Practical backward unlinkable revocation in FIDO, German e-ID, Idemix and U-Prove

Eric R. Verheul

Radboud University Nijmegen, KeyControls
P.O. Box 9010, NL-6500 GL Nijmegen, The Netherlands.
eric.verheul@[cs.ru.nl,keycontrols.nl]
Draft 29th February 2016

Abstract FIDO, German e-ID, Idemix and U-Prove constitute privacy-enhanced public-key infrastructures allowing users to authenticate in an anonymous way. This however hampers timely revocation in a privacy friendly way. From a legal perspective, revocation typically should be effective within 24 hours after user reporting. It should also be backward unlinkable, i.e. user anonymity cannot be removed after revocation. We describe a new, generic revocation mechanism based on pairing based encryption and apply it to supplement the systems mentioned. This allows for both flexible and privacy friendly revocation. Protocol execution takes less than a quarter of a second on modern smartcards. An additional property is that usage after revocation is linkable, allowing users to identify fraudulent usage after revocation. Our technique is the first Verifier Local Revocation scheme with backwards unlinkable revocation for the systems mentioned. This also allows for a setup resembling the well-known Online Certificate Status Protocol (OCSP). Here the service provider sends a pseudonym to a revocation provider that returns its status. As the information required for this is not secret the status service can be distributed over many cloud services. In addition to the status service our technique also supports the publication of a central revocation list.

Keywords: ABCs, electronic authentication backward unlinkable revocation, pairings, Verifier Local Revocation

1 Introduction

In an electronic identity (e-ID) scheme a user is issued a device to electronically authenticate to on-line service providers. A first e-ID requirement is *reliability*: only the user is able to authenticate itself to a service provider. Reliability also relates to revocation. Regulations [17] and guidelines [13] stipulate that revocation should be effective within 24 hours after reporting. The first generation of European e-ID infrastructures accomplish reliability by issuing user digital certificates containing user personal data and storing the private key on an e-ID

card. As these cards can also be used to authenticate to private service providers the latter always receive user personal data regardless of their necessity. This contradicts the personal data minimisation principle, cf. [18]. We have identified a second e-ID requirement: *privacy*. Only minimal personal data should be provided to service providers. Privacy risks also relate to providers being able to indirectly identify users by combining attributes or using unique identifiers, e.g. social security numbers. The FIDO [20,21], German e-ID [11], Idemix [14,23] and U-Prove [9] systems allow for data minimisation. This however hampers timely revocation in a user and privacy friendly way. In this paper we describe a generic technique that can supplement these systems with such revocation. We briefly explain these systems to identify the desired privacy properties.

With "Restricted Identification" [11, Section 3.6], the German e-ID card provides persistent pseudonyms to service providers that cannot be cryptographically linked. Technically the pseudonyms take the form of hashed public keys, cf. Appendix A. The pseudonyms can be supplemented with attributes stored on the card by the user. The persistence of the pseudonyms is related to revocation (cf. [6]). Through a revocation provider, each service provider is provided its own list of revoked pseudonyms (Certificate Revocation List or CRL) allowing him to verify the validity of the card. To revoke, a user sends a revocation key to the revocation provider. This allows the latter to calculate all pseudonyms of the user and to provide the service provider CRLs. Due to its setup, CRL management in the German e-ID is a big burden. CRLs also leak potentially privacy sensitive data, indicating they should not be publicly readable, further complicating CRL management. To illustrate, if only one user revokes its e-ID card, then only one pseudonym is added to the various CRLs, enabling their linkage. By incorporating card expiry dates, prior usage of a revoked card can actually be linked in practice. This setup therefore does not cater for *backward unlinkable revocation*. A more fundamental issue here is that the revocation provider can link all pseudonyms of that user. So by collusion between the revocation provider and the service providers, e-ID card usage before revocation can be linked.

The emerging internet authentication standard FIDO (Fast IDentity Online) [20,21] has similarities with the German e-ID. For each service provider the FIDO user device generates a persistent public/private key pair meant for signing. The resulting public keys cannot be cryptographically linked. The public key is provided to the service provider allowing the user to authenticate against it. It is up to the service provider how to bind this to (personal data of) the user. FIDO does not provide for revocation [21, Section 2.5] and the user needs to revoke its public key at all service providers individually which is cumbersome.

Attributes are encrypted inside the Idemix [14] and U-Prove [9] certificates and the user private key is inside a (secure) device. A service provider can verify the authenticity of the certificate including the encrypted attributes. The user can then selectively disclose the attributes in the certificate to a service provider. To prevent linkability through the Idemix certificate itself, the user is able to blind its certificate. That is, prior to sending it to the service provider, the user can transform the given certificate in an equivalent one that cannot be linked to

other certificate uses. U-Prove unlinkability is addressed by providing the user with several certificates. A drawback of the non-linkability properties is that they do not simply cater for user and privacy friendly revocation. Indicatively a Dutch e-ID trial with Idemix technology did not include revocation. See [31].

Outline of the paper
In this paper we describe a generic revocation technique that is backward unlinkable and apply it to the four systems outlined above. In Section 2 we describe a common context for the systems in scope to form a basis for our revocation technique. In Appendix A we describe how both the FIDO and the German e-ID context can be be supplemented with the described revocation context without changing core protocols. In Section 3 we describe a first, basic revocation mechanism. This also allows us to identify desirable revocation approaches, that the basic mechanism can only support at the cost of privacy. Backwards unlinkability support is possible but both inflexible and user-unfriendly. We then provide a flexible variant of the basic mechanism in Section A based on pairing based encryption. This variant supports all identified revocation approaches without the loss of privacy or user friendliness. In Section 5 we make a comparison of our scheme with existing literature. Section 6 contains further practical considerations and further applications including German e-ID white listing. Section 7 contains conclusions.

