Encryption-Based Second Authentication Factor Solutions for Qualified Server-Side Signature Creation

Christof Rath(✉), Simon Roth, Harald Bratko, and Thomas Zefferer

Institute for Applied Information Processing and Communications,
Graz University of Technology, Graz 8010, Austria
{christof.rath,simon.roth,harald.bratko,thomas.zefferer}@iaik.tugraz.at
http://www.iaik.tugraz.at

Abstract. Electronic identity (eID) and electronic signature (e-signature) are key concepts of transactional e-government solutions. Especially in Europe, server-based eID and e-signature solutions have recently gained popularity, as they provide enhanced usability while still complying with strict security requirements. To implement obligatory two-factor user-authentication schemes, current server-based eID and e-signature solutions typically rely on one-time passwords delivered to the user via short message service (SMS). This raises several issues in practice, as the use of SMS technology can be cost-effective and insecure. To address these issues, we propose an alternative two-factor user-authentication scheme following a challenge-response approach. The feasibility and applicability of the proposed user-authentication scheme is evaluated by means of two concrete implementations. This way, we show that the proposed authentication scheme and its implementations improve both the cost effectiveness and the security of server-based eID and e-signature solutions.

Keywords: Electronic identity · Electronic signature · Server signature · User authentication · Challenge response · Two-factor authentication

1 Introduction

The concepts of electronic identity (eID) and electronic signature (e-signature) are crucial for transactional e-government services. They enable users to securely and reliably authenticate at services and to create electronic signatures. Their relevance is especially given in the European Union (EU), where so-called qualified electronic signatures are legally equivalent to handwritten signatures [10]. This enables users to remotely provide written consent in transactional services.

During the past years, different approaches for the realization of eID and e-signature concepts have been studied, implemented, and deployed. First approaches to provide users eID and e-signature functionality have been based on

A. Kő and E. Francesconi (Eds.): EGOVIS 2015, LNCS 9265, pp. 71–85, 2015.
DOI: 10.1007/978-3-319-22389-6_6

smart-card technology [4,5]. However, these approaches have turned out to suffer from several usability-related limitations and hence from limited user acceptance [11]. As an alternative, mobile eID and e-signature solutions have emerged early. These solution remove the need for smart-card usage by making use of the user's mobile phone instead. Two approaches can be distinguished. The first approach employs the mobile phone's SIM card to store eID data and to implement cryptographic functions required for the creation of electronic signatures. The second approach instead relies on a central hardware security module (HSM) to store eID data and to carry out required cryptographic functions.

During the past years, the second approach, i.e., server-based solutions, has gained relevance and popularity, mainly because it defines fewer requirements for the mobile end-user device and mobile network operators (MNO), which in turn improves applicability, feasibility and usability. The main challenge in designing and developing server-based eID and e-signature solutions is the provision of appropriately secure user-authentication schemes. These schemes are required to restrict access to centrally stored eID data and cryptographic signing keys to the legitimate user. In order to assure a sufficient level of security, two-factor authentication (2FA) is typically the approach of choice.

Current mobile eID and e-signature solutions following the sever-based approach implement 2FA schemes by means of one-time passwords (OTPs) delivered by SMS messages [8,9]. After the user has entered a secret password covering the authentication factor knowledge, he or she receives a OTP via SMS. By proving reception of the OTP, the user proves possession of the mobile phone. This way, the authentication factor possession is covered and the 2FA process is completed.

Unfortunately, reliance on SMS technology raises several issues [6]. First, SMS must not be regarded as secure. This especially applies to smartphones, on which incoming SMS messages can be intercepted by third party applications. Second, the sending of SMS messages containing OTPs can cause significant costs for the service operator, as mobile network operators (MNOs) typically charge the delivery of SMS messages. To overcome these issues, we propose an alternative 2FA scheme for server-based mobile signature solutions. Our proposed scheme renders the use of SMS technology unnecessary. This way, it provides higher cost efficiency and better security compared to existing approaches.

2 Related Work

Two-factor authentication has been a topic of interest for many years. This does not only apply to eID and e-signature solutions but basically to any e-service that requires a secure and reliable remote authentication of users. Numerous approaches and solutions to authenticate users by means of 2FA have been proposed and developed during the past years. Although these approaches and solutions rely on different technologies and communication protocols, they can be classified in a few basic categories, whereas the implementation of the authentication factor possession is used as key classification criterion. Relevant categories of 2FA schemes are briefly sketched in this section.

