To appear in *Advances in Cryptology—Asiacrypt 2001*, *Lecture Notes in Computer Science*, Springer-Verlag. ©IACR 2001

Self-Blindable Credential Certificates from the Weil Pairing

Eric R. Verheul

PricewaterhouseCoopers, GRMS Crypto group, P.O. Box 85096, 3508 AB Utrecht, The Netherlands, eric.verheul@[nl.pwcglobal.com, pobox.com]

Abstract. We describe two simple, efficient and effective credential pseudonymous certificate systems, which also support anonymity without the need for a trusted third party. The second system provides cryptographic protection against the forgery and transfer of credentials. Both systems are based on a new paradigm, called self-blindable certificates. Such certificates can be constructed using the Weil pairing in supersingular elliptic curves.

1 Introduction

Credential pseudonymous certificates (CPCs) were introduced by David Chaum [7] in 1985 to counter some of the privacy problems related to identity certificates. One such problem is that service providers know exactly who they are servicing when a user employs an identity certificate, which for some applications is not required, acceptable or even permissible. Moreover, by combing their logs, service providers can piece together a record of all the user's activities.

A *pseudonym* is a unique identifier (string) by which a user is known by a certain party; typically each party knows the same user by a different pseudonym. These pseudonyms can be references to a user's identity known only by designated parties, or can be completely anonymous, (i.e., known only to the user). Unlike Chaum [7], we do not limit a 'physical' user to only one pseudonym with a given provider. We believe that for some types of providers, e.g., on-line, subscription based, information providers, the use of many different pseudonyms for one physical user, without the provider knowing, can be considered an important feature. However, we do discuss how, if necessary, such unique pseudonyms can be supported by our systems.

A *pseudonymous certificate* binds a user's pseudonym to their public key, the private key to which the user possesses. Such certificates are issued by a trust provider. Identities, pseudonyms and public keys should be unique. A *credential* is a trust provider's statement about the user which is relied upon by other parties, who we simply call service providers. Examples of such statements are properties such as "lives in Amsterdam", qualifications such as "has a PhD in math", or rights such as "can access this secure room". A credential can be *single-use*, such as a prescription, or *multiple-use* such as a driver's license. In this paper we focus on the latter type of credentials.

Finally, *credential pseudonymous certificates* (CPCs) are digital certificates that bind credentials to users, known by a pseudonym. Proof of credential possession is given by proving possession of the private key related to the public key referenced in the certificate. Several credentials may be bound to a single pseudonymous certificate and, thus, pseudonym.

In Chaum's model, pseudonyms are *unlinkable*: parties that know a user by different pseudonyms must not have the ability to combine their logs to assemble a dossier on the user.[1] Another requirement in Chaum's model is that CPCs must be *translatable*: a CPC issued under pseudonym A must be usable under pseudonym B. For example, a user may be given a credential asserting his good health from a doctor under pseudonym A, and show this to its insurance company who knows it by pseudonym B. In addition to these two requirements, the system should fulfill the following three basic security requirements:

Protection against pseudonym/credential forgery It should not be possible for outsiders, malicious users, or other parties involved to generate (credential) pseudonymous certificates without the consent of the relevant trust providers.

Protection against pseudonym/credential sharing A user could be tempted to share its credentials (e.g., a season pass for public transport) with another user. It should therefore be very difficult or awkward for a user to do so.[2] One potential solution to this problem would be to store credentials on tamper resistant devices that are valuable to the user (e.g., smartcard based passports). A better solution would be an *all-or-nothing* concept for credentials: sharing a credential effectively implies sharing a credential that is highly valuable to the user, most notably one enabling him to take over the user's identity and digitally sign contracts that legally binds the user (cf., [6], [5]).

Revocation of pseudonymous certificates and credentials Under certain circumstances, it should be possible for the user and trust providers to revoke pseudonymous certificates as well as credentials bound to them. This could be case, for instance, if a user lost secret (key) information or changes jobs.

CPCs such as those described above, counter the privacy problems of identity certificates to some extent, but not completely. Indeed, in that setting, all user's activities with a provider are related to a pseudonym, so that the provider can link the user's activities with the fixed pseudonym. If the user's identity is compromised, then so are its activities. To prevent this potential problem, a CPC system should preferably support that users can easily and regularly change pseudonyms. A CPC system should also ensure that the translation of

[1] Unlinkability and pseudonymity of credentials are sometimes difficult to enforce simultaneously in practice. Indeed, even if they are anonymous, credentials implicitly narrow down the number of possible users possessing them. To illustrate, how many people have both a degree in cryptography (credential number one) and Swedish citizenship (credential number two)?

[2] Perhaps complete eradication of credential sharing would be impossible in the virtual world, as the end user might give away everything he knows (passwords) or has (smartcards), leaving only the identification factor "what the user is"(e.g., biometrics) to counter credential sharing.

credentials includes as few (trusted) parties as possible. In our CPC system, users themselves can both change pseudonyms and translate their credentials.

We remark that, in the above text, we implicitly define the parties as *users*, *trust providers* (providing credentials and pseudonyms to users) and *service providers* (relying on credentials and pseudonyms), which we use in the remainder of this paper without further explanation.

The goal of this paper is to describe a very simple, effective and efficient CPC system that meets the basic requirements of a CPC system and that is based on the new paradigm of *self-blindable* certificates. With this type of certificates the user can, e.g.:

- generate its own new pseudonymous certificates itself (to which it possesses the private key) based on a valid pseudonymous certificate; and
- translate and combine CPCs issued under one pseudonym to another pseudonym, including a one-time-use pseudonym.

