Selective disclosure blinded credential sets - DRAFT

Jason E. Holt (isrl@lunkwill.org)
Kent E. Seamons (seamons@cs.byu.edu) [*]
Internet Security Research Lab
Brigham Young University
http://isrl.byu.edu/

Abstract

Alice studies computer science at BYU. Her friend Bob is only 17 years old. The local movie theater gives discounts to CS students and minors. This paper describes how Alice can obtain a digital credential from BYU and use it to prove to the theater that she's a computer science student. She doesn't have to reveal anything else to the theater about her identity, and the theater can't discover anything additional even with BYU's help.

Most significantly, Alice and Bob won't be able to pool their credentials to make it look as if Alice is both a CS student and under 18. Revocable anonynimity is also (optionally) available, such that a quorum of authorities can later work with the theater to discover Alice's identity if the need arises.

To implement credential sets, we present several useful primitives. First, we describe a non-interactive form of the cut and choose protocol. Second, we present a method whereby documents can be blindly signed by different signers, and later proved to belong together in a set.

It may be possible to implement credential sets without violating the patents on blind signatures.

Keywords: digital credentials, selective disclosure, credential pooling, blind signatures.

1 Overview

Alice wishes to obtain a service from Steve, a server. Steve will only provide the service if Alice can demonstrate certain attributes about herself as attested by credential issuers. Alice is willing to prove these attributes, but doesn't want Steve to get any additional information about her, even if Steve works together with the credential issuers. Steve wants to make sure that the attributes Alice displays all belong to the same person, and weren't accumulated by Alice and Bob pooling their credentials.

To this end, Alice first creates a Credential Set Request, which contains a matrix of blinded documents. Each row of the matrix contains documents to be signed by the same issuer, and the documents in each column all share a common credential ID which Alice will use to prove to Steve that the documents in the matrix belong together. To prove that she's honest, Alice includes information about certain columns of the matrix and sends the Credential Set Request to each of the issuers of her credentials. Each issuer in-

[*]With special thanks to Ben Laurie, Hilarie Orman, Rich Schroeppel, Robert Sherwood, Mike Stay, Ting Yu, sci.crypt and the Cypherpunks. This research was supported by DARPA through AFRL contract number F33615-01-C-0336 and through Space and Naval Warfare Systems Center San Diego grant number N66001-01-18908.

spects the request then blindly signs the unrevealed columns of his row in the matrix. Alice then removes the blinding factors, leaving her with a valid set of credentials (where each row constitutes an individual credential).

Credentials in credential sets are built from selective-disclosure certificates. These are certificates in which the normal attribute values have been replaced with a bit commitment of the true value. If Alice doesn't choose to reveal the value of a selective disclosure field, Steve doesn't learn anything about that value. But if she does choose to reveal such a field, Steve can verify that she isn't lying about its value.

When Alice shows some subset of the credentials to Steve, Steve checks that all the IDs match, ensuring that all the credentials were in fact issued to the same person. Alice also reveals the preimages of the selective disclosure attributes in each credential which she wishes to show to Steve.

If Alice later shows credentials from the same set to someone else, that person could collude with Steve and determine that they both were dealing with the same person. Thus for maximum privacy Alice should obtain many instances of her credential set, and use each instance in only one transaction.

Revocable anonynimity can optionally be obtained by including a uniquely identifying document in the certificate which can only be decrypted by the cooperation of a quorum of auditing authorities.

2 Related work

Chaum's blind signature techniques [6, 7] made it possible to obtain a certified value from an issuer and show it to a server without the possibility of the server and issuer correlating the issuing and showing events. Chaum presented a credential system based on blind signatures [5] in which users establish a different pseudonym with each potential server and issuer, and can transfer credentials issued under one pseudonym to other pseudonyms they hold. In his system, the exponent used in signing a pseudonym defines the type and value of the credential. His system allows demonstration of mathematical relationships between attributes (such as AND, OR and GREATER THAN). One awkward requirement of Chaum's proposal is that a trusted authority is needed to facilitate the relationship between issuers, users and servers. A particular server and issuer would have to establish a relationship with this authority before users could even obtain credentials from the issuer to show to the server.

Brands [1, 2] presented a system with many of the features Chaum introduced. In his system, an issuer establishes a set of bases $(g_1 ... g_n)$ which are analogous to the fields of a certificate. The bases are raised to a power signifying the value of the field. $(g_1^{x_1} g_2^{x_2} ... g_n^{x_n})$ would comprise (part of) a credential in Brands' system, where x_k is the value for the attribute represented by g_k. Credential holders can reveal exponents or properties of combinations of exponents through proofs of knowledge relative to the credential. Unlike Chaum's system, credentials can be shown to arbitrary servers, and issuers need not have any relationship with a central authority. However, Brands does not address the problem of users pooling credentials together to obtain services no one of them could obtain alone.