Client-generated OTPs represent the first relevant category of 2FA schemes. The basic idea behind this scheme is simple. The user creates a OTP using some kind of hardware token. As creation of the correct OTP is infeasible without this token, proving knowledge of the OTP proves possession of the respective hardware token. This way, the authentication factor possession is covered. During the past years, different implementations of authentication schemes relying on client-generated OTP have been proposed. Examples are SecurID[1] or DIGIPASS[2], which incorporate the current time for the derivation of OTPs. A related standard for the generation of OTPs based on the current time has been proposed in RFC 6238[3]. Alternatively, a counter synchronized between the remote entity and the user's local hardware token can also be used to derive unambiguous OTPs. This has been described in RFC 4226[4]. Recently, OTP-based authentication solutions have been developed that rely on personalized mobile apps instead of hardware tokens. Examples of such app-based approaches are Google Authenticator[5] or a solution developed by the Barada project[6]. In general, client-generated OTPs are a relatively old and hence time-tested approach. However, they have several disadvantages when being used for server-based signature solutions. For example, they do not allow an unambiguous binding between the current transaction and the generated OTP.

To overcome limitations of client-generated OTPs, current server-based signature solutions follow the SMS-OTP Approach, where the authentication factor possession is covered by the user's SIM. Possession of the SIM is verified by sending an OTP to the user's mobile phone via SMS. By proving knowledge of the OTP, the user proves possession of the SIM. Although the SMS-OTP Approach makes use of OTPs as well, there are conceptual differences to solutions relying on client-generated OTPs. Client-generated solutions require only one communication step, in which the locally created OTP is transferred to the remote entity. In contrast, the SMS-OTP Approach implements two consecutive communication steps. A centrally created OTP is first sent to the user's mobile phone. Subsequently, the OTP must be transmitted back to the remote entity. The central generation of the OTP is an important conceptual advantage, as it enables the remote entity to unambiguously bind the OTP to a specific authentication run and hence to a certain transaction. Unfortunately, the central OTP generation also bears a considerable drawback. The SMS-OTP approach demands that the OTP is transferred to the user's mobile end-user device via SMS. Unfortunately, SMS technology cannot guarantee secure data transmissions. This especially applies to modern smartphones, on which incoming SMS messages can be compromised by malware [2]. Still, the SMS-OTP Approach is frequently used in practice. Examples are e-banking solutions and the mobile signature solutions

[1] http://www.emc.com/security/rsa-securid.htm.
[2] https://www.vasco.com/products/products.aspx.
[3] http://tools.ietf.org/html/rfc6238.
[4] http://tools.ietf.org/html/rfc4226.
[5] https://code.google.com/p/google-authenticator/.
[6] http://barada.sourceforge.net/.

Austrian Mobile Phone Signature[7] and ServerBKU [9]. The latter two rely on a concept by Orthacker et al. [8] in order to assure a sufficient level of security to create qualified electronic signatures.

As an alternative to OTP-based approaches, challenge-response approaches have emerged as 2FA schemes during the past years. They represent the third category, in which current 2FA solutions can be classified. Challenge-response approaches resemble the SMS-OTP Approach, as they also rely on two consecutive communication steps. First, the remote entity generates a random challenge, which is transmitted to the user's mobile phone. The mobile phone creates a response from this challenge using cryptographic methods. These methods employ a device-specific cryptographic key. Thus, the capability to create responses from received challenges with this particular key proves possession of the device. Created responses are then returned to the remote entity. The remote entity cryptographically verifies the obtained response. Therefore, it must be aware of the cryptographic key used to create the response. Hence, challenge-response approaches require a pairing process to exchange relevant key material. During the past years, several authentication solutions following challenge-response approaches have been introduced for powerful mobile end-user devices such as smartphones. These solutions can again be classified into two categories. Software-based solutions store required cryptographic key material and implement cryptographic functionality in software. Such a software-based authentication solutions following a challenge-response approach is for instance SQRL[8]. In contrast, hardware-based solutions implement cryptographic functionality in secure hardware elements. While this provides a higher level of security, it also requires mobile end-user devices to provide appropriate hardware components. A well-known example for a hardware-based authentication solution following the challenge-response approach is U2F proposed by the FIDO Alliance[9].

