A Technical Overview of Digital Credentials

Dr. Stefan Brands
Credentica
brands@credentica.com

February 20, 2002

Abstract

Applications that involve the electronic transfer of credentials, value tokens, profiles, and other sensitive information are quickly gaining momentum. Traditional attempts to introduce electronic authentication, such as PKI and biometric verification, expose organizations to potentially unlimited liability, lead to consumer fear, and stifle the adoption of new systems. To overcome these barriers, innovative solutions are needed that address the entire spectrum of security and privacy interests for all parties involved.

This paper is a technical overview of Digital Credentials. Digital Credentials are the digital equivalent of paper documents, plastic tokens, and other tangible objects issued by trusted parties. At the same time, they are much more powerful than their physical counterparts. For example, individuals can selectively disclose properties of the data fields in their Digital Credentials while hiding any other information. Digital Credentials also provide much greater security. As a result, they can be used to securely implement objects that traditionally are made identifiable in order to deal with certain kinds of fraud. Examples are diplomas, work permits, access cards, and drivers' licenses.

1 Introduction

Around the world, influential organizations are rushing to provide individuals with digital identity certificates that will be the sole means of participating in their systems. Among the front-runners are telecommunication organizations, health care providers, mobile service providers, financial institutions, and state and federal governments. A digital identity certificate is just a sequence of zeros and ones, and so it can be transferred electronically and can be verified with 100 percent accuracy by computers. Each certificate contains a unique secret key to enable its holder to prove ownership of the certificate, and is digitally signed by its issuer to guarantee unforgeability.

Digital identity certificates are not appropriate in many situations, because their goal is to enable identification of their holder; they are the digital equivalent of passports. Most people would strongly object to having to identify themselves in every interaction they undertake. Digital identity certificates are in fact much more privacy-invasive than passports and other traditional proofs of identity, because they enable all of an individual's actions to be linked and traced automatically and instantaneously, by a variety of parties. Moreover, the large-scale use of personal identifiers greatly encourages identity theft.

Nevertheless, often at least one of the parties in an interaction needs to know whether the other party is authorized to perform a certain action. Typically, authorization is granted on the basis of a person's accomplishments, privileges, personal characteristics, and so on, rather than on the basis of his or her identity. This is why people use medical prescriptions, stamps, subway tokens, cinema tickets, drivers' licenses, diplomas, coins and banknotes, birth certificates, access cards, and so on.

Digital Credentials

Digital Credentials were proposed in 1993 by Brands [13] (and improved in subsequent work) to provide a secure way to implement all these and other objects in a fully digital manner, without eroding the privacy of their holders. For an up-to-date treatise, see Brands [15]. Digital Credentials enable their holders to determine for themselves when, how, and to what extent information about them is revealed to others, and to what extent others can link or trace this information. Individuals need not trust third parties to protect their privacy: even if all the parties that rely on Digital Credentials and all the Digital Credential issuers actively conspire and have unlimited computing resources, they cannot learn more than what can be inferred from the assertions that individuals willingly and knowingly disclose. This seemingly very strong privacy guarantee is not a new concept: today, when you spend a coin, cast a vote, or use a cinema ticket, you are anonymous.

Conceptually, Digital Credentials borrow heavily from seminal work on electronic privacy by Chaum [21, 22, 23, 27, 29, 31] in the period 1982–1992. In particular, Chaum [19, 20, 24, 34] advocated the use of credentials, which he defined [24] as "statements concerning an individual that are issued by organizations, and are in general shown to other organizations." Enabling individuals to build pseudonymous relations with organizations is just one specific application of Digital Credentials, though, and arguably not among the most interesting ones.

Although it is helpful to think of Digital Credentials as the digital equivalent of stamps, gift certificates, and other tangible non-identity tokens, they are actually much more powerful. Amongst others, the holder of a Digital Credential can selectively disclose a property of the attributes that the issuer encoded into the Digital Credential, without revealing any other information. (As defined by X.501 [18], an attribute is "information of any type.") This goes beyond what can be achieved in the physical world with a paper-based certificate and a marker. For example, the holder of a citizenship Digital Credential can demonstrate that her citizenship is not American or that she is European, without revealing her citizenship. In fact, Digital Credentials encompass identity certificates as a degenerate case, since their issuers can always encode identifier attributes that Digital Credential holders can disclose at their discretion when showing their Digital Credentials.

Security benefits

Digital Credentials offer much greater security than physical non-identity objects. For example, a Digital Credential can contain a built-in identifier that can be uncovered by a central party only if the Digital Credential is shown more than a predetermined number of times. Furthermore, by encoding confidential data of the applicant into a Digital Credential the issuer can discourage lending of that Digital Credential, even though the applicant can always hide the confidential data when using the Digital Credential. We will see later on in this paper how to realize these two counterintuitive properties. Additional security features become available when Digital Credentials are embedded in smartcards or other tamper-resistant devices.

Digital Credentials also offer security benefits over digital identity certificates. By placing identity at the heart of transactions and communications, identity certificates greatly increase the risk of identity fraud. Also, nothing discourages certificate holders from giving away copies of their identity certificates, unless the certificates are accepted in large-scale applications where their owners run a serious personal risk. Furthermore, identity certificates are often used as authenticated pointers to central database entries, but there is no guarantee that these entries are relevant and accurate; Digital Credentials, in contrast, can contain all the data that verifiers are interested in. Digital Credentials overcome these shortcomings.

As a result of their much greater security, Digital Credentials can safely be used in all kinds of applications where the use of physical non-identity objects is considered insecure. That is, they can be used to implement not only gift certificates, railway tickets, and so on, but also diplomas, work permits, birth certificates, and other objects that traditionally identify their holder in order to deter certain kinds of fraud.

Note that Digital Credentials encompass identity certificates as a special case: an identifier is just one of infinitely many attributes that can be encoded into a Digital Credential, and the Digital Credential holder can disclose it whenever desired.

Contents of this paper

The goal of this paper is to provide insight into the cryptographic techniques of Brands [15] for designing Digital Credentials, not to give a formal treatise. Consequently, most of the descriptions center around example cases, and mathematical proofs are replaced by proof outlines. For the same reason, optimizations and variations are not taken into consideration.

Section 2 discusses the basics of Digital Credentials. Section 3 describes how Digital Credential holders can selectively disclose attribute properties while hiding any other information. Section 4 shows how the selective disclosure techniques can be exploited to deter lending and discarding of Digital Credentials. Section 5 describes two protocols for issuing Digital Credentials, and discusses how to combine them with the showing protocol. Section 6 discusses the design of Digital Credentials that can be traced by a central party if and only they are shown more times than allowed. Section 7 explains how Digital Credential verifiers can protect their own privacy. Section 8 discusses the security benefits of smartcards, and shows how to incorporate them without compromising privacy. Section 9 concludes the paper.

2 Basics

Digital Credentials are *issued* to applicants by trusted parties, referred to as Credential Authorities (*CA*s). They are *shown* by Digital Credential holders to verifiers. In a specific application, context-specific names may be used. In an electronic voting system, for example, issuing corresponds to obtaining a ballot and showing is the equivalent of casting a vote. In an electronic cash system, a Digital Credential is an electronic coin or cheque, issuing corresponds to the withdrawal of coins or cheques, and showing corresponds to payment.

In line with cryptographic tradition, we will use the fictitious characters Alice and Bob to refer to a Digital Credential holder and a Digital Credential verifier, respectively.

There can be many *CA*s that each issue their own Digital Credentials, in the same way as voting ballots, drivers' licenses, and credit reports are usually issued by different organizations.

Basic Digital Credential model

The *CA* has its own key pair for digitally signing messages. When issuing a Digital Credential to Alice, the *CA* through its digital signature binds one or more attributes to a Digital Credential public key, the secret key of which only Alice should know. The whole package that Alice receives is called a Digital Credential. A "demographic" Digital Credential, for example, might specify Alice's age, number of kids, marital status, and citizenship, all tied to a single public key by one digital signature of the *CA*.

The primary reason for the presence of a public key within the Digital Credential is to prevent a *replay attack*. This is an attack whereby Bob (or a wiretapper) reuses the Digital Credential data that Alice showed, in an attempt to pass as the owner of the Digital Credential. Replay attacks are prevented as follows. When Alice shows her Digital Credential to Bob, she not only sends her Digital Credential public key and the *CA*'s signature, but she also digitally signs a *nonce* using her secret key. A nonce is a random number, the concatenation of Bob's name and a counter, or any other fresh data provided by Bob. Bob cannot replay the data, since in each showing protocol execution a new nonce must be signed, which requires knowledge of Alice's secret key.

In the showing protocol, Alice also selectively discloses a property of the attributes in her Digital Credential, while hiding any other information about them. For example, when showing her demographic Digital Credential, Alice may decide to disclose the fact that she has kids, without revealing how many and without revealing any information about the other attributes in her Digital Credential. To convince Bob that the claimed property is true, Alice's signature on Bob's nonce doubles up as a proof of correctness. As we will see in the course of this paper, we can in fact exploit Alice's digital signature to achieve many security properties at the same time. This design principle contributes to making Digital Credentials as compact and efficient as possible.

To ensure the *untraceability* of a Digital Credential, Alice must communicate with Bob over an anonymous communication channel. In the issuing protocol, however, reliance on an anonymous channel is inappropriate. Indeed, in many applications Alice will need to identify herself to the *CA* to enable the *CA* to retrieve or to validate the attributes that Alice wants to have certified. Identification to the *CA* may also be required by law, in the same manner as anonymous bank accounts are not accepted in many countries. Even if Alice may obtain Digital Credentials from the *CA* without identifying herself (as is traditionally the case for stamps, cinema tickets, and so on), it may be impractical to remain anonymous.[1] How, then, is the untraceability of Digital Credentials guaranteed? Since Alice always reveals the Digital Credential public key and the *CA*'s signature when showing a Digital Credential, the *CA* must never see these two data elements when

[1] For example, in many situations the *CA* will want to charge a fee for each Digital Credential. Whereas Alice can buy things anonymously in the physical world by using cash, no anonymous payment method is widely available today. (Interestingly, as we will see later on, Digital Credentials can be used straightforwardly to build an anonymous electronic cash system.)

it issues the Digital Credential to Alice. After all, they are unique bit strings, and our privacy threat model allows Bob and the *CA* to collude in trying to trace Alice. At the same time, Alice must not be able to blind the attributes the *CA* wants to encode into her Digital Credential, because that would enable her to modify them.

To achieve these two properties, Section 5 will use a technique called *restrictive blinding*. This technique is not a special case of Chaum's blind signature paradigm [22, 23, 24, 25, 26, 27, 30, 31], which does nothing to limit Alice in the kind of blinding operations she can perform. "Ordinary" blind signatures have many drawbacks that make them unsuitable for most real-life applications. For example, it is not possible for the *CA* to encode attributes that can be disclosed in certain circumstances but may remain hidden in others; blind signatures cannot discourage the unauthorized lending and discarding of certified data; and, it is not know how to securely and efficiently incorporate blind signatures into smartcards without destroying privacy. For an extensive discussion of the drawbacks of ordinary blind signatures, see Brands [15, Section 1.2.2]. The notion of restrictive blinding also differs from Chaum's notion of one-show blinding [21, 29]. The latter concerns a property of an issuing protocol in combination with a showing protocol, while restrictive blinding is a property of the issuing protocol only. In particular, restrictive blinding has nothing to do with restricting the number of times a certificate may be shown. One special use of restrictive blinding is to construct practical one-show blind signature schemes (see Section 6), but its general applicability is much broader.

Note that if Alice shows the same Digital Credential more than once, then Bob learns that these showing protocol executions were performed by the same individual; actions involving the same Digital Credential are *linkable*. Often, linkability is undesirable. If Alice is identified in one protocol execution then all her past and future actions involving the same Digital Credential become traceable as well. Identification can take place indirectly: if Alice discloses a different property each time she shows the same Digital Credential, then each new disclosure reduces the set of Digital Credential applicants that could have been issued that Digital Credential. For this reason, Alice should not reuse a Digital Credential too often; from a privacy perspective, single use is optimal. With today's computers and electronic networks, retrieving several "versions" of the same Digital Credential (i.e., Digital Credentials with the same attributes encoded within them, but each with their own Digital Credential public key) is hardly less efficient than getting a single one, and Alice can request new batches later on as needed (possibly by requesting anonymously refreshed versions of a Digital Credential, as will be explained in Section 5).

Figure 1 provides a schematic overview of this basic model, in which Alice shows one of several versions of a demographic Digital Credential to a medical organization that wants to know her marital status and citizenship.

Optionally, Bob deposits the showing protocol *transcript* (i.e., the data that Alice sends to Bob) to the *CA* (or another central party). This enables the *CA* to detect fraud with Digital Credentials that may be used only a limited number of times (see Section 6), to keep track of the number of customers that Bob serves, to gather statistics on disclosed attribute properties, or to perform other administrative tasks. Deposit can take place either during the showing protocol execution or at a suitable moment afterward; the former is an *online* deposit, the latter is an *off-line* deposit.

