Secret-key certificates (Continued)

S.A. Brands

Computer Science/Department of Algorithmics and Architecture

CS-R9555 1995

Report CS-R9555
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Secret-Key Certificates (Continued)

Stefan Brands

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

A new construction is described for designing secret-key certificate schemes based on signature schemes other than of the Fiat-Shamir type. Also described are practical secret-key certificate issuing protocols that enable the Certification Authority to certify public keys, without being able to compromise the confidentiality of the corresponding secret keys. Furthermore the design of secure showing protocols is discussed, exemplified by secret-key certificate schemes based on Fiat-Shamir type signature schemes.

AMS Subject Classification (1991): 94A60
CR Subject Classification (1991): D.4.6
Keywords and Phrases: Cryptography, Certificates.
Note: Preliminary report, July 18, 1995.

1. Introduction.

This report is a sequel to [3], which introduced and formalized the notion of secret-key certificates, provided motivating examples, and described applications in which secret-key certificates are preferable over public-key certificates. The terminology and notation of that reference will be used freely throughout, without re-introducing it. Recall in particular the notions of certified key pair (a triple, consisting of a secret key, a corresponding public key, and a certificate on the public key) and certified public key (a pair, consisting of a public key and a certificate on it).

A construction principle was introduced in [3, Sect. 3.1.] for deriving practical secret-key certificate schemes from any Fiat-Shamir type signature scheme. In light of the limited applicability of this principle, it may be worthwhile to come up with alternatives. In Section 2 a new construction principle is described that has more general applicability.

The certificate issuing protocols described in [3] allow the Certification Authority to certify public keys without needing to know the corresponding secret keys. However, the Certification Authority can determine the secret keys of certified key pairs once it gets to see the certified key pairs later on. In Sect. 3 it is shown how to design practical secret-key certificate schemes that enable the Certification Authority to certify public keys, yet prevent it from being able to determine the corresponding secret keys afterwards.

Issuance of certified key pairs makes no sense without a corresponding showing protocol, in which the receiver of a certified key pair shows his certified public key to a party and in addition performs a cryptographic action with respect to the public key. A very general description of several exemplary showing protocols has been provided in [3, Sect. 2.1.]. When designing a showing protocol for a specific secret-key certificate scheme, some caution may need to be exercised to ensure its security. In Sect. 4 it is shown how to design secure showing protocols, exemplified by Fiat-Shamir type secret-key certificate schemes.

2. New Secret-Key Certificate Schemes.

The design principle for secret-key certificate schemes detailed in [3, Sect. 3.2] can be applied to signature schemes of the Fiat-Shamir type. It will be clear that there are many variations possible in the exact way the resulting secret-key certificate schemes are defined. Consider for instance the Schnorr-based secret-key certificate scheme described in [3], the certificate verification relation of which is defined in accordance with $g^r = (h_0 h)^c a$. It is easy to come up with a great many modifications of this scheme, by changing the certificate verification relation and/or the key pair definition for the participants. For example, one can define modifications in accordance with any the following verification relations: $g^r = (h_0 h)\, a^c$, $g^c = (h_0 h)^r a$, $g^c = (h_0 h)\, a^r$; or $h^r = h_0^c\, a$, $h^r = h_0\, a^c$, $h^c = h_0^r\, a$, $h^c = h_0\, a^r$; or $g^r = h^c\, a$, $g^r = h\, a^c$, $g^c = h^r\, a$, $g^c = h\, a^r$ (where the public key is defined with respect to the base h_0 instead of g); and so on, ad nauseam. Of course, in all these cases one can also define the public key of a participant with respect to a generator different from g and h_0, or with respect to multiple generators.

What about secret-key certificate schemes based on the DSA [12] or the ElGamal signature scheme [10]? It is easy to see that the construction for Fiat-Shamir type signature schemes can also be applied to these schemes. In fact, both can be thought of as Fiat-Shamir type signature schemes, although there is an important difference in that in the DSA and the ElGamal scheme only the message needs to be hashed. Moreover the hashing can be omitted if valid messages are defined according to an appropriate redundancy scheme. Defining the key generation algorithms for the CA and the participants as in the Schnorr-based secret-key certificate scheme described in [3, Sect. 3.2.], where G_q is a subgroup of $\mathbb{Z}_p^*$ for a prime p such that $q \mid (p-1)$, one can define the certificate verification relation for a DSA-based scheme in accordance with $g^c = (h_0 h)^a a^r$, where $c = \mathcal{H}(h)$. (More precisely, a certificate is a pair $(r, \overline{a}) \in \mathbb{Z}_q \times \mathbb{Z}_q$ such that $(g^{c/r}(h_0 h)^{-\overline{a}/r} \bmod p) \bmod q = \overline{a}$.) And, as detailed above, one can consider a cornucopia of alternative definitions for the certificate verification relation and the key pair definition. Similar secret-key certificate schemes can be based on the ElGamal signature scheme.