All of the mentioned 2FA schemes come with various pros and cons. However, none of them has been explicitly designed for a use with server-based eID and e-signature solutions. Hence, these schemes are not tailored to the special requirements of this use cases. To address this issue, we propose a new 2FA scheme that explicitly takes into account special requirements and characteristic of server-based eID and e-signature solutions. These requirements are identified and discussed in the following section.

3 Requirements

Relevant requirements for the proposed 2FA scheme have been derived from current state-of-the-art solutions, i.e., SMS message-based two-factor authentication, and from requirements defined by relevant legislations. Derived requirements are listed and described below in more detail. Derived requirements will also serve as basis for the evaluation of the proposed solution.

[7] http://www.handy-signatur.at.
[8] http://sqrl.pl.
[9] https://fidoalliance.org/.

R1: Platform independence
In order to avoid exclusion of certain user groups, the proposed solution must be platform independent.

R2: Transaction binding
It must be possible to unambiguously link the document to be signed to the authentication data used to authorize the transaction.

R3: Security
The proposed solution must be at least as secure as established SMS-based solutions.

R4: Usability
User acceptance is a key factor for any eID solution [11]. In general, an alternative 2FA approach must not reduce the usability and, thus, user acceptance. Furthermore, a new solution should be self-explanatory for users and necessary changes and benefits should be easy to communicate from the operator side.

R5: Feasibility
For operators, the proposed solution must be easy to integrate and flexible in its operation. External dependencies should be minimized.

R6: Cost efficiency
The solution must be cost efficient, i.e., investments must pay off and the operation must be cheaper than SMS-based solutions. At the same time, costs must not be transferred to the users.

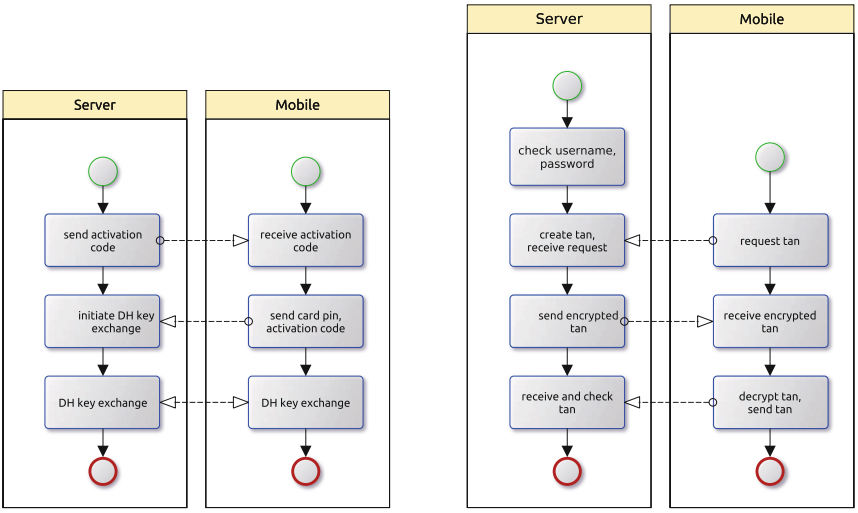
Based on these requirements, we propose an alternative 2FA scheme for server-based eID and e-signature solutions in the next section.

4 Proposed Solution

Over the time, several different approaches as alternatives for the widely used SMS-OTP procedure have been developed. Apart from economical reasons, i.e., the costs for the huge amount of SMS messages to be sent, security considerations were the most important factor to look for a new two-factor authentication approach. Trivially, one could simply change the direction of the SMS message from *being-received* to *has-to-be-sent* by the user. However, this greatly reduces the usability of the system and, thus, the willingness to use such a solution.

The growing popularity of smartphones enables novel solutions that do not rely on SMS messages to prove the possession of an authentication token. Accordingly, our proposed solution does not rely on a separate communication channel, i.e. the GSM protocol and the MNO, but uses a standard network connection and cryptographic key material that is bound to the device.

Overall, the proposed solution consists of two distinct phases, the pairing phase and the authentication phase. The basic concept of the solution and its two phases is depicted in Fig. 1 and detailed in the following subsections.



(a) Abstract Pairing. (b) Abstract Authentication.

Fig. 1. Abstract basic concept of the solution.