2 Common ground in the systems identified

In this section we will define a common e-ID context for the four systems identified in Section 1 allowing us to describe our generic revocation techniques. Throughout this paper we let $G = \langle g \rangle$ be a *revocation group*. This is a multiplicative group of prime order q. By $\mathrm{GF}(q)$ we denote the Galois field of the integers modulo q. The assumed cryptographic security of the revocation group G can be formulated in four problems in the context of the Diffie-Hellman key agreement protocol with respect to g. The first one is the *Diffie-Hellman problem*, which consists of computing the values of the function $DH_g(g^x, g^y) = g^{xy}$. The second problem is the *discrete logarithm* (DL) problem in G with respect to g: given $\alpha = g^x \in G$, with $x \in \mathrm{GF}(q)$ then find $x = DL_g(\alpha)$. The third problem is the *Decision Diffie-Hellman* (DDH) problem with respect to g: given $\alpha, \beta, \delta \in G$ decide whether $\delta = DH_g(\alpha, \beta)$ or not. The DDH problem can be alternatively formulated as: given $h, \alpha, \beta \in G$ decide if $DL_g(\alpha) = DL_h(\beta)$. The DL problem is at least as difficult as the DH problem, which is at least as difficult as the DDH problem. One can easily show that if one can solve the discrete logarithms with respect to one generator, one can solve it with respect to any generator of G. That is, the hardness of the discrete logarithm problem is independent of the generator of the group. In [42] a similar property is shown for the Diffie-Hellman problem. Although not known, it is unlikely that the hardness of the Decision Diffie-Hellman problem is dependent of the generator of the group. To this end, we say that one can solve the *general* Decision Diffie-Hellman problem with respect to G if one can solve it for all generators of G. We assume that all four

introduced problems in G are intractable, i.e. no algorithm exists of complexity polynomial in the size of q that solves them with non-negligible probability.

In our e-ID context the user is issued a credential by an issuer that is bound to the user. The credential contains one or more hidden attributes that allow the user to authenticate to a verifier using a proof of knowledge. In our context we assume that one of the hidden attributes is a *revocation handle* $m_R \in \mathrm{GF}(q)$, that can be jointly generated by the user and the issuer (but without the issuer learning the value). We further assume that the user is able to calculate $P = h^{m_R}$ for any $h \in G$ and is able to prove that. That is, to use (statistical) zero knowledge proofs of knowledge that $DL_h(P)$ is equal to the revocation handle m_R hidden in the credential. Actually, for all systems in scope this is based on Schnorr's three pass protocol [38]. Compare [23, Appendix C]. We have also illustrated this in Appendix A. In the case of Idemix, the revocation handle is a hidden attribute. The revocation group G is used as the so-called pseudonym group described in [23, Section 5.1]. A similar context holds for the DL based U-Prove [9, Section 2.3.2]. In Appendix A we describe how both the FIDO and the German e-ID context can be supplemented with the described revocation handle. The hidden attribute m_R is then the private key of the user and the credential is the public key of the user relative to a verifier specific generator. We note that these supplements do not require changes to the core protocols of FIDO and the German e-ID.

3 A first, basic revocation mechanism

In this section we describe a first, basic revocation mechanism based on one of the revocation mechanisms chosen by the ABC4Trust project [2]. As many variants are possible we make it more concrete by choosing that user revocation should be effective within 24 hours as required in [17,13]. We use the concept of Idemix domain pseudonyms [23, Section 5.5] which are hashes into a group G based on the name of the service provider and signed with a hidden secret $m_R \in \mathrm{GF}(q)$ inside. To this end, we let $\mathcal{H}(.) : \{0,1\}^* \to G$ be a secure hash function. The pseudonyms take the form $\mathcal{H}(\mathrm{Str})^{m_R}$ for any string Str. As explained in Section 2 the user device is able to prove the correct form of these pseudonyms. The general revocation idea from ABC4Trust [2] is to force the user to prove that his credential is *not* revoked. This is achieved by forcing him to provide pseudonyms based on certain strings during the disclosure protocol and to let him prove they are correctly formed. This requires a secret key called revocation handle in [2], the term we also use. If the user wants to revoke its credential, he sends the revocation handle to a *revocation provider*. This provider periodically calculates all pseudonyms corresponding to revoked credentials and places them on a Credential Revocation List (CRL) available for service providers.

Our first revocation idea is to diversify the pseudonyms using strings consisting of a "date/serial number" combination. In more detail, as part of the revocation setup all participants have a common understanding of a *day number*, e.g. the number of days since January 1, 2000. Moreover we assume that

credentials have a validity period and that service providers can assess the validity of the credential. Let us denote the validity period of the credential in day numbers, e.g. $D_1, D_2, \ldots, D_v$ where v is the validity period in days. That is, $v = 1460$ for a four year validity period which in line with the practice. See [13] and [41].

Finally, we let w be the maximum number the card can be used on a daily basis. This should be a large number like $w = 1.000$ to avoid that the user cannot use its card. The revocation supplemented disclosure protocol we suggest runs as follows. A secured channel between the user and the service provider is setup and the service provider sends the day number d. The user device assesses that d (reasonably, see below) corresponds to the current date and that the day number is inside the validate period of its credential. If this assessment is negative an error will be returned. We note that in normal circumstances the service provider will not send a day number outside the validity period. We also note that an user device like a smartcard will typically not have a clock. In that situation we assume that the device stores each day number and validates that new ones sent are larger or equal to the stored ones. If the assessment was positive, the user will generate a random serial number (integer) s inside the interval $[0, w]$. The role of the serial number is to distinguish different sessions at the same day. So if this serial number was used before on the same day number d the user will generate a new serial number until the pair (d, s) is fresh. This implicitly assumes that the device (e.g. card) stores all serial numbers used on one day. A practical way of achieving this is to use an encryption function $\mathcal{E}_K(.) : [0, w] \rightarrow [0, w]$ based on a secret key K only known to the user device. Instead of generating a random serial number, the device then simply encrypts a daily sequence number (that starts with zero every day) and storing that. The key K would need to change daily, so it would be most practical to let K be diversified based on a card specific master key and the day number.

