Chapter 5

Combining Issuing and Showing Protocols

In this chapter we show how to seamlessly combine the showing protocol techniques of Chapter 3 with the issuing protocol techniques of Chapter 4, without adding complexity and without compromising security or privacy. We develop additional privacy techniques, for both certificate holders and certificate verifiers, and design limited-show certificates that reveal all the encoded attributes in case of fraud. We also design software-only techniques that enable the CA to discourage a variety of frauds.

5.1 Integration

Consider a PKI with one CA, many certificate holders (provers), and many verifiers. To gain access to a secure "service" provided by a verifier $\mathcal{V}$, $\mathcal{P}$ must demonstrate to $\mathcal{V}$ its possession of a certificate of the CA on attributes that meet certain criteria. The CA will issue a certificate to $\mathcal{P}$ only after the latter has authenticated itself or proved its right to obtain such a certificate. Clearly, the CA can non-interactively issue an ordinary digital signature on a public key h of $\mathcal{P}$ constructed as described in Chapter 3, but this does not offer any privacy with respect to the CA. In this section we show how to combine the showing protocol techniques of Chapter 3 with the issuing protocol techniques of Chapter 4.

5.1.1 Making the match

The case of DLREP-based scheme I or II or the immunization in Section 4.4.1

Suppose that the CA (playing the role of $\mathcal{P}_0$) uses DLREP-based scheme I or II, or their immunization described in Section 4.4.1. $\mathcal{P}$ (playing the role of $\mathcal{V}_0$) obtains a

certificate of the CA on a public key h' of the form

$$g_1^{x_1} \cdots g_l^{x_l} g_0^{\alpha_1},$$

where $x_1, \ldots, x_l \in \mathbb{Z}_q$ are the encoded attributes and $\alpha_1 \in \mathbb{Z}_q$ is a random number chosen by $\mathcal{P}$. $\mathcal{P}$ can subsequently demonstrate to $\mathcal{V}$ a property about the encoded attributes, in the manner described in Chapter 3. The perfect fit of the issuing and showing protocols is illustrated by the dual role now played by the blinding factor α_1: it ensures that h' is uncorrelated to the view of the CA in the issuing protocol, and also that $\mathcal{P}$ in the showing protocol does not reveal anything about the encoded attributes beyond the validity of the formula it demonstrates (see Proposition 3.6.1(c)).

It is not hard to prove the following generalization of Proposition 4.3.4.

Proposition 5.1.1. *Suppose that $\mathcal{P}$ follows the issuing protocol and uses the resulting certified key pair in a showing protocol execution in which it demonstrates some property of the encoded attributes. The view that $\widetilde{\mathcal{V}}$ can obtain could have originated (with uniform probability) from any one of the issuing protocol executions in which $\widetilde{\mathcal{P}}_0$ encoded attributes that $\widetilde{\mathcal{P}}_0$ knows satisfy this particular property.*

This result applies even when $\widetilde{\mathcal{P}}_0$ and $\widetilde{\mathcal{V}}$ conspire throughout, and applies also to the other protocol combinations considered in this section.

The case of the other DLREP-based schemes in Chapter 4

The match between the showing protocol techniques and the other DLREP-based issuing protocols described in Chapter 4 seems more problematic at first. $\mathcal{P}$ obtains a certificate on a public key h' of the form

$$(g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1},$$

where $\alpha_1 \in \mathbb{Z}_q^*$ is a random number chosen by $\mathcal{P}$. The blinding factor this time spreads across all exponents. This does not limit the applicability of our showing protocol techniques in any way, though. There are two approaches:

- If $\alpha_1 \neq 0 \bmod q$ (which is required) then

  $$h_0^{-1} = g_1^{x_1} \cdots g_l^{x_l} (h')^{-1/\alpha_1},$$

  and so $\mathcal{P}$ can demonstrate Boolean formulae for the DL-representation that it knows of h_0^{-1} with respect to $(g_1, \ldots, g_l, h')$. Although h' is not a fixed generator specified by the CA, the security properties are not adversely affected. Namely, the ability to prove knowledge of a DL-representation of h_0^{-1} with respect to $(g_1, \ldots, g_l, h')$ implies the ability to prove knowledge of a DL-representation of h' with respect to $(g_1, \ldots, g_l, h_0)$. Proposition 3.6.1(b) is easily seen to apply. The showing protocol techniques and results in Chapter 3

apply without change, because the distribution of $-1/\alpha_1 \bmod q$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_q$. Note that $\mathcal{P}$'s ability to demonstrate a Boolean formula, and thus to prove knowledge of a DL-representation of h_0^{-1} with respect to $(g_1, \ldots, g_l, h')$, implicitly demonstrates that $\alpha_1 \neq 0 \bmod q$ (much as in Section 3.4.1).

- The above twist of considering DL-representations of h_0^{-1} with respect to $(g_1, \ldots, g_l, h')$ is not necessary. Consider $h' = g_1^{x_1\alpha_1} \cdots g_l^{x_l\alpha_l} h_0^{\alpha_1}$. Let $z_0 := \alpha_1$, and let z_i denote $x_i\alpha_1 \bmod q$, for all $i \in \{1, \ldots, l\}$. $\mathcal{V}$ can verify that $\alpha_1 \neq 0 \bmod q$ by checking that $h' \neq 1$, and so demonstrating that $x_1 = \beta x_2 + \gamma \bmod q$, say, is equivalent to demonstrating that $z_1 = \beta z_2 + \gamma z_0 \bmod q$. This we can handle by using our demonstration technique for linear relations. It is easy to see that any Boolean formula can be handled in this manner.

As will be shown in Section 5.1.2, both approaches have unique security implications with respect to the delegation strategy in case they are combined with a secret-key certificate issuing protocol that admits arbitrary parallelization.

The second approach results exactly in the first approach if we have $\mathcal{P}$ in addition demonstrate that $z_0 \neq 0$, as part of the formula it is to demonstrate. In the example, the formula would become $(z_1 = \beta z_2 + \gamma z_0)$ AND NOT$(z_0 = 0)$. Of course, $\mathcal{V}$ in that case need no longer check that $h' \neq 1$.

Since $\mathcal{P}_0$ in the issuing protocol in Section 4.5.2 need not know the attributes it encodes, certificate holders can enjoy even greater privacy.

The case of RSAREP-based scheme I or II or the immunization in Section 4.4.1

To encode attributes $x_1, \ldots, x_l \in \mathbb{Z}_v$ into a certified key pair for $\mathcal{P}$, the CA and $\mathcal{P}$ engage in RSAREP-based scheme I or II or their immunization described in Section 4.4.1. $\mathcal{P}$ obtains a certificate of the CA on a public key h' of the form

$$g_1^{x_1} \cdots g_l^{x_l} \alpha_1^v,$$

where $\alpha_1 \in \mathbb{Z}_n^*$ is a random number chosen by $\mathcal{P}$, playing the same dual role as in the case of the DLREP-based schemes. The rest is clear.

The case of the immunized RSAREP-based scheme II in Section 4.4.2

Finally, in the second immunization of RSAREP-based scheme II, described in Section 4.4.2, $\mathcal{P}$ obtains a certificate on a public key h' of the form

$$g_1^{x_1\beta} \cdots g_l^{x_l\beta} h_0^{\beta} \alpha_1^v.$$

In the showing protocol, $\mathcal{P}$ must demonstrate that $\beta \neq 0 \bmod v$. If $\beta \neq 0 \bmod v$, then the RSA-representation $\mathcal{P}$ knows of h_0^{-1} with respect to $((h')^{-1}, g_1, \ldots, g_l, v)$ is

$$(e \bmod v, x_1, \ldots, x_l, z),$$

where $e, f \in \mathbb{Z}$ satisfy $e\beta + fv = 1$, and z is the product of α_1^e and the powers of $l+2$ junk factors resulting from normalization. It is easy to see that proving knowledge of an RSA-representation of h_0^{-1} with respect to $((h')^{-1}, g_1, \ldots, g_l, v)$ suffices to demonstrate $\beta \neq 0 \bmod v$; this is a simple application of the technique described in Section 3.4.2. Therefore, according to the equivalent of Proposition 3.6.1(b) for the RSAREP function, $\mathcal{P}$ demonstrates $\beta \neq 0 \bmod v$ as a by-product of demonstrating any Boolean formula for the RSA-representation it knows of h_0^{-1} with respect to $((h')^{-1}, g_1, \ldots, g_l, v)$. Alternatively, and more efficiently, $\mathcal{P}$ simply demonstrates that $\beta = 1 \bmod v$ as part of the formula it is demonstrating; there is no need to hide β.

Remarks

In many PKIs, only the ability to demonstrate Boolean formulae involving the attributes (i.e., not the more general form of linear relations) is required. In these PKIs the CA can encode attributes that contain more than $|q|$ or $|v|$ bits of information, such as digitally encoded photographs. Consider by way of example a certificate on a DLREP-based public key of the form

$$g_1^{\mathcal{F}_{q,g_l}(x_1)} \cdots g_{l-1}^{\mathcal{F}_{q,g_l}(x_{l-1})} g_l^{x_l}.$$

Assuming that $\mathcal{F}_{q,g_l}(\cdot)$ is a collision-intractable hash function with outputs smaller than q, $\mathcal{P}$ can demonstrate that $x_i = \beta$, for some $\beta \in \mathbb{N}$, by demonstrating that $\mathcal{F}_{q,g_l}(x_i) = \mathcal{F}_{q,g_l}(\beta)$. Likewise, demonstrating that $x_i \neq \beta$ can be done by demonstrating that $\mathcal{F}_{q,g_l}(x_i) \neq \mathcal{F}_{q,g_l}(\beta)$. The usefulness of this variation is limited if the CA sees the attributes it encodes, because attributes with large entropy are in effect identifiers that facilitate tracing and linking when disclosed; however, as we will show in Section 5.2.1, the CA can encode attributes without knowing their values.

In another variation, following Merkle [267] (see also Section 5.4.3), attributes are incorporated in a hash tree. For example, with x_2 chosen at random from $\mathbb{Z}_q$, the number x_1 in $g_1^{x_1} g_2^{x_2}$, could be the root of a Merkle hash tree. To reveal some of the attributes in the tree, $\mathcal{P}$ discloses node values from which the root value can be reconstructed. This approach improves efficiency (and enables limited selective disclosure, assuming $\mathcal{P}$ hashes along a random salt for each attribute to prevent an exhaustive search), but many of the techniques in this book no longer apply.

For efficiency reasons, in practice $\mathcal{V}_0$ may prefer to skip the verification of $\mathcal{P}_0$'s response(s) in the issuing protocol. $\widetilde{\mathcal{P}}_0$ cannot leak more than a single bit of information to $\mathcal{V}$, owing to $\widehat{\mathcal{V}}_0$'s blinding operations. Because $\mathcal{V}$ cannot determine whether $\widetilde{\mathcal{P}}_0$ or $\widehat{\mathcal{V}}_0$ is the source of incorrect certificate data supplied in the showing protocol, it has no choice but to reject incorrect data. Therefore, if $\mathcal{P}_0$'s response in the issuing protocol is incorrect then $\mathcal{V}_0$ will find out in the showing protocol.

From now on we will always assume that the public key h' of the certificate holder is hashed along when determining $\mathcal{V}$'s challenge c in a signed proof.

5.1.2 Coping with delegation

As noted in Section 2.6, care must be taken when the CA issues secret-key certificates. $\mathcal{P}$ may be able to use a simulated certified public key in the showing protocol and delegate part of the requested action to the CA in an execution of the issuing protocol.