4.1 Pairing

Initially, a mobile device, which is later used for authentication, must be bound to a user account by the so-called pairing process. This is shown in Fig. 1(a). After initiating this pairing process, the server component, which maintains the respective user account, generates a random activation code. The activation code gets embedded into an URL that references the server's pairing component and is sent via SMS message to the user's phone. Once this URL gets dereferenced, the user is asked to authenticate by entering his or her personal password that is assigned to his or her user account. If the entered password is correct, the active pairing session can be identified via the embedded activation code and a symmetric key is exchanged between the mobile device and the server component using the Diffie-Hellman key exchange protocol [3]. The exchanged key is securely stored by both the server component and the mobile device and is used in subsequent authentication phases.

4.2 Authentication

The authentication process is shown in Fig. 1(b). It starts with a conventional user name and password authentication to cover the first authentication factor. If this step succeeds, the server component prepares a challenge by generating a random OTP t. The OTP is encrypted using the shared symmetric key ssk generated during the pairing phase.

$$t_{\mathsf{enc}} = \mathrm{encrypt}(t;\, ssk) \tag{1}$$

Upon user interaction, the mobile phone requests the encrypted OTP t_{enc}. On the mobile phone, the OTP is decrypted using the same shared secret ssk'.

$$t' = \text{decrypt}(t_{\mathsf{enc}};\ ssk') \tag{2}$$

By proving the capability to decrypt the OTP t, the user proves possession of the mobile device. This covers the second authentication factor. The decrypted OTP t' is returned to the server component. The server component verifies the correctness of the received OTP t'. If the received OTP t' is correct, the authentication is regarded as successful.

5 Evaluation

The solution proposed in Sect. 4 has been evaluated by means of two concrete implementations. In this section we introduce these two implementations in more detail. Our first implementation has been based on HTML5 and is discussed in Subsect. 5.1. To further increase usability, we have also developed a solution that makes use of QR tags. This solution is described in detail in Subsect. 5.2. Both implementations increase cost efficiency R6 in comparison to traditional SMS-OTP solutions by superseding the need for sending SMS messages every time a signature is created.

To test our approach, we integrated both implementations in the *ServerBKU* [9]. The *ServerBKU* is a secure and flexible server-based mobile eID and e-signature solution. It uses the classical SMS-OTP approach to authenticate users and to authorize the creation of server-side signatures. We have enhanced the *ServerBKU* by replacing its SMS-based user-authentication scheme with the two implementations of the proposed solution. The two developed implementations and their integration into the *ServerBKU* is detailed in the following subsections.

5.1 *TanApp*

Our first implementation of the concept proposed in Sect. 4 is called *TanApp*. The *TanApp* implements all required client functionality by means of HTML5 and JavaScript functionality. According to the proposed concept, the *TanApp* retrieves AES-encrypted [7] OTPs from the *ServerBKU*, decrypts these OTPs and displays them to the user. To further improve security, the *TanApp* additionally features an elaborate sequencing mechanisms. This means, that communication between the *TanApp* and the *ServerBKU* is sequence controlled. Before the first user-authentication process, *ServerBKU* and *TanApp* negotiate an initialization vector (IV), which acts as sequence counter. The sequence counter, i.e. the IV, changes after each user-authentication process.

In total, the *TanApp* implements two use cases: paring and signature creation. During the pairing process, *ServerBKU* and *TanApp* negotiate a secret AES key that is used to encrypt and decrypt OTPs. Furthermore, the IV, which

is required for the implemented sequencing mechanism, is negotiated. During signature creation, the *TanApp* is used to authenticate the user at the *ServerBKU* and to authorize a signature-creation process. Both use cases are described in the following subsections in more detail.

Pairing: After the user is registered at the *ServerBKU* and has activated an eID, in the context of the Austrian e-government also known as mobile citizen card (MCC), he or she may pair this eID to the *TanApp* on his smartphone to use this approach instead of the SMS-based method henceforth.

When the user starts the pairing process an SMS message containing a URL is sent to his or her mobile phone, which guarantees the binding between the mobile phone and its owner. This URL contains a randomly generated activation code as parameter. Clicking this URL opens the default browser where the user is prompted to enter his or her user name and password. This ensures that only the legitimate user can change the authentication method. After that, the *ServerBKU* and *TanApp* use the Diffie-Hellman key exchange protocol to negotiate an AES key, the pairing key ssk. For performance reasons, especially in the context of our JavaScript implementation, we use EC-Diffie-Hellman [1]. On the server side, the created AES key is securely stored. On the smartphone side, the key is stored in the HTML5 local storage.

