Towards a Logic of Privacy-Preserving Selective Disclosure Credential Protocols

Theodoros Balopoulos Stephanos Gritzalis
Dept. of Information and Communication Systems Engineering,
University of the Aegean, Samos, Greece
{*tbalopoulos,sgritz*}*@aegean.gr*

Abstract

This paper presents a first approach towards a logic suited for protocols aiming to achieve selective disclosure of credentials while preserving privacy. The analysis draws from the BAN and related logics [2, 10] that are targeted to aid reasoning about authentication protocols, as well as from formal methods on PKIs [5, 6]. The family of protocols directly covered are built using selective disclosure certificates, blind signatures and oneway hash functions as cryptographic primitives. The logic is able to prove that if the protocol's credentials are properly constructed and signed by trusted issuers, they should convince a verifier; furthermore, it provides a framework on which mechanized attacks against privacy may be attempted by an automatic theorem prover. The running example is a protocol by Holt and Seamons, specified in [4].

1. Overview

Although digital certificates are by far the most promising technique for safeguarding electronic communications and transactions, they cause a severe privacy concern: each digital certificate can be traced uniquely to its owner, and can be followed around as it moves through the system. To come around this situation, we need to build digital certificates, or more generally credentials, that preserve privacy by allowing the owner of a certificate to choose what information to disclose and to whom to disclose it [1]. This paper is about formal reasoning [7, 3] on the protocols specified by such credential systems.

2. The Primitives

This section presents the primitives that can be used to build a privacy-preserving selective disclosure protocol.

2.1. The `oneway` and $\oplus$ Functions

We model the `oneway` function to be a bijection with the extra security property that its inverse is not easily computable, i.e. that it can *only* be evaluated by building the table of mappings of the forward function and looking for the desired image (brute force). Furthermore, we model the $\oplus$ function to be a bijection which takes any number of arguments.

We will use the notation $\Updownarrow\mathtt{f}$ to denote that $\mathtt{f}$ is a bijection, $\mathtt{f}^{-1}$ to denote the inverse function of $\mathtt{f}$, and $\downarrow\mathtt{f}$ to denote that $\mathtt{f}$ is easily computable in the sense described above. We can now state the following rules:

$$\frac{\Updownarrow\mathtt{f}}{\Updownarrow\mathtt{f}^{-1}} \quad \text{(R1)}$$

$$\frac{\Updownarrow\mathtt{f} \quad \mathtt{f(a)} = \mathtt{f(b)}}{\mathtt{a} = \mathtt{b}} \quad \text{(R2)}$$

$$\frac{\Updownarrow\mathtt{f}}{\mathtt{f}(\mathtt{f}^{-1}(\mathtt{x})) = \mathtt{x}} \quad \text{(R3)}$$

We can also state the following axioms for our functions:

$$\frac{}{\Updownarrow\mathtt{oneway} \wedge \downarrow\mathtt{oneway}} \quad \text{(A1)}$$

$$\frac{}{\Updownarrow\oplus \wedge \downarrow\oplus \wedge \downarrow\oplus^{-1}} \quad \text{(A2)}$$

Using (R1), (R2), (A1) and (A2), we can prove the following theorem:

$$\frac{\mathtt{oneway}\left(\oplus_{\mathtt{i}=1}^{\mathtt{n}}\mathtt{a_i}\right) = \mathtt{oneway}\left(\oplus_{\mathtt{i}=1}^{\mathtt{n}}\mathtt{b_i}\right)}{\forall\, \mathtt{i} \in [\mathtt{1..n}]\,.\,\mathtt{a_i} = \mathtt{b_i}} \quad \text{(T1)}$$

In practise, `oneway` will be replaced by a hash function and $\oplus$ will be replaced by a function performing concatenation. A hash function may not be a bijection (because

the image is of fixed size), but it resembles one in the sense that it is (practically) impossible to find two different inputs that map to the same image. Moreover, a function performing concatenation may not be a bijection (because there are many possible ways we can separate the inputs from the image), but we only require it to be a bijection so that its image is able to hold all the information of the inputs—and the concatenation function manages to hold most of this information.

