Anonymous Credentials Light

Foteini Baldimtsi
Brown University
foteini@cs.brown.edu

Anna Lysyanskaya
Brown University
anna@cs.brown.edu

ABSTRACT

We define and propose an efficient and provably secure construction of blind signatures with attributes. Prior notions of blind signatures did not yield themselves to the construction of anonymous credential systems, not even if we drop the unlinkability requirement of anonymous credentials. Our new notion in contrast is a convenient building block for anonymous credential systems. The construction we propose is efficient: it requires just a few exponentiations in a prime-order group in which the decisional Diffie-Hellman problem is hard. Thus, for the first time, we give a provably secure construction of anonymous credentials that can work in the elliptic group setting without bilinear pairings and is based on the DDH assumption. In contrast, prior provably secure constructions were based on the RSA group or on groups with pairings, which made them prohibitively inefficient for mobile devices, RFID cards and smartcards. The only prior efficient construction that could work in such elliptic curve groups, due to Brands, does not have a proof of security.

Categories and Subject Descriptors

D.4.6 [Security and Protection]: [Cryptographic controls, Authentication]

Keywords

Anonymous credentials; attributes; blind signatures; lightweight devices; private identity management.

1. INTRODUCTION

Anonymous credentials, envisioned by David Chaum [17], and first fully realized by Camenisch and Lysyanskaya [13], have proved to be a centrally important building block in privacy-minded identity management systems currently under development [43, 35, 16]. They allow users to prove possession of credentials without revealing any other information about themselves; when such a proof is carried out, it cannot be linked to previous uses of the same credential, or to any other identifying information about the user. Additionally, they give the users the ability to privately obtain credentials. The reason that they have become so popular is that they strictly adhere to data minimization principles [30]: no electronic transaction should require its participants to needlessly reveal private information. These principles are at the core of European privacy standards [37, 36] and also the National Strategy for Trusted Identities in Cyberspace (NSTIC) published by the U.S. government [41].

Efficiency considerations are important when deploying these algorithms in practice. While on a PC, the Camenisch-Lysyanskaya credentials [13] and follow-up work [5, 10, 11, 14] take only a fraction of a second of CPU time, in mobile devices, smartcards and RFID cards, they are not yet as practical as we may wish for certain applications. The issue is that all existing anonymous credential systems use either the RSA group, or groups that admit bilinear pairings, and the security parameter choice needed to provide security under the required assumptions in these groups makes these systems too expensive for smartcards and mobile devices (i.e. original CL credentials require an RSA group with bitlength 2048 or a pairing with 128 bits of security). Bichsel et. al. [6] show an efficient implementation of CL credentials on a standard Java card, where they achieve an execution time for an RSA exponentiation for a base of 1984 bits and an exponent of 1024 bits of 1.3 seconds by using the hardware accelerator for RSA of the chosen card. As a result, more than 16 seconds are required in order to show one credential which is not practical for most applications. Moreover, hardware implementations have high development cost (compared to software) and are not easy to update.

From the efficiency point of view, therefore, the UProve credential system [35] (based on Brands’ work [8, 9], acquired and implemented by Microsoft) seems attractive. UProve does not allow unlinkable reuse of credentials: in order to unlinkably use a credential again, a user must get it reissued. Yet, these “linkable” credentials can serve as the basis for e-cash, where a user is not supposed to use a coin more than once, anyway. The UProve credential issuing protocol, as well as the algorithm for verifying a credential, are practical enough for the more advanced end of RFID technology, since they can work in any group where the discrete logarithm problem is hard (thus can be based on ECC), and require just a few exponentiations in this group. Software implementations of ECC multiplication are much faster than the equivalent RSA exponentiation [27] (hardware accelerators could also speed up both ECC and RSA implementa-
tions by an order of magnitude or more; today there exist RFID chips that perform ECC multiplication in 0.4 seconds [44]).

So perhaps one could use these “linkable” anonymous credentials and just have them reissued as many times as needed. However, UPprove is not provably secure! No proof of security has been given under any reasonable assumption; and in fact, recently, Baldimtsi and Lysyanskaya [4] showed that all known approaches for proving security in the random oracle model will fail for the Brands blind signature scheme on which UPprove is based, under essentially any assumption.

One might ask what happens if, instead of the Brands blind signature, one tries to construct linkable anonymous credentials from a provably secure blind signature scheme. Unfortunately, blind signatures as traditionally defined [40, 39, 29, 1] do not give any consideration to users’ attributes. In a blind signature, a user is free to choose whichever message he wants signed. In a credential scheme, a user should be able to get his particular attribute set signed by the signer. The signers can verify that what they are signing is the correct attribute set, even though they cannot see exactly what they are signing. For example, a signer might want to sign a user’s secret key, his age, and his citizenship information, and wants to be able to verify that this is indeed what she is signing. Attributes are, in fact, essential: without them, it is impossible to tie a credential to a particular user’s identity, and so users would be able to just pool each others’ credentials as desired, instead of tying them to a particular secret key. Moreover, giving the user the ability to choose any message means that the content of the signed messages has not been vetted by the signer at all.

Thus, a better question is: how do we extend the notion of a blind signature so that it can efficiently accommodate anonymous credentials — if not full anonymous credentials a-la Camenisch and Lysyanskaya, then at least linkable lightweight anonymous credentials a-la Brands?

**Blind Signature with Attributes.** In this paper, for the first time, we define blind signatures with attributes. In such a blind signature scheme, the signer and the user both get as input a cryptographic commitment $\tilde{C}$ to the user’s attributes; this way, the user can prove that the commitment contains the correct attributes via a separate zero-knowledge proof. As output, the user obtains another, unlinkable, commitment $\tilde{C}$ to the same attributes, and a signature on this commitment and a message of the user’s choice. **Blindness** ensures that, even upon seeing two signatures obtained this way on commitments of his own choice, the signer cannot link a signature to its issuing. **Unforgeability** ensures that a user cannot produce more signatures than he was issued, and also that the multiset of openings to the input commitments is the same as the multiset of openings to the output commitments.

**From Blind Signatures with Attributes to Anonymous Credentials.** It is easy to see that this is the right building block for linkable anonymous credentials: a user with a particular set of attributes can form a commitment $\tilde{C}$ to these attributes, prove in zero-knowledge that he has committed to the correct attributes, and then obtain a credential by running a blind signature with attributes on input this commitment. He can then prove that he has a credential with the desired attributes by revealing his signature and the output commitment $\tilde{C}$, and proving in zero knowledge that $\tilde{C}$ corresponds to the needed attributes.

For example, suppose that we allow users to obtain and anonymously show age credentials. Then Alice will form a commitment $C$ to her secret key $sk$ and her age $age$, prove to the signer, who serves as an identity validator, that she has committed to $sk$ that corresponds to her $pk$ and to the correct $age$, and run a blind signature with attributes protocol to obtain a fresh commitment $\tilde{C}$ on the same $sk$ and $age$, and a signature $\sigma$. Then, when she needs to prove her age, she can reveal $(\tilde{C}, \sigma)$ and prove that the age inside commitment $\tilde{C}$ allows her entry to some age-restricted resource, for example a video store that requires viewers to be over 13. If she wants to do it again, she needs to run the blind signature with attributes protocol with the signer again. She can further anonymously obtain credentials that are connected to her identity: let’s say that the video store wants to give her a credential for a free movie; she can obtain it by running a blind signature with attributes with the video store. She will form her input commitment $C'$ by committing to the same $sk$ and $age$ as in $C$ and proving to the video store that she did so (by proving equality of committed values); once she runs the protocol with the video store, she receives an output $(\tilde{C}', \sigma)$, which is a credential from the video store (and not from the original signer) on her $(sk, age)$ — even though the video store never saw the public key at all. More interestingly, we can require the free movie to be a single-use credential, by additional clever use of attributes (see Section 4), so that Alice can be traced if she tries to get two free movies using the same single-use credential more than once. Thus we see that blind signatures with attributes are the right building block for linkable anonymous credentials.

**Construction.** In addition to the definition, we solve the open problem of constructing anonymous credentials based on the DDH assumption. We give a lightweight construction of blind signatures with attributes; because of the connection to single use anonymous credentials we call our construction “anonymous credentials light” (ACL). In the random-oracle model, our construction is unlinkable under the decisional Diffie-Hellman assumption, and unforgeable under the discrete-logarithm assumption for sequential composition (we leave extending it to concurrent self-composition as an open problem). Our construction is inspired by the Abe blind signature [1] in which, unlike other provably secure blind signatures [7, 12, 34, 40, 22], blinding still preserves some structural elements into which attributes can be embedded. Unfortunately Abe’s construction, as it was first presented, had a faulty proof of security as it was later found by Abe and Okkubo [33]. The original proof was in the RO model for concurrent composition but it turned out to only be valid for an adversary with overwhelming success probability so they ended up presenting a new proof in the generic model. In the other hand, we manage to proof our scheme in the RO model by limiting our attention to sequential composition only. Moreover, our analysis indicates that Abe’s scheme may be provable under DDH in the RO model for sequential composition.