During the pairing process, *ServerBKU* and *TanApp* also exchange the initial value of the AES initialization vector. In the final step of the pairing process, the *ServerBKU* uses this initialization vector to compute the unique identifier ($tanId$) of the current transaction state as the encrypted concatenation of the initialization vector (IV), the eID alias ($mccAlias$) and the mobile-phone number ($mobileNumber$):

$$tanId = \text{encrypt}(iv \| mccAlias \| mobileNumber; iv, ssk) \tag{3}$$

The *ServerBKU* stores the $tanId$. The pairing process is finished by switching the default user-authentication scheme for the respective eID from SMS-OTP mode to *TanApp* mode.

Signature Creation: Once an eID, i.e. an MCC, has been paired, the *TanApp* can be used to authorize signature-creation processes. The required processing steps are shown in Fig. 2. The *TanApp*-based user authentication of a typical signature-creation process consists of the following steps:

1. The *ServerBKU* receives a signature creation request and prompts the user to enter her user name and password (see Fig. 3(a)).
2. After the user has entered the required data (i.e. user name and password) in the browser, the *ServerBKU* identifies the corresponding eID and retrieves the associated pairing key (ssk).
3. The *ServerBKU* creates a new OTP (t) and the initialization vector for the next round (iv_{new}) and stores both in the card database.

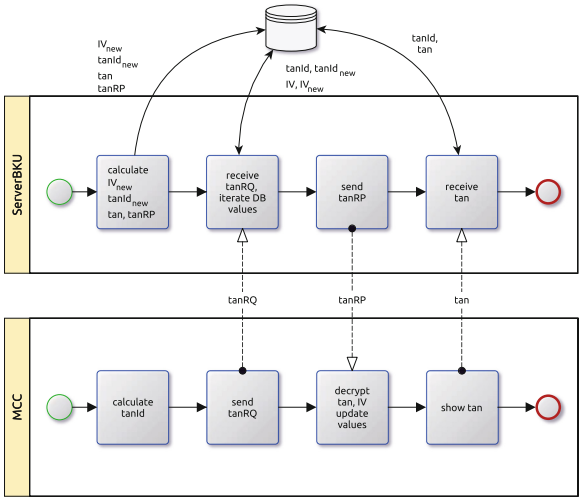


Fig. 2. Tanapp protocol.

4. The *ServerBKU* computes the unique identifier used for transaction binding (cf. Requirement R2) of the next round ($tanId_{\mathsf{new}}$) and stores it as well:

$$tanId_{\mathsf{new}} = \text{encrypt}(iv_{\mathsf{new}} \| mccAlias \| mobilNumber; iv_{\mathsf{new}}, ssk) \tag{4}$$

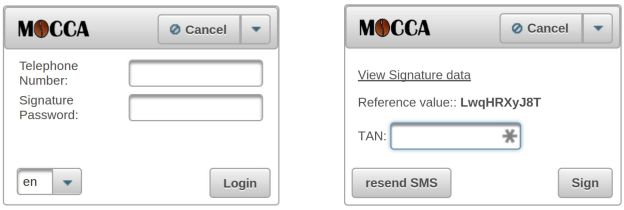
5. The *ServerBKU* computes the OTP response (*tanRP*) for the expected OTP request (*tanRQ*) and stores it. Note that this is the OTP response for the current signature request and, hence, the IV used for encryption is the also the current one:

$$tanRP = \text{encrypt}(iv_{\mathsf{new}} \| t; iv, ssk) \tag{5}$$

   The OTP response represents an encrypted container used for delivering the OTP t (cf. step 3) and the IV for the next round (iv_{new}) from the *ServerBKU* to the *TanApp*.
6. The user's browser window prompts for the OTP, as shown in Fig. 3(b).
7. On the smartphone, the user opens the *TanApp* and selects the eID he or she wants to use for signing. This eID must correspond to the credentials she provided in step 2. Selecting the corresponding icon causes the *TanApp* to compute the value $tanId'$ for the OTP request (*tanRQ*):

$$tanId' = \text{encrypt}(iv' \| mccAlias \| mobileNumber; iv', ssk') \tag{6}$$

   Note that $tanId'$ has the same value as the one that the *ServerBKU* has created and stored in the final step of the pairing process (cf. page 8), or the value $tanId_{\mathsf{new}}$ of a previous signature creation process.
8. The $tanId'$ is embedded in the OTP request and sent to the *ServerBKU*.
9. The *ServerBKU* receives $tanId'$ to identify the corresponding eID. If no match can be found, the *ServerBKU* and the *TanApp* have run out of synchronisation and the corresponding eID has to be reset and paired again.



