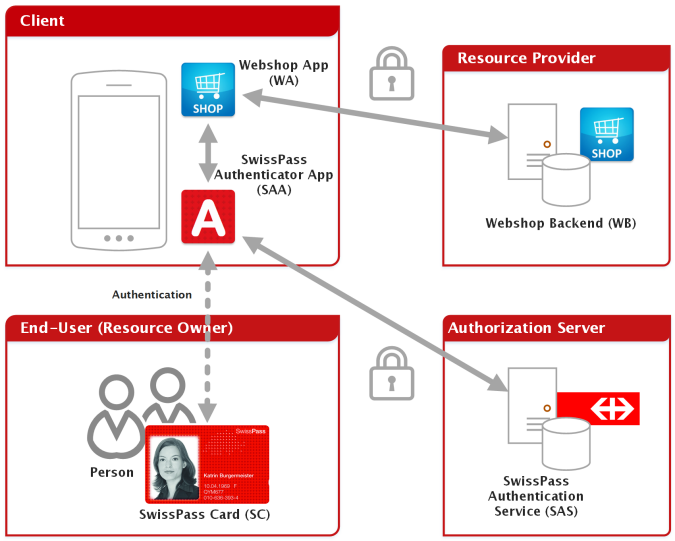

Fig. 1. System context.

The concept rests on the following assumptions:

- Each SwissPass is unique and contains a unique key pair (private and public key) and a unique identifier, called *MediumID*.
- Each user can be uniquely identified by the SwissPass Authentication Service. It is ensured that each user has only one valid SwissPass at a time.
- It is ensured that the SwissPass card is only provided to the correct person (either delivered in person or by registered letter).
- The SwissPass Authentication Service is a trustful entity. It has a database with the public key and MediumID of all valid and invalid SwissPass (e.g. revoked or expired) and their related user.
- If a user does not register their SwissPass within 2 weeks upon reception, the SwissPass is suspended from the usage with the SwissPass Authentication Service.
- The SwissPass Authenticator App running on the smart phone is considered as secure and not manipulable.

IV. Protocols

We distinguish between two processes: (1) the one-time *SwissPass Registration* which can be achieved via the SwissPass Authenticator App (SAA) (Section IV-A) or a Web Application, e.g. the SwissPass Authenticator Portal (SAP) (Section IV-B); and (2) the *Authentication* directly in the SAA (Section IV-C), in the SAP (Section IV-D) or federated in a webshop consisting of Webshop App (WA) and Webshop Backend (WB) (Section IV-E).

A. SwissPass Registration SAS using SAA

Use Case: Before a user can enjoy the SwissPass two-factor authentication, the registration using the SwissPass Authenticator App has to be done. During the one-time registration process, the user has to scan their SwissPass, must select a password and has to provide an e-mail address. The registration is finished when the user has confirmed their e-mail address (which is later only used to reset the password).

Sequence: In Fig. 2, the entire registration protocol is depicted. It consists of the following steps[2]:

1) The user starts the SwissPass Authenticator App (SAA).
2) The SAA requests the *login* endpoint of the SwissPass Authentication Service (SAS):
   `GET https://SERVER/api/login`
3) The SAS generates the $Nonce_{SAS}$ and encloses it in the JSON WebToken $Token1FA_{SAS}$. The token is valid for 5 minutes and signed by the SAS. The token has the following payload:

```
{
  ''iss'': ''swisspass.ch'',
  ''sub'': ''login_1FA'',
  ''nonce'': ''9beb4efa067fda4f66540a9935...'',
  ''iat'': 1482939481,
  ''exp'': 1482939781
}
```

[2]All protocols in the paper contain only the happy path.