Next, as part of the disclosure protocol to a service provider the user also produces a "date/serial number" pseudonym which takes the form

$$P_{(d,s)} = (d, s, \mathcal{H}(d||s)^{m_R}).$$

Moreover, as part of the disclosure protocol the user provides the pseudonym and also proves that the hidden attribute m_R is also equal to the discrete logarithm of the third component of $P_{(d,s)}$ with respect to $\mathcal{H}(d||s)$. We have included the day number d in the pseudonym description only for clarity; the service provider is aware of d already.

If this disclosure protocol was successful the service provider then checks if the "date/serial number" pseudonym $P_{(d,s)}$ is on the CRL of day number d that is available from the revocation provider. If it is, the credential was revoked and the service provider will break off the protocol. If the user wants to revoke its credential, the "date/serial number" pseudonym of the user will be placed on each daily CRL as follows. The user sends its revocation handle m_R and the validity period of its credential to the revocation provider. During the remaining days of the validity period, the revocation provider will then calculate all w

"date/serial number" pseudonyms of that user and puts them on the daily CRL. This means that if the number of revoked (but not expired) credentials is z, the revocation provider will need to generate a CRL with $w \times z$ entries with an equal amount of calculations. See Section 6 for more practicabilities.

We will now show that the CRL entries of non-revoked credentials cannot be linked by introducing a game. With respect to the rules of this game, we observe that if two different users provide pseudonyms based on the same date and serial number, they can be distinguished as these pseudonyms will then be different. However, as long as pseudonyms based on different date and serial number combinations cannot be linked this is not a concern because this information does not accumulate. Theoretically this issue can be addressed by choosing w so large that the probability of this occurring is negligible.

The game we introduce is between a party P and an adversary A consisting of a learning phase and a final phase. In the learning phase the adversary can request any "date/serial number" pseudonym for any day d for a number of times polynomial in the size of q. P follows two possible scenarios. The first scenario is that P will consequently give a "date/serial number" pseudonym for one user, i.e. corresponding with one secret key m_R. The second scenario is that P will alternately give the "date/serial number" pseudonym for two users, i.e. corresponding with two different secret keys m_R, m'_R. We assume that P will never give a pseudonym based on the same date/serial number twice for the reason explained above. In the final phase of the game, A needs to state which of the two scenarios P chose. The next result states that the "date/serial number" pseudonyms do not allow linking of users.

Proposition 3.1 *We assume the random oracle model [5] and we assume that the hash function $\mathcal{H}$ behaves like a random oracle. If there exists a polynomially bounded algorithm that can win the above game with non-negligible probability of success, then the general DDH problem in G can be broken.*

Proof: We use the alternative formulation of the general DDH problem. That is, we are given any $h_1, h_2, \alpha, \beta \in G$ and we need to decide if $DL_{h_1}(\alpha) = DL_{h_2}(\beta)$. For each request i we generate a unique pair (s_i, d_i) and a unique $x_i \in_R \mathrm{GF}(q)$. For an even request we define the hash function to be equal to $h_1^{x_i}$ and let the pseudonym be equal to (s_i, d_i, α^{x_i}). For an odd request we let the hash function be equal to $h_2^{x_i}$ and the pseudonym be equal to (s_i, d_i, β^{x_i}). The scenario implicitly chosen by the simulation corresponds with the answer to the DDH problem with respect to $h_1, h_2, \alpha, \beta \in G$. This means that a polynomially bounded adversary that can predict the scenario chosen with non-negligible probability of success can break the general DDH problem in G. □

The basic setup discussed has a privacy drawback. Indeed, if service providers store the "date/serial number" pseudonyms of the user they are able to link all uses of the credential of the user in the past with the revocation handle m_R of the user. As this revocation handle is provided by the user to the revocation provider this effectively means that collusion between the revocation provider and the service providers would result in linkability issues. Of course, one might

consider that this risk is acceptable, but it does seem to be in contrast with the philosophy around the systems in scope.

Communication in the basic scheme and the variant described above, could be reduced if we let the service providers simply send $P_{(d,s)}$, to the revocation provider and request the status of the credential. The service provider would then do a lookup on the CRL and respond accordingly. This resembles the setup of the Online Certificate Status Protocol (OCSP) [26]. However, in this setup the privacy role of the revocation provider becomes much larger. Indeed, if he would store all requests of the service providers then he would be able to link all uses of the credential in the past of a user that revoked his card. That is, the revocation provider would become a single point of failure for user privacy. The most flexible approach would be to simply give the service providers the revocation handles m_R instead of the CRL. Indeed, this approach would allow them to calculate all pseudonyms corresponding to the given date and serial number and look for a match with the one given by the user. That is, to have a Verifier Local Revocation setup, cf. [32,33]. See also Section 5. The service provider would then also be able to calculate the CRL itself. Although flexible, this setup is particularly privacy unfriendly as this would allow the service providers to directly link all revoked card usages in the past.

In the next section we develop an extension of the basic scheme based on pairings. The extension then yields a flexible Verifier Local Revocation setup similar to the last approach but without the privacy unfriendliness. This will then also allow for all other approaches mentioned also.

4 The proposed revocation mechanism

We introduce three groups $G_1 = \langle g_1 \rangle$, $G_2 = \langle g_2 \rangle$ and $G_T = \langle g_T \rangle$ that are all of prime order q. It is assumed that the Diffie-Hellman Decision problem is hard in all these groups. We also introduce a *pairing* map $e : G_1 \times G_2 \to G_T$ that satisfies the following two properties:

Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{a \cdot b}$ for all integers a, b
Non-Degenerate: $g_T = e(g_1, g_2) \neq 1$.

We further assume that the pairing is of Type 3 meaning that there are no efficiently computable homomorphisms between G_1 and G_2. The group G_T will play the role of the revocation group from Section 2. In [22] a framework for pairing friendly elliptic curves is developed. This allows a flexible selection of groups G_1, G_2 and G_T and to make a trade-off between security and efficiency of computing the pairing.