(a) User identification. (b) OTP prompt.

Fig. 3. Web user interface

10. The *ServerBKU* verifies if there is a pending signature request for the eID by checking whether the OTP entry is not `null`. The *ServerBKU* updates the database ($iv = iv_{\text{new}}$, $tanId = tanId_{\text{new}}$, $iv_{\text{new}} = tanId_{new} =$ `null`) and sends the OTP response ($tanRP$), pre-computed in step 5, to the *TanApp*.
11. The *TanApp* receives the OTP response $tanRP$ and decrypts it using the pairing key (ssk) for extracting the OTP t and the IV for the next round iv_{new}.
$$iv'_{\text{new}} \| t' = \text{decrypt}(tanRP; iv', ssk') \tag{7}$$
12. The *TanApp* updates the initialization vector ($iv' = iv'_{\text{new}}$) and displays the OTP t' to the user as shown in Fig. 4.
13. The user enters the OTP t' in the browser window shown in Fig. 3(b), which has been displayed in step 6.
14. The *ServerBKU* checks the OTP, and if $t' = t$ the authentication is successful. It cleans up the database ($tan = tanRP =$ `null`), issues the signature and returns the signed document.

Synchronisation. As already mentioned, the *ServerBKU* and *TanApp* may become unsynchronized. Especially, this can happen if the connection drops during the OTP response delivery. In that case, the *TanApp* will not update the

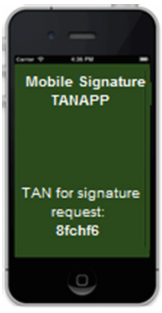


Fig. 4. OTP displayed on smartphone

AES IV and, hence, the next OTP request will use the same $tanId$ as the interrupted request. However, since the *ServerBKU* has performed the IV update before sending the OTP response, no matching record would be found and, thus, the signature creation fail. Once *ServerBKU* and *TanApp* are asynchronous, they must be reset and paired again. If the user is still able to log on to the *ServerBKU*, e.g., by means of an additional eID, he can perform this re-pairing himself. Otherwise, a *ServerBKU* administrator must perform this step. For reasons of usability, efficiency and operational costs this is an undesirable situation. To mitigate this effect, the *ServerBKU* keeps the previous value $tanId_{\mathsf{old}}$. If the *TanApp* for some reason has not updated the IV and, hence, uses the same $tanId$ as in the previous OTP request, the *ServerBKU* will be able to identify the corresponding eID by checking the values of $tanId$ and $tanId_{\mathsf{old}}$. Once a signature creation request succeeds the *ServerBKU* deletes $tanId_{\mathsf{old}}$ and keeps only the value for the next round ($tanId = tanId_{\mathsf{new}}$), because in that case it is guaranteed, that the *TanApp* has updated the IV. Obviously, if the same OTP request can be sent twice, we must ask if this could bring the potential for a replay attack. However, we could show that yields no advantage for an adversary.

5.2 *QR TanApp*

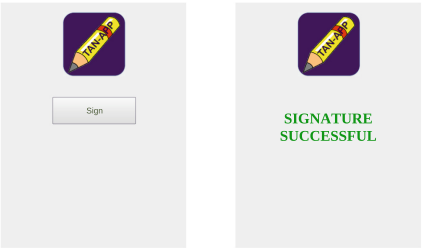
The pairing process for the *QR TanApp* is akin to the paring process for the *TanApp* described in Subsect. 5.1.

Signing on the other hand differs in some aspects albeit maintaining the same level of security (R3). After the *ServerBKU* receives a signature request, the user is prompted to enter her phone number of an already paired smartphone and signature password to access her eID. Similar to other approaches, the eID is locked for some time if repeatedly wrong login credentials are provided to avoid brute-force attacks (R3). After successful identification, the parameters necessary for the signature creation process are calculated and temporarily stored in the database. Finally a QR code is generated and presented to the user as shown in Fig. 5.

Fig. 5. Second authentication factor QR code prompt

This code contains an URL to the *ServerBKU* signature servlet. Subsequently, the user has to take a photo of this code using a QR reader and continue

the signature creation process by opening the contained URL. Since QR code readers are available on all major platforms the platform independence requirement R1 is fulfilled. To complete the signature creation the user has to click a sign button, as shown in Fig. 6(a). Although not strictly necessary, the button was integrated to improve the user experience (R4). Finally, a message indicating the success status is shown on the smartphone, as in Fig. 6(b) and the signature gets issued on the server side.