Camenisch and Lysyanskaya [3] proposed a system allowing a credential to be shown multiple times without allowing showing instances to be linked. Their system focuses on limiting the information revealed to servers during the showing protocol rather than restricting what information the issuer gets during the signing process. Revocable anonymity is possible along with several other desirable features. Their system provides many of the features of previous systems without using blinding a la Chaum, but also relies heavily on proofs of knowledge. Credentials are issued to a user relative to their public key, so like ours, their system prevents users from pooling credentials.

Credential sets are more related to the current X.509 certificate infrastructure than the exponent-based systems listed here. That is, credential sets use signed documents containing multiple name-value pairs, rather than using separate signing exponents for each attribute.

3 Preliminaries

3.1 Credentials and Certificates

A certificate is a document containing various attributes and their values. Certificates are issued by issuers (often called certificate authorities) to users (often called subjects). X.509v3 certificates have a number of standard fields such as the issuer and subject names, and "extensions" which can specify arbitrary other kinds of information about the user.

A user typically creates a certificate to be signed, called a certificate request, and sends it to the issuer. The issuer signs the request by hashing the document with a collision-resistant one-way function and signing the hash with his private key. Later the user can show the signed certificate to a verifier, who verifies that the certificate is valid. In an "on-line" system, this verification takes place with the help of a central authority. In an "off-line" system, the server can verify a certificate (or more generally, a credential) without outside help.

We use the term "credential" when we wish to speak of attribute-demonstrating information in general. A credential is something which establishes one or more attributes of its owner. Credential sets use multiple X.509v3-like certificates together to form a single credential. Proper X.509v3 certificates signed in the traditional way could also be considered credentials.

3.1.1 Selective disclosure credentials

A selective disclosure credential has several attributes. When the user shows the credential to a verifier, she can choose to reveal only some of the attributes to the verifier.

Credential sets accomplish this with the help of bit commitment. Bit commitment allows our user, Alice, to commit to a value without revealing it. One way to do this is with the help of collision-resistant one-way functions. Alice's commitment c is the output of a one-way function *oneway*() operating on her secret value s and a random string r:

$$c = commit(s) = oneway(s.r)$$

(. denotes concatenation). Alice first sends c to Victor the verifier. If she chooses not to reveal the value, Victor can't determine what the value was. If she does choose to reveal her secret, she sends Victor s and r, who runs them through *oneway*() and checks that the result equals c. If *oneway*() is collision-resistant, Alice can't easily find any other values for s and r which will produce c as output.

A normal certificate can easily be made into a selective disclosure certificate by replacing actual attribute values with commitments to those values. Victor can verify that the certificate is valid in the usual way, but gains no information about the selective disclosure values unless Alice reveals the value and random string used in the commitment function.

Selective disclosure credentials are even more powerful when used in conjunction with blind signatures. If Izzy (a credential issuer) blindly signs a credential for Alice, she can show it to Steve without revealing all the selective disclosure fields. Steve can pass the credential back to Izzy, but unless Alice has revealed enough information to uniquely identify herself, Izzy won't be able to determine what the unrevealed fields were, even though he signed the credential.

3.2 Blind signatures

Chaum introduced the idea of blind signatures. Blind signatures allow Izzy, the issuer, to sign a document without knowing its contents. To accomplish this, Alice applies a blinding factor to her document before submitting it to Izzy for signing. Izzy signs the blinded document and returns it to Alice. Alice can then remove the blinding factor without invalidating the signature. Anyone can verify that the signature is valid, but Izzy can't prove any association between the blinded document he signed and the one Alice presents unless he knows the blinding factor Alice used.

Since Chaum has a patent on blind signatures until 2005, we'll later mention an alternative which might not be covered by his patents. Camenisch [4] described blind signature systems based on other public-key cryptosystems which might also be useful for implementing credential sets.

3.2.1 Cut and choose

Since Izzy is unwilling to sign documents without knowing what they contain, Alice uses a cut and choose protocol to demonstrate that the credentials she wants signed are valid[1]. Alice will create n blinded copies of a certificate, and allow Izzy to examine some of them. In a simple implementation, Izzy examines n-1 of the certificates, and signs the remaining one unexamined. Alice has a 1 in n chance of successfully cheating by guessing which certificate he won't examine. This implementation can be useful if there are strong penalties for trying to cheat, but isn't suitable for the non-interactive version of the protocol we present in the next section, since Alice can attempt to cheat without communicating with any other parties.

