A Short Paper on How to Improve U-Prove Using Self-Blindable Certificates*

Lucjan Hanzlik and Kamil Kluczniak

Faculty of Fundamental Problems of Technology,
Wrocław University of Technology
{firstname.secondname}@pwr.wroc.pl

Abstract. U-Prove is a credential system that allows users to disclose information about themselves in a minimalistic way. Roughly speaking, in the U-Prove system a user obtains certified cryptographic tokens containing a set of attributes and is able to disclose a subset of his attributes to a verifier, while hiding the undisclosed attributes. In U-prove the actual identity of a token holder is hidden from verifiers, however each token has a static public key (i.e. token pseudonym), which makes a single token traceable, by what we mean that, if a token is presented twice to a verifier, then the verifier knows that it is the same token. We propose an extension to the U-Prove system which enables users to show U-Prove tokens in a blinded form, so even if a single token is presented twice, a verifier is not able to tell whether it is the same token or two distinct tokens. Our proposition is an optional extension, not changing the core of the U-Prove system. A verifier decides whether to use issuer signatures from U-Prove, or the blind certificates from the extension.

Key words: U-Prove, Anonymous Credentials, Self-Blindable Certificates

1 Introduction

David Chaum in [1] sketched some of the problems related to identity certificates. One of them is that service providers are able to track the activity of users. The idea to hide the actual identity of a user is based on pseudonyms. A pseudonym is a unique identifier by which a user can authenticate against some parties in the system. Typically pseudonyms are issued by service providers in order to blind the actual identity of a user. Pseudonymity can be differently understood. In some systems users appear under just one pseudonym which sometimes is called a token. Other systems provide unique pseudonyms for a user which are different in distinct service providers and even if these service providers cooperate, the pseudonyms cannot be linked. This means that having two or more pseudonyms it is infeasible to decide whether the pseudonym is related to one user or many different users. The notion of unlinkability was also described in [1] and can be differently understood. One situation is, as described above, when a user presents different pseudonyms in different domains and the identity cannot

* This research was conducted within project 2012-9/4 of the Ventures programme of Foundation for Polish Science, cofinanced from European Union, Regional Development Fund.

be linked between this domains, however within one domain a user appears under just one pseudonym and thus is tracable in it [2]. Another situation is when each authentication session provides a new pseudonym. We will call the second situation untraceability since the data which a user passes during two or more authentication sessions are unlinkable, so it cannot be used to trace the activity of a particular user.

Generally, the anonymity notion appears in a range of different protocols and schemes. The main goal of group signatures [3,4,5] , for instance, is to identify that a signer belongs to a group and the signatures made by any group member are unlinkable in the sense that, a verifier checks only if a signature was made by a relevant group member, but it is infeasible to determine who exactly produced that signature.

A similar notion of anonymity can be observed in anonymous credential systems where a user can prove different statements about himself, but without revealing any other information to a verifier. Such credential systems based on CL-Signatures [6] were designed in [7] and are constructed for algebraic groups of unknown order. Another credential system, designed by Microsoft, is called U-Prove [8] where a user obtains authentication tokens and is able to proof statements about himself, which are contained in that token. The token contains a public key, so in some sense it is a pseudonym of a user, and an issuer certificate on that public key. Presenting one U-Prove token twice or more requires to show the token public key and the certificate in an unblinded form, so a set of verifiers can easily track a single token.

In this short paper we study the possibility to improve the U-Prove credential system by providing the untraceability property for a U-Prove token. So in effect, many presentations of a single token should be unlinkable. We believe that an interesting building block introduced by Verhuel in [9], called self-blindable certificate, can naturally provide the untraceability property for credential systems such as U-Prove. The idea behind a self-blindable certificate is, that a issuer generates a certificate under a users public key, and the user can present such certificate in an blinded form to a verifier.