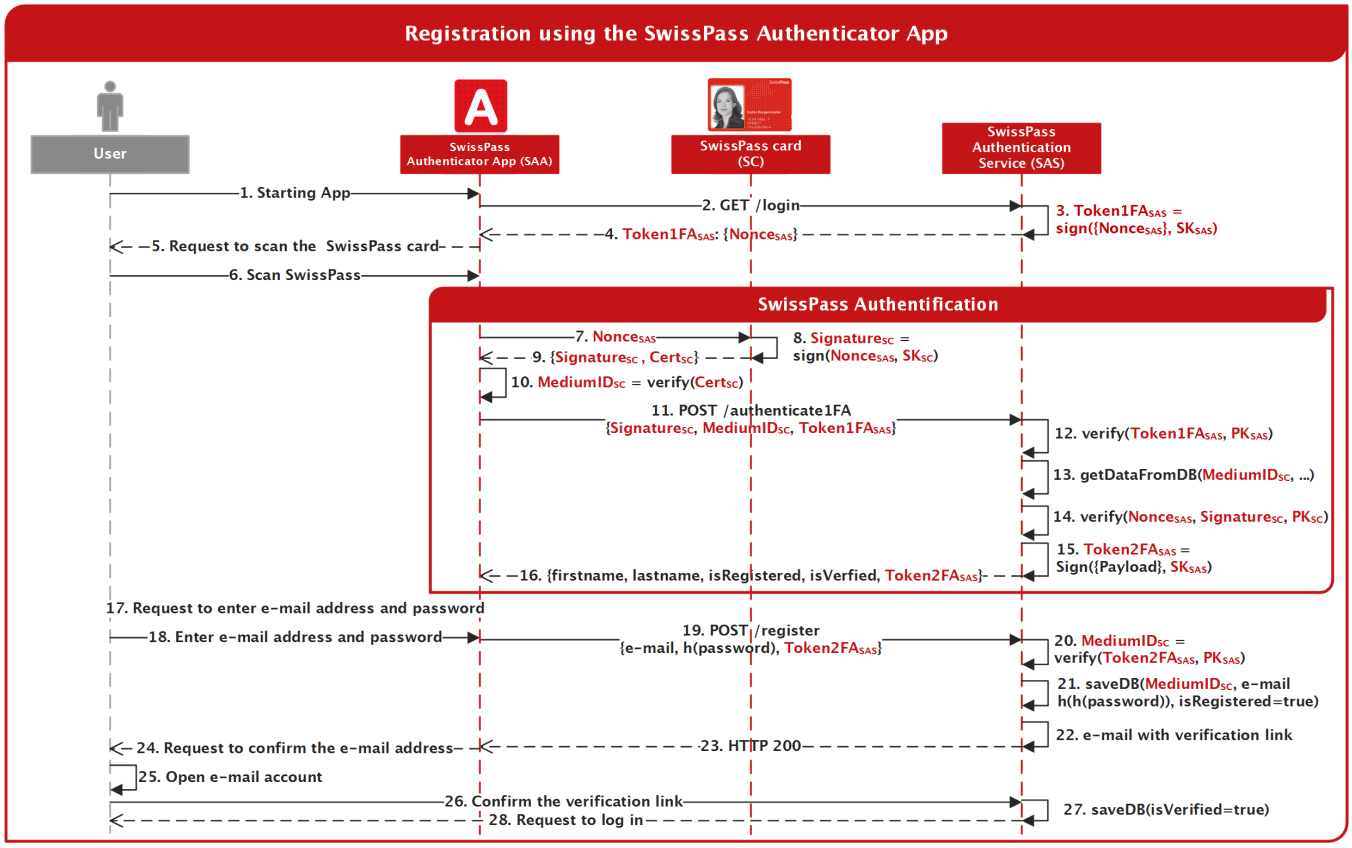

Fig. 2. Registration protocol.

4) The SAS returns the token $Token1FA_{SAS}$ to the SAA.
5) The SAA requests the user to scan their SwissPass.
6) The user scans their card using the NFC reader of the smart phone.
7) During the scan, the SAA communicates with SwissPass card (SC). It sends the token $Nonce_{SAS}$ to the card.
8) The SC signs the token with its private key (SK_{SC}).
9) The SC sends the $Signature_{SC}$ and the certificate $Cert_{SC}$ back to the SAA.
10) The SAA checks the CA signature of $Cert_{SC}$ and extracts the $MediumID_{SC}$.
11) Now, the SAA sends a second request to the SAS's *Authenticate1FA* endpoint including the $Token1FA_{SAS}$, $Signature_{SC}$ and $MediumID_{SC}$.

```
POST https://SERVER/api/authenticate1FA
{
  ''token'': ''eyJhbGciOiJSUzUxMiIsInR5cCI6
      IkpXVCJ9.QnrfMZOuAgL260vw...'',
  ''mediumid'': ''275245
      C44CF4EFFE4F04B09B19D1CD0C'',
  ''signature'': ''30450220684
      B7ED9A3BAF2138E7C...''
}
```