Related Work

As we could probably write an entire paper just discussing and comparing all of the CPC schemes that have been published, we will be brief. The first scheme was introduced by Chaum and Evertse [10] and is based on having a semi-trusted third party involved in all credential translations. Both from an efficiency and a security point of view, this is undesirable. Chen's scheme [12], envisions a trusted party who, amongst other things, should be trusted to refrain from transferring credentials between different users. Damgård's scheme [13], is based on general complexity-theoretic primitives and is therefore not applicable for practical use. The scheme developed by Lysyanskaya, Rivest, Sahai and Wolf [19] is based on one-way functions and general zero-knowledge proofs which also makes it inappropriate for practical use. Our CPC system can be considered as the opposite of the credential scheme [6] constructed by Camenisch and Lysyanskaya, which in effect issues one secret CPC for each trust provider; the scheme's properties of anonymity and untraceablity arise from the zero-knowledge protocols that confirm that a user indeed has such a certificate without revealing it. Although the scheme [6] appears to be of practical use, it is based on rather complex (zero-knowledge) protocols. Our scheme and the required proofs of knowledge are basic (Schnorr and Okamoto). Finally, we mention the work of Brands [5], which deals with the related subject of privacy protecting attribute certificates. In our system, the user itself can translate or combine credentials received from different trust providers without the interaction of any trusted party, generating a new certificate. This is an important distinction from Brands' scheme [5] when applied to the special case of a credential certificate system. As a final note, we remark that the privacy of our scheme can be further improved by the use of "Wallet with Observer" techniques, cf., [5], [11].

Outline of the paper

- In Section 2, we describe a variant of the Chaum-Pedersen digital signature

scheme which is of crucial importance for our constructions of self-blindable certificates.
- In Section 3, we provide a functional description of our model for CPSs.
- In Section 4.1, we present the first technical construction of our model, which assumes that secret key information is stored on tamper-proof devices to provide resistance to credential transfer.
- In Section 4.2, we present the second technical construction of our model, which is more resistant to the transfer of credentials, without requiring the use of tamper-proof devices. The transfer of any credential in this construction to another person will actually result in the transfer of a very valuable signing key, e.g., one enabling the holder to sign legally binding contracts in the user's name.
- In Section 5, we summarize our results.

2 A proofless variant of the Chaum-Pedersen signature scheme

A digital signature s formed by an entity is a data string, based on a private key under control of the entity, that associates a message m (in digital form) to enable a proof that it originates from the entity and that it has not been changed. If the actual message comprises a public key plus some optional additional attributes, then (m, s) is called a certificate and the entity issuing it is called a Certification Authority (CA).

In this section, we describe a digital signature scheme that enables a CA to issue certificates that are "self-blindable". This will be explained further in Section 3. The digital signature scheme is based on the Chaum-Pedersen signature scheme (cf., [11]). The setting of our scheme is not standard but is based on a group, G, of prime order q, with generator g, in which the Decision Diffie-Hellman problem is simple, while the discrete logarithm and the Diffie-Hellman problems are practically intractable. In the section below, we further explain these notions and indicate how such groups can be constructed. In Section 2.2, we describe our digital signature scheme and its properties.

2.1 Groups in which the DDH problem is simple and DH, DL are hard

Recall, that the *Diffie-Hellman* (DH) problem with respect to a generator g of a group G of (prime) order q, is the problem of computing the values of the function $DH_g(g^x, g^y) = g^{xy}$. Two other problems are related to the DH problem. The first one is the *Decision Diffie-Hellman* (DDH) problem with respect to g: given $a, b, c \in G$ decide whether $c = DH_g(a, b)$ or not. An alternative formulation of the Decision Diffie-Hellman problem is: given a quadruple g, g^x, h, h^y in the group G decide whether $x = y$. The second problem related to the DH problem, is the *discrete logarithm* (DL) problem in G with respect to g: given $a = g^x \in G$, with $0 \leq x < q$, find $x = DL(\alpha)$. The DL problem is at least as difficult as the DH problem. It is widely assumed that if the DL problem G is hard, then so is the DH problem. Currently, cf. [16], [27], [17], a large class of groups has been discovered in which the DDH problem is simple, while the Diffie-Hellman and discrete logarithm problems are presumably not. This class consists of certain groups of points on supersingular elliptic curves in which setting the DDH problem can be efficiently computed (in polynomial time i.e., in polynomial time and space in length of input) by using the so-called Weil pairing.

As an illustration of such groups and techniques, consider the curve $C_a : y^2 = x^3 + a$ with $p = 2 \bmod 3$ and a any non-zero element in $\mathrm{GF}(p)$. Then, the Frobenius trace over $\mathrm{GF}(p)$ is equal to 0 (hence the curve is supersingular) and the number of points on the curve in $\mathrm{GF}(p)$ is equal to $p+1$. Moreover, as $p = 2 \bmod 3$, the equation $x^3 = 1$ only has solutions in $\mathrm{GF}(p^2)$ other than $x = 1$; let ω be such a solution. Now, if $\langle P \rangle$ is a group of points of (prime) order q on the curve in $\mathrm{GF}(p)$ (i.e., q divides $p+1$) and A, B, C is an instance of the DDH problem with respect to P. Then $C = DH_P(A, B)$ if and only if $e_q(A, D(B)) = e_q(P, D(C))$, where $D(.)$ is the endomorphism (called a *distortion map* in [27]) on C_a that maps a point (x, y) on the curve to the point $(\omega \cdot x, y)$ also on the curve (over $\mathrm{GF}(p^2)$) and where $e_q(.,.)$ is the so-called Weil pairing. See [1], [20] or [26]. As the Weil pairing is efficiently computable, the DDH problem is also efficiently computable in this situation. It is well-known that the DL problem in the group of points on the curve in $\mathrm{GF}(p)$ reduces to the DL problem in a subgroup of order q in $\mathrm{GF}(p^2)^*$ (cf. [21]). That is, to make the DH and DL problems practically intractable against attacks known today, the length of the prime number q should be at least 160 bit and the length of the prime number should be at least 512 bits.

A practical construction of a group in which the DDH problem is efficiently computable and the DH and DL problems are presumably not, is as follows. Choose a 512 bit prime number p of type $p = 6q - 1$ where q is also a prime number and consider the curve $C_1 : y^2 = x^3 + 1$. Let P be any $\mathrm{GF}(p)$-rational point on the curve of order q. This construction is used in [2] in the setting of an identity-based encryption scheme that is also based on the Weil pairing. This paper also analyzes the work needed to solve the DDH problem in the group $\langle P \rangle$, which amounts to a small number of multiplications on the curve.

These techniques generalize to groups of points on supersingular elliptic curves over a finite field, say F, and the work required to compute the DDH problem is asymptotically bounded by $O(k^3 \log(\|F\|)$ bit operations, i.e., the complexity of calculating a Weil pairing. The parameter k is the so-called MOV degree (cf. [21]) and is equal to either 1, 2, 3, 4 or 6 in the setting of supersingular curves.