Contribution We show an extension to the Microsoft U-Prove credential system providing the untraceability property for U-Prove tokens using self-blindable certificates. In short, instead of obtaining a linkable certificate on a token, we issue a self-blindable certificate on the token public key, so a token holder can show statements related to the token without revealing the token public key, i.e. show the token public key in an blinded form and prove that it is genuine by showing the blinded certificate. In effect two or more authentication sessions become unlinkable and verifiers cannot track one particular token. Our extension don't changes the U-Prove system substantially. A verifier can choose, depending on his intend, whether to verify the standard U-Prove certificate or the self-blindable certificate. In the first case, the protocols goes unchanged as described in the specification [8]. When self-blindable certificates are used, then some steps of the protocol are modified. First in section 2 we describe a construction for self-blindable certificates from [9]. Then, we give a high-level description of the U-Prove system and indicate the changes between our contribution and the original protocol in section 3. Finally in section 4 we give a brief security analysis of our proposed extension.

2 Self-Blindable Certificates

We first recall the definition of *Self-Blindable Certificates* as described in [9] by Verheul. Then, we present a construction that implements this definition.

2.1 Definition

We assume that the system consists of users and a trust provider. We define a certificate on a users public key $P_U \in \mathcal{U}$, signed with the trust providers secret key S_T, as:

$$\{P_U, Sig(P_U, S_T)\}.$$

Let $\mathcal{C}$ be the set of all possible certificates and let F be a set called *transformation factor space*. We call the certificates $\mathcal{C}$ *self-blindable* if there exist a efficiently computable *transformation map* $D : \mathcal{C} \times F \to \mathcal{C}$ such that:

- For any certificate $C \in \mathcal{C}$ and $f \in F$ the certificate $D(C, f)$ is signed with the same trust provider secret key as the certificate C.
- Let C_1, C_2 be certificates and let $f \in F$ is known. If $C_2 = D(C_1, f)$ then one can efficiently compute a transformation factor $f' \in F$ such that $C_1 = D(C_2, f')$.
- The mapping $D(.,.)$ induces a mapping $D' : \mathcal{U} \times F \to \mathcal{U}$ namely if C_1, C_2 are certificates on a users public key P_U, then $D(C_1, f)$ and $D(C_2, f)$ are certificates for the public key $D'(P_U, f)$, for any transformation factor $f \in F$.
- Let P_U be the public key of a user and let $f \in F$ be a transformation factor known by the user. If the user possesses the private key for P_U, then the user also knows the private key for $D'(P_U, f)$.
- If the users public key P_U is fixed and the transformation factor f is a uniformly random element in F, then $D'(P_U, f)$ is a uniformly random element in $\mathcal{U}$.

2.2 Instantiation

Definition 1. *Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ be cyclic groups of prime order q. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a map with the following properties:*

- *for $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_q$, we have $e(aP, bQ) = e(P, Q)^{a \cdot b}$,*
- *if P is a generator of $\mathbb{G}_1$ and Q is a generator of $\mathbb{G}_2$, then $e(P, Q)$ generates $\mathbb{G}_T$,*
- *there is an efficient algorithm to compute $e(P, Q)$ for $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$.*

We now say that the function e is a:

- Type 1 pairing function if $\mathbb{G}_1 = \mathbb{G}_2$,
- Type 2 pairing function if $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct groups and there exists a efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$,
- Type 3 pairing function if $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct groups and there is no known isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.

Type 1 pairing is also called symmetric, because $\mathbb{G}_1 = \mathbb{G}_2$. Type 2 and type 3 are called asymmetric.

From now on, we will only use the multiplicative notation (even when the group is additive) to simplify the description and to remain compatible with the U-Prove Crypto Specification V1.1 [8].

Construction We present the second construction from [9]. However, we extend the construction to type 2 pairing (so the security analysis in [9] is valid) and not only type 1 as is the original description.

In this case we define the set $\mathcal{U}$ as $\mathbb{G}_1^3$, the transformation factor space F as $\mathbb{Z}_q^2$ and the certificate space $\mathcal{C}$ as $\mathbb{G}_1$. In addition let P_1 be the generator of $\mathbb{G}_1$ and P_2 be the generator of $\mathbb{G}_2$.