The RSA signature scheme [15] is of a different structure, and indeed the design principle of [3] cannot be applied to it. The same holds for many other types of signature schemes. In light of this, it is of interest to have a different construction principle.

2.1 A New Construction For Secret-Key Certificate Schemes

In many signature schemes, one has to sign a one-way function $f(\cdot)$ of the message instead of the message itself (or allow only negligibly few messages that conform to some appropriate redundancy scheme, but this case is not of interest here), for otherwise message-signature pairs can be forged. Of course, in these schemes it is really the message m itself that is signed and not $f(m)$, it's just that the signing algorithm has to apply $f(\cdot)$ as a first step before processing m, but for the sake of this description we will think of it in the latter way. For example, in the RSA scheme the signature on a message m is the v-th root of $f(m)$ in an RSA group; but we will think of the signature on m as being $m^{1/v}$, in which case for security one must sign $f(m)$ instead of m itself.

From any such signature scheme one can derive a secret-key certificate scheme by defining a certified key pair as follows: a message m is a secret key for a participant, $f(m)$ is the corresponding public key, and the secret-key certificate is the signature on $f(m)$. To be able to perform "useful" cryptographic actions (such as decryption, digital signing, or identification) with respect to a public key of a participant, the function $f(\cdot)$ should typically be chosen to be a number-theoretic one-way function, such as discrete exponentiation. Although the function $f(\cdot)$ for the underlying signature scheme usually is defined to be a hash-function such as MD5 [14] or SHA [1], so that it can be evaluated much more efficiently, most choices for number-theoretic one-way functions should make the underlying signature scheme at least as secure. As will be shown in example 2 below, one in addition may have to make a minor adaptation to cope with the fact that in practice the range of $f(\cdot)$ in the certificate scheme typically must be taken much larger than considered sufficient for the underlying signature scheme.

By defining the issuing protocol in the obvious way (the CA generates a random secret key x, and signs the corresponding public key $f(x)$), it follows immediately that forgery of certified key pairs is infeasible if a known message attack on the underlying signature scheme is infeasible. (Other security results hold for other types of issuing protocols, clearly.) The certificate simulation algorithm for the resulting secret-key certificate scheme can be defined in accordance with how signatures can be forged in the underlying signature scheme when the one-way function $f(\cdot)$ is not applied; trial and error must typically be used by the certificate simulation algorithm to ensure that the simulated public key is generated according to an (approximately) uniform distribution over the range of $f(\cdot)$.

Example 1. Applying the new construction principle to the RSA signature scheme results in the following definition for a certified key pair. A secret key for a participant is a number x, the corresponding public key is $f(x)$, and the secret-key certificate is $f(x)^{1/v} \bmod n$. Here, n is the product of two large prime numbers, and v is co-prime to $\lambda(n)$. Certified key pairs can be issued in the obvious way, and the certificate simulation algorithm is defined as follows:

Step 1. Generate c at random from $\mathbb{Z}_n^*$.

Step 2. Compute $h := c^v \bmod n$. If h is not in the range of $f(\cdot)$, go to Step 1.

Step 3. Output public key h and certificate c.

Any key pair definition for the participants will do to make a secure secret-key certificate scheme, as long as the corresponding $f(\cdot)$ is such that it is believed to make the RSA signature scheme secure. For example, it seems to be secure to define the key pairs according to the Guillou-Quisquater signature scheme [11], with respect to a modulus n' less than n; a certified key pair then is a triple $(x \in \mathbb{Z}_{n'}, h = x^w \bmod n', h^{1/v} \bmod n)$, where w is the public encryption exponent for the participant(s). Note that a key pair definition that sets $f(x) = x^w \bmod n$ is not secure: apart from the fact that the CA can compute the secret key of any key pair because it knows the factorization of n, certified key pairs can easily be forged.

Another key pair definition would be to use an RSA set-up, by defining $f(\cdot)$ as multiplication of two prime numbers. More precisely, a participant generates his own RSA modulus $n' < n$; the certificate of the CA then is $(n')^{1/v} \bmod n$. In case the participants can also choose their own encryption exponent and (the description of) a hash-function, the concatenation of this information with n should be signed. A certificate simulation algorithm for this secret-key certificate scheme does not seem to exist, however.

The certificate scheme in example 1 offers *public-key recovery*, meaning that the public key can be computed from the certificate; in a showing protocol it hence suffices to transfer only the certificate instead of the certified public key. In general, public-key recovery is possible if and only if the underlying signature scheme offers message recovery.