2.2. Signatures and Blind Signatures

Signatures can be modelled with a signing function `sign` that accepts as arguments a message and a private key and returns the corresponding signature, and a signature validation boolean function `sign_valid` that accepts as arguments the message, the public key and the signature. These two functions are related by the following rule ($\texttt{key}^{-1}$ denotes the private key that corresponds to the public key `key`):

$$\frac{}{\texttt{sign\_valid}(\texttt{msg}, \texttt{key}, \texttt{sign}(\texttt{msg}, \texttt{key}^{-1}))} \quad \text{(R4)}$$

Blind signatures can be modelled based on Chaum's blinding [8, 9]. We need the blinding function `blind` that accepts as arguments a message, a blinding factor and the public key of the entity that will do the signing, and returns the message blinded. We also need the unblinding function `unblind` that accepts as arguments the signature of the blinded message and the blinding factor, and returns the signature of the (unblinded) message. These functions are related as shown below:

$$\frac{\texttt{sign\_valid}(\texttt{blind}(\texttt{msg}, \texttt{factor}, \texttt{key}), \texttt{key}, \texttt{sgn})}{\texttt{sign\_valid}(\texttt{msg}, \texttt{key}, \texttt{unblind}(\texttt{sgn}, \texttt{factor}))}$$

This rule might be satisfying in providing a primitive for blind signatures; however, the mathematics behind Chaum's blinding allow us to use the more general rule given below:

$$\frac{\texttt{sign\_valid}\left(\prod_{\texttt{i}=1}^{\texttt{n}} \texttt{blind}(\texttt{msg}_\texttt{i}, \texttt{factor}, \texttt{key}), \texttt{key}, \texttt{sgn}\right)}{\texttt{sign\_valid}\left(\prod_{\texttt{i}=1}^{\texttt{n}} \texttt{msg}_\texttt{i}, \texttt{key}, \texttt{unblind}(\texttt{sgn}, \texttt{factor})\right)} \quad \text{(R5)}$$

We also include an axiom that says that all these functions are easily computable:

$$\frac{}{\downarrow\texttt{sign} \wedge \downarrow\texttt{sign\_valid} \wedge \downarrow\texttt{blind} \wedge \downarrow\texttt{unblind}} \quad \text{(A3)}$$

2.3. The Selective Disclosure Certificate

A selective disclosure certificate `cert` will be modelled as having three fields: the issuer—denoted by $\texttt{I}(\texttt{cert})$, the committed attribute—denoted by $\texttt{A}(\texttt{cert})$, and the credential-id—denoted by $\texttt{C}(\texttt{cert})$. The signature will not be part of the certificate.

The committed attribute is intended to be used in such a way that, on one hand, the real attribute cannot be deduced from it, and, on the other hand, the entity that has committed to it is able to convince other entities of the real attribute's value. In the discussion that follows, we will assume that an entity that creates a certificate `cert` and wants to commit to a real attribute denoted by $\texttt{attrib}(\texttt{cert})$, will compute the committed attribute as illustrated by the formula

$$\texttt{A}(\texttt{cert}) = \texttt{oneway}(\texttt{attrib}(\texttt{cert}) \oplus \texttt{random}(\texttt{cert}))$$

where $\texttt{random}(\texttt{cert})$ denotes a random value the entity generates for this particular certificate and does not disclose to any other entity.

Notice that one disclosed attribute is enough for our modelling purposes: if more attributes are needed, one may use more certificates in his model.

2.4. Relationships between Entities and Data

Having `P` can *have* the value of an expression if he can compute the expression by himself, or if he received this value during the execution of a protocol[1].