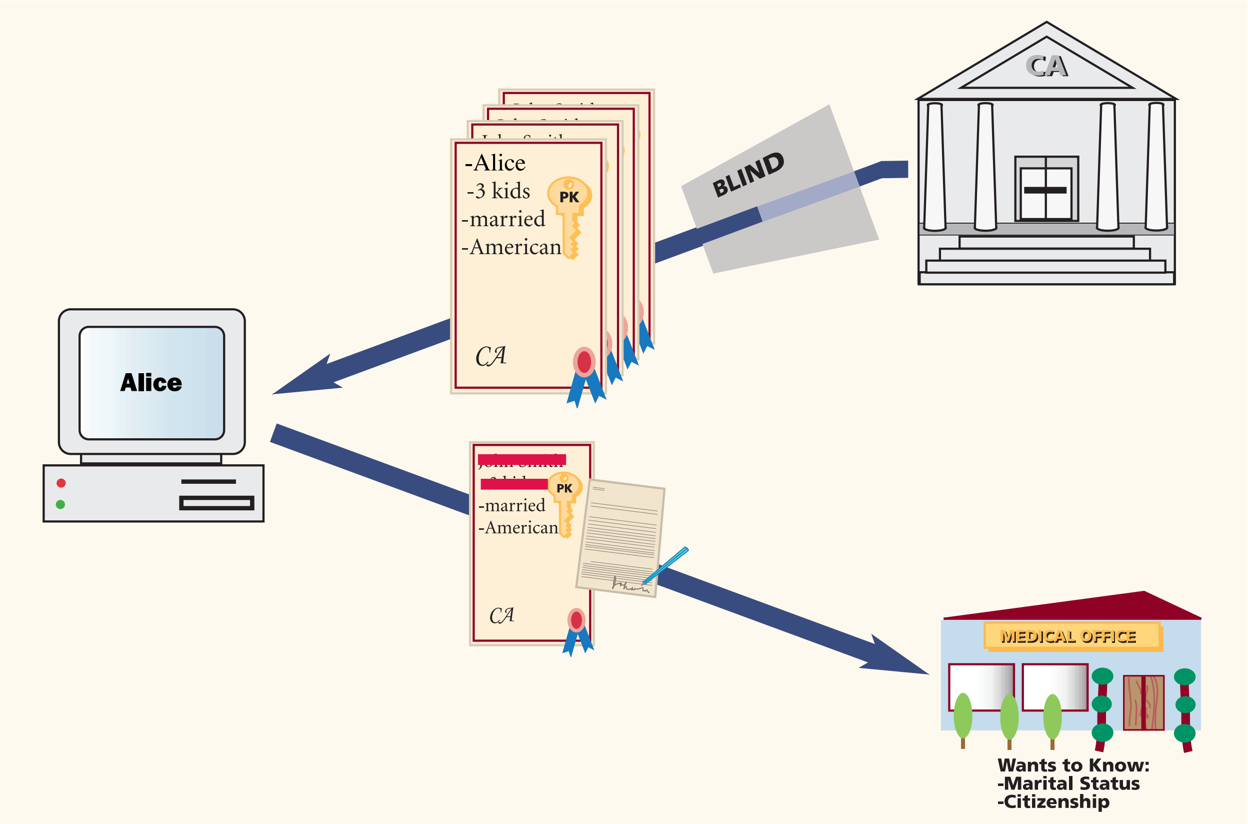

Figure 1: Basic Digital Credential model.

Encoding attributes into a Digital Credential

The issuing protocol that will be described in Section 5 enables the *CA* to encode an arbitrary number of attributes into a single Digital Credential. Here is how $l \geq 1$ attributes, $(x_1, \ldots, x_l)$, end up encoded in a Digital Credential:

- The tuple

  $$(x_1, \ldots, x_l, \alpha)$$

  is Alice's secret key for the Digital Credential. Alice generates α at random from $\mathbb{Z}_q$ during the issuing protocol. Even though Alice may disclose some or even all of the attributes to Bob in the showing protocol, she keeps α secret at all times; this ensure that only she knows the entire secret key.

- The Digital Credential public key is equal to the product

  $$g_1^{x_1} \cdots g_l^{x_l} h_0^{\alpha}.$$

  The elements $g_1, \ldots, g_l, h_0$ have been generated by the *CA* at random from a group G_q of prime order q; they are part of its public key. For a technical reason that will become apparent shortly, h_0 should not be equal to 1, so that it is a generator of G_q.

(The actual form will be slightly more complex, for reasons that will be addressed in Section 5.) The Digital Credential public key reveals no information about $x_1, \ldots, x_l$, because many secret keys correspond to the same public key. Specifically, for any public key $h \in G_q$ and any attribute tuple $(x_1, \ldots, x_l)$, there is exactly one $\alpha \in \mathbb{Z}_q$ that would make the match: take α as the discrete logarithm of $h/(g_1^{x_1} \cdots g_l^{x_l})$ with respect to the generator h_0. Since Alice generates α at random and keeps it secret, h leaks no information about $x_1, \ldots, x_l$. For this reason, the number α is called a *blinding factor*. (This is a generalization of the special case $l = 1$, which has been used as a commitment scheme by, amongst others, Chaum and Crepeau [16], Chaum and van Antwerpen [36], Chaum [32], Boyar, Kurtz, and Krentel [8], Pedersen [49], van Heijst and Pedersen [56], and Okamoto [48].)

At the same time, the following basic security property holds (see Brands [15, Proposition 2.3.3.]).

Property 1 *Regardless of the choice of l, assuming it is infeasible to compute discrete logarithms in G_q, Alice cannot compute a Digital Credential public key for which she knows more than one secret key.*

(This property was first proved by Brands [10]; earlier on, Chaum, van Heijst, and Pfitzmann [37, 38] proved collision-intractability for the case of fixed l.) Consequently, by signing the Digital Credential public key the *CA* binds a unique attribute tuple to Alice's Digital Credential, albeit in an indirect manner: the *CA*'s signature binds Alice's public key, which in turn binds Alice's secret key, which contains all the attributes.

The following property is an immediate consequence of Property 1.

Property 2 *Assuming it is infeasible to compute discrete logarithms in G_q, Bob cannot compute any secret key when presented with Alice's Digital Credential public key, regardless of which property of $(x_1, \ldots, x_l)$ Alice discloses to him.*

Note that even if Alice discloses all of $(x_1, \ldots, x_l)$, computing Alice's secret key still requires Bob to compute the discrete logarithm of $h/(g_1^{x_1} \cdots g_l^{x_l})$ with respect to h_0. Therefore, Alice can disclose arbitrary attribute properties without facilitating a replay attack.

Many constructions for G_q are known for which it is believed infeasible to compute discrete logarithms. One choice for G_q is a subgroup of $\mathbb{Z}_p^*$, where p is a prime such that q evenly divides $(p - 1)$. Another choice is an elliptic curve of order q over a finite field. In any case, it is recommended that q is at least 160 bits.

Note that $x_1, \ldots, x_l$ must all be numbers in $\mathbb{Z}_q$, because q is the order of G_q. How to map meaningful attribute information to numbers in $\mathbb{Z}_q$ is up to the *CA*. It could specify efficient mappings in a public list (e.g., Alice's gender can be expressed by a single bit, and her citizenship by a number between 0 and 266) or use encodings that enable anyone to infer the meaning of x_i from x_i itself (e.g., Alice's citizenship represented in ASCII). The interpretation of each x_i depends on the application at hand. Here are two examples:

- In an electronic coin, x_1 could specify the denomination, currency, and expiry date of the coin, and x_2 could specify a negotiable quality that Alice might want to reveal to Bob in return for a discount.

- In a national ID card, x_1 could represent Alice's name, x_2 her place of birth, x_3 her gender, x_4 her child support payments, x_5 her divorce status, x_6 her age, and so on.

The *CA* can hash attributes that cannot be represented by a number in $\mathbb{Z}_q$ by using a collision-intractable hash function; to disclose these attributes, Alice discloses the preimages for the corresponding x_i's. However, the number of values that an attribute can take on must be limited (unless Alice can hide the attribute when showing the Digital Credential), otherwise the *CA* can identify Alice in the showing protocol by encoding a unique attribute value into her Digital Credential. In light of this, the approach of publishing a list of public mappings is usually preferable; the *CA* could publish this list along with its public key, and sign it using its secret key.

3 Showing a Digital Credential

At this stage it is not obvious at all how the *CA* can digitally sign a public key without seeing that public key and its own signature, while being convinced that the public key is bound to the right attributes. For the moment we will simply assume that a Digital Credential can be issued in this way, and defer a discussion to Section 5. The reason we first consider the showing protocol is to simplify the exposition.

Signed proofs

As explained in the previous section, to show a Digital Credential to Bob, Alice transmits to him the Digital Credential public key and the *CA*'s digital signature. In addition, she selectively discloses a property of the attributes and digitally signs a nonce using her secret key, to prevent replay attacks.

In our design, Alice's digital signature is derived from a *proof of knowledge*. Informally, this is a protocol by means of which one party can convince another that it "knows" a secret key corresponding to its public key; see Goldwasser, Micali, and Rackoff [45] for general reflections. In our case, Alice must convince Bob that she knows a secret key $(x_1, \ldots, x_l, \alpha)$ corresponding to a Digital Credential public key, $h = g_1^{x_1} \cdots g_l^{x_l} h_0^{\alpha}$. Figure 2 describes a showing protocol using a simple proof of knowledge that does not leak Alice's secret key. The symbol $\in_{\mathcal{R}}$ means that the variable on its left is chosen independently and uniformly at random from the set specified on its right, and "sign(h)" denotes the signature of the *CA* on the Digital Credential public key. The number c is Bob's *challenge*, and the numbers $r_1, \ldots, r_{l+1}$ are Alice's *response(s)*. It is easy to see that the verification relation for $r_1, \ldots, r_{l+1}$ holds if Alice knows $(x_1, \ldots, x_l, \alpha)$ and follows the protocol. The number a is called Alice's *initial witness*. The special case $l = 0$ is the Schnorr proof of knowledge [52], while the case $l = 1$ is an optimization of Okamoto's extension [48, page 36] of the Schnorr proof of knowledge.

Bob cannot compute Alice's secret key, nor any other secret key corresponding to h:

- A *simulator* can produce transcripts of protocol executions without knowing a secret key: generate a random challenge c, generate $(r_1, \ldots, r_{l+1})$ at random, and compute a such that the verification relation holds. This resulting transcript has the same probability distribution as what Bob obtains when he follows the protocol. Bob could choose c from another probability distribution that cannot be simulated, though, and so this is not a proof that Bob

$$\begin{array}{lcl}
\boxed{\text{Alice}} & & \boxed{\text{Bob}} \\
w_1,\ldots,w_{l+1} \in_{\mathcal{R}} \mathbb{Z}_q & & \\
a := g_1^{w_1}\cdots g_l^{w_l} h_0^{w_{l+1}} & & \\
& \xrightarrow{h,\,\text{sign}(h),\,a} & \\
& & c \in_{\mathcal{R}} \mathbb{Z}_q \\
& \xleftarrow{c} & \\
r_1 := cx_1 + w_1 \bmod q & & \\
\vdots & & \\
r_l := cx_l + w_l \bmod q & & \\
r_{l+1} := c\alpha + w_{l+1} \bmod q & & \\
& \xrightarrow{r_1,\ldots,r_{l+1}} & \\
& & h^c a \overset{?}{=} g_1^{r_1}\cdots g_l^{r_l} h_0^{r_{l+1}}
\end{array}$$

Figure 2: Proof of knowledge for a Digital Credential public key.

cannot learn Alice's secret key. If c would come from a small challenge set, instead of from $\mathbb{Z}_q$, then simulation is possible regardless of how Bob generates c: the simulator guesses the challenge c that Bob will produce and forms $a := h^{-c}g_1^{r_1}\cdots g_l^{r_l}h_0^{r_{l+1}}$ for randomly chosen "responses," and repeats this experiment until it correctly guesses c. This proves that for small challenge sets Bob cannot learn anything, since everything he obtains from Alice he could have generated himself. In our case the challenge set is large, which causes the simulation to take much too long, but it nevertheless suggests that Alice's secret key does not leak.

- If Bob does not know all the x_i's a priori, then we can rigorously prove that Alice's secret key does not leak. Namely, Bob cannot distinguish which particular secret key Alice uses in the protocol; see Brands [15, Proposition 2.4.8] for a simple proof. We already know that h itself does not leak this information. Therefore, if after a certain number of protocol executions Bob can compute a secret key, that key will differ with significant probability from the one that Alice knows. This would provide a way for Alice and Bob to jointly generate a Digital Credential public key for which they know more than one secret key, which according to Property 1 is infeasible. (The notions of so-called witness-indistinguishability and witness-hiding, and the general technique of using the first to prove the second, are due to Feige and Shamir [42].)

At the same time, Alice cannot trick Bob into accepting her responses without her actually knowing a Digital Credential secret key, for the following reason. Suppose that Alice can come up with an initial witness a for which she can provide correct responses for two different challenges, c and c'. Denote the two response sets by $(r_1,\ldots,r_{l+1})$ and $(r'_1,\ldots,r'_{l+1})$, respectively. From

$h^c a = g_1^{r_1} \cdots g_l^{r_l} h_0^{r_{l+1}}$ and $h^{c'} a = g_1^{r'_1} \cdots g_l^{r'_l} h_0^{r'_{l+1}}$ it follows that

$$h = g_1^{(r_1 - r'_1)/(c-c')} \cdots g_l^{(r_l - r'_l)/(c-c')} h_0^{(r_{l+1} - r'_{l+1})/(c-c')}.$$

That is, $((r_1 - r'_1)/(c - c'), \ldots, (r_l - r'_l)/(c - c'), (r_{l+1} - r'_{l+1})/(c - c'))$ is a secret key corresponding to the Digital Credential public key h, and Alice can compute it from the two challenges and the corresponding response sets. In other words, Alice cannot provide responses to more than one challenge unless she actually knows a secret key corresponding to the Digital Credential public key. The only way in which Alice can cheat is by correctly guessing c and sending $a := h^{-c} g_1^{r_1} \cdots g_l^{r_l} h_0^{r_{l+1}}$ to Bob, for arbitrarily chosen "responses" $(r_1, \ldots, r_{l+1})$; this has success probability $1/q$, which is negligible because q is very large.

To transform the proof of knowledge into a digital signature by Alice, the challenge is provided by a "virtual" verifier instead of by Bob, in accordance with a technique due to Fiat and Shamir [44]. Specifically, Alice sets $c := \mathcal{H}(h, a, m)$, where $\mathcal{H}(\cdot)$ is a strong one-way hash function specified in public and m is an arbitrary message that at the minimum contains a nonce of Bob to prevent replay. The outputs of $\mathcal{H}(\cdot)$ are less than q. For reasons that will become apparent in Section 6, we will assume that it is infeasible to find collisions for $\mathcal{H}(\cdot)$. The tuple $(a, r_1, \ldots, r_{l+1})$ is Alice's digital signature on the message m.[2] See Figure 3 for the resulting showing protocol; it is a generalization of the Schnorr signature scheme [52]. (The symbol "$||$" denotes string concatenation.) Because Alice's digital signature is derived from a proof of knowledge, it is also called a *signed proof*.

Note that a is hashed along when forming c. This ensures that a cannot be formed in a manner that depends on c. Following Pointcheval and Stern [51] and Pointcheval [50], one can prove the unforgeability of the resulting digital signatures in the so-called random oracle model; see Brands [15, Proposition 2.5.4]. The only way to produce digital signatures with respect to the Digital Credential public key h is by knowing a secret key, which cannot be inferred by examining previously produced digital signatures (even if Bob chooses his messages m in a clever fashion). Thus replay attacks are prevented.