A convenient choice is the Barreto-Naehrig curve BN254 from [4]. In this case G_1 is a group of points on an ordinary elliptic curve over $\mathrm{GF}(p)$, G_2 is a group of points on an ordinary elliptic curve over $\mathrm{GF}(p^2)$ and G_T is a multiplicative subgroup of $\mathrm{GF}(p^{12})$. Here p is a 254 bit prime number that is 3 modulo 4 making square root calculation convenient. The order q is also 254 bits in size. To prove a credential is not revoked, our techniques require the user (device) to

calculate at most one pairing operation. See below and Section 6 for practical considerations.

We also let $\mathcal{H}_1(.)$ (respectively $\mathcal{H}_2(.)$) be a one-way function $\{0,1\}^* \to G_1$ (respectively $\{0,1\}^* \to G_2$). A straightforward construction for such functions in the context of elliptic curves is probabilistic, cf. [28]. Here one uses a standard secure hash function to map the string to an element in the field over which the elliptic curve is defined and verifies if there exists a curve point with this x-coordinate. If this is not the case one varies the string in a deterministic fashion, e.g. by concatenating a string corresponding to an incrementing counter that starts with 1 and one tries again. Each try has a fifty percent of success so eventually one will find a point on the curve. A deterministic polynomial-time algorithm to embed strings in elliptic curves can be found in [37]. A simplified algorithm can be found in [8, Section 7.] which reduces the complexity of embedding a string in a curve to one quadratic residue check and one square root operation in GF(p). In the electronic passport specification [25, Appendix B] a further speedup of this algorithm is provided. The algorithm [8, Section 7.] can also be applied to elliptic curves over extension fields, e.g. over GF(p^2), and speedups similar to [25, Appendix B] can be devised. To summarize, the algorithm in [8, Section 7.] provides us with effective embeddings in both G_1 and G_2 in the context of Barreto-Naehrig curves.

We now use the same setup as in Section 3 but we let the "date/serial number" pseudonyms related to the pair (d, s) take the slightly alternative form

$$P_{(d,s)} = (d, s, e(\mathcal{H}_1(d||s)^{m_R}, \mathcal{H}_2(d))) = (d, s, e(\mathcal{H}_1(d||s), \mathcal{H}_2(d))^{m_R}). \quad (1)$$

As before, the user device calculates $P_{(d,s)}$ during the disclosure protocol and proves that the hidden attribute m_R is also equal to the discrete logarithm of the third element in $P_{(d,s)}$ with respect to $e(\mathcal{H}_1(d||s), \mathcal{H}_2(d))$.

The essential difference with the scheme in Section 3 is how the user revokes its credential. Instead of providing m_R to the revocation provider, it only provides a list of the *remaining* day keys, i.e. corresponding to the days D_x, $D_{x+1}, \ldots, D_v$. This list takes the form:

$$\{(D_x, \mathcal{H}_2(D_x)^{m_R}), (D_{x+1}, \mathcal{H}_2(D_{x+1})^{m_R}), \ldots (D_v, \mathcal{H}_2(D_v)^{m_R})\}. \quad (2)$$

The revocation provider creates a list of all day keys and publishes them. This provides a Verifier Local Revocation setup, cf. Section 5, allowing a service provider to validate if a pseudonym $P_{(d,s)}$ given by a user corresponds to a revoked credential. Indeed, based on the pseudonym the service provider can calculate

$$e(\mathcal{H}_1(d||s), H),$$

for every day key H on the day key list and compare that with the third element in $P_{(d,s)}$. As

$$e(\mathcal{H}_1(d||s), H) = e(\mathcal{H}_1(d||s), \mathcal{H}_2(D_d)^{m_R}) = e(\mathcal{H}_1(d||s), \mathcal{H}_2(D_d))^{m_R}$$

the "date/serial number" pseudonym calculated by the user coincides with the one calculated by the service provider. This validation requires the service provider to perform a number of pairings linear to the number of revoked credentials.

The Verifier Local Revocation setup allows any of the approaches mentioned at the end of Section 3 in a privacy friendly fashion. For instance, the revocation provider (or in fact any party) can create the CRL of day d consisting of all "date/serial number" pseudonyms for all revoked credentials by calculating expressions of the form

$$P_{(d,s)} = (d, s, e(\mathcal{H}_1(d||s), H)),$$

for every serial number s and key H on the list of the day. The revocation provider can publish the list and make them available for the service providers. In Section 6 we give estimations of the size of the list and show that it is manageable. The revocation provider (or in fact any party) can base a revocation check service on the CRL. Here a service provider would send a pseudonym to the revocation provider that would check if this is on the list. This is reminiscent of the Online Certificate Status Protocol (OCSP) [26]. As the day key list is not secret one can distribute the OCSP service over many cloud services by dividing the set of serial numbers.

From the first equality of Equation (1) it follows that the pseudonyms are a composition of the pseudonyms in Section 3 related to $\mathcal{H}_1$. It follows from Proposition 3.1 that these pseudonyms do not link users due to the presumed hardness of the Diffie-Hellman Decision problem in G_1. In the following proposition we show backward unlinkability: possession of the day keys does not enable to link the "date/serial number" pseudonyms after revocation with those before revocation. This property is due to the presumed hardness of the Diffie-Hellman Decision problem in G_2. So the hardness of both Diffie-Hellman Decision problems (in G_1 and in G_2) is essential in our scheme. As in Section 3 the result is based on a game between a party P and an adversary A consisting of a learning phase and a final phase. At the start of the game the adversary is given all day keys of U after a certain day d. The adversary can then request "date/serial number" pseudonyms before day d. P follows two possible scenarios. The first scenario is that P will consequently give the "date/serial number" pseudonym for user U, i.e. corresponding with revocation handle m_R. The second scenario is that P will give the "date/serial number" pseudonym for another user U', i.e. corresponding with a different revocation handle m'_R. In the final phase of the game, A needs to state which of the two scenarios P chose. The next result states backward unlinkabilty of the scheme.