We end this section with two remarks for later reference. A group of points, G, on a supersingular elliptic curves has the property that there exists an efficiently computable embedding, i.e., an injective homomorphism, of the group in a second group G' where all three of the DDH, DH and the DL problems are believed to be hard. Indeed, this embedding is given by the MOV embedding (cf. [21]) and the second group, G', is a subgroup of the multiplicative group of a finite field.

It is shown in [27] that inverting such embeddings is hard; in fact, as hard as the DH problem in the group G. Note that by using a specific choice of G, the group G' could be the XTR group. Compare [18] and [27]. A group of points on a (supersingular) elliptic curve over a finite field used in cryptography is typically chosen in such a way that its order is a prime number times a small number (e.g., 6 in the example above). This means that choosing provable random elements in the subgroup without knowledge of relative discrete logarithms is very simple, e.g., by mapping a hash value into a point on the curve and then mapping it to a point in the subgroup. See also [3].

2.2 The 'proofless' variant of the Chaum-Pedersen scheme

As explained in the previous section, we consider a group, G, of prime order q, with generator g, in which the DDH problem is simple, while the discrete logarithm and the Diffie-Hellman problems are practically intractable. The public key of a participant in the Chaum-Pedersen scheme takes the form $y = g^x$ where $0 \leq x < q$ is the participant's randomly chosen private key. A signature on a message $m \in G$ in the original Chaum-Pedersen scheme, consists of $z = m^x$ plus a proof that $\log_g(y) = \log_m(z)$. Resolving the latter problem is just an instance of the *Decision Diffie-Hellman* (DDH) problem with respect to g. Indeed, one can easily verify that $\log_g(y) = \log_m(z)$ if and only if $z = DH_g(m, y)$. That is, if one applies the Chaum-Pedersen scheme to the group G, one is not required to send along an explicit proof that $\log_g(y) = \log_m(z)$, as anyone can validate that themselves. Or, in other words, the signature on a message $m \in G$ only consists of an element $z = m^x$ of the group G, without the additional proof of knowledge. This is the variant of the Chaum-Pedersen scheme that we use in our schemes. It follows that by choosing a group of points on a supersingular elliptic curve of MOV degree 6 (cf. [21] and the previous section), the representation of the element z requires only $1024/6 \approx 171$ bits to obtain a security level comparable with 1024 bit RSA (with respect to attacks known today). See [3], where it is also shown that the above digital signature scheme is secure in the random oracle model.

An interesting property of this variant is that it is *self-blindable*: it enables easy randomization without losing the verification property and without requiring knowledge of the signing key z. Indeed, given the signed message m, m^z, then by choosing a randomizing factor, k, it can be transformed into m^k, m^{kz}. This property becomes useful when the message m has a property that is inherited by m^k, e.g., knowledge of a certain discrete logarithm, and is explored in the following sections. Another interesting property of this variant (as pointed out to us by Stefan Brands), is its easy blinding property, cf. [7]. When a party wants to obtain a blind signature on a message (typically a hash), M, from a signing party with public key g^x in our variant of the Chaum-Pedersen, it asks the signing party to sign M^r, for a random $0 \leq r < q$, resulting in M^{rx}. The user can deduce M^x from this using r and verify that it is a correct signature on M, which is publicly verifiable. We will delve no further into this property in this paper.

In the terminology we introduced above, we formulate the security assumption that we require for our variant of the Chaum-Pedersen scheme (cf. [8], [9]).

Assumption 2.1 *If the Diffie-Hellman problem with respect to g is hard, then without knowledge of the private signing key z, the only forged message an attacker can make on the basis of signed messages $(m_1, m_1^z), (m_2, m_2^z), \ldots, (m_n, m_n^z)$ with respect to the public key g^z is of the form $(g^{i_0} \prod_{j=1}^{n} m_n^{i_n}, (g^{i_0} \prod_{j=1}^{n} m_n^{i_n})^z)$, for any integers $i_0, i_1, \ldots i_n$, i.e., a power product of the signed messages.*

3 Our functional model for CPCs

In this section, we describe our functional model for CPCs. To this end, we first formulate the requirements for self-blindable pseudonymous certificates and credentials based upon them. Then we explain how these elements can be used to build a CPC system.

3.1 Self-blindable certificates

In this section we introduce the notion of *self-blindable* certificates, which is of crucial importance for our schemes. Our introduction is somewhat informal, but can be made formal without much effort.

We assume that one public key crypto system is employed by all users and we denote the collection of all possible user public keys by $\mathcal{U}$. We also assume that one signing public key crypto system is employed by all trust providers for certificate issuance. For simplicity's sake, we also assume that certificate signing is deterministic, i.e., there is only one possible valid certificate on a fixed public key, plus optional fields. We let $\mathcal{T}$ denote the collection of possible verification public keys of trust providers. Our description of a credential on a user public key $P_U \in \mathcal{U}$ from a trust provider with public verification key P_T takes the form

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

where S_T stands for the private signing key of the trust provider relating to P_T. This certificate is typically accompanied by a higher-level certificate

$$Cert(P_U, \text{“Trust statement”})$$

on the public verification key P_T. We do not further elaborate on this, but this certificate can be thought of as a standard X.509 certificate with the "Trust statement" in one of its extension fields. We denote the collection of all possible certificates by $\mathcal{C}$.

The certificates are called *self-blindable*, provided there exists a set called *transformation factor space* F and an efficiently computable *transformation map* $D : \mathcal{C} \times F \to \mathcal{C}$ with the following properties:

1. For any certificate $C \in \mathcal{C}$ and $f \in F$ the certificate $D(C, f)$ is signed with the same trust provider public key as C.

2. Let C_1, C_2 be certificates and $f \in F$ known. If $C_2 = D(C_1, f)$ then one can efficiently compute a transformation factor $f' \in F$ such that $C_1 = D(C_2, f')$.
3. If $C_1, C_2 \in \mathcal{C}$ are two different certificates on the same user public key, then so are $D(C_1, f)$ and $D(C_2, f)$. That is, the mapping $D(.,.)$ induces a mapping $\mathcal{U} \times F \to \mathcal{U}$ and although abusive, we also use the notation $D(P_U, f)$ for any user public key P_U and transformation factor f.
4. Let P_U be a user public key and let $f \in F$ be a known transformation factor. Then, a user possesses the private key relating to P_U if and only if it possesses the private key relating to $D(P_U, f)$.
5. If the user's public key $P_U \in \mathcal{U}$ is fixed and if $f \in F$ is a uniformly random element in F, then $D(P_U, f)$ is a uniformly random element in $\mathcal{U}$.