If `P` *has* the arguments of an easily computable function, then he also *has* their image under this function. Using the notation $\texttt{x}_{1..\texttt{n}}$ or $(\texttt{x}_1, \ldots, \texttt{x}_\texttt{n})$ to denote the `n` arguments of a function, we have the following rules:

$$\texttt{P}\ \textit{has}\ \texttt{x}_{1..\texttt{n}} \Leftrightarrow \forall\, \texttt{x} \in [\texttt{1..n}]\,.\,\texttt{P}\ \textit{has}\ \texttt{x}_\texttt{i} \quad \text{(R6)}$$

$$\frac{\texttt{P}\ \textit{has}\ \texttt{x}_{1..\texttt{n}} \quad \downarrow\texttt{f}}{\texttt{P}\ \textit{has}\ \texttt{f}(\texttt{x}_{1..\texttt{n}})} \quad \text{(R7)}$$

Using (R3) and (R7), we can prove the following theorem:

$$\frac{\texttt{P}\ \textit{has}\ \texttt{f}(\texttt{x}_{1..\texttt{n}}) \quad \Updownarrow\texttt{f} \quad \downarrow\texttt{f}^{-1}}{\texttt{P}\ \textit{has}\ \texttt{x}_{1..\texttt{n}}} \quad \text{(T2)}$$

Notice that if `P` *has* $\texttt{oneway}(\texttt{x})$, it does not follow that `P` *has* `x`, because $\downarrow\texttt{oneway}^{-1}$ is not true.

[1] (M1) and (M2) formalize the latter case.

Deducing P can *deduce* that an expression is true, either because he trusts someone else to have verified it for him[2], or because he is able to evaluate the expression by himself. In the latter case, in order for P to deduce the expression, the expression must be true, and P must *have* the expression, in the sense of rule (R7):

$$\frac{\mathtt{x} \quad \mathtt{P}\ \textit{has}\ \mathtt{x}}{\mathtt{P}\ \textit{deduces}\ \mathtt{x}} \tag{R8}$$

The notion of this rule is that if P and Q participate in a protocol and Q deduces an expression, then P will be able to deduce it as well if Q sends P all the necessary information.

We also include the following rules about deduction in our logic:

$$\mathtt{P}\ \textit{deduces}\ (\mathtt{x} \wedge \mathtt{y}) \Leftrightarrow \mathtt{P}\ \textit{deduces}\ \mathtt{x} \wedge \mathtt{P}\ \textit{deduces}\ \mathtt{y} \tag{R9}$$

$$\frac{\mathtt{P}\ \textit{deduces}\ \mathtt{x} \quad \mathtt{x} \Rightarrow \mathtt{y}}{\mathtt{P}\ \textit{deduces}\ \mathtt{y}} \tag{R10}$$

Identifying P can *identify* a piece of data. If this is the case, P can also *identify* the image of this piece of data under a bijection, assuming the bijection is easily computable:

$$\frac{\mathtt{P}\ \textit{identifies}\ \mathtt{x} \quad \Updownarrow\mathtt{f} \quad \downarrow\mathtt{f}}{\mathtt{P}\ \textit{identifies}\ \mathtt{f}(\mathtt{x})}$$

More generally, he can *identify* the image of a number of arguments under an easily computable bijection, if he can *identify* at least one of these arguments and *have* the rest:

$$\frac{\begin{array}{c}\exists\ \mathtt{i} \in [\mathtt{1..n}]\ .\ \mathtt{P}\ \textit{identifies}\ \mathtt{x_i} \quad \Updownarrow\mathtt{f} \quad \downarrow\mathtt{f} \\ \forall\ \mathtt{i} \in [\mathtt{1..n}]\ .\ (\mathtt{P}\ \textit{identifies}\ \mathtt{x_i} \vee\ \mathtt{P}\ \textit{has}\ \mathtt{x_i})\end{array}}{\mathtt{P}\ \textit{identifies}\ \mathtt{f}(\mathtt{x}_{\mathtt{1..n}})} \tag{R11}$$