Example 5.1.2. *Consider the immunized DLREP-based certificate scheme I in Section 4.4.2. To simulate a certified public key, $\widehat{\mathcal{P}}$ generates a number $\beta \in \mathbb{Z}_q$, sets $h' := g_0^\beta$, and computes a corresponding certificate. To issue a signed proof of the formula* TRUE*, say, $\widehat{\mathcal{P}}$ must be able to come up with $a \in G_q$ and $r_1, \ldots, r_{l+1} \in \mathbb{Z}_q$ such that*

$$\prod_{i=1}^{l} g_i^{r_i} h_0^{r_{l+1}} = (h')^c a \quad \textit{and} \quad c = \mathcal{H}_{q,g_l}(h', a, \ldots).$$

Hereto $\widehat{\mathcal{P}}$ engages in an execution of the certificate issuing protocol with the CA*, in the following manner. Upon receiving a_0, $\widehat{\mathcal{P}}$ sets $a := a_0$, $c := \mathcal{H}_{q,g_l}(h', a, \ldots)$, and sends the challenge $c_0 := -\beta c \bmod q$ to the* CA*. Upon receiving r_0 from the* CA*, $\widehat{\mathcal{P}}$ sets $r_i := x_i r_0 \bmod q$, for all $i \in \{1, \ldots, l\}$, and $r_{l+1} := r_0$. (Additional blinding operations can be applied to a, c_0, and $r_0, \ldots, r_{l+1}$.) It is easy to verify that the result convinces $\overline{\mathcal{V}}$. Note that this works also if the hash function in the showing protocol is different from that used in the issuing protocol.*

The delegation strategy in this example is without security consequences, because $\widehat{\mathcal{P}}$ demonstrates a formula that pertains to the attributes $(x_1, \ldots, x_l)$ that the CA "encodes" in the protocol execution to which $\widehat{\mathcal{P}}$ delegates the cryptographic action. In effect, $\mathcal{P}$ simply follows a shortcut that circumvents the need to compute a as a product of powers of $g_1, \ldots, g_l, h_0$. In most circumstances this shortcut is not preferable: it requires an online connection with the CA at the time of the showing protocol execution; simulated certified public keys cannot be reused; and, for interactive showing protocols, timing of issuing and showing protocol executions would reveal the link between them.

When using the certificate issuing protocol in Section 4.5.2, the possibility of delegation simply does not arise, because public-key certificates are issued. Whenever delegation is harmless, though, there is no reason to prefer public-key certificates. In fact, secret-key certificates are preferable for reason of their greater efficiency, provable security, and privacy. (The privacy benefits will be clarified in Section 5.2.2.) In the remainder of this section we examine for all the secret-key certificate issuing protocols in Chapter 4 whether $\widehat{\mathcal{P}}$ can abuse delegation to demonstrate a formula that does not apply to the attributes that the CA encodes in the issuing protocol execution with $\widehat{\mathcal{P}}$. We will show that in many situations the delegation strategy is not possible at all, and in those cases where it cannot be prevented it can easily be rendered harmless.

The case of the four issuing schemes in Section 4.2

Assumption 5.1.3. *For any of the four secret-key certificate schemes in Section 4.2, if $\overline{\mathcal{V}}$ accepts a formula demonstration by $\widehat{\mathcal{P}}$, then $\widehat{\mathcal{P}}$ must have engaged in an issuing protocol execution in which the* CA *encoded an attribute tuple that satisfies the formula.*

In other words, delegation may be feasible, but it does not enable $\widehat{\mathcal{P}}$ to pretend to have attributes for which it cannot (at the same moment) obtain a certificate from the CA. We now argue why this should be true.

Consider first DLREP-based certificate scheme I in Section 4.2.1. According to the DL-based analogue to Assumption 4.3.9, the only way to simulate a certified public key is to set $h' := h_0^{-1} g_0^{\beta}$, for some $\beta \in \mathbb{Z}_q$. Assume for the moment that the CA uses the same attribute tuple $(x_1, \ldots, x_l)$ in all protocol executions. In the same manner as in Proposition 4.3.10, the role of the CA in these protocol executions with $\mathcal{P}$ can be simulated by generating a random $\gamma \in \mathbb{Z}_q$ and setting $h_0 := h^{-1} g_0^{\gamma}$, on input $l+1$ random numbers $g_0, \ldots, g_l \in G_q$. Now, the demonstration of a Boolean formula by means of the showing protocol techniques of Chapter 3 is a proof of knowledge of a DL-representation $(y_0, \ldots, y_l)$ of h' with respect to $(g_0, g_1, \ldots, g_l)$; see Proposition 3.6.1(b). (For signed proofs, the validity of Proposition 3.6.1(b) follows from the DL-based analogue to Assumption 4.3.9.) From the four relations

$$\begin{aligned} h &= g_1^{x_1} \cdots g_l^{x_l} \\ h' &= h_0^{-1} g_0^{\beta} \\ h_0 &= h^{-1} g_0^{\gamma} \\ h' &= g_0^{y_0} g_1^{y_1} \cdots g_l^{y_l}, \end{aligned}$$

we get

$$g_0^{\beta-\gamma} g_1^{x_1-y_1} \cdots g_l^{x_l-y_l} = 1.$$

Thus, a polynomial-time collision-finding algorithm for the DLREP function can be constructed, unless $\beta = \gamma \bmod q$ and $y_i = x_i \bmod q$, for all $i \in \{1, \ldots, l\}$. Therefore, any choice for β should result in $\widehat{\mathcal{P}}$ demonstrating a property for a DL-representation

$$(\beta - \gamma \bmod q, x_1, \ldots, x_l)$$

of h' with respect to $(g_0, g_1, \ldots, g_l)$. This is the same as what $\overline{\mathcal{P}}$ can demonstrate by simply following the issuing and the showing protocol.

When issuing protocol executions are run with respect to different attribute tuples, the CA can no longer be simulated by setting $h_0 := h^{-1} g_0^{\gamma}$. However, the fact that these protocol executions cannot be run in parallel is believed to ensure that the security argument remains valid; the situation is reminiscent of that in Section 4.3.3.

The security argument can be easily generalized to the scenario considered by Proposition 4.3.15, which provides a limited degree of parallelization of protocol executions with respect to different attribute tuples.

Interestingly, DLREP-based certificate scheme I seems to admit delegation only when the challenge c_0 in the verification relation of the issuing protocol appears at the same position as $\mathcal{V}$'s challenge c in the verification relation of the showing protocol. In particular, delegation is believed to be infeasible for atomic formulae without a "NOT" connective: the verification relations for these formulae have $\mathcal{V}$'s challenge situated as an exponent of the "wrong" base number.

The same considerations apply to the other three secret-key certificate schemes in Section 4.2.

The case of the immunized DLREP-based scheme I in Section 4.4.2

In case all protocol executions involve the same attribute tuple, the security argument for DLREP-based scheme I this time results in the relation

$$g_0^{\beta - y_0\gamma} g_1^{y_0 x_1 - y_1} \cdots g_l^{y_0 x_l - y_l} = 1.$$

Consequently, any choice for β should result in $\widehat{\mathcal{P}}$ demonstrating a property for a DL-representation

$$((\beta/\gamma)x_1 \bmod q, \ldots, (\beta/\gamma)x_l \bmod q, \beta/\gamma \bmod q)$$

of h' with respect to $(g_1, \ldots, g_l, h_0)$; this is the same as what $\overline{\mathcal{P}}$ can demonstrate by simply following the issuing and the showing protocol. On the basis of this, it seems that any delegation strategy with undesirable security consequences must exploit the ability of $\widehat{\mathcal{P}}$ to engage in parallel in two (or more) issuing protocol executions with respect to distinct attribute tuples.

Consider now the following parallel delegation strategy, involving two parallel issuing protocol executions with respect to different tuples. The verification relation in the i-th issuing protocol execution, for $i \in \{0, 1\}$, is

$$g_0^{c_{0i}} = (h_0 g_1^{x_{1i}} \cdots g_l^{x_{li}})^{r_{0i}} a_{0i}.$$

$\widehat{\mathcal{P}}$ can combine the information obtained in the two issuing protocol executions into information that satisfies the combined verification relation

$$h_0^{-(\gamma_0 r_{00} + \gamma_1 r_{01})} g_0^{\gamma_0 c_{00} + \gamma_1 c_{01}} \prod_{i=1}^{l} g_i^{-(\gamma_0 x_{i0} r_{00} + \gamma_1 x_{i1} r_{01})} = a_{00}^{\gamma_0} a_{01}^{\gamma_1},$$

for some $\gamma_0, \gamma_1 \in \mathbb{Z}_q$ of its own choice. According to the DL-based analogue to Assumption 4.3.9, the only way to simulate a certified public key is to set $h' := g_0^{\beta}$, for some $\beta \in \mathbb{Z}_q$. According to Figure 3.1, the verification relation in the showing protocol for an atomic formula without a "NOT" connective is of the form

$$h_0^{r_0} (g_0^{\beta})^{-c} \prod_{i=1}^{l} g_i^{r_i} = a.$$

Consequently, $\widehat{\mathcal{P}}$ in the issuing protocol executions can choose c_{00} and c_{01} subject to $\gamma_0 c_{00} + \gamma_1 c_{01} = -\beta c \bmod q$. With $a := a_{00}^{\gamma_0} a_{01}^{\gamma_1}$, $\widehat{\mathcal{P}}$ can use the responses in the two issuing protocol executions to satisfy the verification relation in the showing protocol, by setting $r_0 := -(\gamma_0 r_{00} + \gamma_1 r_{01}) \bmod q$ and, for all $i \in \{1, \ldots, l\}$, $r_i := -(\gamma_0 x_{i0} r_{00} + \gamma_1 x_{i1} r_{01}) \bmod q$. Now, if the showing protocol execution is to demonstrate the formula TRUE, $\widehat{\mathcal{P}}$ does not gain anything by this strategy. For any other atomic formula without a "NOT" connective, the responses $r_1, \ldots, r_l$ must satisfy at least one linear relation. Since the responses r_{00}, r_{01} are outside of $\widehat{\mathcal{P}}$'s control, in the sense that they should be as unpredictable as two independent random variables, $\widehat{\mathcal{P}}$ can satisfy this linear relation only if it may select the formula coefficients by itself. In that case, the parallel delegation strategy enables $\widehat{\mathcal{P}}$ to pretend to have certified attributes that it in fact has never been issued.

Either of the following two measures is believed to suffice to make the parallel delegation strategy harmless:

- A description of the formula F is hashed along when forming $\mathcal{V}$'s challenge in the showing protocol. With $c = \mathcal{H}_{q,g_l}(g_0^{\beta}, a_{00}^{\gamma_0} a_{01}^{\gamma_1}, F, \ldots)$, it should be infeasible to determine c as an algebraic function of $\beta, \gamma_0, \gamma_1$ and the formula coefficients. (This is provably true in the random oracle model.) Therefore, it seems that $\widehat{\mathcal{P}}$, upon receiving (a_{00}, a_{01}), has no choice but to select $(\beta, \gamma_0, \gamma_1)$ and the formula coefficients before forming c. The rest of the security argument is similar to that in Section 4.4.2 for the restrictive blinding property of the immunized DLREP-based scheme I; the only workable choices seem to be those that result in formulae that $\overline{\mathcal{P}}$ can demonstrate by following the issuing and showing protocols.

- All the matrix entries (coefficients) that specify the Boolean formulae are restricted to sets V such that $|V|/q$ is negligible in k. (The set V may differ for each formula coefficient.) The idea is that it should be infeasible to come up with $(\beta, \gamma_0, \gamma_1)$ and admissible formula coefficients (favorable to $\widehat{\mathcal{P}}$) that satisfy the linear relations. (Note that $c = \mathcal{H}_{q,g_l}(g_0^{\beta}, a_{00}^{\gamma_0} a_{01}^{\gamma_1}, \ldots)$ must be satisfied as well.) For security reasons, it is recommendable to apply this measure in combination with the preceding measure.

These measures were also recommended in Chapter 3 to guarantee unmodifiability of signed proofs for atomic formulae without a "NOT" connective. (Recall, though, that they are not always necessary to guarantee unforgeability of signed proofs.)

For atomic formulae that contain one "NOT" connective, the delegation strategy (be it the sequential or the parallel variant) is believed to be infeasible altogether, without any additional measures; $\mathcal{V}$'s challenge in the verification relations for these formulae is an exponent of the wrong base numbers. As we have seen in Section 5.1.1, in the showing protocol $\mathcal{P}$ must demonstrate anyway that $\alpha_1 \neq 0$; in case it does so as part of the formula that it is demonstrating to $\mathcal{V}$, the whole formula

demonstration will always include a "NOT" connective. In other words, the issue of delegation can be circumvented altogether by requiring that $\mathcal{P}$ always demonstrate Boolean formulae for the DL-representation it knows of h_0^{-1} with respect to $(g_1, \ldots, g_l, h')$, rather than having $\mathcal{V}$ check that $h' \neq 1$ and having $\mathcal{P}$ demonstrate formulae without "NOT" connectives.

The case of the remaining secret-key certificate schemes

The observations with respect to the immunized DLREP-based scheme I in Section 4.4.2 apply also to the immunized RSAREP-based scheme II in Section 4.4.2 and to the DSA-like scheme in Section 4.5.1.