Example 2. It is well-known that DSA and ElGamal signatures on random messages can be forged, and that one must therefore first apply a one-way function before signing.

Applying the new construction principle to the DSA (using the same notation as at the beginning of this section) results in the following definition for a certified key pair. A secret key for a participant is a number x, the corresponding public key is $f(x)$, and a secret-key certificate is a pair $(r, \overline{a}) \in \mathbb{Z}_q \times \mathbb{Z}_q$ such that $(g^{f(x)/r} h_0^{-\overline{a}/r} \bmod p) \bmod q = \overline{a}$. Certified key pairs can be issued in the obvious way, and the certificate simulation algorithm is defined as follows:

Step 1. Generate α, β at random from $\mathbb{Z}_q$.

Step 2. Compute $r := -\beta^{-1}(g^{\alpha} h_0^{\beta} \bmod p) \bmod q$, and $c := \alpha\, r \bmod q$. If c is not in the range of $f(\cdot)$, go to Step 1.

Step 3. Output public key c and certificate $(r, (g^{\alpha} h_0^{\beta} \bmod p) \bmod q)$.

As in example 1, any key pair definition will do to make a secure secret-key certificate scheme, as long as the corresponding $f(\cdot)$ is such that it is believed to make the DSA signature scheme secure. To allow parameter sizes that are comparable to those of the CA, p and q can be defined such that $p - 1 = Aq$ for a small integer A (although the omission of this adjustment need not necessarily result in an insecure scheme when large parameter sizes are allowed).

A similar scheme can be based on the ElGamal signature scheme. Note that in the ElGamal signature scheme generators of the group $\mathbb{Z}_p^*$ are used, and so no additional adjustment is needed to allow for comparable parameter sizes.

One can also construct variations for which public-key recovery is possible, by applying the construction principle to variations that offer message recovery (such as described by Nyberg and Rueppel [13]).

Example 3. Consider the undeniable signature scheme of Chaum [8]. In this scheme the secret key x_0 and public key $(G_q, g, h_0 = g^{x_0}, f(\cdot))$ of the signer are defined as in the Schnorr signature scheme, and an undeniable signature on a message m is m^{x_0}. To verify a signature, a confirmation protocol with the signer must be performed. Undeniable signatures on arbitrary messages m can be forged, by generating m as a known power of g; but forging such signatures should be infeasible if $f(m)$ is signed instead of m. We can therefore design an (undeniable) secret-key certificate scheme by defining a secret key to be a number x, the corresponding public key to be $h := f(x)$, and the undeniable certificate to be h^{x_0}.

Note that the new construction principle can even be applied to Fiat-Shamir type signature schemes, although the mandatory inclusion of the randomly generated component of the signature into the one-way function $f(\cdot)$ makes the construction for these schemes somewhat less useful.

2.2 Comparison Between Construction Principles

Whether one should design a certificate scheme according to the new construction principle or according to that of [3, Sect. 3.1.] depends chiefly on what kind of issuing protocol one is interested in. The particular properties that an issuing protocol needs to have depend on the application at hand, and one can distinguish between a virtually unlimited number of different kinds of issuing protocols. Nevertheless, cryptographic research over the years has singled out several types of issuing protocols that have proven to be of particular interest.

The most basic of all types of issuing protocols is that in which there is no interaction at all: the CA simply generates the issued certified key pairs by itself. Clearly this can be done for any secret-key certificate scheme, and so neither of the two construction principle is favorable a priori. However, certain realizations derived by using the new construction principle are more efficient than the Fiat-Shamir type secret-key certificate schemes, and so are preferable in general. In particular secret-key certificates defined as in example 1 can be verified much more efficiently if v is very small.

As will be shown in the next section, secret-key certificates designed by applying the new construction principle can always be issued in such a way that the secret keys corresponding to certified public keys can never be compromised. When certificate schemes are designed in accordance with the principle of [3, Sect. 3.2.], practical such issuing protocols are more cumbersome to define, and may not even exist when the certificate verification relation and/or the key pair definition for participants is modified (see next section).

Ordinary blind issuing protocols (in which completely blinded certified key pairs can be retrieved) can be designed for many secret-key certificate schemes, regardless of the particular construction principle that has been applied. For instance, the RSA-based secret-key certificates of example 1 can be issued by means of a blind issuing protocol by using the idea of Chaum [7]. A participant hereto generates a random secret key x, and blinds the public key $h = f(x)$ to $s^v h \bmod n$ for a random $s \in \mathbb{Z}_n^*$. The result is provided to the CA, which returns $(s^v h)^{1/v} \bmod n$, from which the participant can extract $h^{1/v} \bmod n$. In general, a practical blind secret-key issuing protocol can be designed if a practical blind signature issuing protocol can be designed for the underlying signature scheme.