Proposition 4.1 *We assume the random oracle model [5] and we assume that the hash functions $\mathcal{H}_1, \mathcal{H}_2$ behave like random oracles. If there exists a polynomially bounded algorithm that can win the above game with non-negligible probability of success, then the general DDH problem in G_2 can be broken.*

Proof: We use the alternative formulation of the general DDH problem in G_2. That is, we are given any $\gamma_1, \gamma_2, \alpha, \beta \in G_2$ and we need to decide if $DL_{\gamma_1}(\alpha) =$

$DL_{\gamma_2}(\beta)$. We define the hash function $\mathcal{H}_2$ on day d after the revocation day to be equal to $\gamma_1^{x_d}$ where $x_d \in \mathrm{GF}(q)$ is randomly chosen. We also provide the adversary with the day keys formed as α^{x_d} for all remaining days after the revocation. For each request i we generate a unique pair (s_i, d_i) and a unique $x_i \in_R \mathrm{GF}(q)$. In both scenarios we define the hash function $\mathcal{H}_1$ on $d||s$ to be equal to a random element g_i in G_1. In the first scenario we let $\mathcal{H}_2(d_i)$ to be equal to $\gamma_1^{x_i}$ and in the second scenario we let it be equal to $\gamma_2^{x_i}$. The pseudonym returned in the first scenario takes the form $(s_i, d_i, e(g_i, \alpha^{x_i}))$ and in the second it takes the form $(s_i, d_i, e(g_i, \beta^{x_i}))$. The scenario implicitly chosen by the simulation corresponds with the answer to the DDH problem with respect to $\gamma_1, \gamma_2, \alpha, \beta \in G_2$. This means that a polynomially bounded adversary that can predict the scenario chosen with non-negligible probability of success can break the general DDH problem in G_2. □

From the previous result it follows that one is not able to link the user credential usages *before* revocation. However, with the day keys one is able to link the credential usages *after* revocation. This could facilitate the user to see what happened with his card *after* revocation. A revocation provider would be the obvious position for such an inspection service. Moreover, if the user did discover the loss of its card only after some time, it is possible to identify the service providers that were visited with the card in the intermediary period between card loss and reporting. Indeed, the user then provides the day keys from the date the card was lost instead of from the date he found out.

5 Comparison with related work

In [31] another application of the ABC4Trust revocation mechanism is used based on pseudonyms diversified to time periods (epochs) and service providers identities. This is reminiscent to the German e-ID revocation setup and with its functional drawback that each service provider needs to be provided with a separate CRL. An additional drawback of [31] is that it needs to provide CRLs to service providers every epoch and the user can use his credential only once in an epoch as otherwise he will be linkable by his non-revoked-proving pseudonym. Further improvements are suggested in [31]. Although the privacy properties achieved are better than those of the German e-ID its practical usability seems worse.

In [29,30] an overview of various (Idemix) revocation techniques is given. Other than using credentials with short lifetimes it identifies three approaches for backward unlinkable revocation: Accumulators [15], Signature Lists [34], and Verifier Local Revocation [32,33]. The first two approaches require the user to download and process an entire set of update information before a revocation check can take place. As the size of this information is linear to the number of revoked credentials this can be time consuming for the user. In the situation of Signature Lists the main computational burden lies at the issuer and is linear to the number of revoked credentials. The actual revocation check for the user (on the updated Signature List) is in constant time. With Accumulators the

main computational burden lies at the user and is linear to the number of newly revoked credentials. This means that with Accumulators the user is burdened with both a download and an update process.

From the user perspective, Verifier Local Revocation seems more practical as it does not require the user to download or update its information to facilitate the revocation check prior to authentication. Here the burden of download and computation lies at the verifier, i.e. the service provider in our context. This complexity is linear to the number of revoked credentials. As performing pre-computation is not possible, the verifier needs to perform a computation linear to the number of revoked credentials as part of the user authentication process. This can result in a time consuming authentication process of the user. In addition to this drawback, it seems difficult to design Verifier Local Revocation schemes that are backward unlinkable. For this [29] refers to the group signature scheme with backward unlinkable revocation from [32] and its improvement in [33]. However [29] fails to explain how the user's group signature secret key would be bound to an Idemix credential or in fact any other privacy-enhanced public-key infrastructure mentioned in this paper. The actual conducted Verifier Local Revocation experiments in [29] are based on a variant of the VLR scheme from [10] which does not provide for backward unlinkable revocation.

Our technique can also be considered as a VLR mechanism with backward unlinkable revocation. Like the techniques from [32,33] our technique is based on pairing based cryptography. However, our technique is based on a Type III pairing whereas the technique of [32,33] is based on Type II pairings. Not taking into account the (scheme dependent) effort to prove credential ownership, our technique only requires elliptic curve multiplication and one pairing to be calculated by the user device. See Section 6, where we also show that the pairing calculation can in fact be avoided with the aid of a helping user computer. In contrast, the generation of a group signature and the proof that it is not revoked requires 6 exponentiations, 3 pairings and two isomorphism computations (inherent to a Type II pairing) with the technique of [33]. The verification of both schemes is similar and consists of a number of pairing operations linear to the number of revoked credentials. The verification of both schemes is similar and consists of a number of pairing operations linear to the number of revoked credentials. However, like Signature Lists our scheme also allows for pre-computation which faciliates that the verification of the user proof is simply a table lookup.

We conclude that, regardless whether the technique from [33] can actually be used as an convenient credential revocation mechanism, our technique is about twenty times more efficient on the user side. As a more personal opinion we also state we have more trust in the security of Type III pairings then in Type II pairings. This due to our believe that the isomorphism in Type II pairings will raise the risk of potential attacks. The security of our mechanism is based on the well-known Decision Diffie-Hellman assumptions whereas that of [33] is based on a weakening thereof (called q-SDH and DLDH).

6 Practical considerations and further applications

In the proposed mechanism of Section A the group G_T can be different from the group in which the basic scheme takes place. This means that the basic scheme does not need to become completely dependent on the security of paring friendly curves but only for the revocation. It would however, be most practical in the situation of U-Prove, FIDO and the German e-ID scheme to let the basic scheme take place in the group G_1.