12) The SAS checks the $Token1FA_{SAS}$: it verifies the signature and checks the claims *iss* (issuer), *exp* (validity) and *sub* (subject).
13) If the token was valid, the first name, the last name and the card's public key (PK_{SC}) is retrieved from the SAS's database using the MediumID as primary key.
14) The signature $Signature_{SC}$ from the SC is verified with PK_{SC}.
15) To get the second authentication factor (password), the SAS creates a second token $Token2FA_{SAS}$ with the following payload:

```
{
  ''iss'': ''swisspass.ch'',
  ''sub'': ''register'',
  ''mediumid'':
        ''275245C44CF4EFFE4F04B09B19D1CD0C'',
  ''iat'': 1482939790,
  ''exp'': 1482940090
}
```

16) The SAS returns the token $Token2FA_{SAS}$ together with first and last name of the user and the two flags *isRegistered* and *isVerified* to the SAA.

```
{
  ''jwt'': ''eyJhbGciOiJSUzUxMiIsInR5cCI6
      IkpXVCJ9.eyJpc3MiOiJzd2lzc3Bh...'',
  ''firstname'': ''Urs'',
  ''lastname'': ''Kuettel'',
  ''isRegistered'': false,
  ''isVerified'': false
}
```

17) The SAA checks the flags *isRegistered* and *isVerified*. If not, it requests the user to enter an e-mail address and a password.
18) The user does so.

19) The SAA creates a salted hash of the entered password (client-side hashing) and sends it together with the password and $Token2FA_{SAS}$ to the *register* endpoint of the SAS.

```
POST https://SERVER/api/register
{
 ''token'': ''
      eyJhbGciOiJSUzUxMiIsInR5cCI6IkpXVCJ9.
      eyJpc3MiOiJzd2lzc3Bh..."',
 ''email'': ''user@domain.ch",
 ''password'': ''5994471abb01112afcc18159f6
      ..."
}
```

20) The SAS checks the $Token2FA_{SAS}$: it verifies the signature and checks the claims *exp* (validity) and *sub* (subject). The SAS extracts the MediumID from the token $Token2FA_{SAS}$ and passes it to the next step.
21) The SAS salts and hashes the received password hash (server-side hashing) and stores it together with the e-mail address in its database. The value of the flag *isRegistered* is set to *true* in the database using the extracted MediumID as primary key.
22) The user is now registered, but has yet to confirm their e-mail address. The SAS generates a link containing the token $TokenVerify_{SAS}$ which is sent to the user per e-mail.
23) The SAS returns HTTP 200 (OK) to the SAA.
24) The SAA requests the user to go to their e-mail account and to confirm the link.
25) The user goes into their e-mail account.
26) The user clicks on the provided link and in the browser the following URL is opened: `https://SERVER/api/verify?token=eyJhbGciOiJSUzUxMiIsInR5cCI6IkpXVCJ9.ey...`
27) The SAS verifies the contained token $TokenVerify_{SAS}$ and changes the value of the flag *isVerified* to *true* in the database.
28) The user gets a message that the SwissPass authentication has been activated.

Remarks/Features:

- The SAA is mainly used to establish a secure channel (based on TLS) between the SAS and the SwissPass. No data is persistently stored in this app.
- The mentioned tokens $Token1FA_{SAS}$ and $Token2FA_{SAS}$ are excepted only once in the SAS to avoid replay attacks (Step 12).
- The signed nonce $Nonce_{SAS}$ ensures that the $Signature_{SC}$ and $MediumID_{SC}$ belong together. It is verified that the nonce was signed by the private key (SK) belonging to the public key found in the database to the given $MediumID_{SC}$ (Step 14).
- $Token2FA_{SAS}$ links the two factors of the authentication protocol together. Only a user with such a valid token can assign a password and an e-mail address to the referenced SwissPass.
- Because nothing is stored in the SAA or on the smart phone, the SwissPass 2FA will work with any other NFC enable device as well. There is no need for backups and a device replacement is possible without any effort.

B. SwissPass Registration SAS using SAP

Use Case: The second possibility to make the SwissPass Registration is via a Web application, e.g. the SwissPass Authenticator Portal (SAP). The process is very similar to the registration using the SwissPass Authenticator App (SAA) shown Section IV-A).