We briefly explain the rationale behind these properties. The first property enables one to transform a user certificate into another one from the same certificate authority; the fourth property ensures that the user still has possession of the private key referenced in the transformed certificate provided he knows the transformation factor. The fifth property states that all user public keys are equally possible in the transformed certificate. As we will explain below, a user typically collects credentials on different certificates formed as transformations of one fixed certificate. Now, the second property enables to invert transformations, allowing to translate all credentials to the fixed certificate and then to other certificates. Finally, the third property is technical and in fact emerged from our constructions. We have chosen it as part of our formal definition, as it enables simple proofs and formulation of other properties, e.g., properties four and five. More complicated requirements are possible to arrive at a more general notion of self-blindable certificates, but we will not explore this.

3.2 A CPC system based on the building blocks

We use the terminology introduced above and we assume that the certificates are self-blindable. Our notion of a pseudonymous credential is the simplest possible and takes the form

$$\{P_U, [Sig(P_U, S_N), Cert(P_N, \text{“PP statement”})]\},$$

where P_U stands for the public key of the user (with related private key S_U). Moreover, $Sig(P_U, S_N)$ is a signature on the user's public key with a signing key of the *pseudonym provider* (PP) and $Cert(P_N, \text{“PP statement”})$ is a (conventional) certificate on the public verification key of the pseudonym provider, with a statement on its applicability included among the usual fields (e.g., expiration date). For evident reasons, this PP certificate must be used by the pseudonym provider for many users to prevent linkage of the issued pseudonymous certificate. Also note that the pseudonym of a user is in fact the user's public key in its certificate, which is reminiscent of the SPKI (Simple Public Key Infrastructure) approach, cf. [24].

Note that the self-blinding properties of the certificates enable the users themselves to generate a new pseudonymous certificate validly signed by the same PP, by choosing a (random) factor and transforming an initially issued pseudonymous certificate.

Our description of a CPC is based upon that of a pseudonymous credential, say $\{P_U, [Sig(P_U, S_N), Cert(P_N, \text{“PP statement”})]\}$ and its simplest form is:

$$\{P_U, [Sig(P_U, S_N), Cert(P_N, \text{“PP statement”})],\\ [Sig(P_U, S_C), Cert(P_C, \text{“CP statement”})]\}.$$

Here, $[Sig(P_U, S_C), Cert(P_C, \text{“CP statement”})]$ is called the *credential field*. In this, $Sig(P_U, S_U)$ is a signature on the public key of the owner with a signing key S_C of the *credential provider* (CP). Also, $Cert(P_C, \text{“CP statement”})$ is a (conventional) certificate on the related credential provider's public verification key, that has a statement on its credential applicability, e.g., “the person having possession of the private key is over 18 years old” included among the usual fields (e.g., expiration date). In a natural fashion one can have several credential fields attached to a pseudonymous credential in the above way, which is in fact the general form of a CPC.

Based on the building blocks explained above, one can now construct a wide variety of types of CPC systems. We provide a high-level description of one such system on which many variations are possible (cf. Figure 1).

System description 3.1

Initial Registration The user registers, typically in a non-anonymous fashion, with a pseudonym provider. After registration a First Pseudonymous Certificate (FPC) issuing protocol between the user and the pseudonym provider is started. This protocol is system specific. The pseudonym provider puts the FPC in a public directory. When unique pseudonyms are required, the provider has the option to maintain a private list of physical persons that were issued a pseudonymous certificate; this ensures that at most one such certificate is issued to a physical person.

Credential Issuance By using a random transformation factor, the user transforms its FPC into a random pseudonymous certificates (RPC). The user securely stores the used transformation factor. Then the user registers with a credential provider using this RPC which includes a proof of possession of the private key referenced in the RPC. This registration need not be anonymous. The user does what is required to obtain a credential (e.g., takes a driver's exam, shows other credentials) and up-on succeeding, is issued a credential on the RPC, that is the CPC. The pseudonym provider has the option to put the CPC in a public directory.

Credential Use The user registers (typically anonymously) with a service provider using a new RPC, which includes a proof of possession of the private key referenced in the new RPC. The user combines all of the CPCs relating to credentials required by the service provider into one CPC under the registered pseudonym. This is possible by using the second property of self-blinding certificates on the transformation factors related with the individual, original CPCs. That is, a CPC is first translated to the First Pseudonym and then translated to the registered pseudonym (in our constructions these two steps can be performed in one operation). This certificate is presented to the service provider, together with a proof of possession of the private key referenced in this CPC. Once the user is successful in doing so, he will be serviced.

If the service provider wants to be certain that the user has not already been issued another pseudonym, the service provider has the option to require that the user contact a specific trust provider which we refer to as "unicity" provider. The user sends this trust provider the transformation factor(s), transforming the new RPC to the first issued pseudonymous certificate stored in the pseudonym provider's directory (i.e., the FPC). This trust provider then validates that these factor(s) transform the RPC into a FPC on the PP's directory, and that this FPC was not registered before. The trusted party then reports to the service provider that the user has not registered before. Note that the PP directory does not specify user identities, only FPCs, also note that the specific trust provider need not be the user's pseudonym provider.

In the system description above, we have used the FPC list of the pseudonym provider as the reference data for all trust providers that need to verify that a 'physical' user cannot register twice (under different pseudonyms) with a service provider. This means that if two such trust providers conspire, they can link together the different pseudonyms of a user. One can prevent this linkage with a flexible secret sharing technique as follows. During registration, the pseudonym provider and the user, say U, exchange a secret, S. If a trust provider, say T, wants to provide assurance on unique pseudonyms, then provider A is provided a list consisting of transformed FPCs, in such a way that:

- user U's FPC is transformed using a transformation factor based on a secure hash of the name of the provider T and the secret S; and
- the order of the FPCs is randomly permuted.