**Related Work and Comparisons.** Let’s now take a closer look in our ACL construction and compare it with other well known blind signature and credential schemes. For our comparisons we focus on the efficiency of signing (which is analogous to the credential issuing), the cost of
<table>
<thead>
<tr>
<th>Brands</th>
<th>Abe</th>
<th>ACL Scheme</th>
<th>CL credentials [13]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>Signing</td>
<td>Verification</td>
<td>Sign. size</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5+6p</td>
<td>12</td>
</tr>
</tbody>
</table>

1. Signer, 2. User, 3. In number of exponentiations
4. Non-comparable, 5. It can be used multiple times
6. Generic group model

Table 1: Comparison of anonymous credentials

verification of the signature/credential, the size of the signature/credential, the security of the schemes i.e. blindness and unforgeability and whether the schemes support attributes which is obviously not going to be the case for a “traditional” blind signature scheme.

The ACL construction consists of three phases: registration, preparation and validation. Registration is the most expensive phase, but it need not be repeated for the same user/signer pair should the user need his credential reissued (as long as his attribute set is the same). Preparations and validation require the signer to perform 7 exponentiations, while the user performs 13 verification requires 8 exponentiations. These numbers are essentially the same as Abe’s blind signature; Brands’ blind signature at the heart of UProve requires essentially the same number of exponentiations for signature issue, and 4 exponentiations fewer for verification. Thus, we get comparable efficiency to the most efficient examples of protocols that either lack essential features (such as previously provable secure blind signatures) or provable security (such as UProve). We also compare ourselves to the Camenisch-Lysyanskaya [13] which is significantly less efficient during issuing due to the use of pairings or RSA group but is a multi-use credential, thus we denote the signature/credential size to be close to zero since it can be used multiple times. See Table 1 for the comparisons.

A related work is due to Guajardo et al. [25] where they proposed “encrypted anonymous credentials”. The idea is that an issuer can certify encrypted attributes in such a way that none of the involved parties, including the user, can learn the value of the attributes. This is an interesting application but cannot be used as a building block so it is incomparable to our work. Moreover, their construction [25] is based on Brands scheme which as mentioned above cannot be proven one-more unforgeable [4].

Recommended parameter setting. For 128 bit security we recommend using a group of 576 bits\(^1\). In order to achieve an equivalent security level with CL credentials keys of 3072 bits would be required.

Implementation. In an independent work [28] we have implemented the ACL scheme using an NFC enabled smartphone: the BlackBerry Bold 9000. The implementation has been tailored for payments in transportation systems where our ACL construction is being used as an e-cash scheme (which, as we mentioned above, is a direct application of “single-use” credentials). Our implementation is mostly a proof of concept and we used a 160 bit elliptic curve group.

Briefly, the signing/issuing takes a total of 300 milliseconds including terminal, communication and smartphone execution time, while spending (i.e. verification of the signature) takes about 380 milliseconds when 2 attributes are revealed which would be less if no attributes were revealed.

2. PRELIMINARIES

2.1 Basic Assumptions

The following two standard cryptographic assumptions are useful in the paper.

Definition 1. (Discrete Logarithm Assumption) Let \( k \) be the security parameter. Let \( G_\ell \) be a set of groups such that \( G_\ell \) is a group of order \( q \) (k-bit prime) and generator \( g \). Then, for every polynomial time algorithm \( A \) it holds that:

\[
Pr[h \leftarrow G; x \leftarrow A(h) : x = log_g h] \leq \nu(k)
\]

where \( \nu(k) \) is a negligible function.

Definition 2. (Decisional Diffie-Hellman Assumption) Let \( DH \) be an instance generator that, on input the security parameter \( k \) outputs \( (G, q, g, A, B, C) \) where \( G \) is a cyclic group of order \( q \) (k-bit prime) with generator \( g \), and, for randomly chosen \( a, b \in \mathbb{Z}_q \), \( A = g^a \), \( B = g^b \) and \( C = g^{ab} \). Let \( Rand \) be an instance generator that, on input \( k \) outputs \( (G, q, g, A, B, C) \) where \( g, g, A \) and \( B \) are as above, while \( C = g^c \) for a random \( c \in \mathbb{Z}_q \). We assume that \( DH \) and \( Rand \) are computationally indistinguishable.

2.2 Witness Relations and Interactive Proofs

A witness relation for a language \( L \in \mathcal{NP} \) is defined as:

Definition 3. (Witness relation [23]) A witness relation for a language \( L \in \mathcal{NP} \) is a binary relation \( R_L \) that is polynomially bounded, i.e., \((h, x) \in R_L \) implies \( |x| \leq \text{poly}(|h|) \), is polynomial-time-recognizable and characterizes \( L \) by

\[
L = \{ x : \exists h \text{ s.t. } (h, x) \in R_L \}.
\]

For \( h \in L \), any \( x \) satisfying \((h, x) \in R_L \) is called a witness (for the membership of \( h \in L \)). By \( R_L(h) \) we denote the set of witnesses for the membership \( h \in L \); that is, \( R_L(h) = \{ x : (h, x) \in R_L \} \).

Definition 4. (Interactive Proof System [24]) An interactive proof system with soundness error \( s \in [0, 1] \) for a language \( L \) with witness relation \( R_L \) is a pair of algorithms \((P, V)\) where \( v \) is probabilistic polynomial time and the following properties hold:

1. Completeness. For every \( h \in L \) and every \( x \in R_L(h) \),

\[
Pr[(P(x), V(h)] = 1.
\]

2. \( s \)-Soundness. For every \( h \notin L \), every \( z \in \{0, 1\}^* \) and every interactive algorithm \( P^* \)

\[
Pr[(P^*(z), V(h)] = 0 \geq 1 - s.
\]

A useful property in the above setting would be if the verifier \( V \) wouldn’t learn anything useful from \( P \) about the witness \( x \) besides the fact that \( P \) knows \( x \). This property is called zero knowledge. If soundness error \( s \) is negligible, then this interactive proof system has strong soundness.
An interactive proof system \((P, V)\) for a language \(L\) is said to be honest verifiable zero knowledge if there exists a probabilistic polynomial time algorithm \(S\) (the Simulator) such that for all \(h \in L:\)

\[
\text{view}_V \left( (P, V) \right)(h) \approx S(h),
\]

where \(\text{view}_V\) is the view of the honest verifier \(V\) of the interaction between \(V\) and \(P\) on input \(h\).

### 2.3 \(\Sigma\)-Protocols

\(\Sigma\)-protocols are a class of interactive proofs where the Prover and the Verifier \((P, V)\) have a common input \(h\) and \(P\) proves in zero-knowledge that he knows a value \(x\) such that \((h, x) \in R_L\). Their main characteristic is that they have exactly 3 rounds of the following type: (1) \(P\) sends a message \(a\) to \(V\), (2) \(V\) responds with a random challenge \(c\) chosen from a domain of size \(\Theta(k)\) and (3) \(P\) resents a reply \(r\). \(V\) decides whether to accept or not given the information he has seen: \((h, a, c, r)\). Formally:

**Definition 6. (\(\Sigma\)-Protocol)** A protocol \(P\) is said to be a \(\Sigma\)-protocol for a relation \(R_L\) if:

- \(P\) is of the above three rounds form, and if \((P, V)\) follow the protocol, the verifier always accepts.
- From any \(h\) and any pair of accepting conversations on input \(h, (a, c, r), (a', c', r')\) where \(c \neq c'\), one can efficiently compute \(x\) such that \((h, x) \in R_L\) (special soundness).
- There exists a polynomial-time simulator \(S\), which on input \(h\) and a random \(c\) outputs an accepting conversation of the form \((a, c, r)\), with the same probability distribution as conversations between the honest \(P, V\) on input \(h\) (special honest-verifier zero-knowledge).

An example of a \(\Sigma\)-protocol is the Schnorr Identification scheme [42]. Let \(G\) be a group of prime order \(q\) with generator \(g\), and let \(\mathbb{Z}_q\) denote the field of integers modulo \(q\). Schnorr’s identification scheme works as follows:

\[
\begin{align*}
\text{Prover}(q, g, h = g^x) & \quad \text{Verifier}(g, h, g) \\
y \leftarrow \mathbb{Z}_q, a = g^y & \quad a \rightarrow c \\
r = y + cx \mod q & \quad c \leftarrow \mathbb{Z}_q \\
g^r \equiv ah^c &
\end{align*}
\]

\(\Sigma\)-protocols are an essential building block for blind signatures and anonymous credentials. For example Brands’ [8] and Abe’s [1] schemes are based on a \(\Sigma\)-protocol, while CL anonymous credentials [13] uses ZK proofs which are based on \(\Sigma\)-protocols.

### 2.4 Fiat-Shamir Heuristic

Fiat and Shamir [21] proposed a method to transform any three-round interactive proof system with negligible soundness error, such as \(\Sigma\)-protocols, into a digital signature using a hash function, modeled as a random oracle.

To transform a three-round proof system into a signature scheme, one could, instead of a random \(c\), compute \(c = H(a, m)\), where \(H\) is a hash function. Then, the Fiat-Shamir transformation uses \(c\) to create a signature \(\sigma(m) = (a, r)\). Famous digital signatures that have been constructed from \(\Sigma\)-protocols using the Fiat-Shamir heuristic include Schnorr’s [42] and GQ signatures [26] and they have been proven secure in the RO model [39].