As part of the proof that a credential is not revoked, the user device needs to calculate the pseudonym $P_{(d,s)}$ from Equation (1) and to prove it is well-formed. For this we have suggested Schnorr's three pass protocol for the systems in scope. Used in a straightforward way, this requires two pairings: one for formula Equation (1) and one in the Schnorr commitment of type

$$e(\mathcal{H}_1(d||s), \mathcal{H}_2(d))^r, \tag{3}$$

for a random r in GF(q). Compare Appendix A. However, the latter pairing one can be avoided by providing $\mathcal{H}_1(d||s)^r$ to the verifier and letting him calculate the pairing in Formula (3). For a less powerful device such as a smartcard, additional work can be done by a helping computer. For instance, all secret keys (including the revocation handle m_R) could be inside a contactless smartcard that is driven by an "APP" on a smartphone or a tablet. In this situation, the "APP" could send the day number d to the smartcard that would send back a fresh serial number after validation of the day number. The "APP" then calculates the expression $e(\mathcal{H}_1(d||s), \mathcal{H}_2(d))$ for the smartcard an let that be raised to the power m_R in G_T. In combination with the discussion above, we conclude that with the aid of a helping user computer, the calculation of pairings on the user device (smartcard) can be avoided. Not taking into account the (scheme dependent) effort to prove credential ownership, the total computational effort for the device is then limited to a 256 bit exponentiation in G_T and G_1. In the context of BN254 curves the latter corresponds to an elliptic curve multiplication in a curve over GF(p) which take about 50 milliseconds on a modern smartcard. Using [7] multiplicative operations in GF(p^{12}) can be estimated to be about 50 times slower than in GF(p). An exponentiation in GF(p) of size 256 bits corresponds to an RSA encryption of modulus size 256 bits and an exponent of 256 bits. This corresponds to a factor $(256/1.280)^2 \cdot 200/256 = 0,03125$ of an RSA encryption of modulus size 1280 bits and a random exponent of 200 bits. This is reported [39] in 2009 (!) to take 120 milliseconds on a Java smartcard. That is, we can estimate that an exponentiation in GF(p) of size 256 bits takes less than 4 millisecond and an exponentiation in GF(p^{12}) takes less than 200 milliseconds on a modern smartcard. With the aid of an helping computer it would thus take less than a quarter of a second in total to prove that a credential is not revoked.

One could also let all operations take place in the user device, i.e. without the help of an additional user computer. This would mean that the device would also calculate the embeddings in G_1 and G_2 as the verifier can not be trused with this. Based on the discussion above it follows that the device would only need

to calculate the pairing from Equation (1) and leave the commitment pairing to the verifier.

In [16], [36], [40] it is indicated that the calculation of pairings is within the reach of modern smartcards and embedded systems. From [16] it follows that already in 2007 it was possible to calculate BN254 pairing on a smartcard in 5 seconds. As demonstrated in 2014 [40] a plain Cortex-M0+ without any accelerators can compute a BN254 pairing in less than a second. It seems a safe assumption that a smartcard with suitable accelerators for big-integer arithmetic, such as the soon-to-be-released OpenCard [35] can compute such a pairing in less than half a second.

The list in (2) could in practice be formed during credential issuance and sent in encrypted form to the revocation provider as follows:

$$(D_x, \mathcal{E}'_{K_x}(\mathcal{H}_2(D_x)^{m_R})), (D_{x+1}, \mathcal{E}'_{K_{x+1}}(\mathcal{H}_2(D_{x+1})^{m_R})), \ldots, (D_v \mathcal{E}'_{K_m}(\mathcal{H}_2(D_v)^{m_R})).$$

Here $\mathcal{E}'_K(.)$ is a symmetric encryption under key K with the appropriate domain and where key K_{i+1} is the secure hash of its predecessor K_i. Ideally this list is generated by the user device where the symmetric encryption can be done outside the device. For a four year validity period this essentially corresponds to 1.460 elliptic curve multiplications in the curve over GF(p^2). Based on [7] we estimate that on average GF(p^2) arithmetic is about three times slower than GF(p) arithmetic. As elliptic curve multiplications in a curve over GF(p) take about 50 milliseconds on a modern smartcard, it takes about 4 minutes to calculate the above list on a modern smartcard. This seems acceptable as this is a one-time operation. If in this approach the user wants to revoke its credential on day $i \geq x$ he would only need to provide K_i as all other required keys follow from this. Moreover, the user would only need to store the first one, i.e. K_x (e.g. a 128 bit value, i.e. 32 HEX nibbles) as K_i can be calculated from this by repeated hashing. Such a value could be placed inside a user PIN mailer. For a four year validity period and using curve BN254, the encrypted list above would be of size less than 100 Kilobytes. We note that one could also let the issuer generate the list during card personalisation, although one then needs to trust the issuer not to store the revocation handle.

We now give an indication of the practical complexity of our techniques assuming credentials with a four year validity. About 13 million people of the Dutch population is over 20 years of age. From [41, Figure 12] we estimate that 10 percent of all non-expired credentials is revoked at a given day. This means that if all Dutch people over 20 would have a credential, about 1,3 million non-expired credentials would have a revoked status at a given day. Based on [3], [7] we estimate that we can calculate 4.800 BN254 pairings a second on the four cores of an Intel i7 2.8GHz processor. This means it would take a service provider about five minutes to validate a credential by running through the list of day keys. If the list would contain the expiry date (available in the German e-ID card) this would reduce the time with a factor of 1.460 and the check can be done under half a second.

For a large serial number range of 1.000 the calculation of a daily CRL would require 1,3 billion pairings. So the computational capacity of four Intel

i7 computers would suffice to calculate the daily CRL for the Netherlands. As the day keys are non-secret, this work can be robustly performed through cloud services where it represents little effort. Here we have not taken into account that the revocation provider can work in advance by calculating pseudonyms for existing revoked credentials. Ballpark estimates show that this approach can lower the daily computational effort by half. In the context of BN254 elements in G_T will be of size 12 x 254 = 3.048 bits. This size can be reduced by using secure hash values of these elements instead. A 128 (16 bytes) bit hash output will suffice to withstand birthday paradox based issues. The daily CRL will then be about 20 Gigabytes in size which is manageable, especially as this work can be placed in the cloud. Especially for pan European e-ID cards such as envisioned in [17] this is an interesting add-on as it gives users an indication of the intended fraud of the card taking place in another country through the national PEPS.