If user U wants to assure the trust provider T that it is not registering twice (under different pseudonyms) with a service provider via T, then it provides the provider with the transform factor transforming the RPC (see above) into the transformed FPC stored at the provider T. This technique can be iterated: user U can (after proving possession of a transformed FPC at T) be issued another secret by T, and the transformed FPC can be re-transformed by T and stored at another trust provider T_2, etc.. By combining transformation factors, user U can employ provider T_2's service without any interference from provider T. Moreover, in such a setting, linkage requires that all such trust providers and the pseudonym provider conspire.

In Figure 1 we have depicted the (five) steps from pseudonym issuance to CPC application in a sample voting application. The communication between the "unicity" provider and the service provider (the voting application) is not depicted.

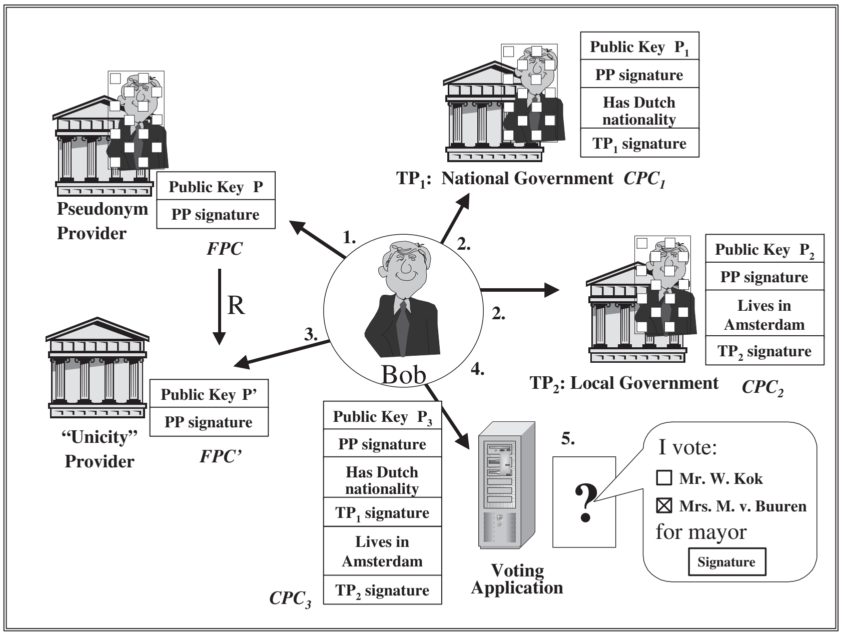

Fig. 1. Overview of system description 3.1

3.3 Revocation of Certificate Bases

As users typically will not present the originally issued certificates to service providers, certificates cannot be revoked in the conventional way. A primary concern is that the revocation process should not make it possible to link credential use, except, possibly, by certain trusted parties.

There are several methods to address revocation in our model, but we outline only two. The first method is pro-active, and consists of letting the trust providers employ signing keys with a short expiration time (e.g., a week). If a pseudonymous certificate or a credential relating to such a certificate has not been revoked, then the trust provider automatically updates the certificates or credentials in its directory with newly signed ones. A user can collect the updated pseudonymous certificates and credentials, preferably via an anonymous channel to reduce the chances of linkage. To achieve this, the user can, for example, collect many certificates, including the required ones. By revoking its FPC, the user can effectively revoke all credentials based on it.

The second method for revocation we outline consists of sending along specific transformation factors with a (credential) pseudonymous certificate, to a specific trust provider. This trust provider can then retrieve the original issued (credential) pseudonymous certificates and find out if they have been revoked. The trust provider then provides a statement on the status of the (credential) pseudonymous certificate to the service provider. This functionality resembles the use of an On-line Certificate Status Protocol (OCSP) request, commonly used on the Internet (cf. [23]). Of course, the service provider still needs to verify that the user is in possession of the private key referenced in the used randomized CPC.

The second revocation technique can be supplemented with the flexible secret sharing technique described at the end of the previous section.

4 Constructions for credential pseudonymous certificates

4.1 A simple Construction

In this section we describe an initial and very simple construction for self-blindable certificates and thus CPCs. We describe this scheme merely for purposes of illustration, as it has the serious inherent draw-back of not supporting cryptographic protection against users sharing credentials. Therefore, to implement this construction one would need to trust devices resistant to user tamper to prevent users from sharing credentials. As the construction in Section 4.2 provides cryptographic protection against users sharing credentials this construction is favorable to the scheme presented in this section.

Let $G = \langle g \rangle$ be a group of prime order q in which the DDH problem is efficiently computable, while the discrete logarithm and the Diffie-Hellman problems are practically intractable. We also assume that the (provable) random generation of elements in G without knowing any relative discrete logarithms is also possible (see the end of Section 2.1). The description of the group G, including the g, q are considered as system parameters.

The set $\mathcal{T}$ of all trust provider's public keys takes the form j, j^s where $0 \leq s < q$ is the related private key and where $j \in G \setminus \{1\}$. We assume that each trust provider's public generator j is (provably) randomly chosen, e.g., it could be based on the output of a secure hash algorithm with a fixed input. The set of users public keys $\mathcal{U}$ consists of elements of the form g^x where $0 < x < q$ are all possible user's private keys. There is a subtle reason why $x = 0$ is principally not allowed, see below. Note that a user can prove possession of x in a zero-knowledge fashion with the Schnorr identification protocol [25]. Moreover, several digital signature systems can be based on the user public, private key pairs mentioned above, e.g., DSA [15], ElGamal [14] and Schnorr [25]. Finally, a certificate issued by a trust provider with public key h, h^z on a user public key g^x takes the form:

$$\{g^x, g^{xz}\}.$$

Note that the above certificate is based on the variant of the Chaum-Pedersen signature (as outlined in Section 2) on g^x with respect to the public key h, h^z, i.e., g^{xz}. An important feature of this variant is that it is not required to add an interactive proof that the second component indeed has the form g^{xz} as the DDH problem is assumed to be simple. Due to the restrictions on the first element in the certificate, it cannot be equal to the unity element. If this condition is not also checked by applications, then certificate forgery becomes simple.

The certificates $\mathcal{C}$ constructed in this way are self-blindable. To this end, choose the transformation factor space F equal to $\mathrm{GF}(q)^*$ and define the transformation $D : \mathcal{C} \times F \to \mathcal{C}$ as

$$(\{X, Y\}, f) \to \{X^f, Y^f\}.$$

That is, the certificate $\{g^x, g^{xz}\}$ is transformed to the certificate $\{g^{xf}, g^{xfz}\}$ under factor f. It is a simple verification that $D(.,.)$ satisfies the five properties of a transformation and, thus, that the certificates constructed in this way are self-blindable.