### 2.5 OR-proof

Let \(P\) be a \(\Sigma\)-protocol for a relation \(R_L\). \((h_0, h_1)\) be a common input to \((P, V)\) and \(P\) knows a \(x\) such that \((h_0, x) \in R_L\) for \(b \in \{0, 1\}\). An OR-proof protocol \(P_{OR}\) is a \(\Sigma\)-protocol for proving that either \((h_0, x) \in R_L\) or \((h_1, x) \in R_L\).[2]

The main idea of an OR-proof is that \(P\) will complete two \(\Sigma\) protocols \(\Sigma_0, \Sigma_1\), one for \(h_0\) and one for \(h_1\) in such a way that the verifier will not be able to tell for which of the two \(P\) knows the corresponding witness. For \(h_0\) the prover can actually follow the real \(\Sigma\) protocol while for \(h_1\) he will have to use a simulator \(M\) to create his answers. A \(P_{OR}\) protocol works as follows:

1. \(P\) computes the first message of \(\Sigma_b, a_b\), using \((h_b, x)\) as input. \(P\) randomly chooses \(c_{1-b}\) and runs the simulator \(M\) on input \((h_{1-b}, c_{1-b}, r_{1-b})\) and receives \((a_{1-b}, c_{1-b}, r_{1-b})\) as an output. Then, \(P\) sends \(a_0, a_1\) to \(V\).
2. \(V\) chooses a random string \(c\) and sends it to \(P\).
3. \(P\) sets \(c_b = c \oplus c_{1-b}\) and computes the answer \(r_b\) to challenge \(c_b\) using \(h_b, a_b, c_b, x\) as input. He sends \(c_0, c_1, r_0, r_1\) to \(V\).
4. \(V\) checks that \(c = c_0 \oplus c_1\) and that both \((a_0, c_0, r_0)\) and \((a_1, c_1, r_1)\) are accepting conversations.

Let \(R_{OR} = \left\{(x, (h_0, h_1)) | (h_b, x) \in R_L \text{ or } (h_1, x) \in R_L\right\}\).

Then:

**Theorem 1** ([20]). The \(P_{OR}\) protocol is a \(\Sigma\)-protocol for \(R_{OR}\). Moreover, for any verifier \(V^*\), the probability distribution of conversations between \(P\) and \(V^*\), where \(x\) is such that \((h_0, x) \in R_L\), is independent of \(b\).

### 2.6 Generalized Pedersen Commitment

A non-interactive commitment takes as input a message (or set of messages) \(m\) and randomness \(R\) and outputs a value that, on the one hand, reveals no information about the message but, on the other hand, it is hard to find a \((m', R')\) such that \(\text{Commit}(m; R) = \text{Commit}(m', R')\) but \(m \neq m'\) (see [31] for a standard definition and treatment). The Pedersen commitment is a commitment scheme based on the discrete logarithm assumption [38]. Here we give a generalized version that allows a commitment to a set of messages \((L_1, \ldots, L_n)\). The scheme is defined as follows:

1. **Setup**: On input the security parameter \(1^k\) and the maximum number of messages \(n\), pick a group \(G\) of prime order \(q = \Theta(2^k)\) with generators \(k, h_1, \ldots, h_n\).
2. **Commit** \((L_1, \ldots, L_n; R) = \prod_{i=1}^n h_i^{L_i}\); and \(L_i \in \mathbb{Z}_q\).

The Pedersen commitment scheme (constructed from a corresponding \(\Sigma\)-protocol) is information theoretically hiding and computationally binding. It is also important that there are efficient zero-knowledge proof protocols for proving that a commitment \(C\) is to a particular set of values; or to a set of values that satisfy a rich class of relations [8, 15, 19].

### 2.7 Combined Commitment Scheme

Let \(\text{Commit}_1\) be a commitment scheme that takes as input its parameters \(\text{params}_1\), \(n\) messages \((L_1, \ldots, L_n)\) to which we will refer to as attributes from now on and randomness \(R_1\)

[2] This is based on a more general result due to Cramer et. al. [18] where they present a protocol in which the prover demonstrates knowledge of the solution to some subset of \(n\) problem instances out of a collection of subsets.
and outputs a commitment $C_1$. Let $\text{Commit}_2$ be a commitment scheme that takes as input its parameters $\text{params}_2$, an attribute $L_0$ and randomness $R_2$ and outputs a commitment $C_2$. Suppose that another commitment scheme, $\text{Commit}$, is a combination of these two commitments; i.e., on input $C_1$ and $C_2$ it produces a commitment $C$ to the combined attributes $(L_0, L_1, \ldots, L_n)$ with combined randomness $R = R_1 + R_2$.

For example, this can be instantiated by a generalized Pedersen commitment scheme: the parameters for the combined scheme are generators $(h, h_0, \ldots, h_n)$, $\text{Commit}(L_1, \ldots, L_n; R_2) = (\prod_{i=0}^{n} h_i^{L_i}) h^R_2$, and a combined commitment can be obtained either by multiplying together the two component commitments, or by computing it from scratch as $\text{Commit}(L_0, L_1, \ldots, L_n; R) = (\prod_{i=0}^{n} h_i^{L_i}) h^R$.  

**2.8 Blinded Pedersen Commitment Scheme**

Here, we note that the Pedersen commitment scheme can be further extended. Let $(h, h_0, \ldots, h_n)$ be the parameters of the Pedersen commitment. Consider an additional parameter $z \in G$, where $z \neq 1$. Let $C = \text{Commit}(L_1, \ldots, L_n; R)$. Then the values $(z^\gamma, C^\gamma)$ can also be viewed as a commitment to the same $(L_1, \ldots, L_n)$ with randomness $(R, \gamma)$. Let us define a new commitment scheme, which we will call the blinded Pedersen commitment scheme: $\text{Commit}^B(L_1, \ldots, L_n; R; \gamma) = (z^\gamma, \text{Commit}(L_1, \ldots, L_n; R^\gamma))$, where $\text{Commit}$ is the Pedersen commitment. It is easy to see that this commitment is unconditionally hiding, same as Pedersen. It is also easy to see that it is binding: given $(z^\gamma, C^\gamma)$, $\gamma$ is uniquely defined, and therefore so is $C$, which is binding. Finally, using well-known $\Sigma$-protocols, it is easy to see that the same set of relations that can be proven about values inside a Pedersen commitment can be proven about values inside a blinded Pedersen commitment.

**3. DEFINITION OF SECURE BLIND SIGNATURES WITH ATTRIBUTES**

In this section, for the first time, we define what a blind signature scheme with attributes is and its security properties. For definitions of standard (without attributes) blind signatures please refer to [20].

**Definition 7. (Blind signatures with attributes)** Let $\text{Commit}(z; r)$ be a non-interactive commitment scheme whose parameters $\text{params}$ are generated by a trusted party $TP$ and let $n$ be the maximum number of messages the scheme can support. A blind signature scheme with $n$ attributes, for this commitment scheme, consists of three algorithms: $\text{KeyGen}, \text{BlindSign}, \text{Verify}$ where $\text{BlindSign}$ is a protocol between the Signer (S) and the User (U).

- $\text{KeyGen}(1^k, \text{params})$: is a probabilistic polynomial time key generation algorithm. It takes as input the security parameter $k$ and the public parameters of the commitment scheme and outputs $(pk, sk)$ where $pk$ is the signer’s public key and $sk$ is his secret key.
- $\text{BlindSign}$: is an interactive, probabilistic polynomial time protocol between $S$ and $U$. The common input is the Signer’s public key $pk$, the parameters of the commitment scheme $\text{params}$ and $C = \text{Commit}(L_1, \ldots, L_n; R)$ where $(L_1, \ldots, L_n)$ is the set of attributes and $R$ is some randomness. The Signer’s private input is $sk$ and User’s private input is $(L_1, \ldots, L_n, R)$ and the message, $m$, that he wishes to have signed. The User’s output in the protocol is a pair $(\tilde{R}, \sigma)$, where $\sigma = \sigma(m, \tilde{C})$ is the Signer’s signature on $(m, \tilde{C})$, and $\tilde{C} = \text{Commit}(L_1, \ldots, L_n; \tilde{R})$; the Signer’s output is “completed”.
- $\text{Verify}(PK, m, \tilde{C}, \sigma)$: is the signature verification algorithm; i.e. a deterministic polynomial time algorithm that gets as input the public key, the message, a commitment to the attributes and the blind signature on the message and attributes and checks the validity of the signature $\sigma$. If it is valid the algorithm outputs “1”, otherwise outputs “0”.

A blind signature scheme with attributes is secure if it is both blind and unforgeable. Blindness is defined in a similar way as in blind signature schemes without attributes: the Signer is unable to view the messages and the attributes he signs (protection for the User). A malicious Signer, $A$, cannot link a $(m, \tilde{C}, \sigma)$ tuple to any particular execution of the protocol, even if $A$ chooses $m, L_1, \ldots, L_n$.

**Definition 8. (Blindness for blind signatures with attributes)** Let $A$ be a malicious Signer and $b \in \{0, 1\}$ be a randomly chosen bit which is kept secret from $A$. $A$ will try to guess the value $b$ by performing the following steps:

1. $(pk, sk) \leftarrow \text{KeyGen}(1^k)$
2. $(m_0, m_1, L_0, L_1, R_0, R_1) \leftarrow A(1^k, pk, sk)$ (i.e. $A$ produces two messages $(m_0, m_1)$, polynomial in $1^k$, and two attribute vectors $L_0, L_1$ with the corresponding randomness).
3. $A(1^k, pk, sk, m_0, m_1, L_0, L_1, R_0, R_1)$ engages in two parallel (and arbitrarily interleaved as desired by $A$) interactive protocols, the first with $U(pk, \{m_0, m_1, L_0, L_1, R_0\})$ and the second with $U(pk, \{m_1, L_0, L_1, R_1\})$.
4. $A$ only gets signatures and the corresponding blinded commitments for the instances that didn’t fail; for the ones that failed $A$ gets nothing. If neither of the User instances failed, then $A$ gets two signatures and the corresponding blinded commitments: $\sigma(m_0, \tilde{C}_0), \tilde{C}_1$ and $\sigma(m_1, \tilde{C}_1), \tilde{C}_1 \cdot b$. Otherwise, $A$ outputs $\bot$.
5. $A$ outputs a bit $b'$.