Using (A1), (A2) and (R11), we can prove the following theorem:

$$\frac{\begin{array}{c}\exists\ \mathtt{i} \in [\mathtt{1..n}]\ .\ \mathtt{P}\ \textit{identifies}\ \mathtt{x_i} \\ \forall\ \mathtt{i} \in [\mathtt{1..n}]\ .\ (\mathtt{P}\ \textit{identifies}\ \mathtt{x_i} \vee\ \mathtt{P}\ \textit{has}\ \mathtt{x_i})\end{array}}{\mathtt{P}\ \textit{identifies}\ \mathtt{oneway}\left(\oplus_{\mathtt{i=1}}^{\mathtt{n}} \mathtt{x_i}\right)} \tag{T3}$$

The notion of identification is similar to the notion of freshness in the BAN logic [2, 10]; however, we distinguish two important differences:

- Entities are able to identify images of data under any bijection, not only under conjunction (which in BAN logic is a bijection). This is reasonable because a bijection does not throw away any information.
- Identification is a more general notion: whether it is used for freshness and/or another purpose depends on the reasoning behind the initial state's assumptions.

Trusting P can *trust* a certificate's issuer. It is implied that P *has* the public key of the issuer of the certificate he *trusts*:

$$\frac{\mathtt{P}\ \textit{trusts}\ \mathtt{I(cert)}}{\mathtt{P}\ \textit{has}\ \mathtt{key}_{\mathtt{I(cert)}}} \tag{R12}$$

Validating P will *validate* a certificate if he *deduces* that it is validly signed:

$$\frac{\mathtt{P}\ \textit{deduces}\ \mathtt{sign\_valid}(\mathtt{cert}, \mathtt{key}_{\mathtt{I(cert)}}, \mathtt{sgn})}{\mathtt{P}\ \textit{validates}\ \mathtt{cert}} \tag{R13}$$

We will also define the notion of P *validating* a list of certificates signed *together*—such a list of n certificates will be denoted as $\mathtt{cert}_{\mathtt{1..n}}$. All the certificates must include the (same) signer in their issuer field. We also allow the signature to be taken on the product of the certificate images:

$$\begin{array}{l}\mathtt{n} > \mathtt{1} \Rightarrow \Big[\mathtt{P}\ \textit{validates}\ \mathtt{cert}_{\mathtt{1..n}} \Leftrightarrow \\ \qquad \mathtt{P}\ \textit{deduces}\ \mathtt{sign\_valid}\Big(\prod_{\mathtt{i=1}}^{\mathtt{n}} \mathtt{f}(\mathtt{cert_i}), \mathtt{key}_{\mathtt{I}(\mathtt{cert_i})}, \mathtt{sgn}\Big) \\ \qquad\qquad \wedge \Big(\forall\ \mathtt{i}, \mathtt{j} \in [\mathtt{1..n}]\ .\ \mathtt{I}(\mathtt{cert_i}) = \mathtt{I}(\mathtt{cert_j})\Big)\Big]\end{array} \tag{R14}$$

P will also *validate* a certificate if P has *validated* the list of certificates it belongs to:

$$\frac{\mathtt{P}\ \textit{validates}\ \mathtt{cert}_{\mathtt{1..n}}}{\forall\ \mathtt{i} \in [\mathtt{1..n}]\ .\ \mathtt{P}\ \textit{validates}\ \mathtt{cert_i}} \tag{R15}$$

Therefore, P can *validate* a single certificate either by checking its signature and using rule (R13), or by checking the signature of the list this certificate belongs to and using rules (R14) and (R15). Only knowing that P *validates* a certificate does not allow us to distinguish between these two cases.