Finally, in the case of the immunized issuing protocols described in Section 4.4.1, the application of the function $f(\cdot)$ to the CA's initial witness in the issuing protocol is believed to make the delegation infeasible altogether.

5.2 Privacy improvements for certificate holders

In this section we describe various techniques to improve privacy for certificate holders.

5.2.1 Issuing protocol techniques

Users are free to obtain a single batch of certificates from the CA and to thereafter never again retrieve new ones. In effect, a certified public key is a *digital pseudonym*. All communications and transactions conducted using the same pseudonym are linkable. In applications such as online discussion groups, this allows certificate holders to build a reputation with respect to each of their pseudonyms. For reason of privacy, however, it will often be desirable to use each certificate only a limited number of times. In applications where there is no need to build a reputation, single use is optimal. It also has advantages in other respects, as we have seen in Section 1.1.

Issuing many certificates to each applicant does not significantly raise the burden for the CA, if only identifiers and other attributes need not be verified out-of-band each time. The following two methods facilitate the recurring issuance of certificates to the same certificate applicant:

- (Account-based method) The certificate applicant opens an *account* with the CA, and registers under a person identifier (see Section 1.1.2) or a pseudonym. The account holder uses an account key (or, if a digital pseudonym is used, the secret key of the pseudonym) to authenticate its requests for new certificates. Standard authentication techniques can protect against replay and, if desired, ensure non-repudiation.

If the attributes of each account holder are recorded in a database entry associated with the account, then the CA need not verify them more than once. Account holders can minimize the information the CA can learn about their certificate showing behavior by retrieving certificates in batches.

- (Account-less method) The certificate holder proves the right to retrieve new certificates by showing to the CA a previously retrieved certificate (not necessarily of the same type or issued by the same CA). In this manner, the CA can be assured of the authenticity of relevant attributes (identity or otherwise) without needing to keep an account database.

  This model does away with the account database by having each user port his or her own database entries, digitally certified by the CA for security. At the very least, the account-less method is natural for *refreshing* previously issued certificates. More generally, it fits well with the philosophy behind digital certificates to minimize the reliance on central databases; see Section 1.1.3.

In many PKIs it is more efficient to issue many short-lived certificates than to issue a few long-lived certificates and apply online certificate validation. By front-loading the handling of certificates and issuing certificates in batches, the CA greatly reduces the number of accesses to a central database. Also, certificate issuing can easily be scheduled, and in case of a communication or computation fault the same actions can be repeated until successful. Note that even in a PKI with long-lived certificates, the CA must be prepared to reissue certificates; certificates may be lost, stolen, or corrupted, and must be refreshed upon expiry.

The account-based method allows certificate holders to remain anonymous in the issuing protocol, by registering under a pseudonym and using an anonymous channel to retrieve certificates. Unlinkability of certificate retrievals can be achieved by opening multiple anonymous accounts. Full unlinkability, however, is feasible only when using the account-less approach. Hereto certificate applicants must be able to obtain attribute certificates that can be used to authenticate their right to retrieve new certificates.

The account-less approach also enables a certificate holder to anonymously have a previously issued certificate recertified by the CA. Anonymous refreshing of a certificate can be accomplished using any of the RSAREP-based certificate issuing schemes or the DLREP-based scheme described in Section 4.5.2. Suppose in the DLREP-based scheme that the CA is presented with $h' := (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1}$, a certificate on h', and (optionally) a signed proof (or another kind of proof) that discloses some property of the attributes (to enable $\mathcal{P}_0$ to certify unknown attributes that it knows satisfy a certain property) or at least authenticates the request. If the CA agrees to use h' in the new issuing protocol execution (in the role of $h_0 h$), the certificate holder can obtain a certificate on $(h')^{\beta} = (h_0 h)^{\alpha_1 \beta}$, for a random $\beta \neq 0 \bmod q$. The associativity of raising numbers in G_q to a power ensures that the encoded attributes remain intact. Note that different CAs, each with their own signing key, can

perform different recertification stages, assuming they all operate in the same group G_q.

The CA can *update* the attributes of anonymous certificate holders before recertifying them, without knowing the current attribute values. Consider hereto RSAREP-based scheme I. $\mathcal{V}_0$ receives a certificate on a public key h' of the form $g_1^{x_1} \ldots g_l^{x_l} \alpha_1^v$, for a random $\alpha_1 \in \mathbb{Z}_n^*$. Before recertification, the CA multiplies h' by $\prod_{i=1}^{l} g_i^{u_i}$, where $u_1, \ldots, u_l \in \mathbb{Z}_v$ represent the update values. This technique applies also to RSAREP-based scheme II and the immunized RSAREP-based schemes described in Section 4.4. The updating technique does not work when a hash function $\mathcal{F}(\cdot)$ is applied to the attributes (as described at the end of Section 5.1.1), and does not work for the DLREP-based schemes.

A special application of anonymous updating is to prevent the CA from learning the entire set of attributes of a certificate applicant. Attributes that can be assigned to a certificate applicant without requiring the applicant to disclose his or her identity can be encoded in successive anonymous executions of the issuing protocol, in each of which the certificate issued in the previous protocol execution is updated. To prevent linking by timing, there must be ample time between successive updates. Having different CAs certify different attributes further improves privacy.

5.2.2 Showing protocol techniques

To limit the privacy-invading powers of parties who build profiles from lists of certified public keys (obtained from certificate repositories or compiled by monitoring network traffic), one can use secret-key certificates and have certificate holders perform only zero-knowledge proofs. In this manner, nobody is able to obtain digitally signed evidence about the attributes of PKI participants.

Secret-key certificates can also reduce the scope for discrimination on the basis of (the lack of) one's right to participate in a PKI. Certified public keys obtained from a CA in one PKI can be combined with simulated certified public keys for other PKIs in an execution of the showing protocol, to prove knowledge of a secret key (more generally, an attribute property) for at least one of the certified public keys; the "OR" technique of Cramer, Damgård, and Schoenmakers [123] applies straightforwardly to our showing protocol techniques.

Instead of performing a zero-knowledge proof or issuing a signed proof, $\mathcal{P}$ in the showing protocol can also issue a *designated-verifier proof*. This notion, due to Chaum [102], guarantees that $\mathcal{P}$'s proof is convincing only to a designated verifier; this reduces the scope for privacy-invasive practices. (See Jakobsson, Sako, and Impagliazzo [219] and Krawczyk and Rabin [241] for improved constructions.) The idea is for $\mathcal{P}$ to perform its demonstration by proving knowledge of a secret key corresponding to its own public key or to one of the designated verifier, using a witness-indistinguishable proof of knowledge. Hereto our showing protocol techniques can be straightforwardly combined with the "OR" technique of Cramer, Damgård, and

Schoenmakers [123]. The resulting protocol transcript does not convince anyone but the verifier, because the verifier can generate transcripts with indistinguishable probability distribution. An inherent drawback of designated-verifier proofs is that $\mathcal{V}$ must be prevented from using a secret key that is known only to a group of verifiers. To overcome this attack, verifiers must either make a physical appearance in a Faraday cage (to have their public key certified) or a trusted party must securely issue a secret key to each verifier (e.g., by providing a smartcard that stores the key); both approaches have obvious drawbacks. In most PKIs either a signed proof or a zero-knowledge proof will be the preferred choice.

Our showing protocol techniques enable certificate holders to demonstrate properties of attributes that have been encoded into different certified key pairs. Limiting the number of attributes per certificate reduces the need to have attributes recertified (attributes with clearly distinct lifetimes can be encoded into different certified key pairs), improves efficiency,[1] and improves privacy when attributes are certified by different CAs. Any Boolean formula pertaining to the attributes in an arbitrary number of certified public keys can be demonstrated by applying our showing protocol techniques to the appropriate product of powers of the public keys, provided that the attributes specified in atomic formulae are all exponents of the same generator. For example, in the DLREP-based setting, with $h = \prod_{i=1}^{l} g_i^{x_i}$ and $h^* = \prod_{i=1}^{l} g_i^{x_i^*}$, demonstrating that $\alpha_1 x_1 + x_1^* = \alpha_2 \bmod q$, say, can be done by proving knowledge of a DL-representation of $h^{\alpha_1} h^* g_1^{-\alpha_2}$ with respect to $(g_2, \ldots, g_l)$. Depending on the certificate issuing scheme, for each public key an additional proof of knowledge of a representation may need to be performed, for security reasons. To prove knowledge of a representation of each of many public keys $h_1, \ldots, h_t$, knowledge of a representation of the product $h_1 \prod_{i=2}^{t} h_i^{\alpha_i}$ may be proved, for $\alpha_2, \ldots, \alpha_t$ generated at random from a large set. Either the α_i's are generated by application of a sufficiently strong one-way hash function to $h_1, \ldots, h_t$ and the initial witness of the prover, or the protocol is performed interactively and $\mathcal{V}$ generates the α_i's at random after receiving the initial witness.

This technique can be applied not only by a single certificate holder, but also by multiple certificate holders who wish to jointly demonstrate that their combined attributes meet certain criteria, without pooling their attributes. It can even be applied to certificates issued by different CAs, assuming all CAs operate in the same group; in DLREP-based scheme I, for instance, each CA may have its own x_0, but G_q and at least some of the generators should be the same.

The technique is less practical when $\mathcal{P}$ is to demonstrate a formula that involves attributes that are exponents of different g_i's in different certified key pairs. Auxiliary

[1] Attributes that are rarely shown in combination are best encoded into different certified key pairs. Other approaches to shorten certificates include: use of elliptic curve implementations, removing information from certificates, using better data encoding rules, and ensuring that certificate verifiers can derive the certificate holder's public key from its identifier and the CA's public key (Certicom's applies this approach in its "bullet" certificates).

commitments must then be spawned and additional relations must be proved, much as described in Section 3.7. In many PKIs, though, only the ability to demonstrate Boolean formulae involving the attributes (i.e., not the more general form of linear relations) is of interest; for these formulae, our showing protocol techniques readily apply to the attributes encoded into (arbitrarily many) different certified key pairs.

In some PKIs it is desirable that certificate holders can anonymously prove to be the originator of several showing protocol executions. This gives them the power to control the degree to which their communications and transactions can be linked. If the CA encodes into each certified key pair an attribute x_1 (possibly a random number) that is unique to the certificate applicant (applicants may provide $g_1^{x_1}$ or other suitable commitments to hide their x_1 from the CA), certificate holders can provide indisputable linking information for arbitrarily many public keys, $h_1, \ldots, h_t$, as follows. After the initial witnesses have been fixed, $\mathcal{V}$ generates t numbers, $\alpha_1, \ldots, \alpha_t$, at random from a set $V \subseteq \mathbb{Z}_q$, subject to the condition $\sum_{i=1}^{t} \alpha_i = 0 \bmod q$. It is easy to show that if $\mathcal{P}$ can prove knowledge of a representation of $\prod_{i=1}^{t} h_i^{\alpha_i}$ with respect to a tuple that does not contain g_1, then the probability of successful cheating is at most $1/|V|$. The α_i's may be generated as the output of a sufficiently strong one-way hash function to $\mathcal{P}$'s initial witnesses. The same technique applies to the RSAREP-based protocols.

A special application of the latter technique are credential systems in which certificate holders are to build pseudonymous reputations in the showing protocol. Using our techniques, certificate holders can establish digital pseudonyms with organizations; the pseudonyms can be thought of as anonymous accounts. To ensure that certificates can be shown only to organizations at which one has a pseudonym, we have the CA (or different CAs) encode a key holder identifier (e.g., a random number) into each pseudonym and attribute certificate. With $h = g_1^I g_0^{\alpha}$, say, denoting a person's pseudonym at an organization, and $h^* = g_1^{I^*} \prod_{i=2}^{l} g_i^{x_i}$ an attribute certificate, Boolean formulae of h^*/h with respect to tuples in which g_1 does not appear can be demonstrated only if $I = I^* \bmod q$. Separate proofs of knowledge of a representation of h^* and h may be needed; for h this would naturally be done when registering or when using the pseudonym with the organization.

Using our techniques it is also straightforward for several certificate holders to jointly demonstrate that their showing protocol executions did not all originate from the same certificate holder, or for one certificate holder to show that he or she was not involved in a fraudulent transaction; see Section 5.5.1 for details on the latter.

5.3 Privacy improvements for certificate verifiers

In many PKIs, certificate verifiers (must) submit their showing protocol transcripts to the CA (or another central authority), either online or off-line. This enables the CA to detect fraud with limited-show certificates, to keep track of the number of customers

served by $\mathcal{V}$, to gather statistics on how often a certain attribute property is disclosed, or to perform other administrative tasks.