Then the probability, taken over the choice of $b$, over coin-flips of the key-generation algorithm, the coin-flips of $A$, and (private) coin-flips of both users (from step 3), that $b' = b$ is at most $\frac{1}{2} + \nu(k)$, where $\nu(k)$ is a negligible function.

We give the definition of one-more unforgeability for the sequential composition case.

**Definition 9. (Sequential one-more $(\ell, \ell + 1)$ unforgeability for blind signatures with attributes)** ($\text{KeyGen}, \text{BlindSign}, \text{Verify}$) is a one-more unforgeable blind signature scheme with respect to $\text{Commit}$ if $\forall$ probabilistic polynomial time adversaries $A$, the probability that $A$ wins in the following game is negligible:

1. $(pk, sk) \leftarrow \text{KeyGen}(1^k)$
2. $A(pk, \text{params})$ engages in polynomially many $(k)$ adaptive, sequential interactive protocols with polynomially many copies of the signer, where $A$ decides in
an adaptive fashion when to stop. For every execution, \( A \) forms a commitment \( C_i \) and picks a message \( m \), and invokes \( S(pk, sk, C_i) \). Let \( \ell \) be the number of executions, where the Signer output “completed” in the end of the protocol.

3. \( A \) outputs a collection \( \{(\tilde{C}_1, m_1, \sigma_1), \ldots, (\tilde{C}_j, m_j, \sigma_j)\} \) where \((\tilde{C}_i, m_i, \sigma_i)\) for \( 1 \leq i \leq j \) are all accepted by \( \text{Verify}(pk, \tilde{C}_i, m_i, \sigma_i) \), and all \((\tilde{C}_i, m_i)\)'s are distinct.

We say that \( A \) wins the game if either:

1. \( j > \ell \) (i.e. \( A \) outputs more \((\tilde{C}, m, \sigma)\) tuples than he received).
2. \( A \) opens the sets of commitments \( \{C_i\} \) and \( \{\tilde{C}_i\} \) and the resulting multisets do not match.

4. FROM BLIND SIGNATURES WITH ATTRIBUTES TO SINGLE-USE CREDENTIALS

Single-use anonymous credentials ensure that a credential cannot be used more than once: if used more than once, the user’s identity can be discovered, and the user can be penalized. This is a generalization of off-line electronic cash [17], in that such a credential can also contain attributes, while e-cash does not necessarily have any connection to a user’s identity. Here we construct single-use anonymous credentials from blind signatures with attributes using standard techniques [17, 8, 11].

Recall our definition of combined commitment schemes \((\text{Commit}_1, \text{Commit}_2, \text{Commit})\) from Section 2.7. Consider the following construction of a single-use credential with \( n \) attributes from a blind signature scheme with \( n+1 \) attributes. The common inputs to the User and Signer are the Signer’s public key for verifying signatures, and a commitment \( C_1 \) to the User’s attributes \((L_1, \ldots, L_n)\). We assume that the attribute \( L_1 \) contains the user’s identity, so that learning that attribute will allow one to trace the user. The Signer’s private input is its signing key, while the User’s private input is the message \( m \) that the user wants signed (\( m \) will serve as the serial number for this credential, so the user needs to have chosen it at random during the signing protocol) and the opening to \( C_1 \), \((l_1, \ldots, l_n; R_1)\). The commitment \( C_1 \) corresponds to this user’s identity and never changes; each time this user obtains a signature, they input the same \( C_1 \).

First, the User forms a commitment \( C_2 = \text{Commit}_2(L_0; R_2) \) for random \((L_0, R_2)\); this commitment is specific to this credential, and the value \( L_0 \) will be used in the double-spending equation, below. Next, the user submits \( C_2 \) to the signer and provides a zero-knowledge proof of knowledge of its opening.

Next, the User and the Signer carry out the blind signature protocol relative to the combined commitment scheme \text{Commit}, whereby the User’s output is a fresh commitment \( C \) to the values \((L_0, \ldots, L_n)\) and randomness \( R \), and the signature \( \sigma \) on \((m, C)\).

To use this credential to a verifier \( V \), the User reveals \((m, C, \sigma)\), obtains from \( V \) a challenge \( c \), and reveals the double-spending equation \( cl_1 + L_0 \) (the multiplication and addition is in whatever ring the committed values belong to) and whatever other attributes are needed for the transaction; the User then provides a zero-knowledge proof that these values were revealed correctly, relative to commitment \( C \).

To trace a user who spent a single-use credential twice, it is sufficient to examine two double-spending equations for the same credential. If double spending occurred, then the attribute \( L_1 \), which encodes the User’s identity, will be revealed.

The following theorem is stated informally because we omitted a formal definition of single-use anonymous credentials.

**Theorem 2.** Informal Let KeyGen, BlindSign, Verify be a blind signature scheme with \( n+1 \) attributes, associated with the commitment scheme Commit which is a combination of Commit1 and Commit2 as explained above. Then the above construction is a single-use credential scheme with \( n \) attributes.

5. OUR CONSTRUCTION: ACL

Let’s now describe our proposed construction of a blind signature scheme with attributes, called “Anonymous Credentials Light” (ACL). ACL is based on Abe’s blind signature scheme [1] which has been modified appropriately so that it will also allow the users to encode attributes in the signatures they get in a provably secure way. Abe’s scheme is a witness indistinguishable variant of the Schnorr signature scheme where the Signer owns a real public key \( y = g^x \) and a tag public key \( z = g^w \). The idea is that the signature can be issued only with the real secret key \( x \) but no one can distinguish which of the two (real or tag) secret keys was used (remember the OR-proof technique presented above).

Although the modification seems straightforward, proving the security of the resulting scheme turns out to be challenging as we will see later in the next section.

5.1 Setup

Let \( G \) be a group of prime order \( q \) and \( g \) be a generator of this group. Let \( \mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q \) be a hash function. We assume the existence of a trusted party, \( TP \), which chooses \( G, g \) and outputs params \( = (q, g, g, h, h_0, \ldots, h_n) \) where \( h, h_0, \ldots, h_n \in G \) and \( n \) is the maximum number of attributes a User can possibly have. We will use \((h_0, \ldots, h_n)\) as parameters of the combined Pedersen commitment scheme Commit.

The Signer picks his secret key \( x \in_R \mathbb{Z}_q \) and computes his real public key to be \( y = g^x \mod q \). The \( TP \) given \((G, g, g, h, y)\) outputs \( z \) which is the tag public key of the Signer (this can also be done using a hash function \( H_2 : \{0,1\}^* \rightarrow G \) that acts as a random oracle). The public key of the Signer is \((G, g, g, h, y, z)\), and the private key is \( x \). We will use \((z, h_0, \ldots, h_n)\) as parameters for the blinded Pedersen commitment scheme CommitLight.

5.2 Signature/ Credential Issuing

The signature issuing protocol will be described in three phases: registration, preparation and validation. The registration phase actually only needs to happen once for each user/set of attributes and also, preparation and validation phases can happen simultaneously (we choose to present them separately in order to describe the construction in a more modular way).
The signature issuing protocol is basically a \(POW\)-protocol for proving knowledge of one of the following:

- y-side: proof of knowledge \(x\) of \(y = g^x\)
- z-side: proof of knowledge \((w_1, w_2)\) of \(z_1 = g^{w_1}, \ z_2 = h^{w_2}\) (where \(z_1, z_2\) are the so called “one-time” tag keys created by the signer using input from the user).

Notice that the z-side witness in not known to the Signer, so the z-side proof will be done by simulation following the OR-proof technique paradigm described in Appendix 2.5.

5.2.1 Registration

The User’s input includes the system parameters \(\text{params}\), the signer’s real public key \(y\), the message \(m\) to be signed and \((L_1, \ldots, L_n, R)\) where \(L_1, \ldots, L_n\) is a set of attributes and \(R\) is some randomness. The Signer also gets as input the system parameters \(\text{params}\), a commitment \(C = \text{Commit}(L_1, \ldots, L_n; R)\) and his secret key \(x\). During registration the User and the Signer carry out a standard interactive zero-knowledge proof of knowledge protocol where the User creates a proof \(\pi_1\) to convince the Signer that he knows an opening of the commitment \(C\).

5.2.2 Preparation

The Signer prepares \(z_1\) and \(z_2\) (the z-side proof). He first picks \(\text{rand} \in \mathbb{Z}_n\) and creates the “one-time” tag keys: \(z_1 = C \cdot \text{rand}\) and \(z_2 = z/\text{rand}\). The Signer sends \(\text{rand}\) to the User in order to convince him that \(\log z_1\) is not known to him. The User first checks that \(\text{rand} \neq 0\) and then computes himself \(z_1 = C \cdot \text{rand}\) and then picks \(\gamma \in \mathbb{Z}_n^*\) and blinds \(z_1, z_2\) into \(z = z^\gamma, \ z_1 = z_1^\gamma, z_2 = z_2^\gamma\) so that \(\log z_1 = \log z_1^\gamma = z_1\gamma\) holds.

Finally, the User picks \(\tau \in \mathbb{Z}_n^*\) and computes \(\eta = z^\tau\) (this will serve as an element of an additional Schnorr signature that proves knowledge of \(\gamma\) such that \(z = z^\tau\)).