3. Protocol Specification

The protocol will be specified by a list of message passings; each message passing may have one of the following forms:

[2] This is the case with rule (R16). Our logic needs to have such rules to handle trust, hence it is not a logic of knowledge [10] in the sense that $(P\ \textit{deduces}\ x) \Rightarrow x$ is not a theorem.

$$\mathtt{L_i} \;\rightarrow\; \mathtt{R_i} \;:\; \mathtt{m_i}$$
$$\mathtt{L_i} \;\rightarrow\; \mathtt{R_i} \;:\; \mathtt{c_i} \,?\, \mathtt{m_i}$$

The first form denotes that entity $\mathtt{L_i}$ sends to entity $\mathtt{R_i}$ message $\mathtt{m_i}$; the second denotes the same, subject to the condition that $\mathtt{c_i}$ holds. The subscript $\mathtt{i}$ indicates that this is the $\mathtt{i}^{\text{th}}$ message passing specified by the protocol.

The protocol specification is used to augment the set of valid statements, starting with the set of initial assumptions which will be denoted by $\mathcal{T}_0$. When a message passing of the first form is reached, the set of valid statements may be augmented as specified below:

$$\mathcal{T}_\mathtt{i} \equiv \mathcal{T}_{\mathtt{i}-1} \cup \left\{ \frac{\mathcal{T}_{\mathtt{i}-1} \vdash \mathtt{L_i}\ \mathit{has}\ \mathtt{m_i}}{\mathtt{R_i}\ \mathit{has}\ \mathtt{m_i}} \right\} \tag{M1}$$

where $\mathcal{T} \vdash \mathtt{e}$ denotes that $\mathtt{e}$ can be derived from $\mathcal{T}$. Similarly, when a message passing of the second form is reached, the following augmentation applies:

$$\mathcal{T}_\mathtt{i} \equiv \mathcal{T}_{\mathtt{i}-1} \cup \left\{ \frac{\mathcal{T}_{\mathtt{i}-1} \vdash (\mathtt{c_i} \wedge \mathtt{L_i}\ \mathit{has}\ \mathtt{m_i})}{\mathtt{R_i}\ \mathit{has}\ \mathtt{m_i}} \right\} \tag{M2}$$

The fully augmented set of valid statements—denoted by $\lceil \mathcal{T} \rceil$—can be used to prove desirable or undesirable properties of the protocol.

4. The Selective Disclosure Protocol of Holt and Seamons

We will first present a formalization of the selective disclosure credentials used in Holt and Seamons' protocol in Section 4.1, we will then give a formal specification of the protocol in Section 4.2, and finally we will state in our logic desirable and undesirable properties of the protocol in Section 4.3.

4.1. The Selective Disclosure Credential

We need our logic to cover the following types of credentials: a credential consisting of a single certificate, a credential consisting of $\mathtt{n}$ certificates issued by the same issuer, and a credential consisting of $\mathtt{n} \times \mathtt{m}$ certificates issued by $\mathtt{m}$ issuers.

Credential consists of a single Certificate If $\mathtt{P}$ *trusts* an issuer of certificates and $\mathtt{P}$ *validates* a certificate of this issuer, then $\mathtt{P}$ can *deduce* that $\mathtt{cert}$'s fields have appropriate values. Regarding the field denoted by $\mathtt{A(cert)}$, this translates to the following rule:

$$\frac{\mathtt{P}\ \mathit{trusts}\ \mathtt{I(cert)} \quad \mathtt{P}\ \mathit{validates}\ \mathtt{cert}}{\mathtt{P}\ \mathit{deduces}\ \mathtt{A(cert)} = \mathtt{oneway(attrib(cert)} \oplus \mathtt{random(cert))}} \tag{R16}$$