Let $z, f \in \mathbb{Z}_q$ be the private key of the trust provider and let r, r^f, h, h^z (for random $r, h \in \mathbb{G}_2$) be his public key. The users public key takes the following form: $(g_1, g_2, g_1^{x_1} g_2^{x_2})$, where g_1 is a random element in $\mathbb{G}_1$, $g_2 = g_1^f$ and (x_1, x_2) is the private key of the user. The certificate for the users public key is $(g_1^{x_1} g_2^{x_2})^z$. The certificate can be easily verified by checking if:

$$e(g_1^{x_1} g_2^{x_2}, h^z) \stackrel{?}{=} e((g_1^{x_1} g_2^{x_2})^z, h) \quad \text{and} \quad e(g_1, r^f) \stackrel{?}{=} e(g_2, r)$$

and by verifying that the user knows x_1 and x_2, which can be checked using the Okamoto variant of Schnorr's identification scheme [10].

Note that, for a random $(k, l) \in F$, $D(.,.)$ and $D'(.,.)$ defined as follows:

$$\begin{aligned} D((g_1^{x_1} g_2^{x_2})^z, (k, l)) &= (g_1^{x_1} g_2^{x_2})^{z \cdot l \cdot k}, \\ D'((g_1, g_2, g_1^{x_1} g_2^{x_2}), (k, l)) &= (g_1^l, g_2^l, (g_1^{x_1} g_2^{x_2})^{l \cdot k}) \end{aligned}$$

fulfil the above definition of self-blindable certificates.

3 Our Contribution

In this section, we will present our extension. We describe it by embedding it into the U-Prove Crypto Specification V1.1 [8]. Due to space reasons we only show a sketch of the system. Thus, we advise to read this section in conjunction with [8]. Our extension is an optional feature. In short, a token issuer makes an additional self-blindable certificate on the tokens public key. In the proof generation and verification a user or verifier, depending on the use case, can choose whether to show the standard signature specified in [8] or the self-blindable certificate from our proposed extension. We will denote as **[Standard]** the situation when the signature from [8] is used, and as **[Blinding]** when the self-blindable certificate is used. An exception from this is the issuing phase, where both certificates are issued to the token holder.

3.1 System Parameters

The system parameters consist of the standard U-Prove parameters:

$$\begin{aligned} IP = (&UID_P, desc(\mathbb{G}_1), UID_{\mathcal{H}}, (g_0, g_1, \ldots, g_n, g_t), \\ &(e_1, \ldots, e_n), (z_0, z_1, \ldots, z_n, z_t), S) \end{aligned}$$

where

- UID_P is a unique identifier of the token,
- $desc(\mathbb{G}_1)$ is the description of a group of prime order q with a generator $g \in \mathbb{G}_1$

- $UID_{\mathcal{H}}$ is the specification of the hash function $\mathcal{H}$,
- $(g_0, g_1, \ldots, g_n, g_t)$ is the issuers public key, where y_0 is private, $g_0 = g^{y_0}$ and $g_1, \ldots, g_t$ are random group generators.
- $(e_1, \ldots, e_n)$ list of byte values indicating whether or not the attribute values $A_1, \ldots, A_n$ are hashed computing an UProve token.
- $(z_0, z_1, \ldots, z_n, z_t)$ for each $i \in \{1, \ldots, n, t\}$, $z_i = g_i^{y_0}$.
- S - specification for the issuer parameters.

and the additional extension parameters:

$$IP_{\textbf{[Blinded]}} = (q, p, p^r, \mathbb{G}_2, \mathbb{G}_T, e, p_0, p_1).$$

where $\mathbb{G}_2$ is a cyclic group of order q generated by p, r is random in $\mathbb{Z}_q$, e is a Type 2 pairing in sense of Definition 1, $p_0 = p^{r \cdot z}$, $p_1 = p^f$ and (z, f) is the issuers secret key.

3.2 Issuing U-Prove Token

The issuing protocol is similar to the one in the specification [8]. In the issuing procedure the user receives a U-Prove token of the form:

$$\mathcal{T} = (UID_P, h, TI, IP, (\sigma'_z, \sigma'_c, \sigma'_r)_{\textbf{[STANDARD]}}, (\mathcal{B})_{\textbf{[BLINDED]}}).$$

During the issuing procedure the user generates a private key $\alpha \in \mathbb{Z}_q$ which is associated with the public key $h = (g_0 g_1^{x_1} \ldots g_n^{x_n} g_t^{x_t})^{\alpha}$ of the token $\mathcal{T}$. The values σ'_z, σ'_c and σ'_r form the issuer signature on the public key h.