5.2.3 Validation

In this phase two \(\Sigma\) protocols are going to take place and combined according to the OR-proof. By \((a, c, r)\) we are going to denote the transcript of \(\Sigma_y\) and by \((a', c', r')\) we will denote the transcript of \(\Sigma_x\). The Signer will compute \(\Sigma_x\) himself and \(\Sigma_y\) in interaction with the User.

1. The Signer begins by creating \(a\) for \(\Sigma_y\) by picking a random \(u \in \mathbb{Z}_q\) and computing \(a = g^u\). Then, for \(\Sigma_x\), following the OR-proof he picks random \(c' \in \mathbb{Z}_q\) and \(r' = (r_1', r_2') \in \mathbb{Z}_q^2\). and sets \(a'_1 = g^{r_1'} z_1'\) and \(a'_2 = h^{r_2'} z_2'\). Finally he sends to the User \(a, a' = \{a_1, a_2\}\).

2. The User checks whether \(a, a'_1, a'_2 \in G\) and then picks \(\text{blinding factors } t_1, t_2, t_3, t_4, t_5 \in \mathbb{Z}_q\) and blinds a into \(\alpha = a g^{t_1} y^{t_2}\) and \(a_1, a_2\) into \(\alpha' = \alpha' y^{t_3} c_1\) and \(a_2' = a_2 h^{t_5} c_4\). We denote \(a' = \{a'_1, a'_2\}\). Then, computes \(e = \mathcal{H}(\zeta, \zeta', \alpha, \zeta, \gamma, m)\) where \(m\) is the message to be signed. The User sends to the Signer \(e = \langle e - t_2 - t_4 \rangle \mod q\).

3. The Signer, according to the OR-proof technique, computes \(e = c - c' \mod q\) and \(r = u - c x \mod q\). Then, sends \(\langle c, r, c', r' \rangle\) to the User.

4. Finally, the User “unblinds” the received values and gets: \(\rho = r + t_1 \mod q, \ \omega = e + t_2 \mod q, \ \rho_1 = r_1' + t_3 \mod q, \ \rho_2 = r_2' + t_5 \mod q, \ \omega' = c' + t_4 \mod q, \ \mu = \tau - \omega' \gamma \mod q\).

<table>
<thead>
<tr>
<th>Signer(params, x, C)</th>
<th>User(params, y, m, (L_1, ..., L_n, R))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Registration</strong></td>
<td>(C = h^{\mu} h^{\rho_1} h^{\rho_2} \cdots h^{\rho_n} \tau_1)</td>
</tr>
<tr>
<td><strong>Preparation</strong></td>
<td>(\text{rand} \in \mathbb{Z}_n) (z_1 = C \cdot \text{rand}) (z_2 = z/\text{rand}) (\gamma \in \mathbb{Z}_n^*) (\zeta = z^\gamma) (\zeta_1 = z_1^\gamma) (\zeta_2 = z_2^\gamma) (\eta = z^\tau)</td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td>(e = e - c' \mod q) (r = u - c x \mod q) (\rho = r + t_1 \mod q) (\omega = e + t_2 \mod q) (\rho_1 = r_1' + t_3 \mod q) (\rho_2 = r_2' + t_5 \mod q) (\omega' = c' + t_4 \mod q) (\mu = \tau - \omega' \gamma \mod q)</td>
</tr>
</tbody>
</table>

Table 2: Proposed ACL construction

A signature is a 8-tuple \(\sigma = (m, \zeta, \zeta', \rho, \omega, \omega')\) (where \(\zeta_1\) encodes the attributes of the User \((\zeta, \zeta')\) corresponds to \(C\), which is a blinded Pedersen commitment to \((L_1, \ldots, L_n)\) with randomness \((R, \gamma)\)).

In Table 2 we describe all the phases of our construction.

5.2.4 Verification

A signature tuple \((m, \zeta, \sigma)\) (where \(\zeta\) corresponds to \(C\)) verifies if \(\zeta \neq 1\) and \(\omega + \omega' = \mathcal{H}(\zeta, \zeta', \omega, \omega', \mu)\) mod \(q\).

5.3 Discussion

As briefly mentioned in the description of our construction there are certain “tricks” we can do to improve its efficiency. First of all notice that the \text{Registration} phase only needs to happen once for each User and a corresponding set of attributes. The Signer stores the attribute commitment \(C\) together with some identification information asked from the User (e.g. passport, ID). Then, it is sufficient if every time in the beginning of the \text{Signature Issuing} protocol the User identifies himself to the Signer in order to prove ownership of his account. Moreover, notice that \text{Preparation} and \text{Validation} phases can be combined and executed simultaneously. As a result, the whole signing protocol consists of three rounds.

Let’s now discuss the differences between our ACL construction and the blind signature protocol described by Abe [1]. The basic advantage of ACL is that allows for the encoding of users’ attributes in the signatures they receive from
the Signer. In order for this to happen we need the *Registration* phase during which the User commits to his set of attributes. Those attributes are then encoded in the signature in $\zeta$. The crucial difference from Abe’s approach is that $z_1$ can no longer be computed as the result of some hash function given a random input $\text{rnd}$. In ACL, $z_1$ needs to include the attribute commitment $C$ so $z_1 = Cg^{\text{rnd}}$ (we need the $g^{\text{rnd}}$ factor so that two different signature issuings with the same user cannot be linked to each other). This is the step where the blinded signature inherits some structure from the values the signer can see in the signing step, and allows the encoding of attributes. The fact that $z_1$ is not the result of a hash function $H'$ makes our security analysis different from Abe’s. We can no longer define $H'$ so that it will return the output we need, instead we will have to make use of ZK extractors. Unfortunately, we cannot take advantage of Abe’s analysis because it only applied to special types of adversaries [33]. Of independent interest, our analysis indicates that, for sequential composition, Abe’s blind signature may still be provably secure under DDH in the RO model.

The formal analysis follows in the next section.

6. PROOF OF SECURITY

The correctness of the scheme is straightforward, see our full version [3].

**Theorem 3** (Blindness). *The proposed scheme satisfies blindness under the Decisional Diffie-Hellman assumption in the random oracle model.*

**Proof.** We wish to show that, when interacting with a challenger as described in the definition of blindness, the adversary cannot guess which signature is which (i.e. the adversary should not be able to tell which signature corresponds to each instance and each attribute set). We distinguish between two types of signatures that a challenger might output: a signature $\sigma = (m, (\zeta, \zeta_1, \rho, \omega, \rho_1, \rho_2, \omega', \mu))$ is correct for a particular interaction with the signer if there exists a value $\gamma$ such that $\zeta = \zeta' = z_1$ where $z_1$ is the value corresponding to this interaction with the signer, and the signature verifies. A signature is fake if no such $\gamma$ exists. Note that it is easy for a challenger to generate both kinds of signatures: a correct signature can simply be output by correctly following the user’s side of the protocol. With control of the random oracle, a fake signature is computed as follows: first, the challenger picks $\zeta$ and $\zeta_1$ at random from $G$, and let $\zeta_2 = \zeta/\zeta_1$. Next, it picks random $\rho, \omega, \rho_1, \rho_2, \omega', \mu$ from $\mathbb{Z}_q$ and sets the random oracle such that $\omega + \omega' = H(\zeta, \zeta_1, g^\omega g'^\rho, g^{\zeta_2}g^{\zeta_2'}, h^{\rho_2}g^{\zeta_2'}, z^\omega g^{\omega'})$, which ensures that the fake signature verifies.

We will prove the theorem by a hybrid argument. Consider the following three games in which the adversarial signer engages in two signature issuing instances with a challenger who outputs two signatures:

1. **Real:** The challenger outputs two correct signatures, $\sigma_1, \sigma_2$, in random order by honestly following the protocol. A correct signature is one that is honestly generated by interacting with the signer. Thus, in a correct signature the user blinds $z, z_1$ by the same $\gamma$ so that $\log_\gamma(z_1) = \log_\gamma(\zeta_1)$.

2. **Hybrid:** The challenger’s output consists of a randomly picked correct signature and a fake signature, in random order. In a fake signature the challenger doesn’t use the same random $\gamma$ to blind $z$ and $z_1$ so that $\log_\gamma(z_1) \neq \log_\gamma(\zeta_1)$ except for negligible probability. Above, we explained how a fake signature can be generated, assuming that the challenger controls the random oracle.

3. **Fake:** The challenger outputs two fake signatures.

Now, what we need to prove is that $\text{Real} \approx \text{Hybrid}$ and $\text{Hybrid} \approx \text{Fake}$. Then, it holds that $\text{Real} \approx \text{Fake}$ which proves the blindness of the ACL scheme since the view of the adversary is indistinguishable no matter if he receives two correct or two fake signatures.

**Case 1: Real $\approx$ Hybrid**

Suppose that there exists a poly-time adversary $A$ who is successful in distinguishing between $\text{Real}$ and $\text{Hybrid}$ with probability $1/2 + \epsilon$, where the advantage $\epsilon$ is not negligible. We then show that there exists a poly-time algorithm $B$ that solves the DDH problem with non-negligible advantage.