So far we have only considered revocation, i.e. blacklisting. The same techniques can be used for suspension, e.g., automatically suspending a card for business use during weekends, and white listing. The German e-ID [12, Section 2.8.4] introduces white lists in case the shared keys it uses get compromised. These are similar to the black lists (CRLs) and German government already possess these revocation keys and has the ability to link the pseudonyms further hampering privacy of German citizens. To support white listing, we envision an 'existence provider' next to the revocation provider. This provider assures a service provider that a pseudonym (card) 'exists', i.e. is not forged. This only needs to be done once, during user registration at each service provider. The existence provider envisioned consists of two, strictly separated parts: a white list producer and an on-line existence service. The list producer is provided with the 1 January 2000 day keys of all (new) batches of produced cards. These day keys are then used to produce and maintain a white list consisting of the hashed pseudonyms for all serial numbers. This white list and updates (new cards) are then provided to the on-line service. The white list also contains the expiry date of the cards and the on-line service removes expired cards. To prove existence, the user card generates a fresh, random serial number s and provides the "date/serial" pseudonym for 1 January 2000 with respect to s. This is then sent to the on-line provider who will do a look up on the white list and respond accordingly. For 13 million users registered the white list is about 4 Terabytes in size for the serial number range of 10.000, which is manageable and can be distributed over several cloud services.

7 Conclusion

We have described a revocation mechanism for the FIDO, German e-ID and the Idemix and U-Prove systems that is flexible, efficient, timely and backward unlinkable. This also allows for a central revocation provider and card white listing in a privacy friendly fashion.

References

1. American National Standards Institute, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), ANSI X9.62-2005, November 2005.
2. ABC4Trust project, D2.2 - Architecture for Attribute-based Credential Technologies, version 1.0, August 2014.
3. D.F. Aranha et al., Faster Explicit Formulas for Computing Pairings over Ordinary Curves, EUROCRYPT 2011, LNCS 6632, Springer-Verlag, pp. 48-68.
4. Barreto, P., M. Naehrig, Pairing-Friendly Elliptic Curves of Prime Order, SAC 2005, LNCS 3897, Springer-Verlag, pp.319-331.
5. M. Bellare, P. Rogaway, Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, ACM Conference on Computer and Communications Security, 1995, pp.6273.
6. J. Bender et al., Privacy-friendly revocation management without unique chip identifiers for the German national ID card, Computer Fraud & Security, September 2010.
7. J.-L. Beuchat et al., High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves, Pairing 2010, LNCS 6487, Springer-Verlag, pp. 21-39, 2010.
8. E. Brier et al., Efficient Indifferentiable Hashing into Ordinary Elliptic Curves, CRYPTO 2010, Springer-Verlag, LNCS 6223, pp. 237-254, 2010
9. S. A. Brands, Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy, MIT Press, Cambridge, MA, USA, 2000.
10. E. Brickell et al., Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, ACM, New York, pp. 132145, 2004.
11. Bundesamt für Sicherheit in der Informationstechnik (BSI), Technical Guideline TR-03110-2 Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token - Part 2 - Protocols for electronic IDentification, Authentication and trust Services (eIDAS), Version 2.20, 20. February 2015.
12. Bundesamt für Sicherheit in der Informationstechnik (BSI), Technical Guideline TR-03110-3 Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token - Part 3 - Common Specifications Version 2.20, February 2015.
13. CAB Forum, Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates, v.1.3.3, February 2016. Available from https://cabforum.org.
14. J. Camenisch, A. Lysyanskaya, An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation, Proceedings of Eurocrypt 2001, LNCS 2045, Springer-Verlag, pp. 93-118, 2001.
15. J. Camenisch, A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials, CRYPTO 2002, LNCS 2442, Springer-Verlag, pp. 101120, 2002.
16. A. J. Devegili et al., Implementing Cryptographic Pairings over Barreto-Naehrig Curves, Pairing 2007, LNCS 4575, Springer-Verlag, pp.197-207, 2007.
17. European Union, REGULATION No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC.

18. European Union, Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation), 25 January 2012.
19. A. Fiat, A. Shamir. How to prove yourself: Practical solutions to identifcation and signature problems, CRYPTO '86, LNCS 263, Springer Verlag, pages 186-194.
20. FIDO Alliance, FIDO Security Reference, May 2015. Available from https://fidoalliance.org/specifications/download/.
21. FIDO Alliance, FIDO U2F Implementation Considerations, May 2015. Available from https://fidoalliance.org/specifications/download/.
22. D. Freeman et al., A Taxonomy of Pairing-Friendly Elliptic Curves, Journal of Cryptology, April 2010, Volume 23, Issue 2, pp 224-280.
23. IBM Research, Specification of the Identity Mixer Cryptographic Library, version 2.3.4 February 10, 2012. Available from http://www.zurich.ibm.com/idemix/release.html.
24. IETF, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, March 2010. Available from http://www.ietf.org/.
25. International Civil Aviation Organization (ICAO), Security Mechanisms for MRTDs, Machine readable travel documents part 11, version 7, 2015. Available from http://www.icao.int.
26. IETF, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, Request For Comments 2560, June 1999. Available from https://www.ietf.org.
27. I Reveal My Attributes (IRMA), see https://www.irmacard.org/.
28. N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation 48, pp 203-209, 1987.
29. J. Lapon, Anonymous Credential Systems: From Theory Towards Practice, PhD thesis, K.U.Leuven, 2012.
30. J. Lapon et al., Analysis of revocation strategies for anonymous Idemix credentials, Communications and Multimedia Security, LNCS 7025, pp.317.
31. W. Lueks et al., Fast Revocation of Attribute-Based Credentials for Both Users and Verifiers, ICT Systems Security and Privacy Protection, Volume 455 of the series IFIP Advances in Information and Communication Technology pp. 463-478, 2015.
32. T. Nakanishi, N. Funabik, Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps, ASIACTYPT 2005, LNCS 3788, Springer-Verlag, pp.533548.
33. T. Nakanishi, N. Funabik, A Short Verifier-Local Revocation Group Signature Scheme with Backward Unlinkability, International Workshop on Security 2006, LNCS 4266, Springer-Verlag, pp. 1732, 2006.
34. T. Nakanishi et al., Revocable group signature schemes with constant costs for signing and verifying, PKC 2009, LNCS 5443, Springer-Verlag, pp. 463480,2009.
35. Announcement of OpenCard. See https://www.cryptoexperts.com/services/details/125.
36. M. Scott et al., Implementing Cryptographic Pairings on Smartcards, Cryptographic Hardware and Embedded Systems - CHES 2006 LNCS 4249, Springer-Verlag, pp 134-147, 2006.
37. A. Shallue, A., C. van de Woestijne, Construction of rational points on elliptic curves over finite fields, ANTS , LNCS 4076, Springer-Verlag, 2006.
38. C. P. Schnorr, Efficient signature generation for smart cards. Journal of Cryptology, 4(3):239-252, 1991.