Alice can rapidly compute her Digital Credential public key, the initial witness a, and other products of powers of $g_1, \ldots, g_l, h_0$, by using a basic technique known as simultaneous repeated squaring.

Signed proofs with selective disclosure

The above digital signature scheme is a step in the right direction, but it does not support selective disclosure. If Alice claims a property about the attributes in her Digital Credential, how does Bob know that this claim is correct? Alice's digital signature should prove not only that she knows a secret key, but also that the attributes in her Digital Credential meet the property she claims. Generic zero-knowledge solutions, as proposed for instance by Feige and Shamir [43] and Bellare,

[2] If G_q is a subgroup of $\mathbb{Z}_p^*$, it is more efficient, but as secure, for Alice to send $(c, r_1, \ldots, r_{l+1})$ instead of $(a, r_1, \ldots, r_{l+1})$, because the binary length of q can be much smaller than that of p. Bob can verify $(c, r_1, \ldots, r_{l+1})$ by checking that $c = \mathcal{H}(h, g_1^{r_1} \cdots g_l^{r_l} h_0^{r_{l+1}} h^{-c}, m)$. Alternatively, Alice sends a one-way hash of a instead of a, and the verification relation is modified correspondingly. We will not consider optimizations like these any further, because they distract from gaining insight into our techniques.

$$\begin{array}{lcl}
\boxed{\text{Alice}} & & \boxed{\text{Bob}} \\
 & & m := \text{nonce}||\ldots \\
 & \xleftarrow{\qquad m \qquad} & \\
w_1,\ldots,w_{l+1} \in_{\mathcal{R}} \mathbb{Z}_q & & \\
a := g_1^{w_1}\cdots g_l^{w_l} h_0^{w_{l+1}} & & \\
c := \mathcal{H}(h,a,m) & & \\
r_1 := cx_1 + w_1 \bmod q & & \\
\vdots & & \\
r_l := cx_l + w_l \bmod q & & \\
r_{l+1} := c\alpha + w_{l+1} \bmod q & & \\
 & \xrightarrow{h,\, \text{sign}(h),\, (a, r_1,\ldots,r_{l+1})} & \\
 & & c := \mathcal{H}(h,a,m) \\
 & & h^c a \overset{?}{=} g_1^{r_1}\cdots g_l^{r_l} h_0^{r_{l+1}}
\end{array}$$

Figure 3: Alice's digital signature on Bob's message m.

Jakobsson, and Yung [3], are highly impractical, because the claimed properties must be encoded into Boolean circuits and auxiliary commitments must be used for each gate.

The solution, as proposed by Brands [9], exploits the fact that the proof of knowledge in Figure 2 works regardless of the choice of $(g_1,\ldots,g_l,h_0)$. In what follows we use the following terminology: $(x_1,\ldots,x_l,\alpha)$ is called a *representation* of h with respect to the base tuple $(g_1,\ldots,g_l,h_0)$.

A simple example. Suppose for simplicity that Alice wants to do the electronic equivalent of crossing out all but one of the data fields on a paper certificate. That is, she sends h and the *CA*'s digital signature on h to Bob, and discloses a value $y_1 \in \mathbb{Z}_q$ that supposedly equals x_1. If Alice's claim is valid, then $h/g_1^{y_1} = g_2^{x_2}\cdots g_l^{x_l}h_0^{\alpha}$ and so she knows a representation of $h/g_1^{y_1}$ with respect to the base tuple $(g_2,\ldots,g_l,h_0)$. Conversely, if Alice can prove knowledge of a representation of $h/g_1^{y_1}$ with respect to $(g_2,\ldots,g_l,h_0)$, this means she knows $(y_2,\ldots,y_{l+1})$ such that

$$h/g_1^{y_1} = g_2^{y_2}\cdots g_l^{y_l}h_0^{y_{l+1}},$$

and so

$$h = g_1^{y_1}g_2^{y_2}\cdots g_l^{y_l}h_0^{y_{l+1}}.$$

The tuple $(y_1,\ldots,y_{l+1})$ must be Alice's secret key, since according to Property 1 she cannot know more than one secret key for a Digital Credential public key. Therefore, $x_1 = y_1 \bmod q$.

To prove knowledge of $h/g_1^{y_1}$ with respect to the tuple $(g_2,\ldots,g_l,h_0)$, Alice applies the signed proof of Figure 3, but with $h/g_1^{y_1}$ replacing h and $(g_2,\ldots,g_l,h_0)$ replacing $(g_1,\ldots,g_l,h_0)$. By expanding the resulting expressions, so that everything is expressed in terms of products of powers

$\boxed{\text{Alice}}$ $\qquad\qquad\qquad\qquad$ $\boxed{\text{Bob}}$

$m :=$ nonce$||\ldots$

$\xleftarrow{\qquad\quad m \qquad\quad}$

$w_2, \ldots, w_{l+1} \in_{\mathcal{R}} \mathbb{Z}_q$
$a := g_2^{w_2} \cdots g_l^{w_l} h_0^{w_{l+1}}$
$c := \mathcal{H}(h, a, F, m)$
$r_2 := cx_2 + w_2 \bmod q$
$\vdots$
$r_l := cx_l + w_l \bmod q$
$r_{l+1} := c\alpha + w_{l+1} \bmod q$

$\xrightarrow{h, \text{sign}(h), y_1, (a, r_2, \ldots, r_{l+1})}$

$c := \mathcal{H}(h, a, F, m)$
$h^c a \overset{?}{=} g_1^{y_1 c} g_2^{r_2} \cdots g_l^{r_l} h_0^{r_{l+1}}$

Figure 4: Signed proof of $x_1 = y_1$.

of $g_1, \ldots, g_l, h_0$, the protocol in Figure 4 results. (The expansion will make it much easier to understand our later techniques, and is also desirable from a computational viewpoint.)

The symbol F that is hashed along when forming c represents a unique description of the attribute property that Alice demonstrates. If the properties that may be demonstrated are all of the form $x_1 = y_1 \bmod q$, then it suffices to hash along y_1. If, however, the space of attribute properties is larger, more information is needed to identify the demonstrated property. If F would be left out, then Alice could form a and $r_2, \ldots, r_{l+1}$ in the same manner as in Figure 3, and the result would "prove" that $y_1 = x_1 + (w_1/\mathcal{H}(h, a, m)) \bmod q$. Hashing along y_1 ensures that y_1 cannot be formed after c has been formed. In the random oracle model one can prove the security of the resulting signed proof; see Brands [15, Proposition 3.3.6].

Bob cannot learn any information about Alice's attributes beyond the fact that $x_1 = y_1 \bmod q$, regardless of the manner in which $x_1, \ldots, x_l$ have been chosen. In the example this is intuitively clear, since Alice "chops off" $g_1^{x_1}$ and then does a signed proof for the "remainder;" we already saw that the protocol of Figure 3 hides all information about the attributes. For a general proof, see Brands [15, Proposition 3.3.4].

Demonstrating linear relations. It is easy to see how to extend this technique so that Alice can convince Bob of the values of several attributes at once. The resulting protocol, which was first proposed by Brands [13], corresponds to disclosing several data fields on a paper certificate and crossing out the rest. More generally, Alice can demonstrate attribute formulae that connect linear relations by zero or more AND connectives. This ability, which does not have a simple analogue in the physical world, is of practical interest for at least two reasons:

- If the *CA* only certifies attribute values that are 0 or 1 (denoting the absence or the presence of a quality, say), then Alice can demonstrate to Bob that she meets exactly $s \geq 0$ out of $t \geq s$ qualities by proving that $x_1 + \cdots + x_t = s \bmod q$.

- As we will see in Section 5, in our issuing protocol the values that the *CA* encodes into a Digital Credential are in fact not the attribute values themselves but the products of the attribute values with a blinding factor. Consequently, Alice must be able to prove certain linear relations in order to disclose individual attribute values.

By way of example, consider a Digital Credential containing three attributes, $h = g_1^{x_1} g_2^{x_2} g_3^{x_3} h_0^{\alpha}$, and suppose that Alice wants to prove the following attribute property F to Bob:

$$(x_1 = 2x_3 + 3 \bmod q) \text{ AND } (x_2 = 4x_3 + 5 \bmod q).$$

If this formula holds true, then

$$h/(g_1^3 g_2^5) = (g_1^2 g_2^4 g_3)^{x_3} h_0^{\alpha}.$$

Consequently, Alice can prove knowledge of a representation of $h/(g_1^3 g_2^5)$ with respect to the tuple $(g_1^2 g_2^4 g_3, h_0)$. As before, this ability also suffices to convince Bob. After substituting $h/(g_1^3 g_2^5)$ and the new base tuple, $(g_1^2 g_2^4 g_3), h_0)$, into the proof of Figure 3, and expanding the resulting expressions, the protocol in Figure 5 results.

$$\begin{array}{lcl}
\boxed{\text{Alice}} & & \boxed{\text{Bob}} \\
 & & m := \text{nonce}|| \ldots \\
 & \xleftarrow{\qquad\qquad m \qquad\qquad} & \\
w_1, w_2 \in_{\mathcal{R}} \mathbb{Z}_q & & \\
a := g_1^{2w_1} g_2^{4w_1} g_3^{w_1} h_0^{w_2} & & \\
c := \mathcal{H}(h, a, F, m) & & \\
r_1 := cx_3 + w_1 \bmod q & & \\
r_2 := c\alpha + w_2 \bmod q & & \\
 & \xrightarrow{h, \text{sign}(h), F, (a, r_1, r_2)} & \\
 & & c := \mathcal{H}(h, a, F, m) \\
 & & h^c a \overset{?}{=} g_1^{2r_1+3c} g_2^{4r_1+5c} g_3^{r_1} h_0^{r_2}
\end{array}$$

Figure 5: Signed proof of $(x_1 = 2x_3 + 3)$ AND $(x_2 = 4x_3 + 5)$.

Demonstrating a negation. On now to another selective disclosure technique, for proving that an attribute in a Digital Credential does *not* equal a certain value. Suppose by way of example that Alice has been issued a Digital Credential containing a single attribute, $h = g_1^{x_1} h_0^{\alpha}$, and she wants to prove that $x_1 \neq y_1 \bmod q$. If indeed $x_1 \neq y_1 \bmod q$ then there exists an $\epsilon \neq 0 \bmod q$ so that $x_1 = y_1 - \epsilon \bmod q$, and so

$$g_1 = (g_1^{y_1}/h)^{1/\epsilon} h_0^{\alpha/\epsilon}.$$

On the other hand, if Alice knows z_1, z_2 such that $g_1 = (g_1^{y_1}/h)^{z_1} h_0^{z_2}$, then $h^{z_1} = g_1^{y_1 z_1 - 1} h_0^{z_2}$. If $z_1 = 0 \bmod q$ then Alice would know the discrete logarithm of g_1 with respect to h_0; it equals z_2. Therefore, $z_1 \neq 0 \bmod q$ and so $h = g_1^{y_1 - 1/z_1} h_0^{z_2/z_1}$. But then it follows from Property 1 that $x_1 = y_1 - 1/z_1 \bmod q$, which is not equal to y_1, as Alice intended to demonstrate. In other words, Alice can demonstrate that $x_1 \neq y_1 \bmod q$ by proving knowledge of a representation of g_1 with respect to $(g_1^{y_1}/h, h_0)$.

Again, for security a description of the formula must be hashed along when forming c.

Demonstrating properties with AND and NOT connectives. By combining the previous techniques, Alice can demonstrate attribute formulae combining linear relations by one NOT connective and arbitrarily many AND connectives. By way of example, consider a Digital Credential containing three attributes, $h = g_1^{x_1} g_2^{x_2} g_3^{x_3} h_0^{\alpha}$, and the following attribute formula F:

$$\text{NOT}(x_1 + 3x_2 + 5x_3 = 7 \bmod q) \text{ AND } (3x_1 + 10x_2 + 18x_3 = 23 \bmod q).$$

With ϵ denoting $7 - (x_1 + 3x_2 + 5x_3) \bmod q$, this formula is equivalent to the statement that there exists an $\epsilon \neq 0 \bmod q$ such that

$$(x_1 = 1 + 4x_3 - 10\epsilon) \text{ AND } (x_2 = 2 - 3x_3 + 3\epsilon).$$

By substitution we get

$$g_1^{10} g_2^{-3} \quad = \quad (g_1^1 g_2^2/h)^{1/\epsilon} (g_1^4 g_2^{-3} g_3)^{x_3/\epsilon} h_0^{\alpha/\epsilon}.$$

Consequently, if F holds true then Alice can prove knowledge of a representation of $g_1^{10} g_2^{-3}$ with respect to $(g_1^1 g_2^2/h, g_1^4 g_2^{-3} g_3, h_0)$. This is also sufficient to convince Bob, as can be seen by following the same line of reasoning as before. The resulting protocol is depicted in Figure 6.

Other selective disclosure techniques. Alice is not confined to demonstrating properties of attributes contained in a single Digital Credential. She can also demonstrate properties pertaining to attributes encoded in different Digital Credentials, although some flexibility is lost. The basic idea, which is due to Brands [10, page 22] and was rediscovered by Camenisch and Stadler [17], is to apply the techniques for a single Digital Credential to the appropriate product of powers of the Digital Credential public keys. This works in a straightforward manner if the attributes specified in the formula are all exponents of the same base number. For example, with $h_1 = g_1^{x_{11}} \cdots g_l^{x_{1l}}$ and $h_2 = g_1^{x_{21}} \cdots g_l^{x_{2l}}$, Alice can demonstrate that $\alpha_1 x_{11} + x_{21} = \alpha_2 \bmod q$ by proving knowledge of a representation of $h_1^{\alpha_1} h_2 g_1^{-\alpha_2}$ with respect to $(g_2, \ldots, g_l)$. For each Digital Credential public key Alice may need to perform an additional proof of knowledge of a representation, for security reasons. To prove knowledge of a representation of each of many public keys $h_1, \ldots, h_t$, Alice can prove knowledge of a representation of the product $h_1 h_2^{\alpha_2} \cdots h_t^{\alpha_t}$, for $\alpha_2, \ldots, \alpha_t$ randomly generated by Bob from a large set.

We leave it at this and refer to Brands [15, Chapter 3] for a treatise of how Alice can demonstrate arbitrary attribute formulae connecting linear relations by AND, OR, and NOT connectives, and how she can efficiently demonstrate that an attribute is contained in a large interval.