So instead, Alice will allow Izzy to examine n/2 of the certificates. He'll sign the other n/2 copies. Alice can unblind them and show them to Steve, who will examine them to ensure that they're all consistent. Alice can only cheat by guessing which n/2 copies Izzy won't examine, and falsifying exactly those. Now she has a 1 in (n choose n/2) chance of successfully cheating.

To begin, Alice creates n equivalent copies of the certificate and hashes each one with *oneway*(). She blinds each hash with a different blinding factor. If the documents were normal certificates, each copy would be completely identical (and blinding wouldn't serve any useful purpose – the issuer would know that the documents revealed were byte-for-byte identical to the unrevealed documents). If we use selective disclosure certificates, however, each certificate will have the same actual attribute values, but the commitments to those values which go into the certificate will be different because of the random strings used in *commit*(). Since Izzy won't know the random strings, he won't know what the actual attribute values are if he later sees the signed documents.

Alice makes all n blinded hashes available to Izzy, and n/2 of them are selected at random. In order for Izzy to verify that the corresponding n/2 certificates are correct, Alice reveals the blinding factors for the n/2 selected hashes, as well as the random strings used in the *commit*() function when constructing the selective disclosure fields Izzy needs to see. Izzy uses *oneway*() on the values and random strings to verify the selective disclosure fields. Then he hashes the certificates and blinds them with the supplied blinding factors to see if they match the blinded values. If Alice hasn't tried to cheat, he multiplies the remaining n/2 hashes together, signs the result and returns it to Alice.

Alice unblinds the signed product, and can show it to Victor as the signature for the certificates. To reveal a selective disclosure attribute to Victor, Alice must send the true attribute value along with the random string used by *commit*() in each of the n/2 certificates. Victor verifies that the *oneway*() function returns the proper value for each copy of the certificate. If any of the values are different, Victor knows Alice was trying to cheat.

Alice has only a 1 in (n choose n/2) chance that she can successfully defeat the cut and choose mechanism. To do this, she must get Izzy to sign n/2 documents all consistent with each other and all different from the n/2 valid documents which he inspects.

3.2.2 Non-interactive cut-and-choose

Borrowing a concept from non-interactive zero knowledge proofs [2], we can use a collision-resistant one way function to select the n/2 documents to be revealed

[1]See Applied Cryptography[9], pp.113-115.

[2]Applied Cryptography[9], pp.106-107.

in the cut-and-choose protocol. The *oneway*() function is called on the concatenation of the n blinded documents described above, and its output it used to select which n/2 documents Alice must reveal. Izzy can also hash the n documents to verify that Alice properly applied *oneway*().

The following algorithm can be used to select the n/2 columns: Use the output of *oneway*() to seed a suitable PRNG. Divide the PRNG's output into strings which are each $ceil(log_2 n)$ bits long and use each string as the index of an ID to reveal, until n/2 IDs have been selected.

3.2.3 Chaum blinding

Chaum's blinding is simple. Here's how we can blindly compute an RSA signature using Izzy's RSA secret key d, public key e and public modulus n. Alice chooses a random blinding value b, raises it to Izzy's public exponent and multiplies it with her document h (h since we typically sign the hash of a message, not the message itself):

$$r = hb^e \pmod n$$

Izzy signs r by raising it to his secret exponent d and returns $r^d \ (mod\ n)$ to Alice. Alice can now remove the blinding factor to obtain h^d, the signature for her document. Note that $(b^e)^d$ is equivalent to encrypting then decrypting b: $(b^e)^d = b \ (mod\ n)$.

$$\begin{aligned} r^d &= h^d b^{ed} && (mod\ n) \\ &= h^d b && (mod\ n) \\ h^d &= \tfrac{h^d b}{b} && (mod\ n) \end{aligned}$$

Alice is left with a normal RSA signature on h which anyone can verify if they know Izzy's public key. Thus, if credential sets are implemented using Chaum's scheme, verifiers will be able to verify the validity of credentials off-line, without Izzy's help.

This technique works with our cut-and-choose algorithm. In this case, Alice has a collection of hashed certificates $h_1 ... h_n$, and chooses corresponding blinding factors $b_1 ... b_n$.

$$r_k = h_k b_k^e \pmod n$$

Izzy signs the product of $r_1 .. r_n$, and Alice unblinds as before:

$$(h_1 h_2 ...)^d = \frac{h_1^d b_1^{ed} h_2^d b_2^{ed} ...}{b_1 b_2 ...} \pmod n$$

Alice is left with the product of the signatures of the hashes.

3.2.4 Laurie blinding

Laurie [8] proposed an alternative to Chaum's blinding technique in an effort to avoid the patent on blind signatures. Since the end result is not a signature which can be verified by a third party, it might not be covered by Chaum's patent. Laurie's technique isn't as well established or tested as Chaum's technique.