Notice that the transfer of credentials is simple in this construction, if the user is able to retrieve (and transfer) the private key related to the public key of a (transformed) pseudonymous certificate. This problem can be controlled by ensuring that all security operations with respect to credentials take place on a tamper resistant signing device in such a way that private key information of (transformed) certificates can be used ('addressed') but not retrieved. The use of such devices needs to be addressed in the FPC issuing protocol for these certificates, for instance as follows.

1. The user registers, typically in a non-anonymous fashion, with a pseudonym provider.
2. The pseudonym provider generates a random $0 < x < q$, and forms the user public key g^x and the certificate $\{g^x, g^{xz}\}$. All information is put on a tamper resistant signing device, in such a way that private key information of (transformed) certificates can be used but not retrieved.
3. The secure signing device is handed over to the user in a secure fashion.

Having filled in this issuing protocol, our CPC scheme now follows system description 3.1. Protection against pseudonym/credential linking and pseudonym/credential translation are obvious consequences of the properties of self-blindable certificates. For the other two security properties (protection against forgery and transfer), one needs to trust devices resistant to user tampering.

4.2 A more robust construction

This construction is based on the technique in Brands' e-cash scheme to trace double spenders (cf. from [4]). Just as in the previous section our construction is based on the variant of the Chaum-Pedersen signature scheme as introduced in Section 2. So, again, let $G = \langle g \rangle$ be a group of prime order q in which the DDH problem is efficiently computable, while the discrete logarithm and the Diffie-Hellman problems are practically intractable. We also assume that the (provable) random generation of elements in G without knowing any relative discrete logarithms is also possible. In addition to this, we assume that there exists an efficiently computable embedding $E(.)$ from G into a group G' where all three problems DDH, DH and DL are practically intractable. All these requirements are met by suitable groups of points on supersingular elliptic curves, cf. the end of Section 2.1. The description of the groups G, including the g, q, the group G' and the embedding are considered to be system parameters.

As before, the set of all trust providers' public keys, $\mathcal{T}$ takes the form j, j^s where $0 \leq s < q$ is the related private key and where $j \in G \setminus \{1\}$. We assume that each trust provider's public generator j is (provably) randomly chosen, e.g., it could be based on the output of a secure hash algorithm with a fixed input. In addition we assume that the pseudonym provider publishes a certified pair $(r, s) = (r, r^f)$ where $r, s \in G$ and for some $0 < f < q$ which is unknown by all parties. Generation of such a pair consists of choosing two (provable) random r, s which determines f. Alternatively, the pseudonym provider can choose the element r in a provable random fashion and generate a random element $0 < f < q$ and form $s = r^f$. We prefer the first construction, for two reasons. First, it is difficult for the pseudonym provider to convince others that f has been chosen randomly and, second, it is good practice to have as few secret keys in a system as possible.

The set of users public keys $\mathcal{U}$ consists of elements of the form $g_1, g_2, g_1^{x_1} g_2^{x_2}$. Here $0 \leq x_1, x_2 < q$ is the related private key, g_1 is a random generator and $\log_{g_1}(g_2) = f$. As in the previous scheme, we require that $g_1^{x_1} g_2^{x_2}$ be unequal to the unity element. Note that a participant can prove possession of x_1, x_2 in a zero-knowledge fashion with the Okamoto variant of Schnorr's identification protocol [22]. In the same paper, a variant of Schnorr's signature scheme is described based on the user public, private key pairs mentioned above. Finally, a certificate issued by a trust provider with public key h, h^z on a user's public key $g_1, g_2, g_1^{x_1} g_2^{x_2}$ takes the form:

$$\{g_1, g_2, g_1^{x_1} g_2^{x_2}, (g_1^{x_1} g_2^{x_2})^z\}.$$

Again, this is precisely the variant of the Chaum-Pedersen signature (as outlined in Section 2) on the user's public key with respect to the public key h, h^z. As the DDH problem is simple, on basis of the certified pair (r, r^f), anyone can and should verify that the first two parameters in the certificate are indeed correctly formed, i.e., the second one is an f-th power of the first one (cf. the alternative description of the DDH problem in Section 2.1). Due to the restrictions on the three elements in the certificate, none of them can be equal to the unity element. If this condition is not also checked by applications, then certificate forgery becomes simple.

The certificates $\mathcal{C}$ constructed in this way are self-blindable. To this end, define the transformation factor space by $F = \mathrm{GF}(q)^* \times \mathrm{GF}(q)^*$ and the transformation $D : \mathcal{C} \times F \to \mathcal{C}$ as:

$$(\{X, Y, W, Z\}, (k, l)) \to \{X^l, Y^l, W^{kl}, Z^{kl}\}.$$

That is, the certificate $\{g_1, g_2, g_1^{x_1} g_2^{x_2}, (g_1^{x_1} g_2^{x_2})^z\}$ is transformed into the certificate

$$\{g_1^l, g_2^l, g_1^{x_1 kl} g_2^{x_2 kl}, (g_1^{x_1 kl} g_2^{x_2 kl})^z\}$$

under the transformation factor (k,l). It is a simple verification that $D(.,.)$ satisfies the five properties of a transformation. Notice that two transformation factors (k,l) are used to ensure that a randomly transformed public key is indeed a random element in the user's public key space.

The FPC issuing protocol for these certificates can be filled in as follows, but many variations are possible; the pseudonym provider's public key is denoted as h, h^z, where $h \in G \setminus \{1\}$ is (provably) randomly chosen.