For competitive or other reasons, $\mathcal{V}$ may want to hide from the CA all or part of the formulae demonstrated. On the other hand, with the possible exception of closed PKIs in which verifiers are tamper-resistant terminals representing the security interests of the CA, the CA is unlikely to trust verifiers with truthfully submitting summary indications of the required information. That is, $\widehat{\mathcal{V}}$ should not be able to provide false information to the CA.

As an example application, consider an untraceable electronic coin system with "earmarks." Each electronic coin is a certified key pair encoding an attribute x_1 that specifies (at least) the expiry date and the denomination of the coin, and attributes $x_2, \ldots, x_l$ that specify personal data (such as age, gender, hobbies, income, marital status, and perhaps the identity of the coin holder). To make a payment, the payer discloses x_1 to the shop. Depending on the conditions, the payer may in addition decide to disclose some of $x_2, \ldots, x_l$, for example to get a discount. To get its account credited, the shop deposits the coin to the bank. The shop is required to deposit a signed proof (to enable the bank to detect and trace double-spending) that discloses x_1 (to enable the bank to determine the redeemable amount), but the personal data that the shop acquired from the payer is none of the bank's business.

We now show how $\mathcal{V}$ can obtain a signed proof that hides all or part of the formula demonstrated by $\mathcal{P}$, while $\mathcal{V}$ itself becomes convinced of the entire formula.

Consider first the case in which $\mathcal{P}$ interactively demonstrates to $\mathcal{V}$ an atomic formula F of the form (3.2), in the manner depicted in Figure 3.1 and subject to the conditions of Proposition 3.3.7. Let F^* be any atomic formula obtained from F by pruning "AND" connectives from F (i.e., by deleting rows from the left-hand side matrix in (3.2)). We claim that $\mathcal{V}$ can obtain a signed proof that serves as self-authenticating evidence that F^* applies to $\mathcal{P}$'s DL-representation but that unconditionally hides all other information about F. The basic idea is for $\mathcal{V}$ to hash along a description of F^* instead of F when determining c, and to make it appear as if some of $r_{l-t+1}, \ldots, r_l$ were provided by $\mathcal{P}$. For instance, the signed proof $(c, (r_1, \ldots, r_{l-t+1}))$ corresponds to the formula F^* that is the result of deleting the first row of the left-hand side matrix in (3.2), assuming that a description of F^* is hashed along instead of F when forming c. This signed proof does not unconditionally hide all other information about F, though, because r_{l+1} does not have an independent random distribution; for any choice of $l-t$ elements of the tuple $(b_1, \alpha_{11}, \ldots, \alpha_{1,l-t})$, the remaining element is uniquely defined. To take care of this aspect, $\mathcal{V}$ must randomize r_{l-t+1} by adding a random element $\beta \in \mathbb{Z}_q$; this must be compensated by hashing along $ag^{\beta}_{\pi(l-t+1)}$ instead of a when forming c.

More generally, with $\mathcal{P}$ demonstrating to $\mathcal{V}$ a formula F of the form (3.2), the following protocol steps result:

Step 1. (This step is identical to Step 1 of the protocol in Section 3.3.) $\mathcal{P}$ generates

at random $l-t$ numbers, $w_1, \ldots, w_{l-t} \in \mathbb{Z}_q$, and computes

$$a := \prod_{i=1}^{l-t} g_{\pi(i)}^{w_i} \prod_{i=1}^{t} g_{\pi(l-t+i)}^{-\sum_{j=1}^{l-t} \alpha_{ij} w_j}.$$

$\mathcal{P}$ then sends the initial witness a to $\mathcal{V}$.

Step 2. $\mathcal{V}$ decides on F^* by deleting rows from the left-hand side matrix in (3.2). Without loss of generality, we assume that F^* is the result of pruning the first $t^* \leq t$ rows.

$\mathcal{V}$ generates at random t^* blinding factors $\beta_1, \ldots, \beta_{t^*} \in \mathbb{Z}_q$, and computes $a^* := a \prod_{i=1}^{t^*} g_{\pi(l-t+i)}^{\beta_i}$. $\mathcal{V}$ then forms $c := \mathcal{H}_{q,g_l}(h, m, F^*, a^*)$, for some (optional) message m, and sends c to $\mathcal{P}$.

Step 3. (This step is identical to Step 2 of the protocol in Section 3.3.) $\mathcal{P}$ computes a set of responses, responsive to $\mathcal{V}$'s challenge $c \in \mathbb{Z}_s$, as follows:

$$r_i := c x_{\pi(i)} + w_i \bmod q \quad \forall i \in \{1, \ldots, l-t\}.$$

$\mathcal{P}$ then sends $(r_1, \ldots, r_{l-t})$ to $\mathcal{V}$.

As in the protocol in Section 3.3, $\mathcal{V}$ computes

$$r_{l-t+i} := b_i c - \sum_{j=1}^{l-t} \alpha_{ij} r_j \bmod q \quad \forall i \in \{1, \ldots, t\},$$

and accepts the demonstration of F if and only if the verification relation

$$\prod_{i=1}^{l} g_{\pi(i)}^{r_i} h^{-c} = a$$

holds. If $\mathcal{V}$ accepts, it computes $r^*_{l-t+i} := r_{l-t+i} + \beta_i \bmod q$, for all $i \in \{1, \ldots, t^*\}$, and outputs the signed proof

$$c, r_{\pi^{-1}(1)}, \ldots, r_{\pi^{-1}(l-t)}, r^*_{\pi^{-1}(l-t+1)}, \ldots, r^*_{\pi^{-1}(l-t+t^*)}.$$

The CA (or anyone else) accepts the signed proof if and only if

$$c = \mathcal{H}_{q,g_l}(h, m, F^*, \prod_{i=1}^{l-t} g_i^{r_{\pi^{-1}(i)}} \prod_{i=1}^{t^*} g_{l-t+i}^{r^*_{\pi^{-1}(i)}} h^{-c}).$$

Note that if $t^* = t$ then the signed proof hides F in its entirety.

According to Proposition 3.6.3, if $\mathcal{V}$'s challenge is formed by hashing at least h, any initial witnesses, and a unique description of a formula F, then a signed proof

serves as self-authenticating evidence that $\mathcal{P}$ knows (knew) a representation of h, and that $\mathcal{P}$'s representation satisfies the formula F. It follows that $\mathcal{V}$ can hide an arbitrary part of the formula demonstrated, but cannot obtain a signed proof for a formula that does not apply to $\mathcal{P}$'s representation: F^* is implied by the formula F demonstrated by $\mathcal{P}$ in the showing protocol execution from which the signed proof resulted.

The same technique applies to atomic formulae of the form (3.4). Peculiarly, though, $\mathcal{V}$ can only obtain signed proofs for formulae F^* obtained by pruning linear equalities from F^*; it is unclear how to prune the linear inequality.

More generally, if $\mathcal{P}$ interactively demonstrates an arbitrary formula F of the form (3.7), then $\mathcal{V}$ can obtain a signed proof for any formula that is obtained by pruning one or more of the j subformulae of F. Hereto $\mathcal{V}$ simply omits hashing along the initial witness sets for those subformulae when forming c.

In sum, when $\mathcal{P}$ is interactively demonstrating to $\mathcal{V}$ a formula F of the form (3.7), $\mathcal{V}$ can obtain a signed proof for any formula F^* that is obtained from the formula F demonstrated by $\mathcal{P}$ by pruning "AND" connectives. The signed proof unconditionally hides all other information about F. How much $\mathcal{V}$ may hide in the signed proof it deposits to the CA could depend on the policies set by the latter.

In case $\mathcal{P}$ wants to prevent $\mathcal{V}$ from obtaining a signed proof for a formula other than the formula F it is demonstrating to $\mathcal{V}$, it must form c itself by hashing along a description of F. More generally, $\mathcal{P}$ can ensure that $\mathcal{V}$ obtains a signed proof that convinces of a formula F^* obtained from F by pruning "AND" connectives by hashing along a description of F^* itself (and possibly a^* instead of a, for blinding factors supplied by $\mathcal{V}$).

In case of disputes, it may be desirable that $\mathcal{V}$ can "open up" the signed proof to prove that $\mathcal{P}$ demonstrated F, not just F^*. Hereto $\mathcal{V}$ should form c by hashing along a commitment to (the hidden part of) F, and save the blinding factors it used in Step 2 (in order to reconstruct those responses of $\mathcal{P}$ that it blinded). $\mathcal{P}$ has leverage to settle disputes as well: it can always prove to the CA to be the originator of the signed proof, by revealing h and proving knowledge of a corresponding secret key.

An efficiency improvement is possible in case $\mathcal{P}$ demonstrates a formula F of the form (3.6) containing only atomic subformulae of the form (3.2), and $\mathcal{V}$ wants to obtain a signed proof that hides F entirely. Let $s := q$. With $\prod_{i=1}^{l} g_{\pi(i)}^{r_{it}} h^{-c_t} = a_t$ denoting the verification relation for the t-th atomic subformula, for $t \in \{1, \ldots, j\}$, we have

$$\prod_{i=1}^{l} g_{\pi(i)}^{\sum_{j=1}^{t} r_{ij}} h^{-c} = \prod_{i=1}^{t} a_i.$$

Consequently, c together with the responses in this compound verification relation form a signed proof for the formula TRUE. $\mathcal{V}$ must hash along $a := \prod_{i=1}^{t} a_i$ (instead of all a_i's) and a description of the formula TRUE when determining c. There is no need for blinding factors or interaction.

Our techniques thus far do not enable $\mathcal{V}$ to obtain a signed proof for any formula

F^* implied by F. To get around this limitation, $\mathcal{P}$ should submit two proofs to $\mathcal{V}$: a signed proof that discloses the minimal information required by the CA (perhaps just the formula TRUE) and a proof (not necessarily a signed proof) that discloses the formula $\mathcal{V}$ is interested in. A drawback of this approach is increased computation and communication complexity.

All the techniques in this section apply straightforwardly to the RSAREP-based showing protocols.

5.4 Limited-show certificates

In this section we show how to ensure that the CA (or another central party to which showing protocol transcripts are submitted) can compute all the attributes encoded into a certified key pair once the certificate is shown more than a predetermined number of times. Benefits of this technique will be explained in Section 5.5.

5.4.1 Static one-show certificates

Consider the setting in Chapter 3. Suppose that $\mathcal{P}$ engages in a showing protocol execution, demonstrating a Boolean formula F for its DL-representation of h with respect to $(g_1, \ldots, g_l)$. We do not care whether $\mathcal{P}$ gives a signed proof, a zero-knowledge proof, or otherwise. We claim the following.

Proposition 5.4.1. *Suppose that formulae are demonstrated using the showing protocol described in Section 3.6.1. If $\widehat{\mathcal{P}}$ demonstrates the same formula twice using the same public key, responsive to any two different challenges of $\overline{\mathcal{V}}$ but with respect to the same initial witness (sets), then with overwhelming probability $\mathcal{P}$'s secret key can be efficiently computed from the two accepting views of $\overline{\mathcal{V}}$.*

Proof. To prove the claim, we consider the following four cases for the formula F that is demonstrated:

Case 1. F is an atomic formula consisting of zero or more "AND" connectives and no other connectives, in the form (3.3). Recall from Figure 3.1 that the verification relation in the protocol for demonstrating F is

$$\prod_{i=1}^{l} g_{\pi(i)}^{r_i} h^{-c} = a,$$

where a is the initial witness. The verification relation with respect to a challenge $c^* \neq c \bmod s$ is

$$\prod_{i=1}^{l} g_{\pi(i)}^{r_i^*} h^{-c^*} = a.$$

Division of the two relations results in

$$h^{c-c^*} = g_{\pi(1)}^{r_1-r_1^*} \cdots g_{\pi(l)}^{r_l-r_l^*},$$

and from $s \leq q$ it follows that

$$h = g_{\pi(1)}^{(r_1-r_1^*)/(c-c^*)} \cdots g_{\pi(l)}^{(r_l-r_l^*)/(c-c^*)}.$$

The DL-representation $((r_1-r_1^*)/(c-c^*) \bmod q, \ldots, (r_l-r_l^*)/(c-c^*) \bmod q)$ can be efficiently computed from the two views of $\overline{\mathcal{V}}$. With overwhelming probability, this DL-representation is $\mathcal{P}$'s secret key, $(x_{\pi(1)}, \ldots, x_{\pi(l)})$. Namely, according to Corollary 3.3.2 $\mathcal{P}$ must know at least one secret key, and if this is different then $<\widehat{\mathcal{P}}, \overline{\mathcal{V}}>$ is a collision-finding algorithm for the DLREP function used to implement $\mathcal{P}$'s commitment; according to Proposition 2.3.3 this contradicts the assumption that the underlying DL function is one-way.