$\boxed{\text{Alice}}$ $\boxed{\text{Bob}}$

$m := \text{nonce}|| \ldots$

$\xleftarrow{\quad m \quad}$

$w_1, w_2, w_3 \in_{\mathcal{R}} \mathbb{Z}_q$
$a := g_1^{w_1+4w_2} g_2^{2w_1-3w_2} g_3^{w_2} h_0^{w_3} h^{-w_1}$
$c := \mathcal{H}(h, a, F, m)$
$\epsilon := 7 - (x_1 + 3x_2 + 5x_3) \bmod q$
$\delta := 1/\epsilon \bmod q$
$r_1 := c\delta + w_1 \bmod q$
$r_2 := cx_3\delta + w_2 \bmod q$
$r_3 := c\alpha\delta + w_3 \bmod q$

$\xrightarrow{h, \text{sign}(h), F, (a, r_1, r_2, r_3)}$

$c := \mathcal{H}(h, a, F, m)$
$h^{r_1} a \overset{?}{=} g_1^{r_1+4r_2-10c} g_2^{2r_1-3r_2+3c} g_3^{r_2} h_0^{r_3}$

Figure 6: Signed proof of NOT$(x_1{+}3x_2{+}5x_3 = 7 \bmod q)$ AND $(3x_1{+}10x_2{+}18x_3 = 23 \bmod q)$.

4 Cryptographic security measures

Our selective disclosure techniques benefit not only Alice's privacy. They also enable the *CA* to strongly deter lending and discarding of Digital Credentials.

Lending protection

Nothing is easier to copy than a string of bits. To discourage Alice from lending (or giving away copies of) a personal Digital Credential, such as a drivers' license or a diploma, the *CA* can encode into the Digital Credential a confidential attribute of Alice. This approach originates from Dwork, Lotspiech, and Naor [41], who proposed it in the context of discouraging the distribution of copyrighted materials. An issuer of Digital Credentials for gaining access to gender-specific online discussions, say, could encode into Alice's Digital Credential not only a bit indicating her gender, but also her credit card data. This ensures that Alice cannot give copies of her Digital Credential to other individuals without disclosing her credit card data to them. This is because signing Bob's message requires knowledge of Alice's entire secret key, which includes all the attributes. At the same time, Alice's privacy is protected because she can hide her confidential attribute when showing the Digital Credential.

Alternative attributes that the *CA* could encode to discourage lending are Alice's identity (her reputation may be damaged if it is disclosed), an electronic coin or some other electronic token of value (it need not fit into a single attribute), or a secret key of Alice (e.g., to sign e-mail or to authenticate access to a bank account). In Section 5 we will see that the *CA* does not need to know the attribute in order to be able to encode it into the Digital Credential.

Stronger protection against lending requires smartcards or other tamper-resistant devices for Digital Credential holders. We will examine this in Section 8.

Note that X.509 certificates and other identity certificates do not offer this level of security. Nothing discourages the holder of an identity certificate from giving away copies of his or her certificate, unless the holder runs a serious personal risk. This typically means that the certificate must be accepted in applications with liability issues that are much more complicated than in closed systems with not a lot at stake. This makes it hard to bootstrap certificate applications in practice. For example, if identity certificates are to be secure for establishing whether Web site visitors are over eighteen, the same certificates should also give access to bank accounts or the like.

Encoding unfavorable attributes

The *CA* can prevent Alice from discarding Digital Credentials that contain unfavorable attributes by encoding these attributes into Digital Credentials that contain related favorable attributes. Information on late payments at a health club, for example, can be encoded into Alice's membership Digital Credential instead of into a separate Digital Credential. Likewise, a mark for drunk driving can be encoded into Alice's drivers' license Digital Credential. This ensures that Alice cannot discard unfavorable attributes without discarding the whole Digital Credential, which would exclude her from further participation in the system. At the same time, Alice's privacy is not violated because she can choose to hide any unfavorable attributes from Bob if he has no clear need to know them.

To limit Alice's ability to reuse old Digital Credentials that are more favorable to her, the *CA* can assign short validity periods to Digital Credentials. Alternatively, or in addition, the *CA* could issue Digital Credentials that cannot be used more than once; see Section 6 for details. The strongest cryptographic protection is for the *CA* to issue only a single non-reusable Digital Credential to Alice, and to recertify it online each time it is shown to a verifier; in Section 5 we will see how to do anonymous recertification.

Stronger protection against discarding requires the use of smartcards or other tamper-resistant devices. These can be programmed to show Digital Credentials in the order in which they were retrieved, and can enter into a suspension mode in case of tampering. See Section 8 for details.

5 Issuing a Digital Credential

Since Alice reveals the Digital Credential public key and the *CA*'s digital signature when showing a Digital Credential, these two elements must be statistically uncorrelated to the information that the *CA* sees when it issues the Digital Credential, even if the *CA* tries to cheat. At the same time, the *CA* must be able to encode the desired attributes into the secret key of the Digital Credential, even if Alice tries to cheat. We now examine an issuing protocol that achieves these two goals. No insight is given into why the protocol is secure; see Brands [15, Chapter 4] for this. As mentioned, Alice need not communicate over an anonymous channel; in fact, in many applications she will be required to identify herself to the *CA*.

Basic issuing protocol

In the following issuing protocol, Alice will obtain the *CA*'s signature on a public key $h \neq 1$ 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 blinding factor chosen by Alice during the protocol execution. This form is different from that considered in Sections 2 and 3, in that the blinding factor, α_1, affects all the components of Alice's secret key. All the techniques still apply, though:

- In the same way as in Section 3, h reveals no information about $x_1, \ldots, x_l$: for any $h \in G_q$ and any attribute tuple $(x_1, \ldots, x_l)$, there is exactly one blinding factor $\alpha_1 \in \mathbb{Z}_q$ that would make the match. (The case $g_1^{x_1} \cdots g_l^{x_l} h_0 = 1$ can be excluded.) Since Alice generates α_1 at random and keeps it secret, no attribute information leaks.

- With $z_0 := \alpha_1$ and $z_i := x_i\alpha_1 \bmod q$, for all $i \in \{1, \ldots, l\}$, demonstrating that $x_1 = \beta x_2 + \gamma \bmod q$, say, is equivalent to demonstrating that $(z_1 = \beta z_2 + \gamma z_0)$ AND NOT$(z_0 = 0)$. (Having Alice demonstrate that $\alpha_1 \neq 0 \bmod q$ is preferable over having Bob check that $h \neq 1$, for a highly technical reason; see Brands [15, Section 5.1.2].) More generally, our showing protocol techniques still apply.

In the issuing protocol, the *CA* must know the discrete logarithm of each g_i with respect to a random base $g_0 \in G_q$. Hereto it generates in the key set-up phase l numbers $y_1, \ldots, y_l$ at random from $\mathbb{Z}_q$, and computes $g_i := g_0^{y_i}$ for all $i \in \{1, \ldots, l\}$. In the same way, it generates $h_0 := g_0^{x_0}$ for a random $x_0 \in \mathbb{Z}_q$. The public key of the *CA* is $g_1, \ldots, g_l, h_0$ and its secret key is $y_1, \ldots, y_l, x_0$. The Digital Credential verification relation is defined as

$$c'_0 = \mathcal{H}(h, g_0^{c'_0} h^{r'_0}),$$

where (c'_0, r'_0) is the *CA*'s digital signature.[3] The Digital Credential issuing protocol is as follows:

Step 1. The *CA* generates a random number $w_0 \in \mathbb{Z}_q$, and sends $a_0 := g_0^{w_0}$ to Alice.

Step 2. Alice generates three random numbers $\alpha_1 \in \mathbb{Z}_q^*$ and $\alpha_2, \alpha_3 \in \mathbb{Z}_q$. She then computes $h := (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1}$, $c'_0 := \mathcal{H}(h, g_0^{\alpha_2}(g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_3} a_0)$, and sends $c_0 := c'_0 - \alpha_2 \bmod q$ to the *CA*.

Step 3. The *CA* sends $r_0 := (w_0 - c_0)/(x_0 + x_1 y_1 + \cdots + x_l y_l) \bmod q$ to Alice.

Alice accepts if and only if $a_0 = g_0^{c_0}(g_1^{x_1} \cdots g_l^{x_l} h_0)^{r_0}$. If this verification holds, she computes $r'_0 := (r_0 + \alpha_3)/\alpha_1 \bmod q$. See Figure 7 for the resulting protocol.

If Alice follows the issuing protocol and accepts, then the following holds: for any Digital Credential public key h and any *CA* signature (c'_0, r'_0) (Alice reveals these to Bob), and for any (a_0, c_0, r_0) satisfying $g_0^{c_0}(g_1^{x_1} \cdots g_l^{x_l} h_0)^{r_0} = a_0$ (this is what the *CA* sees in the issuing protocol execution) there is exactly one choice of blinding factors $(\alpha_1, \alpha_2, \alpha_3)$ that would make the match. Therefore, the public key and the *CA*'s signature that Alice reveals in the showing protocol do

[3] Technically, (c'_0, r'_0) is not a signature on h but a secret-key certificate, a notion originating from Brands [14]; see also Brands [15, Section 2.6.2].

$$
\begin{array}{lcl}
\boxed{\text{Alice}} & & \boxed{\mathit{CA}} \\
 & & w_0 \in_{\mathcal{R}} \mathbb{Z}_q \\
 & & a_0 := g_0^{w_0} \\
 & \xleftarrow{a_0} & \\
\alpha_1 \in \mathbb{Z}_q^* & & \\
\alpha_2, \alpha_3 \in_{\mathcal{R}} \mathbb{Z}_q & & \\
h := (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1} & & \\
c_0' := \mathcal{H}(h, g_0^{\alpha_2} (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_3} a_0) & & \\
c_0 := c_0' - \alpha_2 \bmod q & & \\
 & \xrightarrow{c_0} & \\
 & & r_0 := (w_0 - c_0)/(x_0 + x_1 y_1 + \cdots + x_l y_l) \bmod q \\
 & \xleftarrow{r_0} & \\
a_0 \stackrel{?}{=} g_0^{c_0} (g_1^{x_1} \cdots g_l^{x_l} h_0)^{r_0} & & \\
r_0' := (r_0 + \alpha_3)/\alpha_1 \bmod q & &
\end{array}
$$

Figure 7: Basic issuing protocol.

not enable Bob and the *CA* to find out in which issuing protocol execution that Digital Credential was issued. In particular, even if Alice identified herself to the *CA*, the Digital Credential showing cannot be traced to her identity. More generally, Bob and the *CA* cannot infer more than that the individual who showed the Digital Credential is one of all the applicants that requested Digital Credentials for attributes satisfying the attribute property that Alice disclosed in the showing protocol.

Alice can preprocess almost all her computations. In addition, Alice may prefer to skip the verification of r_0. The *CA* cannot leak information through an incorrect r_0, because Bob never sees r_0, only its blinded form r_0'. In fact, Bob will not be able to determine whether Alice or the *CA* is the source of incorrect Digital Credential data supplied in the showing protocol.

This protocol is a modification of a protocol by Brands [13]. In contrast to the original version, the modified protocol allows the *CA* to run different executions in parallel in an arbitrary fashion; see Brands [15, Section 4.4.2]. The modification, which is brought about by swapping the challenge and the response in the Digital Credential verification relation, is due to Schoenmakers [53], who proposed it in the context of the electronic cash withdrawal protocol of Brands [12].

Note that different versions of the same Digital Credential are unlinkable, because Alice generates $(\alpha_1, \alpha_2, \alpha_3)$ independently at random in each execution of the issuing protocol.

Issuing protocol with attribute hiding

In the basic issuing protocol, the *CA* must know $(x_1, \ldots, x_l)$ in order to form its response, r_0. The following issuing protocol gives the *CA* the ability recertify previously issued Digital Credentials and to issue new ones without knowing the attributes they contain. This is useful when attributes are verified by Registration Authorities that rely on the *CA* for certification but do not want it to know the attributes, and when the *CA* wants to deter lending and discarding of Digital Credentials. It can also be used in all kinds of applications, for example to increment counters in loyalty schemes. The protocol is based on a protocol by Chaum and Pedersen [35] for demonstrating the equality of two discrete logarithms, and was first used by Brands [10, 11] for the special case $l = 1$ in order to design an off-line electronic cash scheme.

The *CA*'s digital signature on a Digital Credential public key $h \neq 1$ this time consists of three elements, $z' \in G_q$, $c'_0 \in Z_q$, and $r_0 \in \mathbb{Z}_q$, satisfying

$$c'_0 = \mathcal{H}(h, z', g_0^{r'_0} h_0^{-c'_0}, h^{r'_0}(z')^{-c'_0}).$$

This time, Alice must know $z := (g_1^{x_1} \cdots g_l^{x_l} h_0)^{x_0}$. To enable Alice to compute z herself, the *CA* publishes $(g_1^{x_0}, \ldots, g_l^{x_0}, h_0^{x_0})$ together with the rest of its public key. The issuing protocol is as follows:

Step 1. The *CA* generates a random number $w_0 \in \mathbb{Z}_q$, forms $a_0 := g_0^{w_0}$ and $b_0 := (g_1^{x_1} \cdots g_l^{x_l} h_0)^{w_0}$, and sends (a_0, b_0) to Alice.

Step 2. Alice generates a random number $\alpha_1 \in \mathbb{Z}_q^*$ and two random numbers $\alpha_2, \alpha_3 \in \mathbb{Z}_q$. She then computes $h := (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1}$, $z' := z^{\alpha_1}$, $a'_0 := h_0^{\alpha_2} g_0^{\alpha_3} a_0$, $b'_0 := (z')^{\alpha_2} h^{\alpha_3} b_0^{\alpha_1}$, $c'_0 := \mathcal{H}(h, z', a'_0, b'_0)$, and sends $c_0 := c'_0 + \alpha_2 \bmod q$ to the *CA*.

Step 3. The *CA* sends $r_0 := c_0 x_0 + w_0 \bmod q$ to Alice.

Alice computes $r'_0 := r_0 + \alpha_3 \bmod q$, and accepts if and only if $a'_0 b'_0 = (g_0 h)^{r'_0} (h_0 z')^{-c'_0}$. (This verification relation is essentially the two separate verification relations for a_0 and b_0 batched into one.) The resulting protocol is depicted in Figure 8.