The reduction $B$ gets as input an instance $(g, A, B, D)$ where $(g, A, B, D)$ is a DDH instance. $B$ first fixes $z = A$ and then sets $h_0 = g^{x_0}$, and $h_1, \ldots, h_n$ to $A^{x_0}, \ldots, A^{x_n}$ for randomly chosen $e_i \in \mathbb{Z}_q$. $B$ sends the parameters to $A$. The adversary $A$ creates his public key $y$ and sends $y$ to $B$ along with messages $m_0, m_1$, commitments $C_0, C_1$ and openings $(L_0, L_1)$ which are two attributes vectors of size $n$. $A$ and $B$ engage in two interleaving registration and signature issuing instances. During the registration phase we have: $C_j = A^{k_1(j)}$, $(k_1(j) = e_0 R + e_1 L_{j,1} + \cdots + e_n L_{j,n})$ honestly created, for $j \in \{0, 1\}$. Note that the value $k_1^{(j)}$ is known to the reduction $B$.

Then, during the two signature issuing instances, $A$ computes $z_1^{(j)} = A^{k_1^{(j)}} g^{\text{rnd}(j)}$ where $\text{rnd}(j)$ was sent to $B$. $B$ responds with random $c^{(j)}$ and then computes two signatures $\sigma_1, \sigma_2$ in the following way:

- $B$ flips two random coins: $\text{coin}, b \in \{0, 1\}$. Then, let $k_2 = \text{coin}$. $B$ is the correct signature on $(m_0, C_0)$ (where $C$ is denoted as $\zeta_1$ in our scheme) for the corresponding instance, and $\sigma_{\text{coin}+1}$ be a signature on $(m_1, C_1+)$ generated from the input to $B$, as follows: set $\zeta_1 = D$ and $\zeta_2 = E^{k_1^{(j)}} B^{\text{rnd}(j)}$; let $\zeta_2^{(j)} = \zeta_2^{(j)} / \zeta_1^{(j)}$, randomly choose $\rho, \omega, \rho_1, \rho_2, \omega', \mu$, and then define $H$ so that the signature verifies.

After receiving the signatures, $A$ outputs $\text{Real}$ or $\text{Hybrid}$. If $A$ outputs $\text{Real}$ then $B$ outputs “DH” or $B$ outputs “random” otherwise.

Let us analyze the success of $B$ in distinguishing Diffie-Hellman tuples from random tuples. If the input $(g, A, B, D)$ is a DH tuple, then: $\zeta_1^{(j)} = D = g^{ab} = A^b = z_1^b$ and $\zeta_2^{(j)} = E^{k_1^{(j)}} B^{\text{rnd}(j)} = g^{ab_1} g^{\text{rnd}(j)} = (g^{ak_1^{(j)}} g^{\text{rnd}(j)})^b = (A^{k_1^{(j)}} g^{\text{rnd}(j)})^b = 1$, so the signature $\sigma_{\text{coin}+1}$ is distributed identically to a correct signature, and this is precisely the $\text{Real}$ game. Similarly, if the input is not a DH tuple then it is the $\text{Hybrid}$ game. Therefore, $B$ will be correct exactly when $A$ is and will distinguish a DH tuple with probability $1/2 + \epsilon$, which contradicts to the DDH assumption.

**Case 2: Hybrid $\approx$ Fake**

Similarly, we build a reduction $B$ which given an adversary $A$, who distinguishes between $\text{Hybrid}$ and $\text{Fake}$ with probability $1/2 + \epsilon$, solves the DDH problem.

Working in a similar way we did before, $B$ will now create a fake signature and one using the input tuple. If $A$ outputs
Hybrid then B will output “DH” or “not DH” otherwise. If
the input is a DH instance then the output will correspond to
the Hybrid game and to the Fake otherwise. So, B will be
successful whenever A is and will break the DDH assumption
with non-negligible advantage $\epsilon$ which is a contradiction.

Finally, since Real$\approx$Hybrid and Hybrid$\approx$Fake it follows that
Real$\approx$Fake. $\square$

**Theorem 4 (One-More Unforgeability).** The signature
issuing protocol is $(\ell, \ell+1)$-unforgeable for polynomially
bounded $\ell$ if the discrete logarithm problem is intractable
and $\mathcal{H}$ is a random oracle.

Outline of the proof We first discuss the witness indistingui-
shability of the protocol (Lemma 1), which allows us to
to simulate the Signer with either y-side or z-side witness(es)
to extract the witness of the other side. Then, in Lemma 2
we prove that in order for the User to get a valid signature,
he has to blind $(z, z_1)$ into $(\zeta, \zeta_1)$ only in such a way that
$\log_z \zeta = \log_{z_1} \zeta_1$. In Lemma 3 we prove that is infeasible
to create a valid signature without engaging in the issuing pro-
tocol with the legitimate Signer. From Lemmas 2 and 3 we see
that if a User engages in the signature issuing protocol $\ell$
times and outputs $\ell + 1$ signatures, then, there exist at least
two valid signatures linked to a particular run of the issuing
protocol. Finally, it needs to be proven that a forger who
manages to produce two signatures from a single protocol
run can be used to solve the discrete logarithm problem.
We will denote the $i$-th execution of the issuing protocol by
run$_i$. Recall that a transcript of run$_i$ of the issuing protocol
contains values $(z_1, z_2)\text{;}$; by the z-side witness of run$_i$ we
denote $(w_1, i, w_2, i)$ such that $z_1 = g^{w_1}$ and $z_2 = h^{w_2}$.

Consider an alternative signing algorithm that, instead of
using the y-side witness in the issuing protocol, uses the z-
side witness. Let $\text{params} = (q, G, h, z, \{h_1\})$ be the public
parameters, and $y$ be the public key of the Signer. Sup-
pose that the registration and the preparation phases of the
signing protocol for run$_i$, resulted in the signer and the user
setting up the values $z_1 = g^{w_1}$ and $z_2 = h^{w_2}$. The alter-
native signing algorithm takes as input the public parameters $\text{params}$,
the public key $y$, the preparation phase output $(z_1, z_2)$ and the values $(w_1, i, w_2, i)$ instead of the secret key
$x$, and works as follows:

1. Generate $c_i, r_i \in_U Z_q$ and set $a_i := g^{r_i}$
2. Compute $a'_i := g^{w_1}$ and $a''_i := h^{w_2}$ with $w_1, i, w_2, i

$1$: \in_U Z_q$.
3. Sends $a_i, a'_i, a''_i$ to the user.
4. Given $e_i$ from user, compute $c'_i := e_i - c_i \mod q, r'_i :=
w_1 - c'_i \mod q$ and $r''_i := w_2 - c'_i \mod q$.
5. Send $r_i, c_i, r'_i, r''_i, c'_i$ to the user.

Definition 10. The signing protocol that was described
above is called the z-side signer.

**Lemma 1.** The signer’s output is perfectly indistinguish-
able from the output of a z-side signer.

**Proof.** Follows from the result of Cramer [18]. $\square$

Note that, in order to run the z-side signer, it is necessary
to somehow get the corresponding witness. We will see later
that, with black-box access to an adversarial user in the
registration phase, a reduction can use standard $\Sigma$-protocol
extraction techniques in order to learn the representations
of $C$; if the reduction is, in addition, given the discrete log-
arithms of all the public parameters to the base $g$, it will be
able to compute the z-side witness, and so, inside a reduc-
tion, the z-side signer can be invoked.

**Lemma 2. (Restrictive Blinding) Let $A$ be a User that
gains in the signature issuing protocol $\ell$ times, and outputs
a valid signature $\sigma = (m, (\zeta, \zeta_1, \rho, \omega, \rho_1, \rho_2, \omega', \mu))$. Let $\text{run}_i$ 

denote the $i$-th execution of the protocol and $z_1, i$ denote $z_1$
used by the signer $S$ in $\text{run}_i$. For polynomially bounded $\ell$
and for all polynomial-time $A$, the probability that $\log_z \zeta \neq
\log_{z_1} \zeta_1$ holds for all $i$, is negligible if the discrete logarithm
problem is intractable and $\mathcal{H}$ is a random oracle.

**Proof.** Let $A$ have at most $q_h$ accesses to $\mathcal{H}$ and ask at
most $f$ signatures to $B$. Let $\sigma = (m, (\zeta, \zeta_1, \rho, \omega, \rho_1, \rho_2, \omega', \mu))$
be a signature that $A$ outputs and satisfies $\log_z \zeta \neq \log_{z_1} \zeta_1$
for all $i$ with probability $e_0$ which is not negligible in $n$ ($q_h$
and $\ell$ are bounded by a polynomial in the security parameter
$n$). We randomly fix an index $i \in \{1, \ldots, \ell\}$ and regard $A$ as
successful only if the RO query for the resulting signature
was during the $I$-th run. (If the resulting signature does
not correspond to any query, then $A$ is successful only with
negligible probability due to the randomness of $\mathcal{H}$.) Let
$(G; q; g; Y)$ be an instance of the DL problem.

**Reduction Algorithm: $B$ first sets $(p, q, g) := (p, q, g)$.

Then, it flips a coin $\chi \in \{0, 1\}$ to either select $y := Y$
(case $\chi = 0$), or $h := Y$ (case $\chi = 1$) and also guesses
a random index $I \in \{1, \ldots, \ell\}$.

Case $y = Y$: (extracting y-side witness)

1. **Key Generation:** $B$ selects $k, k', \{k_j\} \in_U Z_n$, for $0 \leq

j \leq n$, and sets $h := g^k$, and $h_j := g^{k_j}$, $z = g^{k'}$.

As a result of setting the parameters this way, $B$ will always be
able to compute z-side witnesses, as long as $B$ can successfully
extract the user’s attributes and randomness in the registration
phase.