Case 2. F is an atomic formula consisting of zero or more "AND" connectives, one "NOT" connective, and no other connectives, in the form (3.4). Recall from Figure 3.5 that the verification relation in the protocol for demonstrating F is

$$\prod_{i=1}^{l} g_{\pi(i)}^{r_i} h^{-r_0} = a.$$

The verification relation with respect to a challenge $c^* \neq c \bmod s$ is

$$\prod_{i=1}^{l} g_{\pi(i)}^{r_i^*} h^{-r_0^*} = a.$$

Division of the two relations results in

$$h^{r_0-r_0^*} = g_{\pi(1)}^{r_1-r_1^*} \cdots g_{\pi(l)}^{r_l-r_l^*}.$$

If $r_0 = r_0^* \bmod q$, then

$$(r_1 - r_1^* \bmod q, \ldots, r_l - r_l^* \bmod q)$$

is a DL-representation of 1 with respect to $(g_{\pi(1)}, \ldots, g_{\pi(l)})$, and two cases can be distinguished:

- If this is a non-trivial DL-representation of 1, then $<\widehat{\mathcal{P}}, \overline{\mathcal{V}}>$ has found a DL-representation of 1 other than the trivial one. According to Proposition 2.3.3 this contradicts the assumption that the DL function is one-way.

- If this is the trivial DL-representation, then $r_i = r_i^* \bmod q$ for all $i \in \{1, \ldots, l\}$. Recall that $\mathcal{V}$ computes $(r_{l-t+1}, \ldots, r_l)$ from $(r_0, \ldots, r_{l-t})$ according to

$$r_{l-t+i} := b_i r_0 - f_i c - \sum_{j=1}^{l-t} \alpha_{ij} r_j \bmod q \quad \forall i \in \{1, \ldots, t\},$$

  where $t \geq 1$. From the linearity of these t responses it follows that $f_i(c - c^*) = 0 \bmod q$, for all $i \in \{1, \ldots, t\}$. From $s \leq q$ it follows that $c \neq c^* \bmod q$, and so $f_1, \ldots, f_t$ are all zero. This contradicts the presence of one "NOT" connective.

Therefore, with overwhelming probability $r_0 \neq r_0^* \bmod q$. It follows that

$$((r_1 - r_1^*)/(r_0 - r_0^*) \bmod q, \ldots, (r_l - r_l^*)/(r_0 - r_0^*) \bmod q)$$

is a DL-representation of h with respect to $(g_{\pi(1)}, \ldots, g_{\pi(l)})$. The same argument as in Case 1 completes the proof.

Case 3. F is a formula connecting atomic subformulae by zero or more "OR" connectives and no other connectives, in the form (3.6). According to the protocol in Section 3.5, $\mathcal{V}$ accepts if and only if $c = \sum_{i=1}^{j} c_i \bmod s$ and the verification relations for all the atomic subformulae hold. Let c^* denote the challenge of $\mathcal{V}$ in the second protocol execution involving the same initial witness (set), and c_i^* the challenge used by $\widehat{\mathcal{P}}$ for the i-th subformulae in that protocol execution. We have $\sum_{i=0}^{t} c_i^* = c^* \bmod s$. From $c^* \neq c \bmod s$ it follows that there exists at least one i such that $c_i^* \neq c_i \bmod s$. This index i corresponds to an atomic subformula that $\widehat{\mathcal{P}}$ has demonstrated twice with respect to the same initial witness but different challenges. We are now in Case 1 or 2, and refer to these cases for the completion of the proof.

Case 4. F is an arbitrary Boolean formula, in the form (3.7). According to the protocol in Section 3.6, $\mathcal{V}$ accepts if and only if it accepts $\mathcal{P}$'s demonstration of each of the j subformulae. We can therefore consider $\widehat{\mathcal{P}}$'s demonstration of any one of these subformulae, and apply the proof of Case 3.

This completes the proof. □

The same result can be proved for the RSAREP-based showing protocols.

Based on Proposition 5.4.1 it is straightforward to construct a showing protocol that protects $\mathcal{P}$'s privacy if and only if $\mathcal{P}$ does not use the same certified public key more than once. $\mathcal{P}$ must hereto commit already in the certificate issuing protocol to the initial witness (sets) that it will use in the showing protocol, so that the CA can certify these along. Specifically, for any of the restrictive blind certificate issuing

protocols in Chapter 4, $\mathcal{P}$ (i.e., $\mathcal{V}_0$) must hash along its initial witness (sets) for the showing protocol when computing c'_0 in Step 2 of the issuing protocol.

It is natural to think of h (or h', in the notation used in the issuing protocols in Chapter 4) together with the initial witness (sets) of $\mathcal{P}$ as a one-time public key of $\mathcal{P}$. Any two showing protocol executions by $\mathcal{P}$, with respect to the same one-time public key but different challenges, reveal its secret key. In case $\mathcal{P}$ non-interactively issues signed proofs, the use of different challenges can be forced by having $\mathcal{P}$ hash along a unique message m when forming $\mathcal{V}$'s challenge; if the hash function used to form $\mathcal{V}$'s challenge is collision-intractable, it is infeasible to compute two different messages that are mapped to the same challenge, regardless of any additional inputs to the hash function. Note that m need not be generated at random: Proposition 5.4.1 does not require $\mathcal{V}$'s challenge to satisfy a particular probability distribution. In a practical implementation, m could be the concatenation of an identifier for $\mathcal{V}$ (e.g., a true name, a local name, a pseudonym, or a public key) and a nonce. [2]

The limited-show technique applies even if the certificate verifier uses the technique in Section 5.3 to hide (part of) the formulae demonstrated, regardless of the strategy of $\widehat{\mathcal{P}}$ and $\widehat{\mathcal{V}}$.

The proof of Proposition 5.4.1 makes use of the fact that $< \widehat{\mathcal{P}}, \overline{\mathcal{V}} >$ cannot compute a collision for the DL function used to implement $\mathcal{P}$'s commitment. When combining the showing protocol with a certificate issuing protocol, we must see to it that this remains true. For the certificate schemes in Section 4.2 and the immunization in Section 4.4.1, $\widehat{\mathcal{P}}$'s inability to learn a non-trivial representation of 1 follows from Lemma 4.3.5. (Recall that we needed this property to prove blinding-invariance.) For the public-key certificate scheme in Section 4.5.2 the property follows from the fact that the CA need not know a non-trivial DL-representation of 1 to perform the issuing protocol. The other schemes in Chapter 4 are believed to meet the desired property as well.

The delegation strategy is never an issue. $\widehat{\mathcal{P}}$ cannot use the delegation strategy to show a number of certificates that exceeds the number of issuing protocol executions it would need to engage in; "forgery" is not possible. In fact, even if a secret-key certificate issuing protocol is used that admits delegation, reuse of a simulated certified public key is not possible because the CA uses an independent random initial witness in each protocol execution.

The most obvious application of limited-show certificates is to implement certificates that by their very nature may be shown only a limited number of times, such as food stamps, subway tokens, and electronic coins. In Section 5.5 we will see that limited-show certificates are advantageous also to prevent fraud with other types of certificates, such as personal certificates.

[2] The nonce may be a sequence number, a random number provided by $\mathcal{V}$, or a sufficiently accurate estimate of the time and date of the demonstration; in the latter case the showing protocol can be non-interactive, assuming $\mathcal{P}$ can determine the time without the assistance of $\mathcal{V}$.

In many PKIs, only one encoded attribute need be computable in case of fraud. Of special interest is the case where one of the encoded attributes is an identifier of $\mathcal{P}$, which normally need not be shown but must be computable when a limited-show certificate is shown too many times. Instead of storing the entire protocol view, in this case it is more efficient to store those exponents that correspond, in the expanded form of the verification relation, to $h, g_1, \ldots, g_l$. For instance, to be able to compute $x_{\pi(i)}$ in case a formula of the form (3.3) is demonstrated twice, for some $i \in \{1, \ldots, l\}$, it suffices to store (c, r_i) in each showing protocol execution. According to Subsections 2.2.2 and 2.2.3, 25-byte exponents suffice for long-term security, which makes our one-show certificate technique highly practical.

In a typical PKI implementation, there will be many verifiers. To enable detection of reuse of a one-show certificate, verifiers should deposit (relevant data of) the transcripts of their showing protocol executions to the CA (or another central party), in a timely fashion. (Either in real-time during certificate validation or batched at a convenient moment later on.) Verifiers that the CA trusts (such as tamper-proof verifier devices issued by the CA) may accept zero-knowledge demonstrations and deposit only the minimal data needed by the CA, but all others must receive and deposit signed proofs on challenges for which a nonce has been hashed along. The latter group of verifiers must be incited to perform the deposit. How to accomplish this depends on the PKI at hand. Incentives may be either positive (e.g., a financial reward for each deposited transcript) or negative (e.g., a penalty in case audits reveal a discrepancy between the number of customers serviced and the number of transcripts deposited). In some applications, it may be in the best interest of the verifiers themselves to faithfully perform the deposit, for instance if a successful attack leads to damages to their own organization.

By making use of its knowledge of $y_i = \log_{g_l} g_i$, for all $i \in \{1, \ldots, l-1\}$, the CA can speed up the verification of transcripts of showing protocol executions. Instead of verifying whether

$$\prod_{i=1}^{l} g_i^{r_i} h^{-c} = a,$$

say, the CA can simply verify whether

$$g_l^{\sum_{i=1}^{l-1} y_i r_i + r_l} h^{-c} = a.$$

Also, the CA can apply batch-verification to verify one or multiple transcripts, as described in Section 2.5.3.

5.4.2 Dynamic one-show certificates

With the *static* one-show certificate technique in Section 5.4.1, $\mathcal{P}$ must anticipate in the certificate issuing protocol which formula it will demonstrate in the showing

protocol. Depending on the application, this may or may not be a drawback. For instance, the design of a privacy-protecting electronic coin system may be such that each coin encodes two attributes, one to specify (at least) the coin expiry date and the coin denomination and the other to specify the identity of the coin owner; the formula for the showing protocol (payment) would always require the disclosure of just the first attribute. In many PKIs, however, the requirement that the formula must be anticipated is inconvenient or even unrealistic. Typically, the formula to be demonstrated will depend on the verifier or its service, and $\mathcal{P}$ does not know in advance which service of which verifier it will want to get access to. In the case of unlimited-show certificates, $\mathcal{P}$ can postpone the moment of computing its initial witness (sets) until Step 1 of the showing protocol, but this cannot be accomplished for limited-show certificates. We now show how to get around this. Again, we assume the setting of Chapter 3, and detail the technique only for the DLREP-based showing protocols.

Atomic formulae without "NOT" connectives

Consider first the case that $\mathcal{P}$ is to demonstrate an atomic formula of the form (3.3), using the showing protocol depicted in Figure 3.1. In Step 2 of the issuing protocol, instead of generating the initial witness in a formula-dependent manner, $\mathcal{P}$ simply computes $a := \prod_{i=1}^{l} g_i^{w_i}$, for random $w_1, \ldots, w_l \in \mathbb{Z}_q$. In the showing protocol, $\mathcal{P}$ in addition sends to $\mathcal{V}$ a set of t *correction factors*, $e_1, \ldots, e_t$, all in $\mathbb{Z}_q$. These serve to adjust a, in such a manner that the DL-representation known to $\mathcal{P}$ of the adjusted a is suitable to complete the demonstration in the same manner as in the original protocol. Specifically, the adjusted a is computed as

$$a / \prod_{i=1}^{t} g_{\pi(l-t+i)}^{e_i}.$$

By applying the protocol of Figure 3.1 and expanding the resulting terms, the following (generic) protocol steps result:

Step 1. $\mathcal{P}$ computes t correction factors, as follows:

$$e_i := w_{\pi(l-t+i)} + \sum_{j=1}^{l-t} \alpha_{ij} w_{\pi(j)} \bmod q \quad \forall i \in \{1, \ldots, t\}.$$

$\mathcal{P}$ then sends its one-time public key (h, a) and $(e_1, \ldots, e_t)$ to $\mathcal{V}$.