Sequence: The main difference to the SwissPass registration using SAA is the transfer of the nonce generated by the SAS to the SAA via a QR code which the user has to scan with the SAA installed on the smart phone. The QR code contains the token $Token1FA_{SAS}$. After receiving the token via the QR code, the protocol continues with Step 5 from Sect. IV-A.

C. Authentication SAA

Use Case: After a successful SwissPass Registration, the user can now log into the SAA presenting their SwissPass and the password. The SAA provides e.g. the functionality to reset the password, to change the email address or to report the loss of their SwissPass card (see Fig. 5). This use case shows the functionality of the SwissPass 2FA built in in a mobile application.

Sequence: In Fig. 3, the entire authentication protocol is depicted. It consists of the following steps:

1) Step 1 to 16 are identical to Step 1 to 16 of the SwissPass Registration Protocol in Section IV-A.
17) The SAA checks the flags *isRegistered* and *isVerified*. If so, it requests the user to enter their password.
18) The user does so.
19) The SAA creates a hash of the entered password and sends it together with $Token2FA_{SAS}$ to the *register* endpoint of the SAS.
20) The SAS checks the $Token2FA_{SAS}$: it verifies the signature and checks the claims *exp* (validity) and *sub* (subject). The SAS extracts the MediumID from the token $Token2FA_{SAS}$ and passes it to the next step.
21) The SAS compares the password hash with the one found in the database associated to the extracted MediumID.
22) The SAS generates the token $Token_{SAS}$. With this bearer access token (like defined in [14]) the user is authenticated and access to the resources (user data) of the SAS is granted.
23) The SAS returns the token $Token_{SAS}$ to SAA.
24) The SAA can now access the SAS' resources (user data) when presenting the $Token_{SAS}$.

Remarks/Features:

- Note that the SwissPass Authentication (framed in Fig. 2 and 3) is the same in all presented protocols.
- With $Token2FA_{SAS}$ is ensured that only the user, who is in possession of the card and who knows the password gets access to the resource. The token links the two factors of the authentication protocol.