2. **Registration:** $B$ extracts values $(L_1, \ldots, L_n; R)$ using a

knowledge extractor for proof $\pi_1$, (since we only worry
about sequential composition the extractor may rework) and
defines $K = k_0 R + \sum_{i=1}^n h_i L_i$ (so that $C = g^K$).

3. **Preparation:** $B$ selects $w_1, i \in_U Z_q$ and sets $r = w_1, i -

K$. Then, computes $w_2 := (k' - w_1, i)/mod q$ (so we have
$z_1 = g^{w_1, i}, z_2 = h^{w_2}$). $B$ sends $r$ to $A$.

4. **Validation:** $B$ runs $A$ using the z-side signer described
above. At the end of each run, the z-side signer sends:
$r_i, c_i, r'_i, r''_i, c'_i$ to $A$. $B$ simulates $\mathcal{H}$ by returning
$\varepsilon \in_U Z_q$ to the random oracle queries issued by $A$. $A$
outputs a signature $\sigma = (\zeta, \zeta_1, \rho, \omega, \rho_1, \rho_2, \omega', \mu)$, that
corresponds to something $\varepsilon$.

5. If the RO query for the resulting signature $\sigma$ happened
in run$_I$, then move to the next step, otherwise the reduc-
tion fails.

6. **Rewinding:** Reset and restart $A$ with the same setting.

For run$_I$ and after, $B$ simulates $\mathcal{H}$ with $\varepsilon \in_U Z_q$. $A$
outputs a signature, say $\hat{\sigma} = (\zeta, \zeta_1, \hat{\rho}, \hat{\omega}, \hat{\rho}_1, \hat{\rho}_2, \hat{\omega}', \hat{\mu})$.

If $\omega \neq \hat{\omega}$, $B$ outputs $x := (\rho - \hat{\rho})/(\omega - \hat{\omega}) mod q$. The
simulation fails, otherwise.

Case $h = Y$: (extracting z-side witness)

1. **Key generation:** $B$ selects $z, \{k_j\} \in_U Z_q$ and sets $y :=
g^{z}$ and $h_j := g^{k_j}$. It also selects $w_1, w_2 \in_U Z_q$ and sets
$z := g^{w_1} h^{w_2}$.
2. Signature Issuing: $B$ runs $A$ simulating the signer as follows.
   
   (a) For $i \neq 1$, $B$ follows the protocol with $y$-side witness, $x$.
   
   (b) For $i = 1$, $B$ engages in the issuing protocol using the
   z-side witness $(w_1, w_2)$ as follows.
   
   i. Registration: Extract values $(L_1, \ldots, L_\ell; R)$ using
   a ZK extractor and define $K = k_0 R + \sum_{i=1}^{\ell} k_i L_i$ (so that $C = g^K$).
   
   ii. Preparation: Set $rnd = w_1 - K$ (so we have
   $z_{1,1} = g^{w_1}, z_{2,1} = h^{2w_2}$. $B$ sends $r$ to $A$.
   
   iii. Validation: $B$ runs $A$ using the z-side signer
   and at the end of $run_1$ sends $(\gamma_1, \epsilon_1, r_{1,1}, r_{2,1}, c_1)$
   to $A$.
   
   $B$ simulates $\mathcal{H}$ by returning $\epsilon \in \mathbb{U}_m$ to the RO queries
   made by $A$.
   
   3. $A$ outputs a signature, say $\sigma = (\zeta, \xi, (\rho, \omega, \rho_1, \rho_2, \omega', \mu))$, that corresponds to some $\epsilon$. If the RO query for the forged
   signature didn’t happen in $run_1$ then fail, else move to the next step.
   
   4. Rewinding: Rewind and restart $A^*$ with the same setting.
   For $run_1$ and forward, $B$ simulates $\mathcal{H}$ by returning $\epsilon \in \mathbb{U}_m$.
   
   5. $A$ outputs a signature, say $\hat{\sigma} = (\zeta, \xi, (\rho, \omega, \rho_1, \rho_2, \omega', \mu))$, that corresponds to some $\tilde{\epsilon}$. If $\omega' \neq \omega$, $B$ computes
   $w_1' = (\rho_1 - \rho_1')/(\mu - \mu)$ mod $q$, $w_2' = (\rho_2 - \rho_2')/(\mu - \mu)$ mod $q$, and outputs $\hat{w} = (w_1 - w_1')/(w_2 - w_2') \mod q$
   (where $w = \log_2 h$). Simulation fails otherwise.

**Evaluation of success probability** The reduction is successful when it gets $\omega \neq \omega'$ for $\chi = 0$ or when it gets $\omega' \neq \omega'$ for $\chi = 1$. What we need to argue about is that, independently of the choice of $\chi$, the above will happen with non-negligible probability. Note that we construct the reduction by giving different responses after the $I$th run (for a randomly chosen $I \in \{1, \ldots, \ell\}$) when rewinding and we let the reduction fail if the forgery didn’t happen in $run_1$.

Let us now fix the adversary’s view and its RO tape. Once everything is fixed, we notice that $A$’s forgery is uniquely determined by $\epsilon'$ and $\epsilon$. We now consider two cases: (1) $\omega' = \omega$ with non-negligible probability even if $\epsilon$ and $\epsilon'$ change (after rewinding). In this case $\omega' = \omega'$ and thus $\omega' = \omega' = \epsilon = \hat{\omega} + \hat{\omega}'$, so the $y$-side witness can be extracted. (2) $\omega' \neq \omega'$ with non-negligible probability if $\epsilon$ and $\epsilon'$ change. So, when we run the adversary again with access to z-side witness and with different $\hat{\epsilon}$ we get two signatures from him where $\omega = \hat{\omega}$ and thus $\omega' \neq \omega'$ and thus we can extract the z-side witness. Since we fail if the forgery didn’t happen in $run_2$ we do not need to consider what happens when we run $A$ with $y$-side witness for $\chi = 1$ and $i \neq I$.

What is left to show is that for $\chi = 1$, the z-side witness, $(w_1', w_2')$, extracted is not the one that $B$ already knows. Remember that the reduction in the beginning, selects $w_1, w_2 \in \mathbb{U}_m$ and sets $z := g^{w_1} h^{w_2}$ and in $run_1$ sets $r$ so that

$z_{1,1} = g^{w_1}, z_{2,1} = h^{2w_2}$.

The signature that $A$ produces contains proofs that $\hat{\zeta} = \gamma_1, \hat{\xi} = g^{w_1}$ and $\hat{\zeta}_2 = h^{2w_2}$, so $w_1' = \gamma w_1$ and $w_2' = \gamma w_2$ and we can extract those $w_1', w_2'$ by rewinding. For the forgery that $A$ outputs it holds $\log_{q_0} \hat{\zeta} \neq \log_{q_1} \zeta_1$ (by definition of Lemma 2) which guarantees that we will obtain two different representations of $z$. \[ \square \]

**Lemma 3.** Any poly-time adversary $A$ outputs a valid signature without interacting with the Signer only with negligible probability if the discrete logarithm problem is intractable and $H$ is a random oracle.

**Proof. (sketch)** In order to prove this we are going to use the oracle replay attack suggested by Pointcheval and Stern to prove unforgeability of blind signatures [40]. Similar to the proof of Lemma 2 we randomly fix an index $I \in \{1, \ldots, \ell\}$ and regard $A$ as successful only if the RO query for the resulting signature was during the $I$th run. Accordingly, it is equivalent to assuming an adversary, we will call it $A^*$, that asks $\mathcal{H}$ only once and succeeds with probability $\epsilon_1 \geq \epsilon_0/\eta_0$. Given this modified $A^*$ we construct the reduction $B$ that solves the discrete logarithm problem by simulating the interaction. Given $Y \in G$, we construct a reduction $\tilde{B}$ that finds $log_y Y$ in $\mathbb{Z}_q$. During key generation, $B$ first selects $\omega, \xi, (k_i)$ randomly and sets $y = Y, h = g^\omega, h_i = g^\xi z = Y_q$. ($B$ does not need to simulate the signer, so it can put $Y$ in both $y$ and $z$.) Then, $\tilde{B}$ invokes $A^*$ twice with the same initial settings and different $\epsilon$ and $\hat{\epsilon}$ as answers of $\mathcal{H}$. The resulting signatures are $\sigma = (\zeta, \xi, (\rho, \omega, \rho_1, \rho_2, \omega', \mu))$ and $\hat{\sigma} = (\zeta, \xi, (\rho, \omega, \rho_1, \rho_2, \omega', \mu))$. Since $\omega + \omega' = \epsilon \neq \hat{\epsilon} = \hat{\omega} + \hat{\omega}'$, at least one of $\omega \neq \omega'$ or $\omega' \neq \omega$ happens.

In the case that $\omega \neq \omega'$, $B$ computes $\log_{log_{q_0} Y} (\rho - \mu)/(\omega - \omega') \mod q$. For the case $\omega' \neq \omega'$, $B$ computes $\gamma = \log_{log_{q_0} Y} (\rho - \mu)/(\omega' - \omega') \mod q$, $w_1 = \log_{q_0} \zeta_1 = (\rho_1 - \rho_1')/(\omega' - \omega') \mod q$, $w_2 = \log_{q_0} \zeta_2 = (\rho_2 - \rho_2')/(\omega' - \omega') \mod q$, and $\log_{q_0} Y = \log_{q_0} z - \xi = (w_1 + w_2/z)/\gamma - \xi \mod q$. \[ \square \]

**Proof. (Theorem 4).** Suppose that there exists adversary $A$ that outputs $\ell + 1$ valid signatures with probability $\epsilon$ not negligible after interacting with the Signer at most $\ell$ times. The case $\ell = 0$ has been ruled out by Lemma 3. We consider $\ell \geq 1$.