In our extension, a user obtains a self-blindable certificate on the tokens public key h. The issuer computes $h_2 = h^f$. The user then chooses two private keys b_1 and b_2, computes a value $h^{b_1} h_2^{b_2}$ on which the issuer makes his signature using his private key z. Finally, the self-blindable certificate with the corresponding public key, obtained by the user is of the form $\mathcal{B} = (h, h_2, h^{b_1} h_2^{b_2}, (h^{b_1} h_2^{b_2})^z)$ and his private keys associated to the certificate are b_1 and b_2.

3.3 Presenting U-Prove Token

In this subsection we describe the proof presentation procedure.

Input:

1. Disclosed attributes: $D \subset \{1, \ldots, n\}$,
2. Undisclosed attributes: $U \subset \{1, \ldots, n\} \backslash D$,
3. U-Prove token: $\mathcal{T} = (UID_P, h, TI, IP, (\sigma'_z, \sigma'_c \sigma'_r)_{\textbf{[STANDARD]}}, (\mathcal{B})_{\textbf{[BLINDED]}})$,
4. Message: $m \in \{0,1\}^*$,
5. Private key: α,
6. Attribute values: $(A_1, \ldots, A_n) \in (\{0,1\}^*)^n$.

Proof Generation:

1. For each $i \in U$, generate $w_i \in \mathbb{Z}_q$ and generate $w_0 \in \mathbb{Z}_q$,
2. **[Standard]** Compute $a = \mathcal{H}(h^{w_0}(\prod_{i \in U} g_i^{w_i}))$, or

2. **[Blinded]** Choose a random blinding l and compute $a = \mathcal{H}(h^{w_0 \cdot l}(\prod_{i \in U} g_i^{w_i}))$
3. $x_t = ComputeXt(IP, TI)$,
4. For each $i \in \{1, \ldots, n\}$, $x_i = ComputeXi(IP, A_i)$,
5. **[Blinded]** Compute the blinded token
   (a) Blind the U-Prove public key $B_1 = h^l$, where l is chosen randomly,
   (b) Blind the certificate for the token by computing $B_2 = h_2^l$, $B_3 = (h^{b_1} h_2^{b_2})^{l \cdot k}$ and $B_4 = ((h^{b_1} h_2^{b_2})^z)^{l \cdot k}$, where k is chosen randomly.
   (c) Choose r_1, r_2 at random, and compute additionally $B_1' = B_1^{r_1}$ and $B_2' = B_2^{r_2}$.
   (d) The blinded certificate consists of $\mathcal{B}_b = (B_1, B_2, B_3, B_4, B_1', B_2')$
   (e) Set the blinded token as $\mathcal{T} = (UID_P, TI, IP, \mathcal{B}_b)$ (note that the blinded U-Prove token is contained in $\mathcal{B}_b$).
5. $c = GenerateChallenge(IP, \mathcal{T}, a, m, D, \{x_i\}_{i \in D})$,
6. **[Standard]** Compute $r_0 = c\alpha^{-1} + w_0$, or
6. **[Blinded]**
   (a) Compute $r_0 = c\alpha^{-1} \cdot l^{-1} + w_0$,
   (b) Compute $s_1 = r_1 - c \cdot k \cdot b_1$ and $s_2 = r_2 - c \cdot k \cdot b_2$
7. Compute $r_i = -cx_i + w_i$ for each $i \in U$, where w_i is chosen randomly,
8. Return the U-Prove token proof $(\{A_i\}_{i \in D}, a, r_0, \{r_i\}_{i \in U})$.
8. **[Blinded]** Additionally, return s_1 and s_2.