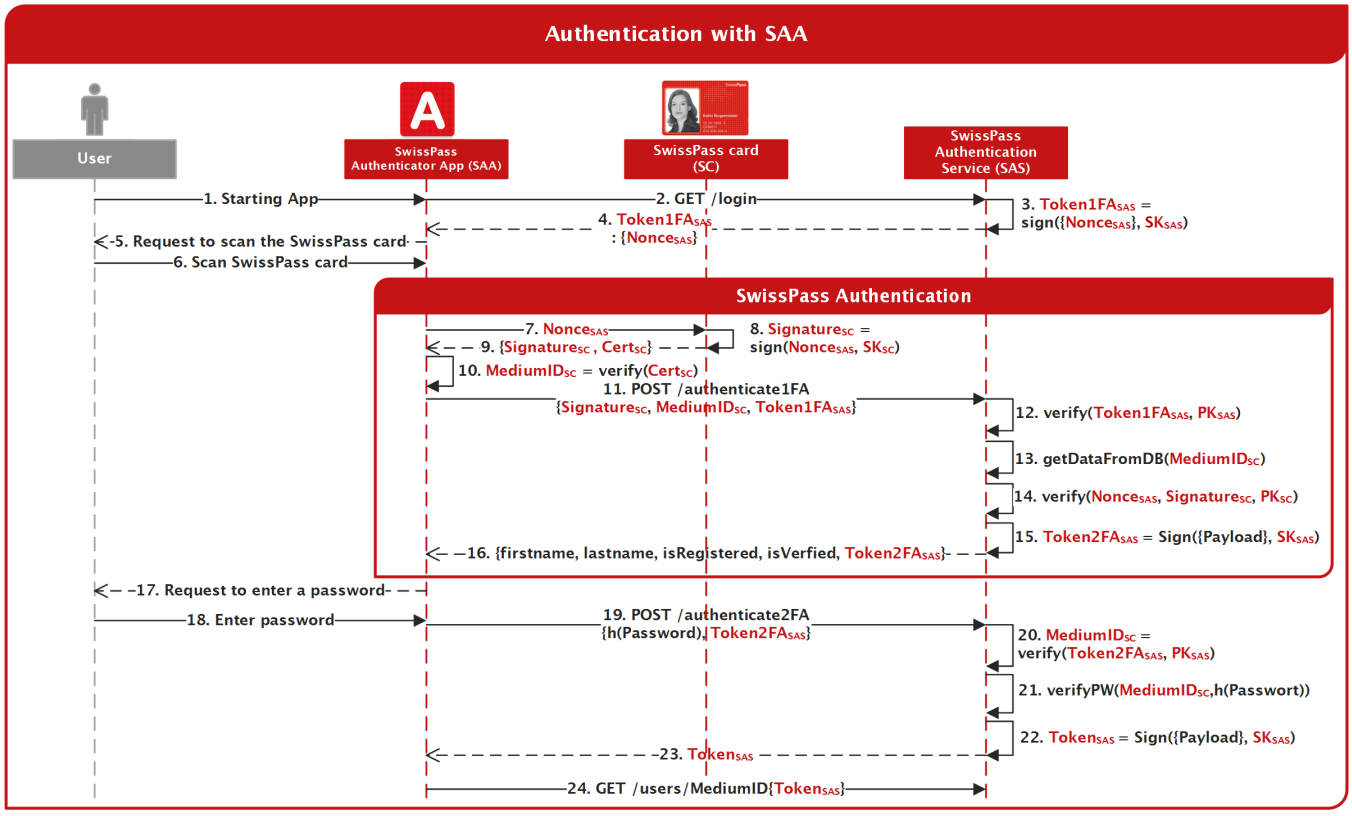

Fig. 3. Authentication protocol.

- The $Token_{SAS}$ is a short-lived access token to ensure that the user is aware of the SwissPass Authentication, esp. in a federated context (see Section IV-E).

D. Authentication SAP

Use Case: The authentication in a Web application, e.g. the SwissPass Authenticator Portal (SAP), on a second device (laptop) is very similar to the authentication in the SAA (Section IV-C). The smart phone is used as NFC card reader and to establish the secure channel from the SwissPass card to the SAS.

Sequence: Like in the SwissPass registration using SAP, the nonce generated by the SAS is transferred to the SAA via a QR code, which the user has to scan with the SAA installed on the smart phone. In parallel, the SAP sends an Access Token Request to the SAS to get the $Token_{SAS}$. The token request includes the nonce formerly transferred to the SAA to uniquely identify the authentication process processed by SAA/SAS. After the successful verification of the two factors (SwissPass and password) the user is authenticated and the SAS returns the access token $Token_{SAS}$.

E. Federated Authentication

Use Case: The user wants to buy some products in a webshop using the smart phone. The resources of the webshop located in Webshop Backend (WB) can be accessed using the Webshop App (WA), that integrates the SwissPass Authenticator to register and to authenticate. This use case shows the functionality of the federated SwissPass 2FA provided by the SAA (SwissPass Authenticator App).

Sequence: The proposed protocol (see Fig. 4) consists of the following steps:

1) The user starts the Webshop App (WA).
2) The WA clicks on the SwissPass Authenticator Login which requests the *login* endpoint of the Webshop Backend (WB).
3) The WB generates the $Nonce_{WB}$ and encloses it in the JSON WebToken $Token_{WB}$. This token has the same content and validity as the token $Token1FA_{SAS}$ from Sect. IV-A, but is signed by the private key of WB.
4) The WB returns the token $Token_{WB}$ to the WA.
5) The WA transfers the token to the SAA.
6) Now the SAA authenticates the user, who has to present their SwissPass and to enter the password. The Steps 5 - 23 of the protocol depicted in Fig. 3 are executed. The only difference is the use of the token $Token_{WB}$ instead of $Token1FA_{SAS}$. In Step 12, the SAS verifies that $Token_{WB}$ was issued by a registered resource provider. This verification is optional and only necessary in closed domains. In Step 14, when the $Token2FA_{SAS}$ is built, $Token_{WB}$ is included in the payload and signed by the SAS. In Step 22, after the successful verification of the users' password, the $Token_{WB}$ is transferred into $Token_{SAS}$.
25) The SAA transfers the $Token_{SAS}$, which contains