To hide (x_1, x_2), say, Alice at the start of the protocol sends $g_1^{x_1} g_2^{x_2} g_l^{\beta}$ to the *CA*, for a random $\beta \in \mathbb{Z}_q$. This is her contribution to $g_1^{x_1} \cdots g_l^{x_l}$; note that the *CA* needs it to form b_0. In addition, she sends a signed proof of knowledge of a representation of $g_1^{x_1} g_2^{x_2} g_l^{\beta}$ with respect to (g_1, g_2, g_l); this could also demonstrate an attribute property, if the *CA* so desires. The *CA* can multiply in $g_3^{x_3} \cdots g_{l-1}^{x_{l-1}}$ by itself when forming b_0, for attributes $x_3, \ldots, x_{l-1}$.

To have a previously issued Digital Credential refreshed, Alice shows it to the *CA* in the same way as she would show it to Bob. That is, she sets up an anonymous channel with the *CA* and sends $h = (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1}$, the *CA*'s signature on h, and a signed proof (possibly demonstrating an attribute property). The *CA* uses h in a new issuing protocol execution, in place of $g_1^{x_1} \cdots g_l^{x_l} h_0$. Alice obtains a new signature on $h^{\beta} = (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1 \beta}$, for a new blinding factor $\beta \neq 0 \bmod q$, and so the encoded attributes remain intact. Note that any lending and discarding protections remain intact as well.

In comparison to the basic issuing protocol, Digital Credentials in the new issuing protocol are larger and more costly to verify, and Alice's computation of $b_0^{\alpha_1}$ in Step 2 cannot be preprocessed. In light of this, the basic issuing protocol is preferable whenever the additional properties are not needed.

$$\begin{array}{lcl}
\boxed{\text{Alice}} & & \boxed{CA} \\
& & w_0 \in_{\mathcal{R}} \mathbb{Z}_q \\
& & a_0 := g_0^{w_0} \\
& & b_0 := (g_1^{x_1} \cdots g_l^{x_l} h_0)^{w_0} \\
& \xleftarrow{a_0, b_0} & \\
\alpha_1 \in \mathbb{Z}_q^* & & \\
\alpha_2, \alpha_3 \in \mathbb{Z}_q & & \\
h := (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1} & & \\
z' := z^{\alpha_1} & & \\
a_0' := h_0^{\alpha_2} g_0^{\alpha_3} a_0 & & \\
b_0' := (z')^{\alpha_2} h^{\alpha_3} b_0^{\alpha_1} & & \\
c_0' := \mathcal{H}(h, z', a_0', b_0') & & \\
c_0 := c_0' + \alpha_2 \bmod q & & \\
& \xrightarrow{c_0} & \\
& & r_0 := c_0 x_0 + w_0 \bmod q \\
& \xleftarrow{r_0} & \\
r_0' := r_0 + \alpha_3 \bmod q & & \\
a_0' b_0' \overset{?}{=} (g_0 h)^{r_0'} (h_0 z')^{-c_0'} & &
\end{array}$$

Figure 8: Issuing protocol with attribute hiding.

6 Limited-show Digital Credentials

Cinema tickets, stamps, coins, and many other tokens are not reusable; showing them means giving them away or making them unusable. Likewise, a drug prescription in general is not usable for more than a limited number of doses, and an access card might impose a limit on the number of times Alice can gain access to a network or to a building. All these physical objects can be implemented electronically using Digital Credentials, but what stops Alice from reusing her Digital Credentials as often as she pleases? Code transformation methods raise the barrier for those trying to reverse-engineer the client software, but typically it will be too dangerous for the *CA* to rely exclusively on software obfuscation methods to deter unauthorized reuse.

Limiting reuse of Digital Credentials

The *CA* has three basic measures at its disposal to deter unauthorized reuse of Digital Credentials:

- Require Bob to clear each transaction online with a central party (possibly the *CA* itself) that keeps track of the number of times each Digital Credential has been shown. This measure has the advantage of preventing fraud rather than merely deterring it. On the downside, third party authorization is a bottleneck in many applications, resulting in transaction delays and high peak load. Also, the cost of the clearing infrastructure might be prohibitive. Consider, for example, a micropayment system in which each electronic coin must be authorized online. Note that the central database does not scale well, because it should be infeasible to spend copies of the same coin simultaneously at different service providers.

- Provide Alice with a tamper-resistant device, such as a smartcard, that does not cooperate in showing a Digital Credential more times than allowed. This measure avoids the bottleneck of online authorization, but the cost of the devices may be significant and a logistical infrastructure is needed for distributing them. Also, criminals may find it worth their while to invest large sums of money into breaking the tamper-resistance of a device, especially if they can do uncontrollable and untraceable damage.

- Issue Digital Credentials in such a way that Alice can be identified by a central party if and only if she shows a Digital Credential more times than allowed. Digital Credentials with this property are called *limited-show* Digital Credentials. This measure does not require Bob to have an online connection with a central party, although Bob will still need to submit protocol data afterwards to enable the central party to detect and trace fraud.

(Another measure is for the *CA* to encode a verifier identifier into Alice's Digital Credential, so that she can only show it to that verifier; this measure is too inflexible in many applications, though.) The first two measures have real-world analogues: they are widely applied today to ensure that multi-use physical tokens are not reused too often. The third security measure has no direct real-world analogue: showing a physical object twice does not reveal more information than showing it once. As we have seen in Section 4, the third measure offers extra security benefits that cannot be achieved by online clearing. For increased security, the *CA* might want to resort to a combination of various measures. For example, it may be preferable to combine the second and third measures, and the *CA* might want to blacklist Digital Credentials that have been shown too often by adding them to a Credential Revocation List.

For Digital Credentials, the cryptographic design of the first measure is straightforward. This is not true for the second security measure, since tamper-resistant devices act on behalf of the *CA* and their internal operations cannot be inspected by their holders; consequently, they cannot be trusted to protect the privacy of their holders anymore than the *CA* can. See Section 8 for details.

On now to the design of limited-show Digital Credentials. Note that Bob learns new information each time Alice shows the same Digital Credential; the presence of the nonce in Bob's message ensures that Alice's signed proof differs each time. This can be exploited to design Digital Credentials satisfying the following two properties:

- One signed proof of Alice does not reveal any information about the attributes beyond what Alice voluntarily discloses.

- Two signed proofs contain enough information for a central party to compute all the attributes that the *CA* encoded into Alice's Digital Credential.

Consequently, if the *CA* encodes into each of Alice's Digital Credentials an identifier attribute (e.g., a Social Security Number, her name, or the number of an account that Alice uses for withdrawing Digital Credentials on an ongoing basis), then Alice can be traced if she reuses a Digital Credential.

As with our cryptographic measures to protect against lending and discarding, limited-show Digital Credentials do not violate privacy, because Alice can hide her identifier attribute when showing a Digital Credential.

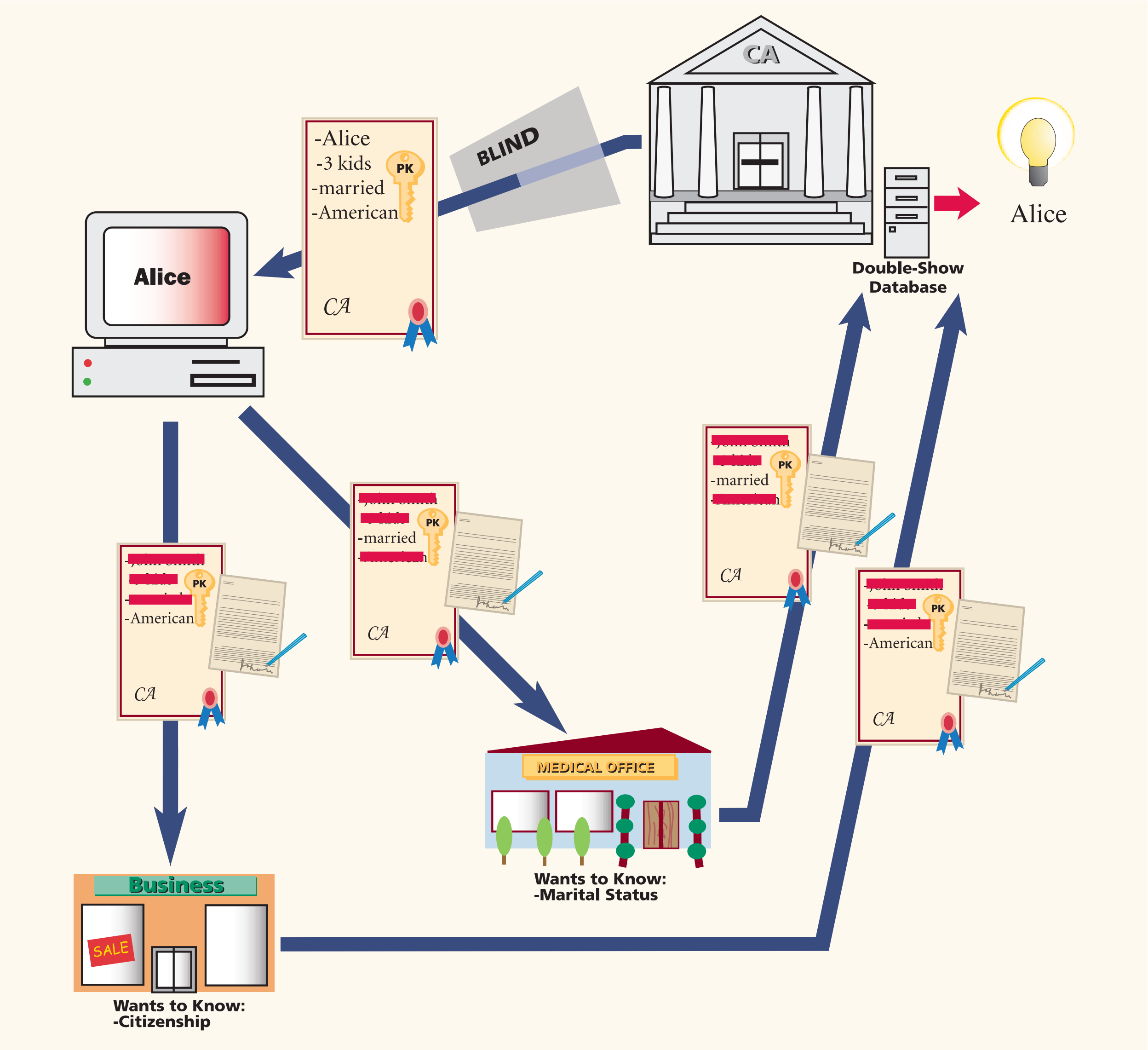

Figure 9: Alice reuses a one-show Digital Credential.

See Figure 9 for a scenario in which Alice shows her one-show demographic Digital Credential to both a medical office and a business.

Basic one-show Digital Credentials

We start with a simple solution that does not accomplish everything we want, but goes a long way. Consider the example protocol in Figure 6. Alice's responses satisfy

$$h^{r_1} a = g_1^{r_1+4r_2-10c} g_2^{2r_1-3r_2+3c} g_3^{r_2} h_0^{r_3}.$$

Suppose that Alice demonstrates the same attribute formula a second time for the same Digital Credential, and in doing so reuses the initial witness a. That is, Alice provides responses (r_1^*, r_2^*, r_3^*) for a challenge c^*, satisfying

$$h^{r_1^*} a = g_1^{r_1^*+4r_2^*-10c^*} g_2^{2r_1^*-3r_2^*+3c^*} g_3^{r_2^*} h_0^{r_3^*}.$$

If $r_1^* = r_1 \bmod q$ then $(r_1 + 4r_2 - 10c, 2r_1 - 3r_2 + 3c, r_2, r_3)$ and $(r_1^* + 4r_2^* - 10c^*, 2r_1^* - 3r_2^* + 3c^*, r_2^*, r_3^*)$ are two different representations of $h^{r_1}a$ with respect to (g_1, g_2, g_3, h_0), since c and c^* are the outcomes of a collision-intractable hash function with different message inputs. According to Property 1 it is infeasible to find two representations of the same number, and so it must be that $r_1^* \neq r_1 \bmod q$. Consequently, h equals

$$g_1^{1+4(r_2^*-r_2)/(r_1^*-r_1)-10(c^*-c)/(r_1^*-r_1)} g_2^{2-3(r_2^*-r_2)/(r_1^*-r_1)+3(c^*-c)/(r_1^*-r_1)} g_3^{(r_2^*-r_2)/(r_1^*-r_1)} h_0^{(r_3^*-r_3)/(r_1^*-r_1)}.$$

Again owing to Property 1, it must be that

$$\begin{aligned} x_1 &= 1 + 4(r_2^* - r_2)/(r_1^* - r_1) - 10(c^* - c)/(r_1^* - r_1) \bmod q \\ x_2 &= 2 - 3(r_2^* - r_2)/(r_1^* - r_1) + 3(c^* - c)/(r_1^* - r_1) \bmod q \\ x_3 &= (r_2^* - r_2)/(r_1^* - r_1) \bmod q \\ \alpha &= (r_3^* - r_3)/(r_1^* - r_1) \bmod q \end{aligned}$$

Therefore, Alice's entire secret key can be computed from the information that she discloses in two showing protocol executions using the same Digital Credential, provided she can be made to reuse her initial witness. This we can accomplish by making the following design change. In step 2 of the issuing protocol, when forming c_0' Alice must hash along an initial witness a for use in the showing protocol. In this manner, the *CA*'s digital signature binds a single initial witness to the Digital Credential, even though the *CA* never sees it. The pair (h, a) can be thought of as a one-time Digital Credential public key. Bob uses the initial witness that Alice hashed along when obtaining the Digital Credential.

The *CA* in the deposit phase registers (a hash of) the one-time Digital Credential public key and the signed proof into a database. In addition, it checks the freshness of Bob's nonce to avoid multiple deposits of the same Digital Credential with respect to the same challenge. Note that the *CA* need not store the entire signed proof in order to be able to trace fraud. In the example, r_3 and c are not needed to compute x_3. More generally, the *CA* needs to store for each deposited Digital Credential merely two numbers in $\mathbb{Z}_q$ and a hash of the Digital Credential public key, regardless of the formula complexity and the number of encoded attributes; see Brands [15, Section 5.4.1].

If the *CA* uses the second Digital Credential issuing protocol in Section 5, then it can encode Alice's identifier without seeing it. In this case, the ability of the *CA* to compute Alice's identifier is evidence that Alice's Digital Credential has been shown too many times.

To incite Bob to deposits the protocol transcript to the *CA* (or another central party) in a timely fashion, for fraud detection, Bob could be made liable for contributing to delays. Alternatively, the central party could provide a financial reward for each signed proof, or a penalty could result if audits reveals a discrepancy between the number of served customers and the number of deposited transcripts.