Step 2. $\mathcal{P}$ computes a set of responses, responsive to $\mathcal{V}$'s challenge $c \in \mathbb{Z}_s$, as follows:

$$r_i := c x_{\pi(i)} + w_{\pi(i)} \bmod q \quad \forall i \in \{1, \ldots, l-t\}.$$

$\mathcal{P}$ then sends $(r_1, \ldots, r_{l-t})$ to $\mathcal{V}$.

$\mathcal{V}$ computes

$$r_{l-t+i} := e_i + b_i c - \sum_{j=1}^{l-t} \alpha_{ij} r_j \bmod q \quad \forall i \in \{1, \ldots, t\},$$

and accepts if and only if

$$\prod_{i=1}^{l} g_{\pi(i)}^{r_i} h^{-c} = a.$$

Assuming s is large and c is generated in a substantially random manner and after the data in Step 1 has been sent, it is easy to prove that $\overline{\mathcal{V}}$ is convinced if and only if $\mathcal{P}$'s attributes satisfy the formula (3.3). Furthermore, all the considerations in Section 3.3.1 apply, with the obvious modifications. Notably, assuming that all the correction factors are hashed along as well when forming c, Propositions 3.3.6, 3.3.7, and 3.3.8 all hold.

Suppose now that $\mathcal{P}$ reuses the one-time public key (h, a), demonstrating a formula of the form (3.3) that need not be the same as in the first demonstration. Clearly, the proof of Case 1 in Proposition 5.4.1 applies as is. Consequently, if the two demonstrations are performed responsive to different challenges, then with overwhelming probability all the attributes of $\widehat{\mathcal{P}}$ can be efficiently computed from the two accepting views of $\mathcal{V}$.

$\mathcal{V}$ can limit the class of atomic formulae that $\mathcal{P}$ is able to demonstrate by imposing restrictions on the number, the position, or the form of the correction factors that $\mathcal{P}$ may use. Complete anticipation of the formula corresponds to the extreme case where $\mathcal{P}$ is not allowed to use any correction factors; this is the case of static one-show certificates. The CA can encode restrictions on the correction factors into one of the x_i's (e.g., as part of a CA policy attribute), which $\mathcal{P}$ must then disclose to $\mathcal{V}$ in the showing protocol as part of the formula it is demonstrating.

The technique in Section 5.3 can be applied as well. Only those correction factors that correspond to F^* should be hashed along when forming c. For dispute settlement, $\mathcal{V}$ should in addition hash along a commitment to both the hidden part of F and the remaining correction factors.

Arbitrary atomic formulae

The situation is slightly more complex in case $\mathcal{P}$ must be able to demonstrate any atomic formula. This time, we have $\mathcal{P}$ in Step 2 of the issuing protocol hash along an initial witness $a := h^{-w_0} \prod_{i=1}^{l} g_i^{w_i}$, for random $w_0, \ldots, w_l \in \mathbb{Z}_q$. To demonstrate a formula of the form (3.4), $\mathcal{P}$ in Step 1 of the showing protocol of Figure 3.5 must be allowed to reveal t correction factors, $(e_1, \ldots, e_t)$, to adjust a to

$$a / \prod_{i=1}^{t} g_{\pi(l-t+i)}^{e_i}.$$

To demonstrate a formula of the form (3.3), $\mathcal{P}$ in Step 1 of the showing protocol depicted in Figure 3.1 must this time be allowed to reveal $t+1$ correction factors, to adjust a to

$$a/h^{e_0}\prod_{i=1}^{t} g_{\pi(l-t+i)}^{e_i},$$

because the exponent $-w_0$ of h must be canceled as well. For both cases, it is easy to describe the resulting protocol. This time, the proof of Case 1 in Proposition 5.4.1 does not apply, because $\mathcal{V}$'s challenge in the verification relation for the demonstration of an atomic formula with a "NOT" connective does not appear as a power of h. Specifically, the verification relation for an atomic formula of the form (3.4) is

$$\prod_{i=1}^{l-t} g_{\pi(i)}^{r_i}\prod_{i=1}^{t} g_{\pi(l-t+i)}^{e_i+b_i r_0-f_i c-\sum_{j=1}^{l-t}\alpha_{ij}r_j} = h^{r_0}a$$

and that for an atomic formula of the form (3.3) with matrix coefficients α_{ij}^{*} is

$$\prod_{i=1}^{l-t^*} g_{\pi(i)}^{r_i^*}\prod_{i=1}^{t^*} g_{\pi(l-t^*+i)}^{e_i^*+b_i^* c^*-\sum_{j=1}^{l-t^*}\alpha_{ij}^* r_j^*} = h^{c^*+e_0^*}a.$$

If $r_0 = c^* + e_0^* \bmod q$, and cancellation takes place also for the exponents to all g_i's (otherwise a non-trivial DL-representation of 1 is obtained; cf. Case 2 of Proposition 5.4.1), then the DL-representation of h with respect to $(g_1,\ldots,g_l)$ cannot be computed even though c^* may differ from c. Indeed, if $\widehat{\mathcal{P}}$ is allowed to select $\mathcal{V}$'s challenge in an arbitrary manner, subject only to the condition that it has to be unique in each formula demonstration, it is not hard to construct a one-time public key (h,a), two atomic formulae, two different challenge messages, and two protocol transcripts for which total cancellation takes place.

We can get around this in a natural manner, by requiring $\mathcal{V}$'s challenge message to be a sufficiently strong hash of at least (h,a), a unique formula description, and the correction factors (in a unique order). Under this condition, which is needed anyway to prove the unforgeability and unmodifiability of signed proofs in the random oracle model (see Section 3.4.1), the following holds: if any two atomic formula demonstrations are performed responsive to different challenges, then with overwhelming probability all the attributes of $\widehat{\mathcal{P}}$ can be efficiently computed from the two accepting views of $\mathcal{V}$. Under the conditions of Proposition 3.3.6 and 3.3.7, respectively, this can be proved in the random oracle model for both non-interactively and interactively issued signed proofs.

Atomic formulae connected by "OR" connectives

Another treacherous aspect enters when $\mathcal{P}$ must be able to demonstrate any formula of the form (3.6), in such a manner that any two demonstrations involving the same

one-time public key disclose all the encoded attributes. The simplest approach is for $\mathcal{P}$ to simulate all but one of the subformulae demonstrations, in the manner described in Section 3.5.1, and to hash along in Step 2 of the issuing protocol an initial witness formed in the same formula-independent manner as in the case of atomic formulae. This fails, however, because $\mathcal{V}$ can infer from a single showing protocol execution which subformula demonstration is genuine. Using the same a for each subformula demonstration, and providing random correction factors for those subformulae demonstrations that are simulated, does not work either: if $j \geq 2$ then $\mathcal{V}$ can compute $\mathcal{P}$'s representation from a single execution of the showing protocol. Yet another flawed approach is for $\mathcal{P}$ to use a one-time public key of the form $(h, a_1, \ldots, a_{j^*})$, where $j^* \geq j$: this requires $\mathcal{P}$ to know an upper bound on j, the number of "OR" connectives in the formula to be demonstrated.

All these problems can be avoided by introducing an extra constraint in the showing protocol described in Section 3.5.1. Let

$$a := h^{-w_0} \prod_{i=1}^{l} g_i^{w_i}$$

be the *master initial witness* that $\mathcal{P}$ in Step 2 of the issuing protocol incorporates into its one-time public key, and let a_i denote the initial witness used in the demonstration of subformula F_i, for all $i \in \{1, \ldots, j\}$. For each of the j subformula demonstrations, we have $\mathcal{P}$ reveal up to l correction factors to adjust the initial witness a_i for that subformula demonstration, as described previously. $\mathcal{P}$ could do without the use of correction factors in the $j-1$ subformula demonstrations that are simulated, but they are needed to hide from $\mathcal{V}$ which subformula demonstration is genuine; the zero-knowledge simulator must generate the required number of correction factors at random (as many as would be needed in a genuine proof). The extra constraint is that the product of the j adjusted initial witnesses must equal the master initial witness, a. $\mathcal{P}$ can easily meet this constraint, since a_i in each of the $j-1$ simulated subformulae demonstrations is generated in such a manner that $\mathcal{P}$ knows a DL-representation with respect to $(g_1, \ldots, g_l, h)$; therefore, $\mathcal{P}$ can determine the remaining set of correction factors (for the subformula that requires a genuine proof) by comparing the representation it knows of a with that of the product of the $j-1$ adjusted initial witnesses. $\mathcal{V}$ must verify the extra constraint as part of its verification process.

To obtain a signed proof, $\mathcal{V}$'s challenge should be formed by hashing at least (h, a, F) and all correction factors (in a unique order). Under the conditions of Proposition 3.3.6 and 3.3.7, respectively, it can be proved in the random oracle model that both non-interactively and interactively issued signed proofs are unforgeable and unmodifiable. If the CA does not trust $\mathcal{V}$, the latter will need to relay the entire transcript of the showing protocol to the CA.

Why does this work? By aggregating the j verification relations (one for each subformula demonstration), $\mathcal{V}$ can obtain a compound verification relation of the

form

$$\prod_{i=1}^{l} g_i^{y_i} = h^{y_0} a.$$

Each y_i, for $i \in \{0, \ldots, l\}$, is an arithmetic expression that involves subformulae coefficients, responses (at most one from each subformula demonstration), and perhaps correction factors and $\mathcal{V}$'s challenge. Suppose now that $\mathcal{P}$ reuses its one-time public key (h, a). This time, $\mathcal{V}$ obtains a compound verification relation

$$\prod_{i=1}^{l} g_i^{y_i^*} = h^{y_0^*} a.$$

From the two compound verification relations, the DL-representation that $\mathcal{P}$ knows of h can be extracted, unless $\widehat{\mathcal{P}}$ has managed to perform its two protocol executions in such a manner that $y_i = y_i^* \bmod q$ for all $i \in \{0, \ldots, l\}$. If $\mathcal{V}$'s challenge is formed by hashing at least (h, a, F) and all correction factors, then it is infeasible to effect this cancellation, assuming only that $\mathcal{V}$'s challenge differs in the two showing protocol executions; this result can be proved in the random oracle model.

An anomaly occurs in case none of the atomic subformulae involve a "NOT" connective. In this case, $\mathcal{V}$ need not hash along a description of F and the correction factors. If $\sum_{i=1}^{j} c_i = c \bmod q$, it follows from the constraint on the adjusted initial witnesses that $\mathcal{V}$ can obtain a compound relation of the form

$$\prod_{i=1}^{l} g_i^{y_i} = h^{c} a.$$

With c being a sufficiently strong hash of at least (h, a), the unforgeability of this compressed signed proof of knowledge of a DL-representation of h follows as in the case of Propositions 3.3.6 and 3.3.7. By applying the technique described in Section 5.3, $\mathcal{V}$ can unconditionally hide which formula F has been demonstrated (from the space of all formulae that connect atomic subformula without "NOT" connectives by zero or more "OR" connectives), yet $\mathcal{V}$ itself becomes convinced of F during the showing protocol. In case (h, a) is reused in a showing protocol execution involving a different c, the CA can extract $\mathcal{P}$'s DL-representation of h. Sending only $(h, c, y_1, \ldots, y_l)$ to the CA is preferable also for reason of efficiency.

Arbitrary Boolean formulae

Finally, consider the case where $\mathcal{P}$ must be able to demonstrate any formula F of the form (3.7). The one-show certificate technique for formulae of the form (3.6) applies here as well. This time, the product of all the adjusted initial witnesses, one for each atomic subsubformula in F, must equal the master initial witness, a. Aggregation

of all the verification relations, one for each subformula demonstration, results in a compound verification relation of the form

$$\prod_{i=1}^{l} g_i^{y_i} = h^{y_0} a.$$

Assuming $\mathcal{V}$'s challenge is formed by hashing at least (h, a, F) and all correction factors, reuse of (h, a) with respect to a different c enables the computation of $\mathcal{P}$'s DL-representation of h. Again, this result can be proved in the random oracle model.

Note that only (y_0, y_i) needs to be stored to be able to compute x_i from $\mathcal{P}$'s DL-representation upon reuse of (h, a), regardless of the formula demonstrated. To be able to detect reuse, in addition the CA must store a hash of $(h, a, \mathsf{cert}(h, a))$. A hash function that is second-preimage resistant may be used (i.e., for a random input, it is infeasible to find another input that maps to the same output). In practice, simply storing the 10 most significant bytes of h, say, should be adequate, assuming that it is agreed on that no certificate applicant retrieves multiple certificates on the same public key. According to Section 2.2.2, 25-byte exponents should suffice for long-term security, which makes our technique for dynamic one-show certificates highly practical: per showing protocol transcript a mere 60 bytes need to be stored, regardless of the formula demonstrated.