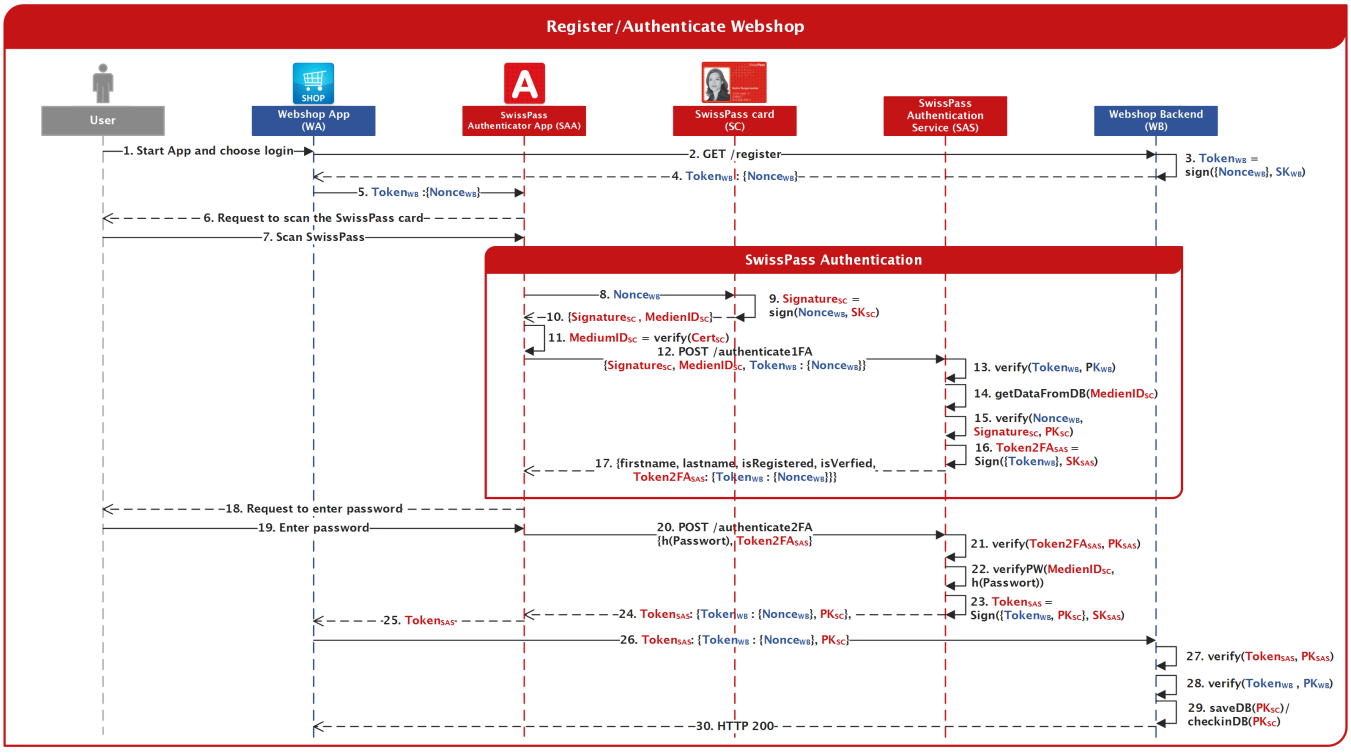

Fig. 4. Federated authentication protocol.

$Token_{WB}$ signed by SAS to the WA.

26) The WA sends this token to the WB.
27) The WB verifies the signature and the validity of token $Token_{SAS}$ to ensure that it is a valid token from the SAS.
28) The WB extracts the $Token_{WB}$ and verifies it. If the validity and signature of the token are correct, the user has successfully been authenticated and can now get access to the resources of the WB.
29) If the user visits the webshop for the first time a user account is created and the public key of the user is stored for identification. Otherwise the user's data are retrieved.
30) The WA returns HTTP 200 (OK) to the WA. The user is now authenticated in the webshop and can fulfill their business.

Remarks/Features:

- The nonce generated by the WB links the access token request (Step 2) to the response with $Token_{SAS}$ (Step 25) from the WA (to avoid reuse).
- The authentication protocol discloses no user information, e.g. first and last name with regard to the WA/WB. The information retrieved from the SAS is only used in the SAA. Only the public key is disclosed to WA/WB (and the information the user enters proactively in the WA) allowing the user to be anonymous to the WB. The transferred public key (which can be salted and hashed) acts as persistent identifier.
- Any resource provider can integrate the SwissPass Authenticator and send an access token request to the SAA. No registration is necessary (except for closed domains). In this way, anyone can get an access token proofing a successful two factor authentication and can verify that no personal information of the user is provided.
- To allow the re-identification of the user when their SwissPass (incl. the public key) has changed, the SAS can transmit the last 5 public keys of the user in $Token_{SAS}$.