An electronic cash system

We now have all the tools available to design a simple off-line electronic cash system. The following system is in essence the same as that proposed by Brands [10, 11]. The *CA* is a bank, Alice is a party with a bank account who wants to make payments, and Bob is a service provider and another acceptor of electronic cash. Each electronic coin is a Digital Credential containing two attributes: x_1 identifies the account from which the coin is withdrawn, and x_2 is the coin denomination. (Other pertinent data, such as an expiry date and a currency, may be included into x_2 as well.)

In the withdrawal protocol, Alice receives the bank's signature on a coin public key $h = g_1^{z_1} g_2^{z_2} h_0^{z_0}$, where $z_1 = x_1\alpha_1 \bmod q$, $z_2 = x_2\,\alpha_1 \bmod q$, and $z_0 = \alpha_1$. Alice may withdraw multiple coins of different denominations in parallel. The bank moves the total amount from Alice's bank account into a float account.

To spend a coin to Bob, Alice discloses $y_2 := x_2$ and demonstrates the attribute formula

$$(z_2 = y_2 z_0 \bmod q) \text{ AND NOT}(z_0 = 0)$$

by giving a signed proof. Hereto she proves knowledge of her representation, $(1/\alpha \bmod q, -x_1)$, of $g_2^{y_2} h_0$ with respect to (h, g_1). The challenge c should specify at least a nonce and Bob's account, and optionally a description of the transaction terms or other data. Multiple coins may be needed to make up the payable amount; all coin payments can share the same challenge, provided all the initial witnesses are hashed along.

The resulting protocols are depicted in Figures 10 and 11, respectively.

At the end of each day, say, Bob deposits all the received "coins" (more accurately, payment transcripts). The bank verifies the deposited coins and, assuming there are no double-deposits, credits Bob by moving the total amount due from the float account into the account specified by the signed proof. The bank requires that Bob's challenge message include the concatenation of a nonce and Bob's account number, so that it can distinguish between double-spending and double-depositing. Double-spending is a case of payer fraud for which Bob cannot reasonably be held liable. Double-depositing is either a fraud or a fault-tolerance feature; in either case the bank will not redeem Bob more than once. Note that wire-tappers and thieves cannot deposit Bob's coins into another account.

One-show Digital Credentials with selective disclosure

The basic construction of one-show Digital Credentials requires Alice to know already in the issuing protocol which attribute property she will demonstrate in the showing protocol. This does not work when Alice must be able to choose an attribute property at the moment of showing the

$$
\begin{array}{lcl}
\boxed{\text{Alice}} & & \boxed{\text{Bank}} \\
 & & w_0 \in_{\mathcal{R}} \mathbb{Z}_q \\
 & & a_0 := g_0^{w_0} \\
 & \xleftarrow{\quad a_0 \quad} & \\
\alpha_1 \in_{\mathcal{R}} \mathbb{Z}_q^* & & \\
\alpha_2, \alpha_3, w_1, w_2 \in_{\mathcal{R}} \mathbb{Z}_q & & \\
h := (g_1^{x_1} g_2^{x_2} h_0)^{\alpha_1} & & \\
a := h^{w_1} g_1^{w_2} & & \\
c_0' := \mathcal{H}(h, a, a_0 g_0^{\alpha_2} (g_1^{x_1} g_2^{x_2} h_0)^{\alpha_3}) & & \\
c_0 := c_0' - \alpha_2 \bmod q & & \\
 & \xrightarrow{\quad c_0 \quad} & \\
 & & r_0 := (w_0 - c_0)/(x_0 + x_1 y_1 + x_2 y_2) \bmod q \\
 & \xleftarrow{\quad r_0 \quad} & \\
g_0^{c_0} (g_1^{x_1} g_2^{x_2} h_0)^{r_0} \stackrel{?}{=} a_0 & & \\
r_0' := (r_0 + \alpha_3)/\alpha_1 \bmod q & &
\end{array}
$$

Figure 10: Withdrawal Protocol.

Digital Credential, because her initial witness depends on the attribute property. In particular, it cannot be used to realize the scenario in Figure 9. We now explain how to get around this.

In Step 2 of the issuing protocol, instead of including a formula-dependent initial witness Alice hashes along a "generic" one, $a^* := g_1^{w_1} \cdots g_l^{w_l} h_0^{w_{l+1}} h^{w_{l+2}}$, for random $w_1, \ldots, w_{l+2} \in \mathbb{Z}_q$. In the showing protocol, Alice in addition sends to Bob up to $l+1$ *correction factors*, $e_1, \ldots, e_{l+1} \in \mathbb{Z}_q$. These serve to adjust a^* so that the representation she knows of the adjusted a^* is suitable to compute the signed proof in the manner explained in Section 3. Here is how Alice would demonstrate two of the example formulae discussed in Section 3, where we assume for simplicity that $h = g_1^{x_1} g_2^{x_2} g_3^{x_3} h_0^{\alpha_1}$ rather than $h = (g_1^{x_1} g_2^{x_2} g_3^{x_3} h_0)^{\alpha_1}$:

- To demonstrate $(x_1 = 2x_3 + 3)$ AND $(x_2 = 4x_3 + 5)$, Alice must use an initial witness of the form $g_1^{2w_1} g_2^{4w_1} g_3^{w_1} h_0^{w_2}$, as we have seen in Figure 5. If Alice sets $e_1 := w_1 - 2w_3 \bmod q$, $e_2 := w_2 - 4w_3 \bmod q$, and $e_3 := w_5$, then the adjusted initial witness $a = a^*/g_1^{e_1} g_2^{e_2} h^{e_3}$ equals $g_1^{2w_3} g_2^{4w_3} g_3^{w_3} h_0^{w_4}$, which is of the right form. The verification relation in Figure 5 changes into $h^c(a^*/g_1^{e_1} g_2^{e_2} h^{e_3}) = g_1^{2r_1+3c} g_2^{4r_1+5c} g_3^{r_1} h_0^{r_2}$. See Figure 12 for the revised protocol.

- To demonstrate the formula NOT$(x_1 + 3x_2 + 5x_3 = 7)$ AND $(3x_1 + 10x_2 + 18x_3 = 23)$, Alice must use an initial witness of the form $g_1^{w_1+4w_2} g_2^{2w_1-3w_2} g_3^{w_2} h_0^{w_3} h^{-w_1}$, as we have seen in Figure 6. If Alice sets $e_1 := w_1 + w_5 - 4w_3 \bmod q$ and $e_2 := w_2 + 2w_5 + 3w_3 \bmod q$, then the adjusted initial witness $a = a^*/g_1^{e_1} g_2^{e_2}$ equals $g_1^{-w_5+4w_3} g_2^{-2w_5-3w_3} g_3^{w_3} h_0^{w_4} h^{w_5}$, which is of the right form. The verification relation in Figure 5 changes into $h^{r_1}(a^*/g_1^{e_1} g_2^{e_2}) = g_1^{r_1+4r_2-10c} g_2^{2r_1-3r_2+3c} g_3^{r_2} h_0^{r_3}$. See Figure 13 for the revised protocol.

$$
\begin{array}{lcl}
\boxed{\text{Alice}} & & \boxed{\text{Bob}} \\
 & & m := \text{nonce}||\text{account}||\ldots \\
 & \xleftarrow{\qquad\qquad m \qquad\qquad} & \\
c := \mathcal{H}(h, a, m) & & \\
r_1 := c/\alpha_1 + w_1 \bmod q & & \\
r_2 := -cx_1 + w_2 \bmod q & & \\
 & \xrightarrow{(h, a), (c_0', r_0'), x_2, (r_1, r_2)} & \\
 & & c_0' \stackrel{?}{=} \mathcal{H}(h, a, g_0^{c_0'} h^{r_0'}) \\
 & & c := \mathcal{H}(h, a, m) \\
 & & h^{-r_1} a \stackrel{?}{=} g_1^{r_2} g_2^{-x_2 c} h_0^{-c}
\end{array}
$$

Figure 11: Payment Protocol.

Note that Alice's correction factors are all hashed along when forming the challenge for the showing protocol. Now, consider what happens if Alice reuses the same one-show Digital Credential, first demonstrating the first example formula, F_1, and then the second, F_2. In the first showing protocol execution, Bob obtains $(c, e_1, e_2, e_3, r_1, r_2)$ such that

$$h^{c-e_3} a^* = g_1^{2r_1+3c+e_1} g_2^{4r_1+5c+e_2} g_3^{r_1} h_0^{r_2}.$$

In the second showing, Bob obtains $(c^*, e_1^*, e_2^*, r_1^*, r_2^*, r_3^*)$ such that

$$h^{r_1^*} a^* = g_1^{r_1^*+4r_2^*-10c^*+e_1^*} g_2^{2r_1^*-3r_2^*+3c^*+e_2^*} g_3^{r_2^*} h_0^{r_3^*}.$$

It follows that

$$h^{c-e_3-r_1^*} = g_1^{2r_1+3c+e_1-(r_1^*+4r_2^*-10c^*+e_1^*)} g_2^{4r_1+5c+e_2-(2r_1^*-3r_2^*+3c^*+e_2^*)} g_3^{r_1-r_2^*} h_0^{r_2-r_3^*}.$$

Therefore, Bob can determine Alice's secret key by raising both sides of the equation to the power $1/(c - e_3 - r_1^*) \bmod q$, unless $c - e_3 - r_1^* = 0 \bmod q$. The latter can happen only if the expressions in the exponents are all equal to $0 \bmod q$, because otherwise Bob has found two different representations of the number 1. This explains why the correction factors must be hashed along into c; if Alice could select c in an arbitrary way, subject only to the condition that it is unique in each showing protocol execution, then she could force all the exponent expressions to be zero, and Bob would learn no more than the aggregate of what Alice selectively disclosed in each individual protocol execution.

In general, Alice can demonstrate to Bob any attribute formula connecting linear relations by AND, OR, and NOT connectives; see Brands [15, Section 5.4.2]. In the random oracle model it can be proved that if c is a strong hash of at least (h, a^*), a unique formula description, and all the correction factors, then with overwhelming probability Alice's secret key can be computed upon reuse of her Digital Credential.

$$\begin{array}{lcr}
\boxed{\text{Alice}} & & \boxed{\text{Bob}} \\
 & & m := \text{nonce}||\ldots \\
 & \xleftarrow{\qquad\qquad m \qquad\qquad} & \\
e_1 := w_1 - 2w_3 \bmod q & & \\
e_2 := w_2 - 4w_3 \bmod q & & \\
e_3 := w_5 & & \\
c := \mathcal{H}((h, a^*), F_1, (e_1, e_2, e_3), m) & & \\
r_1 := cx_3 + w_3 \bmod q & & \\
r_2 := c\alpha + w_4 \bmod q & & \\
 & \xrightarrow{(h, a^*), \text{sign}(h, a^*), (e_1, e_2, e_3), F_1, (a, r_1, r_2)} & \\
 & & c := \mathcal{H}((h, a^*), F_1, (e_1, e_2, e_3), m) \\
 & & h^{c-e_3} a^* \stackrel{?}{=} g_1^{2r_1+3c+e_1} g_2^{4r_1+5c+e_2} g_3^{r_1} h_0^{r_2}
\end{array}$$

Figure 12: One-show signed proof of $(x_1 = 2x_3 + 3)$ AND $(x_2 = 4x_3 + 5)$.

Increasing the threshold

There is nothing magical about a threshold value of 1. To ensure that Alice's secret key can be computed if and only if Alice uses a Digital Credential more than $t > 1$ times, the Digital Credential should comprise t generic initial witnesses. When showing the Digital Credential, Alice uses one of the generic initial witnesses that she has not yet used. Any $t + 1$ uses of the same t-show Digital Credential imply the reuse of one of the generic initial witnesses.

7 Privacy for verifiers

Bob can form a signed proof that hides a part or all of the attribute property that Alice disclosed to him. This is analogous to Bob crossing out data fields on a paper-based certificate that Alice presented to him before he sends a copy to a central party. This prevents the central authority from learning information that Bob's clients disclose, which may be in Bob's interest for competitive or other reasons.

The idea is to hash along into Bob's challenge a description of the property that Bob wants to disclose rather than the property that Alice is demonstrating, and to make it appear as if Alice provided responses that demonstrate Bob's property. By way of example, consider the following enhancement of the electronic cash system described in Section 6. Each of Alice's electronic coins encodes an additional attribute, x_3, that specifies Alice's age or other personal data that may have value to merchants (e.g., for inventory management, to determine target markets, or to determine whether customers are legally authorized to conduct a transaction). When paying Bob, Alice should hide x_1 and must disclose x_2, as in the basic system. In addition, she can opt to disclose x_3 to Bob, for example in return for a discount. In the withdrawal protocol, Alice obtains the

$$\begin{array}{lcl}
\boxed{\text{Alice}} & & \boxed{\text{Bob}} \\
 & & m := \text{nonce}|| \ldots \\
 & \xleftarrow{\qquad\qquad m \qquad\qquad} & \\
e_1 := w_1 + w_5 - 4w_3 \bmod q & & \\
e_2 := w_2 + 2w_5 + 3w_3 \text{mod} & & \\
c := \mathcal{H}((h, a^*), F_2, (e_1, e_2), m) & & \\
r_1 := c\delta - w_5 \bmod q & & \\
r_2 := cx_3\delta + w_3 \bmod q & & \\
r_3 := c\alpha\delta + w_4 \bmod q & & \\
 & \xrightarrow{(h, a^*), \text{sign}(h, a^*), (e_1, e_2), F_2, (a, r_1, r_2)} & \\
 & & c := \mathcal{H}((h, a^*), F_2, (e_1, e_2), m) \\
 & & h^{r_1} a^* \overset{?}{=} g_1^{r_1+4r_2-10c+e_1} g_2^{2r_1-3r_2+3c+e_2} g_3^{r_2} h_0^{r_3}
\end{array}$$

Figure 13: One-show signed proof of NOT$(x_1 + 3x_2 + 5x_3 = 7)$ AND $(3x_1 + 10x_2 + 18x_3 = 23)$.