(a) User consent (b) Status

Fig. 6. *QR TanApp* application

Implementation and Protocol Details. The *QR TanApp* was implemented as native smartphone app for Android. This improves the usability (R4) since we could directly access the camera and include a QR reader library. Furthermore, the security could be increased by making use of the smartphone's system key store (R3). To fulfil the platform independence requirement R1, it would be necessary to implement this native app for all major mobile operating systems. A simplified sequence of a signature creation process using the *QR TanApp* can be seen in Fig. 7.

After the successful identification, several parameters have to be pre-computed and stored in the database. As mentioned before, the pairing key *ssk* is only accessible in the HTTP session where the user has entered her credentials. Therefore, all parameters that require this key have to be pre-computed now. These parameters are:

- iv_{new}: A random initial vector for the next signature.
- $mccTag$: A user can potentially hold multiple eIDs. To distinguish them each eID has an alias $mccAlias$.

$$mccTag = \text{hash}(mccAlias \| phoneNumber) \tag{8}$$

- $tanId$: The $tanId$ is the reference value to verify the possession of the paired mobile device. It contains a random OTP t.

$$tanId = \text{encrypt}(mccTag \| t \| iv; iv_{\mathsf{new}}, ssk) \tag{9}$$

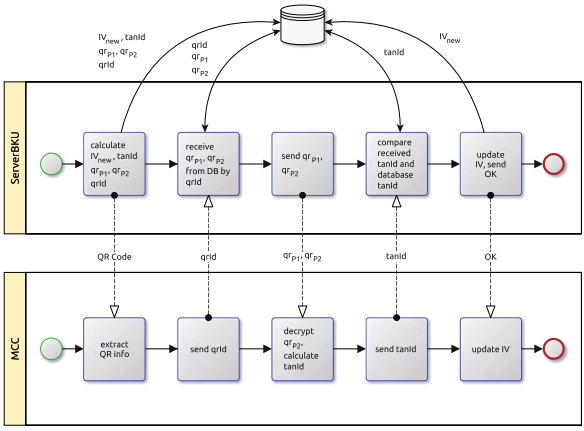


Fig. 7. *QR TanApp* protocol

– qr_{P1}: The encryption result of $mccTag$ concatenated by iv_{new}.

$$qr_{\mathsf{P1}} = \text{encrypt}(mccTag \| iv_{\mathsf{new}}; iv, ssk) \tag{10}$$

– qr_{P2}: The hash value of the $mccAlias$.

$$qr_{\mathsf{P2}} = \text{hash}(mccAlias) \tag{11}$$

– $qrId$: A random value for the identification of the current transaction. This is a reference value to fulfil the transaction binding requirement R2.

These values are stored in the database, except $mccTag$, which is only an intermediate value.

A QR code is generated that contains the URL to the *ServerBKU* servlet processing the signature, including the parameter $qrId$ and the OTP t. This code is shown to the user, as in Fig. 5, who has to take a photo of the code using the *QR TanApp* on her paired mobile device. By decoding the QR code and dereferencing the URL, $qrId$ is transmitted to the *ServerBKU* and the OTP t is stored for later use. Note that we now have a second, independent HTTP session. Using the $qrId$ the *ServerBKU* is able to match the two sessions and returns the parameters qr_{P1} and qr_{P2}.

Using qr_{P2} the *QR TanApp* is able to identify the active eID. Now, qr_{P1} can be decrypted using the corresponding pairing key ssk' and the IV currently stored on the smartphone.

$$mccTag' \| iv'_{\mathsf{new}} = \text{decrypt}(qr_{\mathsf{P1}}; iv', ssk') \tag{12}$$

With these values it is possible to calculate the $tanId'$ similar to the $tanId$ calculated by the *ServerBKU* beforehand.

$$tanId' = \text{encrypt}(mccTag' \| t \| iv'; iv'_{\mathsf{new}}, ssk') \tag{13}$$

This value is transmitted to the *ServerBKU* once the user presses the sign button. The *ServerBKU* verifies that $tanId = tanId'$, in which case the authentication was successful. The new IV gets stored for the next transaction ($iv = iv_{\mathsf{new}}$) and the status is returned to the *QR TanApp*, which in turn also updates the IV ($iv' = iv'_{\mathsf{new}}$). Finally, the *ServerBKU* issues the signature and returns the signed document to the user via the initial HTTP session.