V. Security Discussion

In general, the cryptographic means are considered to hold against the polynomially time/space bound adversary.

A. Specific Security and Trust Assumptions

For the client side, the following security assumptions must hold in order to render the SAA usable: The platform must provide a secure interface between the SAA and the user i.e. secure display and secure keyboard. For the server side, ultimate trust in availability and correct verification has to be given to the SAS. Even though these requirements are heavy in nature, many other existing apps only work under the very same security assumptions (e.g. e-banking apps).

B. Adversary Settings

SwissPass Authentication Service (SAS), SwissPass Authenticator App (SAA) and Webshop App (WA) are allowed to be honest but curious, hence, privacy, integrity and authenticity is guaranteed, as long as all parties follow the protocol and are not colluding.

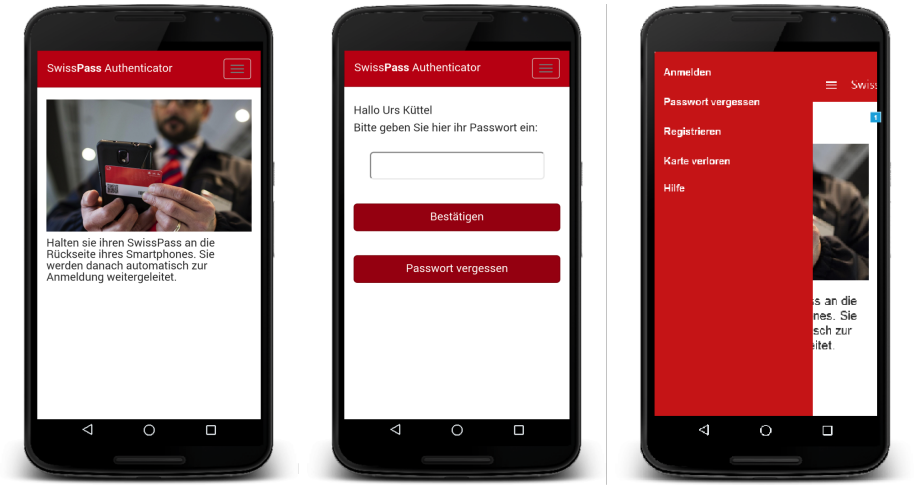

Fig. 5. Screen shots from the SwissPass Authenticator App: (a) Request to scan the SwissPass; (b) Request to enter (or reset) the password; (c) The main menu.

As the sole purpose of the application in question is the secure authentication, the most interesting attack is considered to be the impersonation attack. In order to provoke such an attack, the polynomial bounded external adversary must either gain full control over a running SAA or must get in possession of the SwissPass and the password. Brute-forcing the password is not possible as the password is not stored or verified locally on the user's device. This way, the password validation is an online process (SAS verifies the password) and thus the attack is detected by SAS. The adversary could get the password by key-logger, which however is denied due to the mentioned security assumptions. Gaining the knowledge of the password via social engineering attack is possible but out of scope for this discussion. As the second factor is a hard token, the adversary is required to get direct access to it. Stealing the SwissPass is a possible attack, however in this setting it is not considered a successful attack, as this is always detectable by the legitimate user. As already mentioned, the external adversary might acquire full control over the device where the SAA is running and steal an ongoing transaction. However, this requires the adversary to either break cryptographic means or to tweak the display, which is denied due to the mentioned security assumptions. The external adversary can create a fake SAA which serves its purpose. This is a possible attack which has to be addressed via user awareness.

VI. Implementation

The SAA was implemented as hybrid app (screen shots are provided in Fig. 5) using the Bootstrap Framework, based on HTML5, CSS3 and JavaScript, over Apache Cordova extended with the Phonegap-NFC plugin (to support NFC), the Cordova Vibration plugin (to improve haptics) and Cordova Whitelist plugin (to communicate with the SAS). The native layer is Android.

The SAS is a Node.js application extended with Express.js providing a RESTful API. It contains a mySQL database storing the information about the users and their related SwissPass. More details about the implementation of SAA and SAS can be found in [16].