For the design of efficient privacy-protecting off-line cash systems [5] and pseudonymous credential mechanisms [6], restrictive blind issuing protocols [4] for certified key pairs are imperative. It is highly unclear how to design practical such issuing protocols for secret-key certificates that have been designed in accordance with the new principle. For these applications the schemes based on the principle of [3] are preferable, since a general technique for designing restrictive blind issuing protocols is known for Fiat-Shamir type secret-key certificate schemes (see [4]).

3. Issuing Protocols To Keep Secret Keys Confidential.

Of special interest for certificate schemes are issuing protocols in which the Certification Authority can certify a public key without being able to compromise the confidentiality of the corresponding secret key(s). Although such issuing protocols can be constructed for any secret-key certificate scheme by applying a theoretical construction of Yao [16], the resulting schemes will be far from practical.

Secret-key certificates designed in accordance with the new construction principle can trivially be issued in such a way that the confidentiality of the secret keys of issued certified key pairs cannot be compromised. The CA hereto simply signs the public keys that it is provided with. For example, the RSA-based secret-key certificates described in example 1 of Sect. 2 can be issued by the CA by returning the v-th roots modulo n of the public keys that it is provided with; and the DSA-based and ElGamal-based certificates of example 2 in Sect. 2 can be issued by the CA by returning DSA and ElGamal signatures on the provided public keys.

For secret-key certificate schemes derived by the construction principle of [3, Sect. 3.1.] it is much more difficult to come up with issuing protocols that prevent the compromise of the secret keys of issued certified key pairs. Nevertheless, a technique exists for designing such issuing protocols, which amounts to composing the certificate on a public key of a participant from a signature made with respect to the public key, and a signature of the CA made with respect to its own public key.

Rather than providing an abstract description of this technique, it will now be shown how to design an issuing protocol with the desired property for secret-key certificates based on the Schnorr signature scheme. The resulting secret-key certificate scheme differs from the scheme described in [3, Sect. 3.2.] only in the description of the certificate issuing protocol, which now is as follows:

Certificate issuing algorithm: To retrieve a certified key pair, $\mathcal{U}$ engages in the following issuing protocol with the CA:

Step 1. $\mathcal{U}$ generates at random a number $w \in \mathbb{Z}_q$, and sends $a := g^w$ and its the public key h to the CA.

Step 2. The CA generates at random a number $w_0 \in \mathbb{Z}_q$. It then computes $c := \mathcal{H}(h, I, g^{w_0}a)$ and $r_0 := cx_0 + w_0 \bmod q$, and sends (c, r_0) to $\mathcal{U}$.

$\mathcal{U}$ accepts if and only if $c = \mathcal{H}(h, I, g^{r_0}h_0^{-c}a)$. If this is the case then $\mathcal{U}$ computes $r := r_0 + cx + w \bmod q$.

To prove the security for the CA in the new secret-key certificate scheme, we need to make an assumption. Consider a modification of the Schnorr signature scheme, resulting from the replacement of the issuing protocol by the following protocol: the receiver generates at random a number $w \in \mathbb{Z}_q$, and sends $a := g^w$ and his message m to the signer; the signer generates a number $a_0 := g^{w_0}$ for a random number $w_0 \in \mathbb{Z}_q$, computes $c := \mathcal{H}(m, aa_0)$ and $r_0 := cx_0 + w_0$, where x_0 is the secret key of the signer, and sends (r_0, c) to the receiver; the receiver accepts if and only if $c = \mathcal{H}(m, g^{r_0}h_0^{-c}a)$; and, if this is the case, the receiver computes $r := r_0 + w \bmod q$. Notice that (r, c) is a Schnorr signature on m that has been generated according to the same probability distribution as in the standard Schnorr signature issuing protocol, and that $\log_g aa_0$ is as unpredictable to the receiver as a is in the standard Schnorr issuing protocol. Hence the following assumption seems plausible.

Assumption 1 *It is as hard to forge signatures on messages in this modification of the Schnorr signature scheme as it is to forge signatures on messages in the Schnorr signature scheme.*

We can now prove the following results for the new secret-key certificate scheme, as in [3, Sect. 3.1.]:

1. If $\overline{\mathcal{U}}$ accepts, then $(x, h, (r, c))$ is a certified key pair.
2. If Assumption 1 is true and the Schnorr signature scheme is secure against chosen message attacks, then no polynomial-time conspiracy can forge a certified key pair.
3. The certificate simulation protocol is perfect.

The proof of the second of these properties is similar to that of [5, Proposition 7].