1. The user registers, typically in a non-anonymous fashion, with a pseudonym provider.
2. The pseudonym provider generates a random pair (g_1, g_2) such that $g_2 = g_1^f$, by choosing a (provably) random power of the elements r, s. The pair (g_1, g_2) is sent to the user, or to a party acting on its behalf (e.g., a smart card issuer).
3. The user (or a party acting on its behalf), generates a random private key $0 \leq x < q$ and forms g_2^x. The user sends g_2^x and proves possession of the private key x (i.e., the discrete logarithm with respect to g_2 of the first sent public key), e.g., by using Schnorr's protocol.
4. Based on the elements g_1, g_2 and g_2^x, the pseudonym provider forms the public key $g_1, g_2, g_1 g_2^x$, checks to ensure that the last element is unequal to the unity element and places a Chaum-Pedersen signature on it, i.e., $(g_1 g_2^x)^z$. Moreover, the provider employs the embedding $E : G \to G'$ and determines the elements $E(g_2), E(g_2^x)$ of the group G' (in which the DDH, DH and DL problems are hard). Next the provider determines a random power r of these elements, i.e., $E(g_2)^r, E(g_2^x)^r$. The provider then forms a conventional non-repudiation certificate (e.g., based on the US Digital Signature Algorithm) on $(E(g_2)^r, E(g_2^x)^r)$. The first pseudonymous certificate and the non-repudiation certificate are issued to the user. Both are also stored in separate directories.

Using the terminology of the above protocol; as the embedding $E(.)$ is a homomorphism it directly follows that the private non-repudiation signing key is equal to x. We have used a non-repudiation signing key only as an example of a private key that is highly important to a user. Many more examples exist (e.g., the user's signing key for financial transactions).

There are two reasons why the user's non-repudiation key is embedded in the group G' in the specified way. First of all, using a group where all three of the DDH, DH and DL problems are hard, seems appropriate for a conventional signature scheme. Second, embedding the non-repudiation key in the specified way, prevents linkage between the first pseudonymous certificate and the non-repudiation certificate. Should a party have access to g_2^r, g_2^{xr} (whose $E(.)$ images appear in the non-repudiation key) then this party would be able to link this to the pair g_2, g_2^x as the DDH problem in G is simple. However, inverting the embedding $E(.)$ is hard (cf. the remarks at the end of Section 2.1), so inverting the values $E(g_2)^r, E(g_2^x)^r$ (deducible from the non-repudiation certificate) is not a practical possibility. Moreover, as the DDH problem is presumed to be hard in G' it would be impossible to relate $E(g_2), E(g_2^x)$ (deducible from the first pseudonymous certificate) to $E(g_2)^r, E(g_2^x)^r$ (deducible from the non-repudiation certificate). Strictly speaking, such a linkage might not be an issue, as users will typically employ transformed pseudonymous credentials. However (cf. the generic description 3.1), this might become an issue should a service provider want to be certain that the user has not already been issued another pseudonym. Indeed, the user would then need to provide a trust provider with the transformation factor from its registered pseudonymous certificate to the First Pseudonymous Certificate. We finally note that, in the issuing protocol, the pseudonym can alternatively first calculate random r-powers of the elements g_2, g_2^x in the group G and then utilize the embedding $E(.)$. For the same r, this would give the same result as with the method described above.

Having filled in this issuing protocol, our CPC scheme now follows from the system description 3.1. Protection against pseudonym/credential linking and pseudonym/credential translation are obvious consequences of the properties of self-blindable certificates. We discuss the two other security properties.

Protection against pseudonym/credential forgery
This protection is based on an all-or-nothing concept (see the introduction). The private key in a transformed credential takes the form $(k, k \cdot x \bmod q)$ for some $0 < k < q$. Note that dividing the second part by the first part yields the user's non-repudiation key x. Hence, if the user transfers a credential, then it also transfers a copy of its non-repudiation signing key. We think that this is a sufficient deterrent to transferring credentials (which can be supplemented with the physical security of a signing device).

Protection against pseudonym/credential forgery [*Indication*]
Under Assumption 2.1, we provide a sketched proof in the appendix that an efficient pseudonym/credential forgery algorithm based on all issued certificates and private keys, will in fact provide an algorithm determining hard discrete logarithms with non-negligible probability.

5 Conclusion

We have described two simple, efficient and effective credential pseudonymous certificate systems, which also support anonymity without the need for a trusted third party. Both systems are based on a new paradigm, called self-blindable certificates. Such certificates were constructed using the Weil pairing in supersingular elliptic curves. The second system provides cryptographic protection against the forgery and transfer of credentials.

6 Acknowledgments

We want to thank Stefan Brands and Berry Schoenmakers for stimulating discussions. Berry is specifically thanked for pointing us to the double spending preventing technique from E-cash based on the Okamoto identification protocol

and Stefan is specifically thanked for providing us with the term "self-blinding signatures and certificates".

References

1. I.F. Blake, G. Seroussi, N.P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.
2. D. Boneh, M. Franklin, *Identity-Based Encryption from the Weil Pairing*, Proceedings of Crypto 2001, LNCS 2139, Springer-Verlag 2001, 213-229.
3. D. Boneh, B. Lynn, H. Shacham *Short Signatures from the Weil Pairing*, these proceedings.
4. S. Brands, *Untraceable Off-line Cash in Wallet with Observers*, Proceedings of Crypto '93, LNCS 911, Springer-Verlag 1994, 302-318.
5. S. Brands, *Rethinking Public Key Infrastructures and Digital Signatures; Building in Privacy*, PhD Thesis, Eindhoven University of Technology, the Netherlands, 1999.
6. J. Camenisch, A. Lysyanskaya, *An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation*, Proceedings of Eurocrypt 2001, LNCS 2045, Springer-Verlag 2001, 93-118.
7. D. Chaum, *Security Without Identification: Transaction Systems to Make Big Brother Obsolete*, Communications of the ACM, 1985, 28(10), 1035-1044. See also *Security Without Identification: Card Computers to Make Big Brother Obsolete*, available from www.chaum.com.
8. D. Chaum, *Zero-knowledge Undeniable Signatures*, Proceedings of Eurocrypt'90, LNCS 473, Springer-Verlag 1991, 458-464.
9. D. Chaum, H. van Antwerpen, *Undeniable Signatures*, Proceedings of Crypto'89, LNCS 435, Springer-Verlag 1990, 212-216.
10. D. Chaum, J.-H. Evertse, *A Secure and Privacy-protecting Protocol for Transmitting Personal Information between Organizations*, Proceedings of Crypto '86, LNCS 263, Springer-Verlag 1987, 118-167.
11. D. Chaum, T.P. Pedersen, *Wallet Databases with Observers*, Proceedings of Crypto'92, LNCS 740, Springer-Verlag 1993, 89-105.
12. L. Chen, *Access with Pseudonyms*, In Cryptography: Policy and Algorithms, LNCS 1029,Springer-Verlag 1995, 232-243.
13. I. Damgård, *Efficient Concurrent Zero-knowledge in the Auxiliary String Model*, Proceedings of Eurocrypt 2000, LNCS 1807, Springer-Verlag 2000, 431-444.
14. T. ElGamal *A Public Key Cryptosystem and Signature System Based on Discrete Logarithms*, Proceedings of Crypto '84, LNCS 196, Springer-Verlag 1985, 10-18.
15. FIPS 186, *Digital Signature Standard*, Federal Information Processing Standards publication 186, U.S. Department of Commerce/NIST, 1994.
16. A. Joux, *A One Round Protocol for Tripartite Diffie-Hellman*, 4th International Symposium, Proceedings of ANTS, LNCS 1838, Springer-Verlag, 2000, 385-394.
17. A. Joux, K. Nguyen, *Seperating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups*, in preparation. Available from eprint.iacr.org.
18. A.K. Lenstra, E.R. Verheul, *The XTR Public Key System*, Proceedings of Crypto 2000, LNCS 1880, Springer-Verlag, 2000, 1-19; available from www.ecstr.com.
19. A. Lysyanskaya. R. Rivest, A. Sahai, S. Wolf, *Pseudonym Systems*, In Selected Areas in Cryptography, LNCS 1758, Springer-Verlag 1999.