VII. Conclusion

The paper proposes a 2FA protocol based on the SwissPass smart card. As the SwissPass card is widely spread in Switzerland (approx. 30% of citizen have it today and more are expected) and includes a high quality user registration and issuance process, this card is an ideal hardware based authentication factor. The described protocols show the security and simplicity of the SwissPass registration and (federated) authentication, which also guaranties privacy protection towards the resource providers (no personal information is disclosed without the explicit consent of user). Usability tests have shown that also non-technical people are able to use the SwissPass Authenticator and to scan their SwissPass card (thanks to the example the inspector gives when inspecting the SwissPass in the train.)

The authentication protocols are not bound to a device or SIM card, that means that no setup or recovery processes are needed, which further improves usability.

Still, our solution is restricted to NFC enabled smart phones excluding iPhone users, which are highly relevant in Switzerland. NFC dongles for iPhones are available, but a better solution would be the open release of the iPhone's NFC API by Apple.

At the moment, the SwissPass 2FA is only prototypical implemented. To facilitate the integration with third party resource providers, it should support the standard Access Token Request of oAuth [14]. To increase user experience further, the SAA could also support Refresh Tokens.

The fact that the SwissPass Authenticator does not persist anything on the users' phone has not only advantages. The SwissPass Authenticator can only be used with internet connection to the SAS. A further development could include the generation of a security token for offline usage. This security token is stored on the smart phone and can only be activated with the associated SwissPass card.

ACKNOWLEDGMENT

The authors would like to thank N. Sithampary and F. Sellin, who developed the main parts of the described protocols in their bachelor thesis [16] and implemented the prototype.

REFERENCES

[1] P. A. Grassi et al., "NIST Special Publication 800-63B Digital Identity Guidelines (Draft)," Online, last retrieved May 2017, https://pages.nist.gov/800-63-3/sp800-63b.html, 2017.

[2] Swiss Federal Railways SBB, "SwissPass Portal," Online, last retrieved May 2017, https://www.swisspass.ch.

[3] telebasel, "SBB mit Passagier-Rekord im 2016," Online, last retrieved May 2017, https://telebasel.ch/2017/03/21/sbb-mit-passagier-rekord-im-2016.

[4] Joseph Cox, "We Were Warned About Flaws in the Mobile Data Backbone for Years. Now 2FA Is Screwed." Online, https://motherboard.vice.com/en_us/article/we-were-warned-about-flaws-in-the-mobile-data-backbone-for-years-now-2fa-is-screwed, 2017.

[5] C. Mulliner, N. Golde, and J.-P. Seifert, "Sms of death: From analyzing to attacking mobile phones on a large scale," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 24–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028091

[6] ICICI Bank, "What is SIM-Swap fraud?" Online, last retrieved May 2017, https://www.icicibank.com/online-safe-banking/simswap.html.

[7] L. Davi, A. Dmitrienko, C. Liebchen, and A.-R. Sadeghi, "Over-the-air cross-platform infection for breaking mtan-based online banking authentication," 2012.

[8] Wikipedia. Google authenticator. [Online]. Available: https://en.wikipedia.org/wiki/Google_Authenticator

[9] Twilio Inc. Authy. [Online]. Available: https://authy.com/

[10] D. M'Raihi et al., "TOTP: Time-Based One-Time Password Algorithm," RFC 6238, May 2011. [Online]. Available: https://tools.ietf.org/html/rfc6238

[11] ——, "HOTP: An HMAC-Based One-Time Password Algorithm," RFC 4226, December 2005. [Online]. Available: https://tools.ietf.org/html/rfc4226

[12] FIDO Alliance: S. Srinivas et al., "Universal 2nd Factor (U2F) Overview." Online, last retrieved May 2017, https://fidoalliance.org/specs/fido-u2f-v1.1-id-20160915/fido-u2f-overview-v1.1-id-20160915.html, 2016.

[13] Ori Jacobovitz, "Blockchain for identity management. technical report #16-02." December 2016. [Online]. Available: https://www.cs.bgu.ac.il/~frankel/TechnicalReports/2016/16-02.pdf

[14] E. D. Hardt, "The OAuth 2.0 Authorization Framework [RFC 6749]." Online, last retrieved May 2017, https://tools.ietf.org/html/rfc6749, 2012.

[15] "ISO/IEC 14443: Identification cards, Contactless integrated circuit(s) cards, Proximity cards," 2008.

[16] F. Sellin, N. Sithampary, "2-Faktoren-Authentifizierung mit SwissPass," B.S. Thesis, Bern University of Applied Sciences, 2017.