In addition, we can prove that the secret key x cannot be compromised in the new scheme:

4. Knowledge of the certified public key $(h, (r, c))$ of $\overline{\mathcal{U}}$, and of the view of the CA in the corresponding execution of the issuing protocol, reveals no more information about x than a single execution of the Schnorr identification protocol with respect to public key g^x.

In fact, from the view of the CA and the certified public key $(h, (r, c))$ precisely one Schnorr signature with respect to the public key of $\mathcal{U}$ can be computed. Namely, $(r - r_0 + w_0 \bmod q, c)$ is a Schnorr signature with respect h on the message (h, I).

Note that if we design a similar issuing protocol for a secret-key certificate scheme based on a witness-hiding Fiat-Shamir type signature scheme (such as the Fiat-Shamir scheme, the Feige-Fiat-Shamir scheme, the Brickell-McCurley scheme, or the schemes of Okamoto), then it can be proven unconditionally that the secret key of $\mathcal{U}$ cannot be computed by the CA; partial information may leak, though.

Result 4 does not hold when the issuing protocol described in [3] is used: in that case x can be computed from the certified public key $(h, (r, c))$ and the view of the CA in the issuing protocol, according to $c^{-1}(r - r_0) \bmod q$.

The reader should find little difficulty in designing similar issuing protocols for secret-key certificate schemes based on other Fiat-Shamir type signature schemes, the DSA and the ElGamal scheme. Not any modification of a Fiat-Shamir type secret-key certificate scheme will do, though. For instance, of the many variants of Schnorr-based secret-key certificate schemes listed in Sect. 2, it is unclear how to proceed when a certificate verification relation such as $g^r = h^c a$ is used.

4. Showing protocols.

A protocol in which a party is provided with a certified public key of a participant, and in which in addition the participant performs a cryptographic action with respect to the public key, is called a *showing* protocol. General descriptions of exemplary showing protocols have been described in [3, Sect. 2.1.].

Let us call a cryptographic action *secure* if it can be performed successfully with respect to a public key only if a corresponding secret key is known. The design of a *secure showing protocol* for a secret-key certificate scheme requires somewhat greater care than for a public-key certificate scheme, for which one can use any secure cryptographic action to obtain a secure showing protocol. Namely, if a certificate simulation algorithm for a secret-key certificate scheme outputs certified public keys such that the certificate simulation algorithm and the CA *together* know a corresponding secret key, then a cheater might be able to successfully perform a cryptographic action with respect to a simulated public key by delegating part of the action to the CA; this clearly does not contradict the security of the cryptographic action.

Of course, since the CA is honest the delegation by the cheater must go unnoticed by the CA, and hence must make use of the information that is provided by the CA in executions of its issuing protocol. More generally, if the CA also performs other cryptographic actions besides issuing, these may be put to use by the cheater as well. It will intuitively be clear that such unnoticed delegation can only be successful in case (one of) the cryptographic action(s) performed by the CA "resembles" the cryptographic action in the showing protocol. For the secret-key certificate schemes designed according to the new construction principle of Sect. 2 this problem will usually only arise in contrived schemes, since to prevent forgery the definition of key pairs for participants already may not resemble that for the CA. For instance in the exemplary secret-key certificate schemes in Sect. 2, the certified public keys output by the certificate simulation algorithm do not have the property that the simulation algorithm and the CA together know the corresponding public keys.

The situation is more delicate for Fiat-Shamir type certificate schemes, constructed in accordance with the principle of [3, Sect. 3.2.], because the key pairs for the participants are defined in the same group as the key pair for the CA. The following example illustrates this potential design pitfall. It assumes (as in all but the last example in this section) that the secret-key certificate scheme based on Schnorr signatures, as detailed in [3], is used, and that it is secure.

Example 1. Suppose that a party wants to transfer an encrypted message m in G_q to $\mathcal{U}$, using the ElGamal encryption scheme [10] in the group G_q. The party first retrieves the certified public key $(h,(r,c))$ corresponding to the "identity" description I of $\mathcal{U}$, for instance from a public-key directory. If c is equal to $\mathcal{H}(h,I,g^r(h_0h)^{-c})$, then the party generates at random a number $s \in \mathbb{Z}_q$, and transfers the ciphertext $(g^s, h^s m)$ to $\mathcal{U}$. If the cryptographic action of ElGamal decryption is secure, which is the case assuming that the Diffie-Hellman key exchange assumption [9] in G_q holds for messages chosen in accordance with the message distribution, then only a party that knows $\log_g h$ can decrypt and recover m.