20. A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, Boston 1993.
21. A. Menezes, T. Okamoto, S.A. Vanstone *Reducing Elliptic Curve Logarithms to a Finite Field*, IEEE Trans. Info. Theory, 39, 1639-1646, 1993.
22. T. Okamoto, *Provable Secure and Practical Identifications and Corresponding Signature Schemes*, Proceedings of Crypto'92, LNCS 740, Springer-Verlag 1993, 31-53.
23. RFC 2560, *Online Certificate Status Protocol (OCSP)*, available from www.ietf.org.
24. RFC 2693, *SPKI Certificate Theory*, available from www.ietf.org.
25. C.P. Schnorr, *Efficient Identification and Signatures for Smart Cards*, Proceedings of Crypto'89, LNCS 435, Springer-Verlag 1990, 239-252.
26. J. Silverman, *The Arithmetic on Elliptic Curves*, Springer-Verlag, New York, 1986.
27. E. Verheul, *Evidence that XTR is More Secure than Supersingular Elliptic Curve Cryptosystems*, Proceedings of Eurocrypt 2001, LNCS 2045, Springer-Verlag 2001, 195-210.

A Appendix: forgery protection in the robust construction

Suppose that a total of n-number of certificates under one trust provider are issued, e.g., of type:

$$\{g_{1,i}, g_{2,i}, g_{1,i}^{x_{1,i}} g_{2,i}^{x_{2,i}}, (g_{1,i}^{x_{1,i}} g_2^{x_{2,i}})^z\},$$

where the trust providers public key is of the form h, h^z as usual. Also suppose that a forger has access to all private keys $x_{1,i}$ and $x_{2,i}$ and is able to produce a forged certificate, say

$$\{h_1, h_2, h_1^{y_1} h_2^{y_2}, (h_1^{y_1} h_2^{y_2})^z\},$$

where $0 \leq y_1, y_2 < q$ is known to the forger. Notice that (y_1, y_2) should not be equal to $(0,0)$ as then the certificate contains the unity element. As the h_2 should be an f-th power of h_1, it follows from Assumption 2.1 that h_1 (resp. h_2) is a power product of the $\{g_{1,i}\}$ and r (resp. $\{g_{2,i}\}$ and s). Likewise, $h_1^{y_1} h_2^{y_2}$ is a power product of all $g_{1,i}^{x_{1,i}} g_{2,i}^{x_{2,i}}$ and h. By choosing the right transformation factors, we may assume without loss of generality that $h_1 = r^b \prod_{i \in I} g_{1,i}$, $h_2 = s^b \prod_{i \in I} g_{2,i}$ and

$$h_1^{y_1} h_2^{y_2} = h^c \prod_{j \in J} g_{1,j}^{x_{1,j}} g_{2,j}^{x_{2,j}}, \tag{1}$$

for some subsets I, J of $\{1, 2, \ldots, n\}$ and $b, c \in \{0, 1\}$.

We now sketch that we can rule out the possibility that either b, c is equal to 1. To this end, suppose the probability that the event that $c = 1$ to be non-negligible. Now, if one simulates f, then one can use the forgery algorithm to determine $\log_r(h)$. Indeed, by feeding the algorithm $g_{1,i}$ (resp. $g_{2,i}$) that are of form r^{t_i} (resp. s^{t_i}), where $0 \leq t_i < q$ known and random and by choosing the $0 \leq x_{1,i}, x_{2,i} < q$ in a random way ($i = 1, 2, \ldots n$). As this is 'correct' input, it will lead to equalities of type (1). Now, if $c = 1$ in any of these equalities then the algorithm has produced $\log_r(h)$, which is assumed to be a hard problem.

Likewise, if the probability that $b = 1$ is non-negligible, then simulation of f the forgery algorithm will also enable to determine discrete logarithms with respect to r, by basing all $g_{1,i}, g_{2,i}$ on random powers of an element z for which $\log_r(z)$ is required. Thus we conclude that $b = c = 0$ with overwhelming probability and that actually the equations (1) are of type

$$h_1^{y_1} h_2^{y_2} = \prod_{j \in J} g_{1,j}^{x_{1,j}} g_{2,j}^{x_{2,j}}, \tag{2}$$

where $h_1 = \prod_{i \in I} g_{1,i}$, $h_2 = \prod_{i \in I} g_{2,i}$. Note that the sets I, J cannot be empty as the unity element would then occur in the certificate. Moreover, if the set $I \cup J$ does not contain at least two elements, then $I = J$ is a singleton, and the forgery algorithm has in fact produced a transformed user certificate, which is not considered a forgery. Now, suppose that $\log_a(b)$ is required for some $a, b \in G$, then this can be determined with high probability, by basing 'half' the $g_{i,i}, g_{2,i}$ on random powers of a and the other half on random powers of b. With non-negligible probability, the set $I \cup J$ will contain both a $g_{i,i}, g_{2,i}$ based on a and b, and will hence give a relation providing $\log_a(b)$.

This article was processed using the LaTeX macro package with LLNCS style