Additional Protocol Features. As with the *TanApp*, the protocol has some additional features to make the application more robust and flexible but which are not of significance for the authentication or signature creation itself:

- A mechanism is implemented to allow to exchange the cryptographic primitives.
- There is support to use multiple eIDs on a single device.
- If the IV synchronization gets lost due to failed transactions the protocol has limited possibilities for self recovery. If this recovery fails a new pairing process is necessary.
- It is not possible to issue multiple signatures in parallel. This is a desired behavior and only the latest request is completed in this case.

6 Conclusion

Due to recent technological advances in the mobile sector, authentication schemes based on SMS-delivered OTPs must be regarded as outdated and inappropriate. This raises challenges for server-based signature solutions that still heavily rely on this approach. To overcome this challenge, we have proposed a novel two-factor based authentication scheme that completely avoids SMS technology. The proposed authentication scheme has three basic advantages. First, it provides a higher level of security, by using strong cryptographic algorithms and hardware key stores on the mobile devices. In particular, it prevents Android-specific attacks that employ techniques to intercept incoming SMS messages. Second, it meets all relevant requirements of server-based signature solutions. Third, it provides enhanced usability by integrating QR codes. Concretely, it prevents users from manually copying unintelligible OTPs from received SMS message messages to the browser.

We have successfully evaluated the proposed authentications scheme by means of two different prototype implementations. The applicability of both implementations has been shown by integrating them into and using them together with an existing mobile eID and e-signature solution. This has proven that the two implementations and the underlying concept are feasible. Practical tests of the two prototype implementations have also revealed their strengths and weaknesses. In particular, it turned out that the QR-based implementation is advantageous in terms of usability. Future work, which will mainly focus on a further consolidation of the current prototypes and their integration into productive solutions, will hence mainly focus on the QR-based approach.

The proposed authentication scheme, its prototypical implementation, and the conducted evaluation presented in this paper show that there are secure and usable alternatives for authentication schemes based on SMS-delivered OTPs. This way, this paper contributes to the further improvement of mobile eID and e-signature solutions and paves the way for their for a successful future e-government.

Acknowledgements. The authors have been supported by the European Commission Seventh Framework Programme through project *FutureID*, grant agreement number 318424.

References

1. Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography (2007). http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf. Accessed March 2015
2. Check Point Software Technologies Ltd.: Media Alert: Check Point and Versafe Uncover New Eurograbber Attack (2012). http://www.checkpoint.com/press/2012/120512-media-alert-cp-versafe-eurograbber-attack.html
3. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976). http://dx.doi.org/10.1109/TIT.1976.1055638
4. Fairchild, A., de Vuyst, B.: The Evolution of the e-ID card in Belgium: Data Privacy and Multi-Application Usage. In: The Sixth International Conference on Digital Society, pp. 13–16. Valencia (2012)
5. Leitold, H., Hollosi, A., Posch, R.: Security Architecture of the Austrian Citizen Card Concept. In: 18th Annual Computer Security Applications Conference, 2002, Proceedings, pp. 391–400 (2002)
6. Mulliner, C., Borgaonkar, R., Stewin, P., Seifert, J.-P.: SMS-based one-time passwords: attacks and defense (short paper). In: Rieck, K., Stewin, P., Seifert, J.-P. (eds.) DIMVA 2013. LNCS, vol. 7967, pp. 150–159. Springer, Heidelberg (2013)
7. National Institute of Standards and Technology: Advanced Encryption Standard (AES) (2001). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
8. Orthacker, C., Centner, M., Kittl, C.: Qualified mobile server signature. In: Rannenberg, K., Varadharajan, V., Weber, C. (eds.) SEC 2010. IFIP AICT, vol. 330, pp. 103–111. Springer, Heidelberg (2010)
9. Rath, C., Roth, S., Schallar, M., Zefferer, T.: A secure and flexible server-based mobile eID and e-signature solution. In: Proceedings of the 8th International Conference on Digital Society, ICDS 2014, Barcelona, Spain. pp. 7–12. IARIA (2014)
10. The European Parliament and the Council of the European Union: DIRECTIVE 1999/93/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 December 1999 on a Community framework for electronic signatures (1999). http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2000:013:0012:0020:EN:PDF
11. Zefferer, T., Krnjic, V.: Usability evaluation of electronic signature based E-Government solutions. In: Proceedings of the IADIS International Conference WWW/INTERNET 2012, pp. 227–234 (2012)