One implementation using Laurie's system requires the issuer to help in the credential verification process. Such an implementation must be an on-line system so that the verifier and issuer can work together to verify credentials.

Another option [3] involves Alice returning her blindly signed credentials to the issuer at a later time. He checks that the signature is valid (but, like any other person she could show credentials to, doesn't learn anything about the selective disclosure attributes). Then he signs the document with his RSA key. Now Alice has a traditionally signed credential, just as she would have obtained had she used Chaum's blinding technique. Thus, this approach may not be as immune to Chaum's patents.

[3] Suggested in an anonymous post to the Coderpunks e-mail list on 14 Dec 1999.

To use Laurie's blinding techniques (as described in sections 2.1 and 4.1 of his paper), Izzy creates public values p, g and g^k $(mod\ p)$ such that:

$$\begin{aligned}
&p\ is\ prime\\
&\frac{p-1}{2}\ is\ prime\\
&g^2 \neq 1 \quad (mod\ p)\\
&g^{(p-1)/2} \equiv 1 \quad (mod\ p)\\
&k\ lies\ in\ [2, (p-1)/2)
\end{aligned}$$

k is Izzy's secret exponent.

To blind a value h, Alice chooses a blinding exponent b to use with the generator g:

$$r = hg^b \quad (mod\ p)$$

For completeness, we also specify that:

$$r^{(p-1)/2} \equiv 1 \quad (mod\ p)$$

When Alice sends r to Izzy for signing, Izzy chooses a random integer x such that

x lies in $[(log_g p) + 1, (p-1)/2 - (log_g p) - 1]$

He also calculates:

$$\begin{aligned}
t =& \quad \tfrac{k}{x} && (mod\ \frac{p-1}{2})\\
c =& \quad r^x && (mod\ p)\\
d =& \quad g^x && (mod\ p)
\end{aligned}$$

He sends the values c and d to Alice. Alice requests x or t at random, and Izzy sends it to her. If she requested x, she checks that:

$$c = r^x \quad (mod\ p)$$

and

$$d = g^x \quad (mod\ p)$$

If she requested t, she verifies that:

$$d^t = g^{xt} = g^k \quad (mod\ p)$$

and

$$c^t = r^{xt} = r^k \quad (mod\ p)$$

They repeat this protocol n times to show that Izzy has a probability of cheating of 1 in 2^n. When the protocol terminates, Alice takes one of the values:

$$\begin{aligned}
c^t =& \quad r^k && (mod\ p)\\
=& \quad (hg^b)^k && (mod\ p)
\end{aligned}$$

and unblinds it to produce h^k $(mod\ p)$:

$$\begin{aligned}
(hg^b)^k =& \quad h^k g^{bk} && (mod\ p)\\
=& \quad h^k g^{kb} && (mod\ p)
\end{aligned}$$

$$h^k = \quad \tfrac{h^k g^{kb}}{g^{kb}} \quad (mod\ p)$$

Since g^k is a public value, it's easy for her to compute g^{kb}. Working with the product of hashes works just as in Chaum's system:

$$(r_1 r_2 r_3 ...)^k = h_1^k g^{b_1 k} h_2^k g^{b_2 k} ... \quad (mod\ p)$$

$$h_1^k h_2^k ... = \frac{h_1^k g^{b_1 k} h_2^k g^{b_2 k} ...}{g^{kb_1} g^{kb_2} ...} \quad (mod\ p)$$

Note that only Izzy can verify if his signature is correct, since only he knows k.

4 Credential Sets

Alice may have several credentials issued by different issuers. In certain circumstances she may have to show attributes about herself that reside on different credentials. To prevent Alice and Bob from pooling their credentials to obtain services neither could obtain alone, we require that certificates shown together must have been issued as part of a Credential Set. Credentials in a set can be issued by different issuers, but will all be provably linked. As long as the issuers are trustworthy in following the issuing protocol, proof of set membership is sufficient to show that the credentials were issued to the same person.

Alice will build a credential set request (CSR) to show to the issuers of each of her credentials. Issuers sign the portions of the CSR that correspond to the credentials they issue. Alice can then unblind the credentials and show them to obtain services. The CSR and the showing protocol provide probabilistic evidence that Alice has properly created the credentials she wishes to have signed, and that the credentials were issued to the same person.

4.1 The Credential Set Request

Alice first obtains an identity document, defined as any document sufficient to identify Alice to all her issuers. This could be a selective disclosure certificate, but there's no reason to have it signed blindly since its purpose is to identify its owner. The purpose of the credential set is to prove that all the elements of the set were issued to the person owning the identity document. The identity document might include different forms of identification - for example, a social security number as well as a driver's licence number. Different issuers can use different elements of the identity document to verify Alice's identity, so it's important that the issuer of the identity document ensure that Alice can't get Bob's SSN included in her identity document.