By Lemmas 2 and 3, among the $\ell + 1$ signatures, there exist at least two signature-message pairs which contain $(\zeta, \xi_1)$ and $(\zeta, \xi_2)$ such that $\log_{q_0} \zeta_1 = \log_{q_0} \zeta_2 = \log_{q_0} z_{1,1}$ holds for $z_{1,1}$ used in $run_1$ for some $I \in \{1, \ldots, \ell\}$. Now, there exist two queries to $\mathcal{H}$ that correspond to those signatures. In a similar way as in the proof of Lemma 2, we guess the indices of these queries and regard $A$ as being successful only when the guess is correct. We consider an equivalent adversary $A'$ that asks $\mathcal{H}$ only twice and succeeds with probability $\epsilon' = \epsilon/(\ell + 1)$ in producing two signatures in the expected relation.

We construct a reduction $B$ that, given $(p, q, g, Y)$, solves $\log_{q_0} Y$ in $\mathbb{Z}_q$ by using $A'$.

**Reduction Algorithm:** $B$ sets $(p, q, g, Y) := (p, q, g)$. It then flips a coin, $\chi \in \{0, 1\}$, to select either $y := Y$ (case $\chi = 0$), or $y := g^\omega$ with randomly chosen $x$ (case $\chi = 1$).

1. Setup: If $\chi = 0$, set up and use the z-side signer. Set $h, h_0, \ldots, h_n$ such that their discrete logarithm base $g$ is known. Else, set up the y-side signer by letting $y = g^x$ for a randomly chosen $x$; set up $h, h_0, \ldots, h_n$ such that their discrete logarithm base $Y$ is known.

2. Registration: If $\chi = 0$, extract $(R, L_0, \ldots, L_n)$. Else follow the protocol.

3. Preparation: Follow the protocol.

4. Validation: If $\chi = 0$, use the z-side signer, else follow the protocol.
5. Responding to the RO queries: $B$ simulates $H$ by returning random values, say $\varepsilon_1$ and $\varepsilon_2$, to the two RO queries.

6. Rewinding: When $A^*$ outputs two signatures $(\sigma(1) = (\chi_{1}^{(1)}, \chi_{1}^{(2)}, \omega(1), \rho(1), \omega(1), \mu(1)))$ and $(\sigma(2) = (\chi_{2}^{(2)}, \chi_{2}^{(2)}, \omega(2), \rho(2), \omega(2), \mu(2)))$ corresponding to $\varepsilon_1$ and $\varepsilon_2$, $B$ resets the adversary and this time responds to the two RO queries with $\tilde{\varepsilon}_1$ and $\tilde{\varepsilon}_2$.

7. Interacting with $A^*$ after rewinding is the same as before, according to the value of $\chi$.

8. Extracting: When $A^*$ again outputs two signatures $(\sigma^{\prime}(1) = (\chi_{1}^{(1)}, \chi_{1}^{(2)}, \omega^{\prime}(1), \rho^{\prime}(1), \omega^{\prime}(1), \mu^{\prime}(1)))$ and $(\sigma^{\prime}(2) = (\chi_{2}^{(2)}, \chi_{2}^{(2)}, \omega^{\prime}(2), \rho^{\prime}(2), \omega^{\prime}(2), \mu^{\prime}(2)))$ corresponding to $\tilde{\varepsilon}_1$ and $\tilde{\varepsilon}_2$, do: if $\chi = 0$ and $\omega^{(1)} \neq \omega^{\prime}(1)$ or $\omega^{(2)} \neq \omega^{\prime}(2)$, then extract the y-side witness as described in the proof of Lemma 2; else if $\chi = 1$ and $\omega^{(1)} \neq \omega^{\prime}(1)$ or $\omega^{(2)} \neq \omega^{\prime}(2)$, then extract the z-side witness as described in Lemma 2.

Evaluation of success probability. Here we need to show that $B$ will extract the witness that it doesn’t already know with non-negligible probability (otherwise, $\chi$ is revealed). Consider all the $q^2$ possible $(\varepsilon_1, \varepsilon_2)$ pairs that can be given as responses to $A^*$’s two RO queries. Let $suc$ be the set of pairs $(\varepsilon_1, \varepsilon_2)$ for which $A^*$ succeeds: $suc = \{(\varepsilon_1, \varepsilon_2) \mid A^* \text{ breaks the } q^2 \text{-one more unforgeability}\}$. What we want to show is that for a random pair $(\varepsilon_1, \varepsilon_2) \in suc$ there exists $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2) \in suc$ such that $A^*$ produces the same $\varepsilon$ with non-negligible probability in the run where the first RO query happened. Notice that the transcript (i.e. the messages exchanged between the Signer and $A^*$) of these two runs has to be the same. We call such a pair $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2)$ good for $(\varepsilon_1, \varepsilon_2)$.

Consider $A^*$ is run twice: once with $(\varepsilon_1, \varepsilon_2)$ and once with $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2)$, both times producing two message-signature pairs corresponding to $z_1 i$. Taken together, $(\varepsilon_1, \varepsilon_2)$, $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2)$, the transcript of the runs and the four message-signature pairs allow the reduction to compute either a $y$ side or a $z$ side witness. However, we need to argue that, with non-negligible probability, this will be a witness the reduction doesn’t already know. Since the transcripts are identical, the joint distribution of the two views ($A^*$’s view in the first run and its view, after the rewinding, in the second run) is independent of $\chi$.

First, note that there are only $q$ possible $e$’s but $q^2$ possible $(\varepsilon_1, \varepsilon_2)$, and so there can only be a negligible fraction of $(\varepsilon_1, \varepsilon_2)$ for which no good $(\varepsilon_1, \varepsilon_2)$ exists. Suppose that $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2)$ is good for $(\varepsilon_1, \varepsilon_2)$ but $(\varepsilon_1, \varepsilon_2)$ is not; moreover, the reduction that gives $(\varepsilon_1, \varepsilon_2)$ before rewinding, and $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2)$ after rewinding, can only compute the witness it already knows (i.e. if $\chi = 0$, it can compute the z-side witness, and for $\chi = 1$ the y-side). Then for the same choice of $\chi$, the reduction would be computing the witness it does not already know if, instead, it used $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2)$ and $(\varepsilon_1, \varepsilon_2)$. For example, if $\chi = 0$ we would have $\omega \neq \omega$ and $\omega = \omega^*$ respectively, which implies that $\omega \neq \omega^*$ and thus by choosing $(\varepsilon_1, \varepsilon_2)$ and $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2)$ the y side witness could be extracted. Since it is just as likely to pick $(\varepsilon_1, \varepsilon_2)$ or $(\tilde{\varepsilon}_1, \tilde{\varepsilon}_2)$ for the values before rewinding, it is just as likely to succeed as to fail, independently of $\chi$ (this analysis was first given by Abe and Okamoto [2]).

Finally, the only thing left to prove is that an adversary cannot open the sets of commitments $\{\zeta(i), \sigma(i)\}$ to two different multisets. Note that $\zeta(i)$ uniquely determines $\gamma(i)$, and by Lemma 2, it follows that for every $i$, there exists some $\tilde{z}_2(i)$ such that $\tilde{z}_1(i) = (\tilde{z}(i))^*(i)$. Thus, an alternative opening of the commitment $\{\zeta(i), \sigma(i)\}$ gives rise to an alternative representation of $\tilde{z}_2(i)$ in bases $h, h_1, \ldots, h_n, g$. In turn, by standard techniques, this breaks the discrete logarithm assumption. (The reduction will just honestly emulate the signer, and embed the DL instance into the parameters $h, h_1, \ldots, h_n, g$.)

Setting parameters. Note that our reduction is not tight: an adversary $A$ whose success probability is $\epsilon$ gives rise to the reduction $B$ with the same runtime but success probability $\epsilon/(\epsilon^2) \approx \epsilon/q^2$. This, in other words means that if $A$ takes time $t$ to break unforgeability with probability $\epsilon$, then our reduction, $B$, in order to break the discrete logarithm assumption with the same success probability would need time $t \cdot q^2$. So, if $A$ breaks unforgeability faster than in time $2^k$, then $B$ breaks DL in time faster than $2^{k + 2 \log q} = 2^{k + 2 \log q}$.

Thus, we need to set the parameters to be sufficiently large so that is reasonable to assume that the reduction would take time $2^{k + 2 \log q}$ to break DL directly.

As suggested by NIST [32] in order to tolerate an adversary whose computational power is $2^k$, one needs an elliptic curve group of size $2^{2k}$ which for our reduction should be $2^{k + 2 \log q}$. Now, let’s be very conservative and assume that $q_0 = 2^{30}$, thus the factor $(\frac{q}{q_0})$ we lose is of 160-bits.

Thus, if we want to achieve 128-bit security we would need an elliptic curve group where the bit size of an element is $2(128 + 2 \cdot 80) = 576$-bits. Note that the equivalent security level for RSA requires 3072-bit keys.

7. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments and suggestions. We also thank our collaborators at the Pay-As-You-Go project for discussions on the use of anonymous credential systems in lightweight devices and especially Gesine Hinterwälder. This work was supported by NSF grants 0964379 and 1012060.

8. REFERENCES


[7] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the