Now, suppose that the certified public key $(h,(c,r))$ has been generated by a cheater who is impersonating $\mathcal{U}$. Because we assumed the secret-key certificate scheme and the ElGamal encryption scheme to be secure, the cheater has negligible chance of being able to recover m from the ciphertext by himself. However, the cheater knows a number $t \in \mathbb{Z}_q$ such that $h = h_0^{-1}g^t$, and so the cheater and the CA, viewed as one party, do know the secret key corresponding to h; it is equal to $t - x_0 \bmod q$. Can the cheater exploit this by unnoticeably delegating part of the decryption task to the CA?

If the CA uses its public key not only for the issuing protocol, but also to receive encrypted messages, then the cheater can proceed as follows. He first computes the pair $(g^{-s}, h_0^{-s}m)$ from the ciphertext $(g^s, h^s m)$; this is possible because he can compute g^{st}. The new pair is given to the CA, which will decrypt and recover m. From a response of the CA, the cheater may be able to figure out the purport of the message. Alternatively, it may be sufficient for his malicious purposes to simply pass on the response of the CA to the party that originally provided the ciphertext.

Successful delegation of cryptographic tasks to the CA by making use of actions of the CA *other* than issuing, as in example 1, can always be prevented by letting the CA perform tasks other than issuing with respect to *independently* generated key pairs. This is recommendable anyways, not only in secret-key certificate schemes; if, for instance, the CA certifies public keys by making Schnorr signatures with respect to a public key of its own, and uses the same public key in a Schnorr identification protocol, then impersonation is easy.

To assess in general whether a cryptographic action with respect to a simulated public key can be delegated to the CA, by making use of executions of the issuing protocol, one must consider whether certified public keys can be simulated in such a way that the cryptographic action can be performed by using the issuer as an *oracle*. Note that the definition of a secret-key certificate scheme is not concerned with how the certificate simulation algorithm works, and in particular not with the existence of alternative certificate simulation algorithms, whereas this can be of importance in the assessment of the above issue.

In the issuing protocol in example 1 the CA in effect provides Schnorr signatures on messages of the receiver: if (r, c) is a secret-key certificate on h, and the receiver knows an $x \in \mathbb{Z}_q$ such that $h = g^x$, then $(r - c\,x \bmod q, c)$ is a Schnorr signature with respect to h_0 on the message (h, I). However, a signer that issues Schnorr signatures with respect to public key h_0 seems useless as an oracle to decrypt ElGamal encryptions with respect to h_0 (a rigorous proof of this may be hard to provide, though). Note that it can be shown (if a plausible intractability assumption similar to [5, Assumption 2] holds) that only certified public keys of the form $h_0^{-1}g^t$, for arbitrary $t \in \mathbb{Z}_q$, can be obtained, although the infeasibility of delegating seems to hold regardless of the existence of other certificate simulation algorithms.

Our next example demonstrates a showing protocol that uses a secure cryptographic action and yet is insecure because part of the task with respect to a simulated public key can be delegated by a cheater to the issuing protocol. Note that, to be able to perform kl successful delegations of a cryptographic action in a showing protocol to an issuing protocol in which a participant can only engage k times, l cheaters need to conspire.

Example 2. As in example 1, let $(h, (r, c))$ denote a certified public key corresponding to the identity description I of $\mathcal{U}$, generated by a cheater who wants to make digital signatures, to impersonate $\mathcal{U}$. Suppose that the signature scheme applied in the showing protocol is a minor variant of the Schnorr signature verification relation: a signature with respect to h on a message m is a pair (s, d) such that $d = \mathcal{H}(m, g^s h^d)$. Clearly, it should be as hard to forge such signatures as it is to forge Schnorr signatures. However, the cheater can exploit the fact that the CA in effect is issuing Schnorr signatures with respect to h_0. Namely, if (r_0, c_0) is a secret-key certificate of the CA on a public key g^x, and $h = h_0^{-1} g^t$ for $t \in \mathbb{Z}_q$ known to the cheater, then $(r_0 - c_0(x + t), c_0)$ is a digital signature in the showing protocol on the message (g^x, I_0), where I_0 denotes the identity description of the cheater.

The cheater in example 2 can only sign messages of a very special format, which moreover reveals the identity of the cheater. (In case the cheater is actually a conspiracy consisting of multiple participants, the identity of one the participants is revealed.) The fraud in example 2 can hence easily be prevented if messages signed in executions of the showing protocol have a format different from the format used to certify public keys. To demonstrate that this is not the most general measure, consider the following example.

Example 3. We will now modify the issuing protocol that was used in the previous two example, and make it *interactive*. More specifically, the CA in the modified issuing protocol first sends a number a to the receiver; the receiver then sends a challenge c to the CA; and the CA finally replies by sending a response r such that $g^r = (h_0 h)^{-c} a$.