3.4 Verifying U-Prove Token

Input

1. Issuer parameter fields IP and if the token is blinded then additionally $IP_{\textbf{[Blinded]}}$.
2. Ordered indices of disclosed attributes: $D \subset \{1, \ldots, n\}$,
3. Ordered indices of undisclosed attributes: $U \subset \{1, \ldots, n\} \setminus D$,
4. The UProve token in form
   - **[Standard]** $\mathcal{T} = (UID_P, h_1, TI, IP, \sigma_z', \sigma_c', \sigma_r')$, or
   - **[Blinded]** $\mathcal{T} = (UID_P, TI, IP, \mathcal{B}_b)$.
5. The presentation proof $(\{A_i\}_{i \in D}, a, r_0, \{r_i\}_{i \in U})$,
6. **[Blinded]** The proof of knowledge s_1, s_2.

Proof Verification:

1. **[Standard]** Run the $VerifyTokenSignature(IP, \mathcal{T})$ procedure which verifies $(\sigma_z', \sigma_c', \sigma_r')$ (see [8]), or
1. **[Blinded]** Run $VerifySelfBlindableCertificate(\mathcal{B}_b, \mathcal{T}, s_1, s_2)$.
2. $x_t = ComputeXt(IP, TI)$,
3. For each $i \in D$, $x_i = ComputeXi(IP, A_i)$,
4. Set $c = GenerateChallenge(IP, \mathcal{T}, a, m, D, \{x_i\}_{i \in D})$,
5. **[Standard]** Extract $k = h$ from $\mathcal{T}$, or
5. **[Blinded]** Extract $k = B_1$ from $\mathcal{T}$,
6. Verify that $a \stackrel{?}{=} \mathcal{H}((g_0 g_t^{x_t} \prod_{i \in D} g_i^{x_i})^{-c} k^{r_0} (\prod_{i \in U} g_i^{r_i}))$.

3.5 Verify Self Blindable Certificate

Having as input the system parameters and the issuers parameters $IP_{\textbf{[Blinded]}}$, the challenge c the blinded token $\mathcal{T}$ in particular the values B_1, B_2, B_3, B_4, B'_1 and B'_2, check the following proof of knowledge:

$$B_1^{s_1} B_2^{s_2} B_3^{-c} \stackrel{?}{=} B'_1 B'_2$$

Now, check whether the certificate was indeed issued by the issuer by verifying the following equations:

$$e(B_3, p_0) \stackrel{?}{=} e(B_4, p^r) \quad \text{and} \quad e(B_1, p_1) \stackrel{?}{=} e(B_2, p)$$

4 Security Analysis

Since this is a work in progress, we only present an intuition for the security proof. In particular, we show two things:

1. the adversary, without any U-Prove tokens, cannot create an U-Prove token that passes the verification,
2. having k U-Prove tokens, the adversary cannot forge a $k+1$ U-Prove token which is different then each of the k tokens he possesses and that will pass the verification.
3. having k tokens, the adversary cannot create an U-Prove token which is different then each of the k tokens he possesses and that will pass the verification.

The first statement covers the case when the adversary, without any knowledge of U-Prove tokens in the system, would like to exploit the extension to pass the verification. On the other hand, the second statement covers the case when the adversary would like to exploit the extension to change some attributes in his U-Prove tokens.

Let us first assume that there exists an adversary that without access to any U-Prove token, creates a U-Prove token that passes the verification. However, then we can use such adversary to forge the underlying self-blindable certificates. Thus, since the self-blindable certificates presented in subsection 2.2 are secure against forgery, as shown in [9], so is our extension.

Now we show that the second statement is valid. Let for $i \in \{1, \ldots, k\}$:

$$(h_i, h_{2,i}, h_i^{b_{1,i}} h_{2,i}^{b_{2,i}}, (h_i^{b_{1,i}} h_{2,i}^{b_{2,i}})^z)$$

be the U-Prove token extensions known to the adversary. Note that h_i is the U-Prove tokens public key which contains all attributes. Without loss of generality we assume that the adversary would like to change some attributes in token $i = 1$. We will now show how he can change the token $(h_1, h_{2,1}, h_1^{b_{1,1}} h_{2,1}^{b_{2,1}}, (h_i 1 b_{1,1} h_{2,1}^{b_{2,1}})^z)$ and the contained in it attributes, in such a way that it will pass the verification. Obviously, he can blind this token according to the protocol but then the token contains the same attributes. According to the security proof of the used self-blindable certificates (see appendix in [9]) the adversary can only change h_1 (the tokens public key) in such a way that $h_1 = \prod_{i \in I} h_i^{r_i}$, for $I \subset \{1, \ldots, k\}$ and r_i are known to the adversary.