The dynamic one-show certificate technique can be applied in a straightforward manner to the RSAREP-based issuing and showing protocols. In this case, the CA will be able to compute $\mathcal{P}$'s attributes $(x_1, \ldots, x_l)$ as well as x_{l+1} upon redemonstration of the same formula.

5.4.3 Increasing the threshold

There is nothing magical about a threshold value of 1. To ensure that $\mathcal{P}$'s representation can be computed if and only if $\widehat{\mathcal{P}}$ performs more than t demonstrations using the same certified public key, for a predetermined positive integer t, $\mathcal{P}$'s t-time public key should comprise t master initial witnesses. When performing the i-th execution of the showing protocol, using the same t-time public key, $\overline{\mathcal{P}}$ uses the i-th master initial witness. Alternatively, to obscure in the current showing protocol execution how many showing protocol executions have already been performed using the same public key, $\overline{\mathcal{P}}$ uses a master initial witness that it randomly chooses out of those not yet used. The pigeon-hole principle ensures that any $t + 1$ demonstrations lead to the reuse of a master initial witness. From the corresponding two views $\widehat{\mathcal{P}}$'s representation can be computed.

For large thresholds, it is more efficient to use Merkle's tree authentication technique [267]. Hereto $\mathcal{P}$ builds a binary tree with the master initial witnesses in the leaves. Each node value in the level just above the leaves is computed by compressing the entries in the leaves below it, using a collision-intractable hash function. All

the other node values are computed by compressing the child node values. The value of the root node together with h' serves as the t-time public key; it must be hashed along in Step 2 of the issuing protocol. In the showing protocol, $\mathcal{P}$ must reveal the particular master initial witness it intends to use in the formula demonstration, together with $\lceil {}^2\log t \rceil$ nodes to enable $\mathcal{V}$ to compute its way back to the root node and verify the certificate.

If $\mathcal{V}$ (or the CA) consents, t-show certificates can be turned on the spot into i-show certificates for any $i \in \{1, \ldots, t\}$, by fixing a subset of size i of master initial witnesses that may be used in the showing protocol. Also, a t-show certificate may be converted into an unlimited-show certificate, by allowing $\mathcal{P}$ in the showing protocol to use initial witnesses other than those comprised in its t-time public key. By way of example, consider the technique in Section 5.3 and suppose the CA is interested only in being able to trace certificate holders who show their certificates too many times. $\mathcal{P}$ supplies two proofs to $\mathcal{V}$: a signed proof demonstrating the formula TRUE and a proof demonstrating the formula $\mathcal{V}$ is interested in. The signed proof must use one of the (master) initial witnesses committed to in Step 2 of the issuing protocol, while the other proof may make use of freshly generated initial witnesses and need not be a signed proof.

5.5 Security improvements

In this section we introduce techniques to improve security, without resorting to smartcards or other tamper-resistant devices for certificate holders. Among others, we show how to discourage lending, copying, and discarding of certificates, and how to achieve non-repudiation for limited-show certificates. Most of these software-only techniques are based on the limited-show certification technique of Section 5.4. We also provide measures to protect against leakage and misuse of the CA's secret key.

5.5.1 Benefits of encoding identifiers

Certified key pairs that do not contain information that can be uniquely traced or linked to one person or to a select group are sometimes called *digital bearer certificates*. They are the digital equivalent of paper-based bearer certificates, such as currency and bearer bonds. Anyone may obtain them, show them, and pass them around. Bearer certificates are the opposite of *personal certificates*, which are generally issued only to parties that meet certain criteria. The simplest way for the CA to create digital bearer certificates is to issue blind signatures on public keys. An improved approach is to use restrictive blinding, to enable the CA to encode basic attributes (such as use limitations, expiry dates, and verifier policies) and to provide the ability of selective disclosure.

By definition, non-personal certificates must be limited-show certificates; otherwise everyone could simply use the same certificate. This requirement is at odds with

the ability of computers to instantaneously copy digital certificates in large numbers at virtually no cost, and to freely distribute them over electronic networks at the speed of light. Fraud with digital bearer certificates and other limited-show certificates implemented in software-only devices can be prevented only by clearing all showing protocol executions online with a central party. For each authorization request, this party (typically the CA) must consult an online database that keeps track of the number of times a presented certificate has already been shown. Online clearing suffers from the drawbacks in Section 1.1.3. The scalability problem is even worse, since the process of checking the occurrence of a certificate in the database is inherently sequential; queries may not be performed in parallel, and the online database may not be distributed. In many PKIs, though, it suffices for the CA to strongly discourage fraud and to be able to quickly contain it. Using our technique in Section 5.4, it is possible to realize these security goals and to do away with the need for online clearing. Hereto the CA must personalize all limited-show attribute certificates by encoding into each certified key pair an attribute identifying the certificate applicant. Consider the security benefits:

- A certificate holder who shows a limited-show certificate more times than allowed can be traced by computing the built-in identifier from the transcripts of the showing protocol executions. This enables the CA to stop the fraud by listing the abused certificate on a CRL and by blocking further retrieval of certificates from account. In addition, if the identifier can be linked to the identity of the certificate holder, it may be possible to sue for damages and to prevent the perpetrator from continuing to participate in the system. Of course, traceability of perpetrators often also serves as a powerful deterrent.

- This technique can be made to work even for anonymous accounts. Traced perpetrators can be prevented from opening new anonymous accounts, from which new limited-show certificates can be retrieved, by limiting the number of anonymous accounts to one per certificate applicant. Hereto a special entity should issue digital pseudonyms with an embedded identifier, that can each be used to open one anonymous account. If personal account pseudonyms are issued in such a manner that the recertification technique described in Section 5.2.1 applies, the limited-show property remains intact.

- The number of anonymous accounts need not be limited to one per certificate applicant. In case a traced perpetrator refuses to reveal any other anonymous accounts he or she may have opened, the CA can require all system participants (e.g., the next time they retrieve certificates) to demonstrate that the identifier built into their account pseudonym is not that of the perpetrator. Hereto the showing protocol technique in Section 3.4 can be used. This is not an extra burden, since applicants must use their account pseudonym anyway to authenticate account access requests.

- In the same manner, a perpetrator can be stopped from using any other digital certificates that may have been issued to him or her. For example, an administrator of a pseudonymous chat box can require each participant to digitally sign his or her next message in such a manner that the signed proof demonstrates that the pseudonym is not that of an identified misbehaving participant. (The alternative of issuing only one-show certificates is not always desirable.)

- More generally, to verify that t public keys, $h_1, \ldots, h_t$, are not all owned by the same certificate applicant, the verifier generates t numbers, $\alpha_1, \ldots, \alpha_t$ at random from a set V that is a subset of $\mathbb{Z}_q$, subject to the condition $\sum_{i=1}^{t} \alpha_i = 0 \bmod q$. Let attribute x_1 be unique to the certificate applicant. A cheating prover can prove knowledge of a DL-representation of g_1 with respect to a tuple that contains $\prod_{i=1}^{t} h_i^{\alpha_i}$ with success probability at most $1/|V|$, but an honest group of provers can always (jointly) convince the verifier. (The same technique applies to the RSAREP-based constructions.)

Stronger fraud discouragement without resorting to online clearing requires the use of smartcards or other tamper-resistant devices; for details, see the next chapter.

In sum, even though in regular communications and transactions there may be no need for certificate holders to show identifiers, it is beneficial the CA for to encode them anyway into their certified key pairs.

There is no need for the CA to encode globally unique identifiers; they should merely be unique within the domain of the PKI. Identifiers need not even be determined using an out-of-band method: an identifier may be a random number or a pseudonym known to the CA. In cases where a true name is to be used, the CA may require registrants to present an X.509 or other identity certificate; the CA can copy and paste the subject names from these certificates into the certificates it issues.

The current standards for identity certificates can easily be encapsulated. To encapsulate an X.509v3 certificate, for instance, the CA can take x_1 to be (a collision-intractable hash of) the concatenation of all the fields except the subject's X.500 name (i.e., the format version, serial number, the CA's certification algorithm identifier, the CA's X.500 name, the validity period, and the certificate holder's public key and the algorithm with which it is to be used), and x_2 the certificate holder's X.500 name. The remaining x_i's can specify additional attributes, such as X.509v3 extensions. In the certificate showing protocol, the certificate holder must disclose (the preimage of) x_1, and may be required to demonstrate properties about the other attributes. To ensure that the data encoded into x_1 does not serve as a unique identifier, the entropy of at least the validity period and any extension fields must be restricted. Furthermore, the serial number should be set to a hash of the (certified) public key, since this is what is posted on a CRL; alternatively, it is set to zero, since verifiers can compute the hash themselves if needed.[3]

[3] ANSI draft standard X9.68 for mobile devices proposes to shorten X.509 certificates by assigning to each certificate an implicit serial number formed by hashing the certificate; this matches our approach.

A non-technical objection to the inclusion of identifiers into certified key pairs is that organizations and other verifiers may incite certificate holders to disclose their built-in identifiers, for example by giving discounts to customers who disclose their identifiers or by refusing to serve anonymous certificate holders. Privacy legislation should be adopted that prohibits such discrimination in PKIs where there is no strict need for identification; PKI legislation is needed anyway, so this is not an unreasonable course of action. In Section 6.5.5 we will show how to make it technically infeasible for certificate holders to disclose their identifier attributes.

5.5.2 How to discourage lending

It should not be possible for certificate holder to lend (i.e., distribute copies of) their personal certificates, such as driver's licenses and diplomas, to others. To discourage lending of certified key pairs for personal certificates that are used in face-to-face situations, the CA could encode biometric identifiers into certified key pairs, such as digitized photographs, fingerprints, or retina scans. Privacy legislation should specify when identity disclosure may be required; preferably, verifiers may require disclosure of visual identifiers only in rare cases, sampled at random. Fairness can be enforced by using a cryptographic (biased) coin flipping protocol to determine whether a biometric identifier must be disclosed.

For improved security and privacy, the CA could encode into each certified key pair a plurality of personal characteristics, such as eye color, gender, height, and skin type. Each of these by itself is not a person identifier, but sufficiently many taken as a group are. In the showing protocol the verifier could then request the certificate holder to disclose a random subset of the built-in personal characteristics. Again, fairness can be achieved by means of (biased) coin flipping, to determine the size and the elements of the subset that must be disclosed.

Another measure for the CA to discourage lending of certified key pairs is to encode into each certified key pair an attribute that the certificate holder wants to remain confidential. Examples are the identity of the certificate holder (his or her reputation may be damaged when the borrower discloses the attribute), the secret key of a key pair used to sign messages (such as a PGP secret key), or a redeemable electronic token of significant value. (It should not be possible, though, for a certificate holder to revoke a key or a token within the period that lending is to be discouraged.) According to Proposition 3.6.1(b), the demonstration of any Boolean formula requires knowledge of the entire secret key. Consequently, the lender must reveal to the borrower all the attributes, including the confidential attribute, even though the borrower may be interested only in some of the other attributes. This measure may be combined with the previous measure, but applies also to personal certificates that are not restricted to face-to-face situations. By way of example, consider digital gender certificates needed to gain access to certain online discussion groups, or "over 18" certificates to gain access to adult-oriented Web sites. By encoding along the credit

card data of the legitimate receiver into his or her certificates, the issuer can ensure that the receiver cannot lend (or give out copies of) the certificate without disclosing the credit card data; at the same time, privacy is not compromised because the receiver itself can hide the data when showing the certificate.[4] The CA need not know the confidential attributes it encodes: it can apply our updating or (re)certification technique described in Section 5.2.1 to a public key presented by the certificate applicant (corresponding to the secret the applicant wants to remain confidential).

Yet another measure to discourage lending is for the CA to issue all personal certificates in the form of limited-show certificates.[5] This subjects a lender to the risk that the borrower uses his or her certified key pair more times than allowed, which would result in the lender being traced (and held responsible). It also reduces the number of times the certificate holder can continue to use the certified key pair him or herself. This measure may be applied in conjunction with the measure in the previous paragraph, applies also to certificates that by their nature may be shown an unlimited number of times (before expiry), and works particularly well in conjunction with short validity periods (which are more natural for limited-show certificates; see Section 1.1.5).