A motivation for the use of such an issuing protocol by the CA might be to allow participants to retrieve completely blinded certified key pairs, to be used for instance in an on-line electronic coin system; to spend a coin, a certified key pair must be provided (which is verified on-line and accepted only if it has not been spent before) and a signature must be provided with respect to the public key. (Alternatively, by a suitable change of the key pair definition for participants the issuing protocol can be made *restrictive* blind and can be used in an *off-line* electronic coin system [5] or a pseudonymous credential mechanism [6]; see also example 4.)

With the modified issuing protocol it is no longer possible for the CA to check the format of the signed information that the receiver will end up with. Moreover, by using appropriate blinding techniques (as in [4, 5]) a cheater can even compute genuine Schnorr signatures in the showing protocol, instead of the modified Schnorr signatures considered in example 2. In a similar manner a cheater in the showing protocol can perform the cryptographic action of Schnorr identification with respect to a simulated certified public key.

Successful delegation of cryptographic tasks to the CA by making use of the *issuing* protocol, as in examples 2 and 3, can best be prevented by ensuring that the signature verification relation in the showing protocol cannot be "rewritten" (by substituting the expression in terms of the public key of the CA that is known of the simulated public key) to the verification relation in the issuing protocol. There are many ways in which this can be done, although it may be hard to come up with rigorous proofs of security.

For examples 2 and 3 one can define a signature in the showing protocol on a message m as a DSA signature or as an ElGamal signature. Alternatively one can redefine the definition of key pairs for participants in the secret-key certificate scheme; for example one can define a key pair for a participant to be a pair $(x, h = g_1^x)$, where g_1 has been generated by the CA as g^y for a random $y \in \mathbb{Z}_q$. Yet another alternative is to define the signature in the showing protocol to be a Schnorr signature with a suitably modified verification relation, by defining a signature (s, d) to be equal to one of $\mathcal{H}(m, g^d h^{-r})$, $\mathcal{H}(m, g^{d/r} h^{-1/d})$, or $\mathcal{H}(m, g^{r/d} h^{-1/d})$. Each of these three variants should be as secure as the Schnorr scheme, since their security can be argued in exactly the same way. Note that this alternative can be applied to other Fiat-Shamir type signature schemes as well, and also to Fiat-Shamir type identification schemes.

In certain applications the delegation strategy need not be prevented at all, as is demonstrated by the next two examples.

Example 4. In [5] an off-line electronic coin system is described, in terms of Schnorr-based secret-key certificates. In this system a secret key of the participant corresponding to his one-time public key (h, a) is a pair $(x_1, y_1), (x_2, y_2)$ such that $g^{x_1} g_1^{y_1} = h$ and $g^{x_2} g_1^{y_2} = a$. The participant can retrieve a certified public key in a restrictive blind issuing protocol. In the showing protocol (payment), the participant makes an Okamoto signature on a message (denoted by spec) with respect to its one-time public key (h, a). This signature is a triple (d, r_1, r_2) such that $d = \mathcal{H}(h, g^{r_1} g_1^{r_2} h^{-d}, \text{spec})$. The participant then transfers h, (c, r) and (d, r_1, r_2) to the payee.

Regardless of whether the setting with or without an additional tamper-resistant device is used, a "cheater" can make a payment in a manner similar to that specified in example 3. He hereto simulates a certified public key $((h, a), (c, r))$ by generating h according to $h = h_0^{-1} g^t$ for an arbitrary $t \in \mathbb{Z}_q$, and then uses an execution of the issuing protocol with the CA in order to compute a signature (r_1, r_2) on a specification spec (in a manner similar to that detailed in example 3). However, this strategy costs him exactly the amount that is transferred in the payment protocol (the value of the coin), and so does not bring any gain. Moreover, a coin that is "withdrawn" and "spent" in this way cannot be double-spent (except with negligible probability of success), not even when there is no tamper-resistant device or when it can be compromised, and so actually is disadvantageous to a cheater.

The same applies to the check-based off-line cash system based on secret-key certificates described in [2]. In one modus of operation, an account holder can retrieve certified key pairs for free, and the tamper-resistant device at payment time subtracts the transferred amount from a counter. However, since the delegation cannot be successful without use of the tamper-resistant device, it is of no use here either.

Nevertheless, to prevent the delegation strategy in this example altogether one should apply one of the measures described above. For example one can redefine the signature verification in the payment protocol to $g^{r_1} g_1^{r_2} a^{-d} = h$ (possibly for a hash-function that *never* maps its arguments to zero, although this is not mandatory).