She also generates a random value, and obtains a random value from each of her issuers. The output of *oneway*() given the identity document and random values becomes the Master ID for the credential set. The random values ensure that the Master ID is different for every CSR she creates. This is necessary to prevent attacks which would allow Alice to defeat the credential set mechanism. Later, each issuer will check that Alice used the random value that issuer gave her. Since the issuer doesn't reuse random values, he knows that at least part of the input to *oneway*() was unique, and therefore that the Master ID is also (with overwhelming probability) unique.

Alice uses *commit*() with the Master ID and n different random strings to produce credential IDs. That is, Alice generates $id_1 ... id_n$ such that every $id_k = commit(master\_id)$.

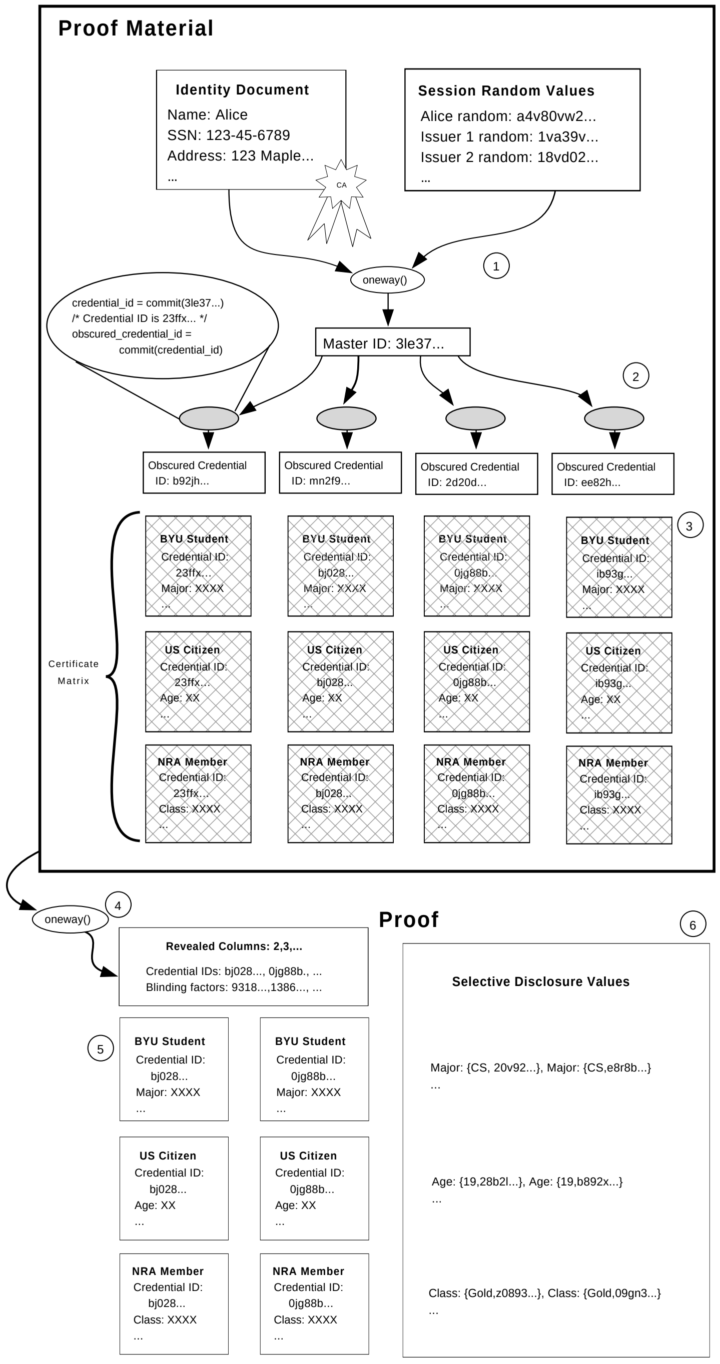
She then calls commit() again to commit to the credentil IDs. That is, she calculates v_k such that $v_k = commit(id_k) = commit(commit(master\_id))$. The first call to commit() prevents anyone seeing a credential ID from knowing what Master ID it descends. The second call prevents the issuer from being able to associate credential IDs Alice reveals during the showing protocol with the unrevealed commitments in her CSR.

Next she creates n certificates (which are actually more closely related to the common notion of certificate *requests*) which will be used together to create a single credential, just as we described in the cut-and-choose protocol. The certificates $1...n$ include the credential IDs $1...n$.