CA's signature on a one-time public key (h, a) where $h = (g_1^{x_1} g_2^{x_2} g_3^{x_3} h_0)^{\alpha_1}$ and $a = h^{w_1} g_1^{w_2} g_3^{w_3}$ for random $w_1, w_2, w_3 \in \mathbb{Z}_q$. Figures 14 and 15 depict the two possible payment protocols, in line with the techniques described in the previous section. In the second payment protocol, w_3 is Alice's correction factor, as explained in the previous section. Note that to reveal x_3, Alice in effect sends to Bob the components x_3, w_3 that she uses to compose r_3 in the case she wants to hide x_3.

Now, if Alice discloses x_3, Bob should be able to end up with a payment transcript that looks like it came from an execution of the payment protocol in which Alice hides x_3 from Bob. To this end, we make the following modifications to the two payment protocols:

- In the first payment protocol, Alice and Bob form c as $\mathcal{H}((h, a), x_2, t, m)$, where $t := \mathcal{H}(s)$ for a random $s \in Z_q$ generated by Bob.

- In the second payment protocol, Alice and Bob form c as $\mathcal{H}((h, a), x_2, t, m)$, where $t := \mathcal{H}((x_3, w_3), s)$ for a random $s \in Z_q$ generated by Bob. Also, Bob computes a fake "response" $r_3 := -cx_3 + w_3 \bmod q$ from the pair (x_3, w_3) disclosed by Alice.

The role of t is to ensure that (x_3, w_3) is included in Bob's challenge in the second payment protocol, as required for security, but not in a manner visible to others. In both cases, Bob deposits $(h, a), (c_0', r_0'), x_2, t, m, (r_1, r_2, r_3)$ to the bank. Clearly, the bank cannot distinguish whether Bob's payment transcript originated from the first or the second payment protocol; the challenge and the responses in Bob's transcript look exactly alike in both cases. By opening the preimage to t, Alice and Bob can at any moment later on prove what property Alice demonstrated to Bob.

For a general treatment, see Brands [15, Section 5.3].

$\boxed{\text{Alice}}$ $\qquad\qquad$ $\boxed{\text{Bob}}$

$m := \text{nonce}||\text{account}||\ldots$

$\xleftarrow{\qquad m \qquad}$

$c := \mathcal{H}((h, a), x_2, m)$
$r_1 := c/\alpha_1 + w_1 \bmod q$
$r_2 := -cx_1 + w_2 \bmod q$
$r_3 := -cx_3 + w_3 \bmod q$

$\xrightarrow{(h, a), (c'_0, r'_0), x_2, (r_1, r_2, r_3)}$

$c'_0 \overset{?}{=} \mathcal{H}(h, a, g_0^{c'_0} h^{r'_0})$
$c := \mathcal{H}((h, a), x_2, m)$
$h^{-r_1} a \overset{?}{=} g_1^{r_2} g_2^{-x_2 c} g_3^{r_3} h_0^{-c}$

Figure 14: Payment Protocol: Reveal x_2, hide x_3.

8 How to integrate tamper-resistant devices

Nothing we discussed thus far assumes special hardware for Alice or Bob; they are both free to use any computing device of their liking to retrieve, verify, store, show, and deposit Digital Credentials. Examples are desktop computers, notebooks, handheld computers, mobile phones, and smart watches. We will refer to computing devices that do not have tamper-resistant components to protect the security interests of the *CA* as *software-only devices*. Using software-only devices has several advantages: the software running on the devices can be manufactured in-house by anyone, and can be distributed via the Internet and other networks (no need for physical transportation); anyone can get into the business of issuing their own kind of Digital Credentials (low start-up cost); and, it is relatively easy for Alice and Bob to be assured that their software are not acting in malicious ways that compromise their privacy or security. Primary contributors to the trust that Alice and Bob can place in their software are compact operating systems, open source code, open market availability, independent code reviews, digital certification of source code and executables, and anti-virus and personal firewall software.

Why and how to use smartcards

There is no security benefit for the *CA* to force Bob to use a special hardware device, with the exception of a secure time-stamping device in some applications. For Alice, the situation is quite different. A software-only device cannot prevent Alice from discarding, reusing, lending, or giving away copies of her Digital Credentials, nor can it prevent other kinds of unauthorized uses. Online authorization prevents Alice from showing a Digital Credential more times than allowed, but does not stop other undesirable behavior. Cryptography-only measures have their limitations as well, as we have seen in Sections 4 and 6.

$\boxed{\text{Alice}}$ $\boxed{\text{Bob}}$

$m := \text{nonce}||\text{account}||\ldots$

$$\xleftarrow{\qquad\qquad m \qquad\qquad}$$

$c := \mathcal{H}((h, a), (x_2, x_3), w_3, m)$
$r_1 := c/\alpha_1 + w_1 \bmod q$
$r_2 := -cx_1 + w_2 \bmod q$

$$\xrightarrow{(h, a), (c_0', r_0'), (x_2, x_3), w_3, (r_1, r_2)}$$

$c_0' \overset{?}{=} \mathcal{H}(h, a, g_0^{c_0'} h^{r_0'})$
$c := \mathcal{H}((h, a), (x_2, x_3), w_3, m)$
$h^{-r_1} a \overset{?}{=} g_1^{r_2} g_2^{-x_2 c} g_3^{-x_3 c + w_3} h_0^{-c}$

Figure 15: Payment Protocol: Reveal (x_2, x_3).

Any Digital Credential system can be implemented using smartcards or other tamper-resistant devices suitable for mass distribution to individuals. These devices can offer strong protection against loss, theft, extortion, lending, copying, and discarding of Digital Credentials, and can prevent all kinds of other unauthorized behavior; see Brands [15, Sections 1.1.6 and 6.4.4.].

However, implementing Alice's part of the protocols entirely in a smartcard is not preferable:

- From a privacy perspective, care must be taken in the way tamper-resistant devices are used. It is virtually impossible to verify that a smartcard does not leak personal data stored inside the card. Even if the issuer and the manufacturer of the devices could be trusted, implementing our protocols entirely in smartcards might not preserve privacy. Namely, the quality and security of noise generators cannot be guaranteed, and pseudorandom number generators derive their outputs from a seed value that is generated by another party than the device holder; this renders any blinding operations ineffective, since the whole point of blinding factors is their secrecy.

- There are serious limitations to the functionality that can be squeezed into a smartcard. Great care must be taken that the addition of complex circuitry and software does not introduce weaknesses in the tamper-resistance characteristics of the hardware. With smartcard components already cramming for space, adding circuitry adversely affects smartcard reliability. A "dead" card at the very least inconveniences and frustrates its holder, and the consequences can be dramatic. Furthermore, smartcards and other devices that do not have their own display and keyboard are vulnerable to fake-terminal attacks.

In light of these considerations, smartcards should not be used as stand-alone devices, but only in conjunction with software-only computing devices. The combination of a tamper-resistant device with a software-only device is natural in many settings. For example, a smartcard can be used over the Internet only when connected to a desktop computer, a mobile phone, or some other device. If Alice's computer has a keyboard and a display, she can enter her password, PIN, or biometric using the keyboard, and read transaction information from her computer's display. By routing all the communications from and to the smartcard through the computer, unexpected data transmissions (i.e., separate from the issuing and showing protocol executions) from and to the smartcard can be prevented.

This is not sufficient to guarantee privacy, though. Alice's tamper-resistant device can try to leak information in the issuing and showing protocols:

- In case Alice's device engages in an interactive protocol, it can leak one or more bits by halting during the protocol execution. For example, one bit can be leaked by deciding whether or not to respond to a challenge before a time-out. This is called the *halting channel*.

- In a similar manner, Alice's device can leak out bits of information by timing the delay before transmitting a message; this is called the *timing channel*.

- A related channel is the *computation error channel*. Here, Alice's device deliberately causes one of its response messages to be incorrect, and encodes information into it.

- Any bits that Alice's device can freely choose can be used to leak information. This so-called *outflow* cannot be detected by Alice's computer, since it complies with the format specifications and verification relations specified by the protocol.

Conversely, any data that is subliminally leaked to Alice's device is called *inflow*. Inflow can cause the smartcard to enter a state of suspension or to default in other ways. In general, Alice will be more concerned about outflow than inflow, since outflow can reveal her smartcard's identifier, access control code, communication and transaction history, attributes, data from other applications, and so on.

Inflow and outflow are examples of subliminal channels, first studied in the context of public-key cryptography by Simmons [54, 55] and Desmedt, Goutier, and Bengio [40, Section 5]. They introduced the idea of using randomization to destroy subliminal channels. In 1988, Chaum [28] proposed very similar randomization techniques to prevent inflow and outflow in the setting of consumer transactions, particularly electronic cash. Chaum advocated the use of smartcards in conjunction with software-only devices, a configuration he referred to as the "wallet-with-observer" setting, and coined the terms inflow and outflow. Chaum's techniques [28, 33, 35] have numerous drawbacks, though, that make them unsuitable for practical use. Namely, in Chaum's wallet-with-observer approach, the tamper-resistant device simply digitally signs statements requested by the user-controlled computer, after checking that the validity of these statements based on the attribute information it maintains. Depending on the application, this can result in enormous damages. For instance, in a financial application the attacker may claim to have been issued digital bearer bonds worth huge sums of money, and may be able to impersonate people by assuming arbitrary identity attributes. Indeed, in the electronic cash system of Chaum and Bos [7, 33], each consumer smartcard has the power to mint untraceable electronic cash. See Brands [15, Section 6.2.2] for a further discussion of the drawbacks of Chaum's wallet-with-observer techniques.

Since in many applications it is completely unacceptable to rely solely on the tamper-resistance of consumer devices (see, for instance, Kocher [46], Anderson and Kuhn [1, 2], Boneh, DeMillo, and Lipton [6], Biham and Shamir [4, 5], and Kocher, Jaffe, and Jun [47]), we will resort to other techniques to prevent subliminal channels. The improved techniques originate from Brands [10, 11, 13].

Secure integration of smartcards

Our first design goal is to make sure that Alice cannot show a Digital Credential without the assistance of the smartcard that was issued to her. Hereto the *CA* encodes into each of Alice's Digital Credentials a random attribute, say x_1, that is known only to her smartcard. Since the ability to compute a signed proof implies knowledge of a secret key corresponding to the Digital Credential public key, this ensures that Alice needs her smartcard's assistance.

Alice's smartcard should be able to assist her without revealing x_1, since that would enable Alice to reuse the Digital Credential without the smartcard's assistance. To see how this can be achieved, consider the payment protocol in Figure 11. Since Alice does not know x_1, she cannot compute $r_2 := cx_1 + w_2 \bmod q$ without the assistance of her smartcard. In Figure 10, Alice forms her initial witness a as $h^{w_1} g_1^{w_2}$. Therefore, if Alice's smartcard generates w_2 and provides $a_s := g_1^{w_2}$ to her, then Alice can form $a := h^{w_1} a_s$, and her smartcard can compute r_2 on her behalf. Note that Alice needs to know $h_s := g_1^{x_1}$ in order to verify the *CA*'s response, r_0, as well as the smartcard's response, r_s; to verify the latter, Alice checks that $h_s^c a_s = g_1^{r_s}$. We will refer to x_1 as the smartcard's secret key, and to h_s as its "public" key.

Looking at the interaction between Alice and her smartcard as a whole, we see that Alice's smartcard in effect is proving knowledge of x_1 to her by means of the protocol of Figure 2. Since this protocol does not reveal x_1, Alice can get around her smartcard only by physically breaking its tamper-resistance and extracting x_1. Extracting x_1 would enable her to double-spend her coins, but then our one-show technique kicks in: the *CA* can trace Alice, because it can associate x_1 with Alice. In other words, our smartcard-enhanced electronic cash system offers two levels of security against double-spending. This is of great importance, since affordable mass consumer devices will likely never be fully tamper-proof.

The bulk of the workload for the smartcard can be shifted to the *CA*, by having the latter instead of the former provide a_s to Alice during the issuing protocol execution. Hereto the *CA* and Alice's smartcard must share the list of random numbers that the smartcard will deploy. In practice these numbers may be generated in a pseudorandom fashion from a seed value. See Brands [15, Section 6.5.1] for details.

Privacy protection

This simple construction lays the foundation for Alice's privacy. Note, for example, that Alice's smartcard does not need to know the Digital Credential public key, the attributes $(x_2, \ldots, x_l)$, the number of encoded attributes, the formula to be demonstrated, or Bob's message. However, the construction does not protect against outflow and inflow. Alice's smartcard can encode its identifier or another message in the "random" number w_2, possibly encrypted under a key of the *CA*. The *CA* can deduct the smartcard's message from (r_2, c) in Bob's transcript by trying all possible smartcard secret keys, x_1, and checking for a redundancy pattern in the candidate messages. Likewise, Bob in the showing protocol can cause inflow through its challenge, c, since at least the nonce is under his control.

To prevent inflow and outflow, we have Alice randomize on the fly any data from and to her smartcard that is not fully under her control. In particular, Alice randomizes Bob's challenge and her smartcard's response by adding random numbers before passing them on. She can adjust for these random numbers by multiplying appropriate powers of g_1 and h_s into her initial witness, a. Application of this technique to the simple electronic cash system depicted in Figures 10 and 11 results in the smartcard-enhanced protocols depicted in Figures 16 and 17. Note that Alice used the blinding factors α_4 and α_5 to randomize c and r_s, respectively.

$$
\begin{array}{lcccl}
\boxed{\text{smartcard}} & & \boxed{\text{Alice}} & & \boxed{\text{Bank}} \\
w_2 \in_{\mathcal{R}} \mathbb{Z}_q & & & & \\
a_s := g_1^{w_2} & & & & \\
 & \xrightarrow{a_s} & & & \\
 & & & & w_0 \in_{\mathcal{R}} \mathbb{Z}_q \\
 & & & & a_0 := g_0^{w_0} \\
 & & & \xleftarrow{a_0} & \\
 & & \alpha_1 \in_{\mathcal{R}} \mathbb{Z}_q^* & & \\
 & & \alpha_2, \alpha_3, w_1, \alpha_4, \alpha_5 \in_{\mathcal{R}} \mathbb{Z}_q & & \\
 & & h := (h_s g_2^{x_2} h_0)^{\alpha_1} & & \\
 & & a := a_s h_s^{\alpha_4} h^{w_1} g_1^{\alpha_5} & & \\
 & & c_0' := \mathcal{H}(h, a, a_0 g_0^{\alpha_2} (h_s g_2^{x_2} h_0)^{\alpha_3}) & & \\
 & & c_0 := c_0' - \alpha_2 \bmod q & & \\
 & & & \xrightarrow{c_0} & \\
 & & & & r_0 := (w_0 - c_0)/(x_0 + x_1 y_1 + x_2 y_2) \bmod q \\
 & & & \xleftarrow{r_0} & \\
 & & g_0^{c_0} (h_s g_2^{x_2} h_0)^{r_0} \stackrel{?}{=} a_0 & & \\
 & & r_0' := (r_0 + \alpha_3)/\alpha_1 \bmod q & &
\end{array}
$$

Figure 16: Smartcard-Enhanced Withdrawal Protocol.