In sum, the following guidelines for the design of secure showing protocols for secret-key certificates can be given. Firstly, delegation to cryptographic actions of the CA other than certification can simply be prevented by letting the CA use independently generated key pairs for those actions. Secondly, delegation to the issuing protocol (which should never apply in case the new construction principle is used, described in Sect. 2) should be assessed on a case-by-case basis, and in general can be prevented by ensuring that the cryptographic task, or the verification relation, in the showing protocol "differs" as much as possible from that of the CA. A rigorous prove of the effectiveness of the measure may be hard to provide, though, since one must hereto prove that the CA cannot be used as an oracle to perform the cryptographic action in the showing protocol with respect to simulated public keys.

5. Conclusion.

As we have seen, the new construction principle for secret-key certificate schemes is much less troublesome than the construction principle described in [3]: the key pair definition for the participants does not need to resemble that for the CA, and in particular does not need to be in the same group; it is easier to design issuing protocols in which the confidentiality of the secret keys of participants cannot be compromised; and in designing a showing protocol one normally need not be worried about the delegation strategy. This shows that secret-key certificate schemes designed by applying the new construction principle are not much more difficult to come up with than public-key certificate schemes, which can be designed straightforwardly from *any* signature scheme for the CA and any key pair definition for the participants, and for which any secure cryptographic action will do to obtain a secure showing protocol.

6. Acknowledgements

I want to thank Berry Schoenmakers at C.W.I., and Ron Rivest and his reading group at M.I.T., for comments that motivated the research for Sect. 4 and Sect. 3, respectively. Berry Schoenmakers noted the possibility of delegation described in example 4 of Section 4, and Ron Rivest and his reading group pointed out that the CA in the issuing protocol described in [3, Sect. 3.2.] can compute secret keys, corresponding to certified public keys, once it actually gets to see the certified public keys.

References

1. Accredited Standards Committee X9, "Working Draft: American National Standard X9.30-1993: Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry: Part 2: The Secure Hash Algorithm (SHA)," 1993.
2. Brands, S., "Off-line Cash Transfer by Smart Cards," Centrum voor Wiskunde en Informatica (CWI), Report CS-R9455, September 1994. Available by anonymous ftp from: ftp.cwi.nl:/pub/CWIreports/AA/CS-R9455.ps.Z. See also: Proceedings of the First Smart Card Research and Advanced Application Conference, Lille (France), Oct. 1994, pp. 101–117.
3. Brands, S., "Secret-Key Certificates," Centrum voor Wiskunde en Informatica (CWI), Report CS-R9510, February 1995. Available by anonymous ftp from: ftp.cwi.nl:/pub/CWIreports/AA/CS-R9510.ps.Z.
4. Brands, S., "Restrictive Blinding of Secret-Key Certificates," Centrum voor Wiskunde en Informatica (CWI), Technical Report, Februari 1995. An extended abstract appeared in: Advances in Cryptology – EUROCRYPT '95, Lecture Notes in Computer Science, Springer-Verlag.
5. Brands, S., "Off-Line Electronic Cash Based on Secret-Key Certificates," Centrum voor Wiskunde en Informatica (CWI), Report CS-R9506, January 1995. Available by anonymous ftp from: ftp.cwi.nl:/pub/CWIreports/AA/CS-R9506.ps.Z. See also: Proc. of the Second International Symposium of Latin American Theoretical Informatics (LATIN '95), Valparaíso, Chili, April 3–7, 1995.
6. Brands, S., "Privacy-protecting Digital Credentials Based on Restrictive Blinding," to appear.
7. Chaum, D., "Blind Signatures for Untraceable Payments," Advances in Cryptology – CRYPTO '82, Lecture Notes in Computer Science, Springer-Verlag, pp. 199–203.
8. Chaum, D., "Zero-knowledge undeniable signatures", Eurocrypt '90, LNCS 473, Springer-Verlag, pp. 458-464.
9. Diffie, W. and Hellman, M., "New Directions in Cryptography", IEEE Transactions on Information Theory 22/6 (1976), pp. 644–654.

10. ElGamal, T., "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, Vol. IT-31, No. 4, July 1985, pp. 469–472.

11. Guillou, L., Quisquater, J.-J., "A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory," Advances in Cryptology – EUROCRYPT '88, Lecture Notes in Computer Science, no. 330, Springer-Verlag, pp. 123-128.

12. NIST, "Specifications for a digital signature standard (DSS)," Federal Information Processing Standards Pub. (draft), Aug. 19, 1991.

13. Nyberg, K., Rueppel, R., "Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem," Pre-proceedings of EUROCRYPT '94, pp. 175–190.

14. Rivest, R., "The MD5 Message-Digest Algorithm," Internet Request for Comments (RFC) 1321, April 1992.

15. Rivest, R., Shamir, A., Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM 21 (1978), pp. 120–126.

16. Yao, A., "How to Generate and Exchange Secrets," Proceedings of the 27th FOCS, 1986, pp. 162-167.