Alice repeats the process for each of the m credentials she wishes to have in her credential set. The mxn blinded certificates can be thought of as an m by n matrix, where each row represents a single credential. Each column consists of one element of each credential, all with the same credential ID. Alice uses the same blinding factor for each certificate in a column.

The identity document, random values, Master ID, obscured credential IDs and blinded certificates are combined to form the proof material of a Credential Set Request. Alice uses this proof material as input to the one-way function used in the non-interactive cut and choose protocol.

The Credential Set Request



Key:

- ⬭ Procedure
- ▭ Data
- ▨ Blinded data

Notes:

1. The identity document is the basis for membership in the credential set. The session random values make the master ID unique for each issuing session.

2. Credential IDs descend from the master ID. Commitments to the IDs are included in the proof material. The credential ID appears in each certificate in a column.

3. Half of the columns in the certificate matrix are used to prove the accuracy of the CSR. The other half will be signed by the issuers. A real certificate matrix would have many more columns than are shown here.

4. All the proof material together feeds oneway(), just as in a noninteractive zero knowledge proof, to determine which columns Alice must reveal.

5. Alice includes the certificates and blinding factors used to create the columns of the certificate matrix to be revealed.

6. Alice must also reveal the selective disclosure values for each certificate and the random strings used in commit() so that the issuers can verify the attributes in each certificate.

The output of the one-way function selects n/2 of the columns of the matrix which Alice must reveal. To wit, she appends the credential IDs for each column and the *commit*() preimages which prove that they descend from the Master ID and produce the values in the proof material, and the blinding factors for the columns of the matrix to be revealed. All of this information forms a complete Credential Set Request.

4.2 Issuing

In the issuing protocol, Alice sends her CSR to each of the issuers of her credentials. Each issuer will examine the CSR to verify its accuracy, then sign his row(s) of the certificate matrix. Alice can then unblind the signature and use it along with the corresponding certificates as a valid, signed credential.

Alice sends the following to each issuer:

- the Credential Set Request
- the certificates which form the preimages of the blinded hashes in the revealed columns and row corresponding to the credential to be issued
- the preimages of the selective disclosure fields in each revealed certificate which the server needs to verify

The issuer verifies the following:

- the random value Alice included was actually issued by that issuer for the current transaction
- the revealed credential IDs were properly generated
- the certificates are of the proper form (ie., each attribute value is accurate, and the ID in each certificate matches the credential ID for that row)
- the certificates hash to the value obtained by unblinding the corresponding element of the matrix

It then multiplies the remaining blinded hashes in the row together, signs the product and returns the signature to Alice. Alice can then divide the blinding factors out of the issuer's signature to obtain the signature for the product of hashes of the certificates. The certificates and signature together form a proper credential.

Alice repeats the process for each issuer.

Each issuer has now signed documents it has never seen, but whose values are almost certainly (for large enough n) either:

- correct, or
- inconsistent among the elements of the row(and the showing protocol states that any such inconsistency invalidates the credential.)

4.3 Proving ownership of credentials

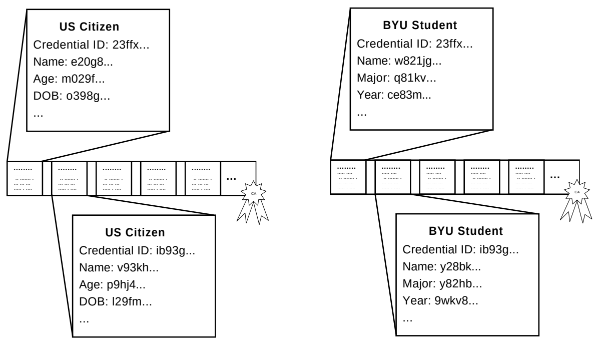
Traditionally, certificates contain a public key whose corresponding private key is known to the rightful certificate owner. This allows Alice to show a certificate to Steve and prove her ownership of it. Steve can keep a copy of the certificate, but can't claim ownership unless he can discover Alice's secret key.

With credential sets, Alice still needs to prove ownership of her credentials, but doesn't want to reveal uniquely identifying information about herself in the process. She can do this by creating a new key pair whose public key will be included in each credential in her set. She stores it as a selective disclosure value in her credentials so that Izzy doesn't see it when signing them. Later she'll reveal it to Steve when he demands that she prove ownership of her credentials.

4.4 Showing

Alice now has a signed credential set, and can show a subset of these credentials to Steve in order to obtain a service. She sends the credentials to Steve, who verifies the issuer signatures on them, possibly with the issuers' help. She also may choose to reveal

Showing Protocol



Selective Disclosure Values

Us Citizen:
Age: {19,t092g...} Age: {19,nl82g...} Age: {19,uzlq8...} Age: {19,t38gb...} ...