It is easy to see that Alice's smartcard can leak at most one bit via the halting channel, the computation error channel, and the timing channel. This is the best result that can be achieved, since Alice's smartcard can always cause a protocol execution to abort by not providing its response. In practice the maximum leakage is much less than one bit, since an incorrect response or a protocol abortion could also be caused by incorrect behavior by Alice or by a communication failure. In fact, any attempt by Alice's smartcard to leak bits through the halting channel or the computation error channel is futile, since Bob and the *CA* cannot distinguish between a fraud by Alice and her smartcard attempting to leak one bit. Therefore, in practice Alice can omit the verification of both r_s and r_0.

This technique to prevent inflow and outflow is highly preferable over that of Chaum [33], which can only detect inflow and outflow after the fact with low probability. It is also preferable

$$
\begin{array}{ccccc}
\boxed{\text{smartcard}} & & \boxed{\text{Alice}} & & \boxed{\text{Bob}} \\
 & & & & m := \text{nonce}||\text{account}||\ldots \\
 & & & \xleftarrow{\qquad\qquad m \qquad\qquad} & \\
 & & c := \mathcal{H}(h, a, m) & & \\
 & & c_s := c + \alpha_4 \bmod q & & \\
 & \xleftarrow{\quad c_s \quad} & & & \\
r_s := c_s x_1 + w_2 \bmod q & & & & \\
 & \xrightarrow{\quad r_s \quad} & & & \\
 & & a_s \stackrel{?}{=} g_1^{r_s} h_s^{-c_s} & & \\
 & & r_1 := -c/\alpha_1 + w_1 \bmod q & & \\
 & & r_2 := r_s + \alpha_5 \bmod q & & \\
 & & r_3 := -c x_3 + w_3 \bmod q & & \\
 & & & \xrightarrow{(h, a), (c_0', r_0'), x_2, (r_1, r_2)} & \\
 & & & & c_0' \stackrel{?}{=} \mathcal{H}(h, a, g_0^{c_0'} h^{r_0'}) \\
 & & & & c := \mathcal{H}(h, a, m) \\
 & & & & h^{-r_1} a \stackrel{?}{=} g_1^{r_2} g_2^{-x_2 c} h_0^{-c}
\end{array}
$$

Figure 17: Smartcard-Enhanced Payment Protocol.

over Chaum's other outflow prevention technique [28, 35] (also applied by Cramer and Pedersen [39]), which in our situation would proceed by letting Alice and her smartcard form a in a mutually random manner by means of a coin flipping protocol; amongst others, this would seriously degrade the communication and computation complexity.

It can be shown that the view of Alice's smartcard is statistically independent from the views of the *CA* and Bob. Therefore, physical access to the contents of Alice's smartcard does not help the *CA* and Bob to retroactively trace and link her showing protocol executions; all that they can learn is what Alice willingly discloses. The importance of preventing mutually known information that is statistically correlated (other than inflow and outflow) was first argued by Cramer and Pedersen [39], who also modified Chaum's protocols to prevent such so-called common coin flips; their technique does not overcome the security and efficiency drawbacks of Chaum's techniques, though.

As shown in Brands [15, Sections 6.3.1., 6.4.1, and 6.4.2], these smartcard techniques can be made to work in general, regardless of the property of $(x_2, \ldots, x_l)$ that Alice demonstrates to Bob. The resulting smartcard-enhanced protocols encompass the software-only protocols as a self-contained subset. This has the advantage that the security protections of the software-only system apply in the (presumably hypothetical) case that the tamper-resistance of a large number of smartcards is compromised.

In many applications Alice's smartcard should be able to make informed decisions as to whether to assist her in showing a Digital Credential. That is, Alice should be unable to hide from her smartcard certain information pertinent to the showing protocol, such as some of the attributes in her Digital Credential, the property that she intends to demonstrate, or Bob's message. For example, if Digital Credentials represent electronic cheques spendable up to a ceiling amount, and Alice's smartcard uses an internal balance to represent the amount of cash held by Alice, then Alice should be unable to lie to her smartcard about the amounts for which she makes out her cheques; after all, her smartcard must deduct these from its internal balance. Brands [15, Section 6.4.4.] describes a general technique for tuning the smartcard-enhanced protocols to accommodate any degree of privacy desired. The basic idea is to let Alice provide the preimage to c to her smartcard, so that the smartcard can compute c by itself. In Figure 17, Alice would provide (h, a, m). The effect is that Alice cannot provide incorrect information to her smartcard, because that would result in a response r_s to an incorrect challenge. By defining c as an arbitrary structure of nested hashes of data fields, the smartcard can be prevented from learning more than than strictly needed.

9 Conclusion

In a computerized world, privacy can be adequately protected only by building it into the architecture of communication and transaction systems. To ensure that users of the Internet and other electronic networks cannot be traced at the data transport layer, anonymous communication channels are needed. In applications where individuals must prove that they are authorized to perform certain actions, anonymous communication is not enough. Namely, individuals may be identifiable at the application layer, through their use of electronic passports and other digital proofs of authorization.

Digital Credentials were designed to mimic the privacy of cinema tickets, stamps, subway tokens, and other non-identifiable authorization proofs. Digital Credentials are much more efficient, secure, and powerful than their tangible counterparts, and can be used to design a virtually unlimited number of applications. They guarantee privacy in the strongest possible sense: even if all verifiers, smartcards, and *CA*s conspire in an active attack, jointly establish secret information in a preparatory phase, and have unlimited computing power, they cannot learn more than what can be inferred from the attribute properties that Digital Credential holders voluntarily disclose.

This introductory paper does not cover all the techniques of Brands [15]. Amongst others, the book also describes techniques based on the RSA problem, and describes how to anonymously update a Digital Credential, how to show a Digital Credential in zero-knowledge, how to design a two-move issuing protocol, how to cope with broken connections and self-revocation, how to protect the *CA*'s secret key, and so on.

The reader should be aware that the primary design goal of Brands [15] has been practicality, so as to make Digital Credentials an appealing alternative to identity certificates and other privacy-invading approaches. In light of this, the Digital Credential protocols were carefully crafted to meet multiple security objectives at the same time. For example, Alice's signed proof to Bob serves not only to prevent replay, it is also exploited to prove the attribute property she discloses to Bob, to enable the *CA* to trace limited-show Digital Credentials that are shown too often, to ensure that Alice cannot show her Digital Credentials without the help of a smartcard, to enable Bob to hide from the *CA* a part of the disclosed attribute property, to avoid simulation of the *CA*'s signature (see Brands [15, Section 2.6.2]), and to avoid a delegation attack (see Brands [15, Section 5.1.2]). The presentation in this introductory paper, however, glosses over many of the security subtleties and dependencies. Consequently, the compactness of the protocols may lead one to believe that Digital Credentials have very simple security properties, but the reality is that seemingly innocent changes (e.g., for optimization purposes) may have far-reaching security implications.

References

[1] Ross Anderson and Markus Kuhn. Tamper resistance - a cautionary note. In *Second USENIX Workshop on Electronic Commerce*, pages 1–11, Oakland, California, November 1996. USENIX Association. ISBN 1-880446-83-9.

[2] Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols, 5th International Workshop, Paris, France*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer-Verlag, April 1997.

[3] Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In Walter Fumy, editor, *Advances in Cryptology–EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 280–305. Springer-Verlag, 1997.

[4] Eli Biham and Adi Shamir. The next stage of differential fault analysis: How to break completely unknown cryptosystems. Distributed on October 30th, 1996.

[5] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology–CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997.

[6] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In Walter Fumy, editor, *Advances in Cryptology–EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.

[7] J.N.E. Bos and D. Chaum. SmartCash: a practical electronic payment system. Technical Report CS-R9035, Centrum voor Wiskunde en Infomatica, August 1990.

[8] Joan Boyar, S.A. Kurtz, and M.W. Krentel. A discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology*, 2(2):63–76, 1990.

[9] Stefan Brands. Cryptographic methods for demonstrating satisfiable formulas from propositional logic. Patent PCT/NL96/00413. Filed November 1995.

[10] Stefan Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, Centrum voor Wiskunde en Informatica, April 1993.

[11] Stefan Brands. Untraceable off-line cash in wallet with observers. In Douglas R. Stinson, editor, *Advances in Cryptology–CRYPTO '93*, volume 911, pages 302–318. Springer-Verlag, 1994.

[12] Stefan Brands. Off-line electronic cash based on secret-key certificates. In R. Baeza-Yates, E. Goles, and P.V. Goblete, editors, *Proceedings of the Second International Symposium of Latin American Theoretical Informatics*, volume 911, pages 131–166. Springer-Verlag, 1995.

[13] Stefan Brands. Privacy-protected transfer of electronic information. U.S. Patent ser. no. 5,604,805, February 1997. Filed August 1993.

[14] Stefan Brands. Secret-key certificates. U.S. Patent ser. no. 5,606,617, February 1997. Filed October 1994.

[15] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, Cambridge, Massachusetts, August 2000. ISBN 0-262-02491-8.

[16] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[17] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burton S. Kaliski Jr., editor, *Advances in Cryptology–CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 1997.

[18] CCITT. Recommendation X.501: The directory–models, 1988.

[19] D. Chaum. Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology–AUSCRYPT '90*, volume 453 of *Lecture Notes in Computer Science*, pages 246–264. Springer-Verlag, 1990.

[20] D. Chaum. Achieving electronic privacy. *Scientific American*, 267(2):96–101, August 1992.

[21] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Advances in Cryptology–CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer-Verlag, 1988.

[22] David Chaum. Blind signatures for untraceable payments. In R.L. Rivest, A. Sherman, and D. Chaum, editors, *Advances in Cryptology–CRYPTO '82*, pages 199–203. Plenum Press, 1983.

[23] David Chaum. Blind signature system. In D. Chaum, editor, *Advances in Cryptology–CRYPTO '83*, page 153, New York, 1984. Plenum Press.

[24] David Chaum. Security without identification: Transaction systems to make Big Brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[25] David Chaum. Blind signature systems. U.S. Patent ser. no. 4,759,063, July 1988. Filed August 1983.

[26] David Chaum. Blind unanticipated signature systems. U.S. Patent ser. no. 4,759,064, July 1988. Filed October 1985.

[27] David Chaum. Privacy protected payments: Unconditional payer and/or payee untraceability. In D. Chaum and I. Schaumüller-Bichl, editors, *SMART CARD 2000*, pages 69–93. Elsevier Science Publishers B.V. (North-Holland), 1989.

[28] David Chaum. Card-computer moderated systems. U.S. Patent ser. no. 4,926,480, May 1990. Filed May 1988.

[29] David Chaum. One-show blind signature systems. U.S. Patent ser. no. 4,987,593, January 1991. Filed April 1990. Continuation of abandoned application Ser. No. 07/168,802, filed March 1988.

[30] David Chaum. Selected-exponent signature systems. U.S. Patent ser. no. 4,996,711, February 1991. Filed June 1989.

[31] David Chaum. Unpredictable blind signature systems. U.S. Patent ser. no. 4,991,210, February 1991. Filed May 1989.

[32] David Chaum. Zero-knowledge undeniable signatures. In I.B. Damgård, editor, *Advances in Cryptology–EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer-Verlag, 1991.

[33] David Chaum. Optionally moderated transaction systems. U.S. Patent ser. no. 5,276,736, January 1994. Filed July 1992.

[34] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In A.M. Odlyzko, editor, *Advances in Cryptology–CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 118–168. Springer-Verlag, 1987.

[35] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology–CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1992.

[36] David Chaum and Hans van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology–CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216, 1990.

[37] David Chaum, Eugène van Heijst, and Birgit Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. Technical report, University of Karlsruhe, February 1991. Interner Bericht 1/91.

[38] David Chaum, Eugène van Heijst, and Birgit Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In J. Feigenbaum, editor, *Advances in Cryptology–CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag, 1992.

[39] R.J.F. Cramer and T.P. Pedersen. Improved privacy in wallets with observers. In Tor Helleseth, editor, *Advances in Cryptology–EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 329–343. Springer-Verlag, 1994.

[40] Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In Carl Pomerance, editor, *Advances in Cryptology–CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 16–20. Springer-Verlag, 1988.

[41] Cynthia Dwork, Jeffrey Lotspiech, and Moni Naor. Digital signets: Self-enforcing protection of digital information. In *Proc.* 28*th ACM Symp. on Theory of Computing*. ACM Press, 1996.

[42] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proc.* 22*nd ACM Symp. on Theory of Computing*, pages 416–426, May 1990.

[43] Uriel Feige and Adi Shamir. Zero-knowledge proofs of knowledge in two rounds. In G. Brassard, editor, *Advances in Cryptology–CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544. Springer-Verlag, 1990.

[44] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, *Advances in Cryptology–CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.

[45] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[46] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In N. Koblitz, editor, *Advances in Cryptology–CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[47] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology–CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[48] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology–CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer-Verlag, 1992.

[49] Torben Pryds Pedersen. *Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem*. PhD thesis, Aarhus University, March 1992. DAIMI PB–388.

[50] David Pointcheval. *Les Preuves de Connaissance et leurs Preuves de Sécurité*. PhD thesis, University of Caen, December 1996.

[51] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology–EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.

[52] Claus P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[53] Berry Schoenmakers. An efficient electronic payment system withstanding parallel attacks. Technical Report CS-R9522, Centrum voor Wiskunde en Informatica, March 1995.

[54] Gustavus J. Simmons. The Prisoners' Problem and the subliminal channel. In D. Chaum, editor, *Advances in Cryptology–CRYPTO '83*, pages 51–67. Plenum Press, 1984.

[55] Gustavus J. Simmons. The subliminal channel and digital signature. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology–EUROCRYPT '84*, volume 209 of *Lecture Notes in Computer Science*, pages 364–378. Springer-Verlag, 1985.

[56] E. van Heijst and T.P. Pedersen. How to make efficient fail-stop signatures. In R.A. Rueppel, editor, *Advances in Cryptology–EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 366–377. Springer-Verlag, 1993.