39. H. Tews, B. Jacobs, Performance Issues of Selective Disclosure and Blinded Issuing Protocols on Java Card, WISTP 2009, LNCS 5746, Springer-Verlag, pp. 95111, 2009.
40. T. Unterluggauer, E. Wenger, Efficient Pairings and ECC for Embedded Systems, Cryptographic Hardware and Embedded Systems, CHES 2014 LNCS 8731, Springer-Verlag, pp 298-315, 2014.
41. D. Walleck et al., Empirical Analysis of Certificate Revocation Lists, Data and Applications Security XXII, LNCS 5094, Springer-Verlag, pp 159-174, 2008.
42. E. Verheul, Evidence that XTR is more secure than supersingular elliptic curve cryptosystems, Journal of Cryptology 17(4), pp. 277-296, 2004.

A Supplementing FIDO and the German e-ID with a revocation handle

In this appendix we describe the required supplements to the FIDO and German e-ID contexts to allow for a revocation handle as described in Section 2. This handle will in fact coincide with a private key already existing on the card in these contexts. In current practice card private keys are often generated by the issuer during card personalization for simplicity. However both the FIDO and German e-ID setup allow generation of such keys by the user device itself, e.g., as part of first usage. This would reduce linkability risks resulting from issuer compromise.

We start with the explanation for FIDO [20,21]. For each service provider the FIDO user device is required to generate a unique persistent public/private key pair meant for signing. The resulting public keys cannot be cryptographically linked. The FIDO specifications include the usage of the Elliptic Curve Digital Signature Algorithm (ECDSA) [1] in an elliptic curve group $\Gamma = \langle\gamma\rangle$ of prime order q. In a straightforward FIDO implementation one makes the public/private key pair unique by using a fixed generator of Γ, e.g. the basepoint γ, and randomly choosing the private key $x \in \mathrm{GF}(q)$. This results in a public/private key of the form (x, γ^x). Another way to create a unique pair is to let the user device randomly generate a private key x only once and then to securely choose a unique generator for each service provider. This private key will serve both as the authentication key as the revocation handle. To reflect this we denote it by m_R.

To further specify, let $\mathcal{H}(.) : \{0,1\}^* \to \Gamma$ be a secure hash function. Compare the discussion at the beginning of Section . We let the generator $\gamma_{SP} \in G$ for a service provider be formed as the hash value of a string Str identifying the service provider, e.g., its domain name from its TLS server certificate. That is, $\gamma_{SP} = \mathcal{H}(Str)$. We then let the public key be $\pi = \gamma_{SP}^{m_R}$ where m_R is the revocation handle. That is, the user uses the same private key for all service providers but with different generators. Note that the FIDO token does not need to store these generators as it calculate them on the fly. This means that the FIDO token becomes stateless which is of independent value. Also note that by letting the Decision Diffie-Hellman problem be intractable in Γ these public keys are not linkable. This now fits in the context described in Section 2. Indeed,

if $G = \langle g \rangle$ is a group of order q. Then for any group G of the same order and $h \in G$ the user can calculate $p = h^{m_R}$ and can prove with Schnorr's three pass protocol [38] that $DL_h(p)$ is equal to the revocation handle m_R hidden in the public key. Indeed, the user device then generates $r \in_R \mathrm{GF}(q)$ and provides $A_1 = \gamma_V^r$ and $A_2 = h^r$ to the service provider as a commitment. The service provider then generates a random $w \in \mathrm{GF}(q)$ and provides that to the user device. The user device then returns $z = w \cdot m_R + r$ to the service provider that then checks if $\gamma_V^z = A_1 \cdot \pi^w$ and $h^z = A_2 \cdot p^w$. We mention that in practice one typically uses non-interactive variants of this protocol, cf. [19].

The "Restricted Identification" (RI) protocol in the German e-ID [11, Section 3.6] is quite similar to the FIDO setup described above. It is also based on a Elliptic Curve group $G = \langle g \rangle$ of some prime order q called Brainpool [24]. The user e-ID card contains a private key known as $SK_{ICC1} \in \mathrm{GF}(q)$, cf. [11, Section 3.5]. It is not specified where this private key can be generated but it can be generated by the e-ID card as part of first usage. The service provider pseudonym takes the form of a hash of $PS_{SP} = PK_{SP}^{SK_{ICC1}} \in G$ where $PK_{SP} \in G$ is the public key of the service provider. This is very close to the FIDO setup we described. If a user wants to prove its RI pseudonym is not revoked (in a backwards unlinkable fashion), it will reveal PS_{SP} which places him in the context of Section 2. We remark that revealing PS_{SP} is already part of the e-ID card as an extension of the "Restricted Identification" protocol called "Pseudonymous Signatures". This protocol allows an e-ID card to additionally bind signatures to its pseudonym. See [11, Section 3.6].