BYU Student:
Major: {CS,2i09g...} Major: {CS, 10vj3...} Major: {CS,g81lg...} Major: {CS,10g93j...} ...
Year: {Senior,r209h...} Year: {Senior,x83hg...} Year: {Senior,pk91k...} Year: {Senior,01kvb...} ...

Notes:

After the issuers sign Alice's credentials, she removes the blinding factors on the signature. She sends Steve the unblinded signature, the certificates which were signed, and the values for the selective disclosure fields she wishes to reveal (along with the random strings used in commit()).

Steve checks to make sure that the Credential IDs match for all the credentials Alice presents. In this example, the ID for the first column is 23ffx... and the ID for the second column (originally the 4th column in the CSR) is ib93g....

For each certificate, Steve runs the actual value and random string provided for each selective disclosure field through oneway() and checks that the result is identical to the value in the certificate. In this example, he would begin by checking the age field, verifying that oneway(19 . t092g...) == m029f..., then that oneway(19 . nl82g...) == p9hj4..., etc.

the values of selective disclosure fields. Steve verifies that the credentials are properly constructed and challenges Alice's ownership of them.

Remember that each credential is comprised of n/2 selective disclosure certificates and the signature on the product of their hashes. Steve must verify the signature as well as that the n/2 certificates are equivalent (the final step of the cut-and-choose protocol).

If Chaum's blinded signatures were used in the issuing process, Steve verifies the credential signature just as for any other RSA signature, by raising the signature to the issuer's public exponent (modulo its public modulus) and checking that the value equals the product of certificate hashes.

If Laurie's alternative blinding technique was used without the option of returning the credential to the issuer for conventional signing (as mentioned in section 3.2.4), Steve forwards the product of hashes and the signature to the issuer for verification.

Steve also checks that the preimages Alice sent for each selective disclosure field specify the same value for each certificate in the row and that they hash to the values in the certificates. This means that he must run *oneway*() on n/2 preimages for every selective disclosure field of every credential which Alice wishes to reveal.

Next Steve verifies that the presented rows came from the same credential set by verifying that the credential ID is the same for all certificates in a column.

Finally, Steve challenges Alice's ownership of the set by means of the public key included in each credential.

4.4.1 Credential re-use

All the credentials issued during the issuing process have the same credential IDs so that Alice can prove they belong together in the set. These IDs are different for each instance of the issuing protocol, however. Alice loses some privacy if she shows credentials from the same set more than once, since the people she shows them to could compare the credential IDs and determine that they were dealing with the same person.

For maximum privacy, then, Alice should go through the issuing protocol with her issuers multiple times and obtain many instances of her credential set. After showing any credential from her set, she discards that entire instance of the set. Credentials from different instances of the issuing protocol can't be linked, except by the attribute values Alice reveals during the showing process. That is, Alice can obviously be traced if she always reveals the SSN field of her credentials, since that attribute is unique to her. But if she reveals only that her hair is brown to both Steve and Sam, they can't tell whether they were dealing with the same person.

4.5 Preventing Credential Pooling

There are several ways Alice might try to pool her credentials with Bob to obtain services neither could obtain alone, or to get an issuer to sign an untrue credential.

Alice could attempt to create a Credential Set Request with fake uninspected columns. Abusing the non-interactive cut and choose protocol, Alice can spend as long as she wishes constructing different Credential Set Requests in which half the columns of the credential matrix have bogus certificates. Each attempt has a 1 in (n choose n/2) chance that the output of *oneway*() will select just the valid columns of the matrix for inspection, leaving the issuer to sign an untrue credential. She can perform this attack offline, without interacting with any other entities. The probability of success can be reduced to an acceptable level by choosing a sufficiently large n.

Alice could try to piece together a bogus credential one column at a time. She creates a CSR with a single column containing bogus certificates. She throws it away and starts over if *oneway*() selects that column for inspection (which it will do with 50% probability). After the issuers sign the credentials in the request, she isolates the signature on just that certificate in each credential. She repeats the process until she has n/2 signatures, then multiplies them together to obtain a signed fake credential. Isolating individual signatures is the hard part, and she has no better chance of success than someone trying to determine the blinding factor applied to a document he has been asked to sign.

Bob could tell Alice the Credential IDs for a Credential Set of his so that Alice can attempt to include those same Credential IDs in a CSR of her own. However, she can't determine before creating the CSR what columns will be required for inspection by *oneway*(). As in the first example, she has only a 1 in (n choose n/2) chance of success with each attempt.