Stronger protection against lending requires smartcards; see Section 6.5.3.

5.5.3 Non-repudiation

To prevent the CA from framing certificate holders by falsely claiming abuse of limited-show certificates, consider a variation of the DLREP-based scheme described in Section 4.5.2. Before issuing certified key pairs to $\mathcal{V}_0$, $\mathcal{P}_0$ requires $\mathcal{V}_0$ to provide $h^* := g_1^I$, for a random $I \in \mathbb{Z}_q$, and to prove knowledge of $\log_{g_1} h^*$ without revealing I; $\mathcal{V}_0$ can apply the Schnorr proof of knowledge or its zero-knowledge variation. In addition, $\mathcal{V}_0$ may be required to sign a statement to agree to the consequences of the use of h^*; this can be combined with the proof of knowledge by giving a signed proof of $\log_{g_1} h^*$. If the proof is convincing (it need only be performed once), $\mathcal{P}_0$ multiplies the desired g_i-powers into h^*, and uses the result as the number h in the issuing protocol to issue to $\mathcal{V}_0$ one or more limited-show certificates. If a certificate is shown more times than allowed, I can be computed. Assuming that (q, g_1, I) has been generated by an invulnerable instance generator, I serves as compact undeniable evidence of fraud.

The evidence can be made unconditionally convincing. Hereto the CA should

[4] This valuable property cannot be ensured by simply certifying a one-way hash structure of the attributes (such as the concatenation of one-way images of all the attribute values). The resulting certificates can be shown without needing to know all the attribute values themselves.

[5] To prevent linkability of showing protocol executions, certificate holders must use their certificates only a limited number of times anyway, and the CA might as well exploit this by issuing limited-show certificates with built-in identifiers. The burden of issuing 100 copies (and new ones when needed) is hardly greater than the burden of issuing a single copy, especially if the CA uses a DLREP-based issuing protocol that admits preprocessing of all exponentiations.

require $\mathcal{V}_0$ to register using a number h^* of the form $g_1^I g_2^\beta$, for an arbitrary I and a random $\beta \in \mathbb{Z}_q$. $\mathcal{V}_0$ must prove knowledge of a DL-representation of h^* with respect to (g_1, g_2); again, this need only be done once. In case of fraud, the CA can compute I, β, which serve as unconditionally undeniable evidence; assuming that $g_1, g_2 \neq 1$, the CA cannot frame $\mathcal{V}_0$, no matter how it generates the system parameters and its public key.

This non-repudiation technique works also in conjunction with anonymous accounts: h^* serves as the account pseudonym, which the certificate applicant must acquire from an account pseudonym issuer (possibly by showing an X.509 certificate or another type of identity certificate) and uses to authenticate account requests.

To apply the non-repudiation technique to an RSAREP-based certificate scheme, $\mathcal{V}_0$ should preferably register using a number h^* of the form $g_1^I \beta^v$.

5.5.4 How to discourage discarding

To discourage certificate holders from discarding certified key pairs that encode unfavorable attributes, the CA can encode favorable attributes into the same certified key pairs. For instance, information on late payments could be encoded into a membership certificate for a health club or the like, and marks for drunk driving into a driver's license certificate. Note that the updating technique described in Section 5.2.1 can be put to use here.

This measure does not work when the attributes encoded into the certified key pairs of a certificate applicant change over time and become less favorable to the applicant. To limit the ability of certificate holders to reuse old certified key pairs that encode attributes that are more favorable, the CA should encode short validity periods. Alternatively, or preferably in addition, the CA could issue only limited-show certificates.

The strongest possible protection in software-only implementations is to issue only one-show certificates, and to ensure that certificate holders cannot show more than one certificate at any time. This can be achieved by having the CA recertify (and update) one-show certificates only when they are shown in executions of the showing protocol. This requires all verifiers to have an online connection with the CA, and applies only to certificates for which the decision as to how to update attributes relies solely on the data disclosed in the showing protocol.

Stronger protection requires the use of smartcards or other tamper-resistant devices. These can be programmed to show certificates in the order in which they were retrieved, and can enter a suspension mode in case of tampering.

5.5.5 Guarding the secret key of the Certificate Authority

In this section we describe measures to guard the CA's secret key. Whether any or all of these measures are actually necessary depends on the PKI at hand.

Elementary measures

To enforce personnel to behave according to guidelines, organizational controls are needed. Examples are security clearance and background checks for new employees, auditing, strict manufacturing and software development procedures, separation of staff responsibilities, frequent change of assignments, and controlled initialization, personalization, and distribution of devices. A discussion of these and other controls is outside the scope of this book.

To raise the barrier to gaining access to the CA's secret key, it is best generated and stored within the confines of a tamper-resistant device and never revealed to the outside world (including the legitimate operators). This prevents CA personnel from being extorted or tempted to misuse the key themselves.[6] The device could even be programmed such as to restrain the rate of certificate issuing to a single person or location in a given time frame.

For CA devices, it is entirely cost-effective and feasible to achieve very strong tamper-resistance, because in contrast to smartcards there are no tight size and cost constraints. CA devices can be made tamper-evident, can be riveted at a secured location, and can maintain unmodifiable audit logs revealing the exact date and time of each task performed. Preferably, they meet at least security level 3 of FIPS 140-2 [275], a U.S. federal standard on the security of cryptographic modules and a de facto industry standard for commercial devices. Adherence to the Common Criteria for Information Technology Security Evaluations (a voluntary international process for developing test requirements, intended to replace ITSEC and other like standards) is recommendable as well.

In addition, it may be desirable to distribute the CA's secret key across multiple tamper-resistant devices, and to use secret-sharing techniques to enable designated subsets to perform the required cryptographic actions without leaking their respective shares.[7] Techniques for shared signature generation are well-known in the cryptographic literature, and can readily be adapted to the issuing protocols described in Chapter 4; see also Section 4.2.3. The secret key of each tamper-resistant device is best generated in a mutually random and verifiable manner between the device and its legitimate operators; this can be accomplished through standard cryptographic techniques.

To cope with cryptanalytic attacks on the CA's secret key, the system design should be based on strong underlying cryptographic primitives that have been publicly scrutinized by experts for decades (such as factoring or computing discrete log-

[6] Jakobsson and Yung [220], in the context of electronic cash, proposed to make each certificate fully traceable and clear each certificate showing online with a central party whenever the CA's secret key has leaked or coins have been extorted. This approach leads to severe system disruption, destroys privacy, does nothing to make extortion less attractive or to prevent it from reoccurring, and does not protect against insider abuse.

[7] CertCo and Spyrus, contracted in May 1997 by MasterCard and Visa to manufacture the CA system for SET [257], were the first to adopt this technique in practice.

arithms). The techniques in this book are believed to meet this criterion. Rather than using key sizes that are just a little beyond reach of currently known algorithms, key sizes should be as large as possible without causing a serious performance devaluation. In this respect, an elliptic curve implementation with keys that are not too short offers a distinct advantage. The CA should update to larger key sizes on a regular basis, and be prepared to do so in short term.

Stronger measures

When much is at stake, the CA should (in addition to the previous measures) enforce deposit of all showing protocol transcripts (not necessarily online), and continuously compare the number of certificates issued against that of certificates deposited. It appears that none of today's commercial certificate systems offer this provision, but this will likely change once the parallels between digital certificate systems and electronic cash systems are widely acknowledged.

This measure is not very effective in a large-scale PKI, because suspicion of forgery does not arise until the number of forged certificates approaches the number of issued certificates that have not yet been shown. The situation can be improved by taking into account the issued and disclosed attributes of each certificate. That is, for each attribute property disclosed the CA should maintain a separate category and compare a running count of each category with the number and combination of attributes issued. In an electronic coin system, for instance, the bank should monitor per coin denomination and per coin version.

In high-risk PKIs, the CA could encode random numbers from small sets into the attribute certificates it issues. Certificate holders are in full control over the secrecy of these attributes, but in case of strong suspicion of forgery, the CA can turn up the heat by requiring verifiers to demand certificate holders to reveal (part of) the encoded random numbers; the CA should then announce that it will no longer honor deposits from verifiers that do not abide by these rules. (Legislation should specify the conditions under which such a revision of deposit rules is legitimate.) In this way, the CA can dynamically shift between a positive list and a negative list approach; in the worst case, certificate holders are required to disclose a built-in identifier each time. (In this fashion, our techniques cover the entire range between privacy-protecting certificates and identity certificates.)

In a similar manner, the CA can dynamically set conditions for which types of certificates may be accepted off-line and which require its on-line approval.

Even stronger measures

In the presence of an attacker with "infinite" computing power, these measures are insufficient. A quantum computer, for instance, would be able to compute discrete logarithms and factorize in about the same time as it takes to exponentiate large

numbers; see Shor [350]. Since forgery by an attacker with unlimited computing power can never be prevented, the primary defense line is to contain the damages.

The ultimate containment measure, if all else fails, is system suspension. The CA should make sure either that its certificate issuance is no longer accepted or that it can recognize and reject forged certificates during online certificate validation. The CA can still accept online executions of the certificate showing protocol by requiring certificate holders to disclose the identifiers that have been encoded into their certificates.

In addition, depending on the nature of the PKI, it may be necessary that all authentic outstanding digital certificates can be securely redeemed, to prevent a meltdown scenario. This fall-back mechanism must be secure even in the presence of an attacker with unlimited computing power. Preferably the fall-back mechanism can be implemented by means of a protocol that does not require certificate holders to show up in person at the CA. To this end, the software-only return protocol that will be described in Section 6.5.2 can be used.

5.6 Bibliographic notes

The techniques in Section 5.1.1 originate from Brands [54]. For an application, see Brands [46, 48]; here, the issuing protocol serves to issue electronic coins that encode an attribute representing the coin denomination (and possibly also an expiry date and other data that must always be disclosed to payees) and an identifier attribute that can be computed if and only if the coin is double-spent (using the static one-show blinding technique).

The discussion of the delegation strategy in Section 5.1.2 is based on Brands [53]. The parallel delegation strategy described for the case of the immunized DLREP-based scheme I in Section 4.4.2 was discovered by Schoenmakers [341, footnote 1]; the proposed measures to make this strategy ineffective are new, though. (In light of this Schoenmakers' remark on the insecurity of the second method of Brands [47] for exchange rates must be nuanced. In fact, in the issuing and showing protocol described by [47] the problem does not arise in the first place because parallel protocol executions by different account holders are excluded.)

The anonymous recertification and updating techniques in Section 5.2.1 were introduced by Brands [54], together with an application of the updating technique to anonymous accounts in the off-line electronic cash system of Brands [46, 48]. Subsequently, Brickell, Gemmell, and Kravitz [62, 240] applied the anonymous updating technique for the purpose of online change-making in the off-line electronic cash system of Brands [48].

The techniques described in Section 5.2.2 are all based on Brands [54]. The application to credential systems predates a credential system proposed by Chen [113], which has many drawbacks over our proposal: Chen's system offers only compu-

tational unlinkability, does not handle attribute certificates and selective disclosure, does not offer traceability of fraud with limited-show credentials, does not provide for smartcard extensions (whereas we can apply the techniques that will be developed in the next chapter), and uses an issuing protocol that is inferior to that described in Section 4.5.2.

The techniques in Section 5.3 appear here for the first time.

The static and dynamic limited-show certification techniques in Section 5.4 are based on Brands [45]. Some details have been filled in, and the formal security statements and their proofs are new.

Most of the security improvements in Section 5.5 originate from Brands [54]. The idea of discouraging lending of certified key pairs by encoding a special secret that the certificate holder wants to remain confidential (see Section 5.5.2) was inspired by the following remark by Garfinkel [180] about identity certificates: "While a few dozen guys might get together and share a single username and password for a cybersex site, there is no way that any of these clowns will let the others share his digital ID's secret key, which will also unlock his bank account and credit cards." The same observation has been used by Dwork, Lotspiech, and Naor [142], by Goldreich, Pfitzmann, and Rivest [192], by Sander and Ta-Shma [333] (to discourage lending in the electronic cash system of Brands [46, 48]), and by Lysyanskaya, Rivest, Sahai, and Wolf [253][8] in the context of digital pseudonyms. However, none of these publications consider the remote lending problem that we will address in Section 6.5.3.

The non-repudiation technique in Section 5.5.3 originates from Brands [46, 48], where it was applied in the context of off-line electronic cash.

[8] The techniques in this book offer a much better solution; the advantages are the same as the advantages over the credential system of Chen [113].