Let us now assume that $|I| = 2$. It follows that h_1 is of the form:

$$(g_0 g_1^{x'_1} \dots g_n^{x'_n} g_t^{x'_t} g_0 g_1^{x''_1} \dots g_n^{x''_n} g_t^{x''_t})^{\alpha}$$

for some key α, encodings $x'_1, \dots, x'_n$ of attributes $A_1, \dots, A_n$ and encodings $x''_1, \dots, x''_n$ of attributes $A'_1, \dots, A'_n$. However, a public key of such form will not pass the standard U-Prove verification. The verifier checks whether:

$$a \stackrel{?}{=} \mathcal{H}((g_0 g_t^{x_t} \prod_{i \in D} g_i^{x_i})^{-c} k^{r_0} (\prod_{i \in U} g_i^{r_i})).$$

Let us consider one disclosed attribute under base g_j, $j \in \{1, \dots, n\}$. The adversary can choose to disclose x'_j or x''_j. Without loss of generality, let the adversary disclose x''_j. Then, the value $(g_j^{x''_j})^{-c}$ will be canceled by the value $(g_j^{x''_j})^{c}$, which will be computed in k^{r_0}. However, note that the value $(g_j^{x'_j})^{c}$ will also be computed, since $(g_j^{x'_j})$ is part of the public key $k = h_1$. Note further, that x'_j cannot be part of the undisclosed attributes since the verifier uses only bases g_i for $i \in U$ and $j \notin U$. It follows that the adversary would have to know $\log_{g_i}(g_j)$ for a $i \in U$ or find a collision for the hash function $\mathcal{H}$ (since c depends on the value of a).

The same argumentation works for $|I| \in \{3, \dots, k\}$. Thus, even if the adversary has k tokens, he cannot create a new U-Prove token that contains a subset of attributes from the k tokens he possesses.

5 Conclusion

We have shown, that it is possible to create an extension for the U-Prove credential system that allows to randomize the token. This extension allows to use the token multiple times in such a way that the verifier cannot link two presentation proofs of the same token. To assure, the validity of the token we use self-blindable certificates instead of blind signatures used in the standard specification. To give some intuition, for the security of this construction, we give a brief rationale. Future work will include a formal security proof of our extension in the sense that this extension is as secure as the standard U-Prove specification (which in fact has no formal security proof).

References

1. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10) (October 1985) 1030–1044 1
2. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM conference on Computer and communications security. CCS '04, New York, NY, USA, ACM (2004) 132–145 2
3. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385 (2005) http://eprint.iacr.org/. 2

4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques. EUROCRYPT'03, Berlin, Heidelberg, Springer-Verlag (2003) 614–629 2
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: In proceedings of CRYPTO'04, LNCS series, Springer-Verlag (2004) 41–55 2
6. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Proceedings of the 3rd international conference on Security in communication networks. SCN'02, Berlin, Heidelberg, Springer-Verlag (2003) 268–289 2
7. Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In: Proceedings of the 9th ACM conference on Computer and communications security. CCS '02, New York, NY, USA, ACM (2002) 21–30 2
8. Paquin, C., Zaverucha, G.: U-prove cryptographic specification v1.1. http://research.microsoft.com/pubs/166969/U-ProveCryptographicSpecificationV1.1Revision2.pdf (April 2013) 2, 3, 4, 5, 6
9. Verheul, E.R.: Self-blindable credential certificates from the weil pairing. In Boyd, C., ed.: ASIACRYPT. Volume 2248 of Lecture Notes in Computer Science., Springer (2001) 533–551 2, 3, 4, 7
10. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In Brickell, E.F., ed.: CRYPTO. Volume 740 of Lecture Notes in Computer Science., Springer (1992) 31–53 4