5 Revocable anonymity

In revocable anonymity, the server and a quorum of authorities can agree to discover additional information about the presenter of a credential.

Revocable anonymity can be implemented by requiring that a field of a credential be a piece of encrypted personally identifying information. For instance, Alice could encrypt her social security number using the public keys of several different government agencies. To prove the field's validity to an issuer as required during the Credential Set issuing protocol, Alice sends the issuer the SSN and random padding used during encryption. The issuer verifies the value by encrypting with the same keys Alice used and ensuring the values are equal.

6 Performance

Here are some size and performance estimates for issuing and showing a CSR with 3 credentials, using $n = 256$ columns and Chaum-style blinding. We assume each certificate in the matrix is 1k bytes in length, and has 8 selective disclosure fields whose actual values are relatively short. The random strings used in *commit*() are each 16 bytes.

The entire CSR contains $3 * 256 = 768$ certificates totalling 768k bytes. Their blinded hashes (which populate the certificate matrix) will each be approximately as large as the modulus of the issuer's signing key. For a 1024 bit key, this would come to $768 * 128 = 96k$ bytes. Each certificate requires $8*16 = 128$ bytes to store its selective disclosure random strings, for a total of another $768 * 128 = 96k$ bytes. Including the rest of the CSR overhead (including the identity document, random strings, blinding factors, etc.), the client needs to store about a megabyte of data for the CSR and all its auxiliary information.

During the issuing protocol, Alice will need to send each issuer the CSR and half the certificates for his row, which comes to a little over half a megabyte of network traffic. To verify the CSR, each issuer must verify, hash and blind the revealed certificates for his row, and sign the unrevealed columns. Checking the selective disclosure fields (assuming they examine all 8) requires $128 * 8 = 3k$ calls to oneway() per issuer. Hashing, blinding and signing require an additional 128 calls to $oneway()$, 128 blinding operations (each consisting of a modular multiplication and exponentiation), and another 128 multiplications plus a single signing operation to generate the signature. To unblind each signature, Alice must then perform 128 modular divisions.

During the issuing protocol, assuming Alice wants to show Steve all three of her credentials, Alice has to send $3*128 = 384$ certificates plus the three credential signatures, for a little over 384k bytes of network traffic. If she discloses all the selective disclosure fields in each credential, she'll also have to send him the $384 * 8 = 3k$ random values totalling $3k * 16 = 48k$ bytes. Steve will have to call $oneway()$ for each of the 3k fields, then once for each certificate. Verifying the signature on each credential requires him to do 128 modular multiplications and one modular exponentiation using the issuer's public key.

This is a rather expensive system by today's standards, especially when using credential sets only once as we recommend. Personal computers with broadband network connections shouldn't have too much trouble with the storage and computational requirements, but they're probably prohibitive for embedded systems such as PDAs and smart cards.

7 Conclusions and future work

The non-interactive cut and choose protocol and notion of a Certificate Set Request are new primitives, and may find application in other areas of public key cryptography.

Though credential sets can be rather expensive computationally, they achieve their goal of protecting users' privacy while solving the problem of credential pooling. Our credentials individually behave much like traditional X.509v3 certificates, making it feasible to adapt existing certificate systems to work with credential sets. Work has begun on a simple free software implementation of the system presented here. We plan to make it available as a completely patent-free solution.

We sincerely hope that our work will encourage the spread of free privacy-protecting security systems. The work of Camenisch and Lysyanskaya is also promising, though quite a different approach from our own. Our systems prevent one of the major forms of credential misuse, making privacy attainable without compromising security.

References

[1] Stefan Brands. A technical introduction to digital credentials. http://www.xs4all.nl/ brands/overview.pdf.

[2] Stefan Brands. *Rethinking Public Key Infrastructures and Digial Certificates — Building in Privacy.* MIT Press, 2000.

[3] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Lecture Notes*

in Computer Science, 2045:93–118, 2001. http://citeseer.nj.nec.com/camenisch01efficient.html.

[4] Jan L. Camenisch, Jean-Marc Piveteau, and Markus A. Stadler. Blind signatures based on the discrete logarithm problem. *Lecture Notes in Computer Science*, 950:428–432, 1995. http://citeseer.nj.nec.com/camenisch94blind.html.

[5] David Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the Association for Computing Machinery*, 28(10):1030–1044, October 1985.

[6] David Chaum. Blind signature systems. *US Patent 4,759,063*, July 1988.

[7] David Chaum. Blind unanticipated signature systems. *US Patent 4,759,064*, July 1988.

[8] Ben Laurie. Lucre: Anonymous electronic tokens. http://anoncvs.aldigital.co.uk/lucre/theory2.pdf.

[9] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, second edition, 1996.