Using axioms (A1) and (A2), rules (R7), (R8), (R9), (R10), (R16), and theorem (T1), we can prove that someone can *deduce* the undisclosed attribute of a certificate if he *trusts* the certificate's issuer, if he *validates* the certificate, and if he has been given an undisclosed attribute and a random value that map correctly to the certificate's committed attribute. We therefore have the following theorem:

$$\frac{\mathtt{P}\ \mathit{trusts}\ \mathtt{I(cert)} \quad \mathtt{P}\ \mathit{validates}\ \mathtt{cert} \quad \mathtt{P}\ \mathit{has}\ (\mathtt{cert}, \mathtt{x}, \mathtt{y}) \quad \mathtt{A(cert)} = \mathtt{oneway(x} \oplus \mathtt{y)}}{\mathtt{P}\ \mathit{deduces}\ \mathtt{attrib(cert)} = \mathtt{x}} \tag{T4}$$

Credential consists of $\mathtt{n}$ Certificates The undisclosed attribute of a credential consisting of $\mathtt{n}$ certificates is defined to be the common undisclosed attribute of these certificates, if such a common value exists:

$$\frac{\exists\, \mathtt{x} \,.\, \forall\, \mathtt{i} \in [\mathtt{1..n}] \,.\, \mathtt{attrib(cert_i)} = \mathtt{x}}{\mathtt{attrib(cert_{1..n})} = \mathtt{x}} \tag{R17}$$

Using rules (R14), (R15), (R17) and theorem (T4), and splitting the two cases $\mathtt{n} = \mathtt{1}$ and $\mathtt{n} > \mathtt{1}$, we can prove the following theorem:

$$\frac{\begin{array}{c}\mathtt{P}\ \mathit{trusts}\ \mathtt{I(cert_1)} \quad \mathtt{P}\ \mathit{validates}\ \mathtt{cert_{1..n}} \\ \mathtt{P}\ \mathit{has}\ \mathtt{x} \quad \forall\, \mathtt{i} \in [\mathtt{1..n}] \,.\, \mathtt{P}\ \mathit{has}\ (\mathtt{cert_i}, \mathtt{y_i}) \\ \forall\, \mathtt{i} \in [\mathtt{1..n}] \,.\, \mathtt{A(cert_i)} = \mathtt{oneway(x} \oplus \mathtt{y_i)}\end{array}}{\mathtt{P}\ \mathit{deduces}\ \mathtt{attrib(cert_{1..n})} = \mathtt{x}} \tag{T5}$$

In the special case $\mathtt{n} = \mathtt{1}$, (T5) is derived trivially from (T4).

Credential consists of $\mathtt{n} \times \mathtt{m}$ Certificates The undisclosed attribute of a credential consisting of $\mathtt{m}$ rows of $\mathtt{n}$ certificates so that the certificates in each row are issued by the same entity, is defined to be the list of the undisclosed attributes of the $\mathtt{m}$ credentials consisting of $\mathtt{n}$ certificates, if the credential-ids of the certificates in each column are mutually *identifiable* by any entity:

$$\frac{\begin{array}{c}\forall\, \mathtt{i} \in [\mathtt{1..m}] \,.\, \mathtt{attrib(cert_{1..n,i})} = \mathtt{x_i} \\ \forall\, \mathtt{i} \in [\mathtt{1..n}] \,.\, \big[\mathtt{Q}\ \mathit{identifies}\ \mathtt{C(cert_{i,1})} \Rightarrow \\ \forall\, \mathtt{j} \in [\mathtt{1..m}] \,.\, \mathtt{Q}\ \mathit{identifies}\ \mathtt{C(cert_{i,j})}\big]\end{array}}{\mathtt{attrib(cert_{1..n,1..m})} = \mathtt{x_{1..m}}} \tag{R18}$$

4.2. Specifying the Protocol

Regarding the formalization of the protocol in the next subsections, the following points should be taken into account:

- In the issuing part of the protocol, we will not deal with how the attribute claimed by the client is verified by the issuer.
- We will not deal with how proof of ownership of the credential is achieved.
- The non-interactive cut-and-choose protocol will be modelled by the boolean function $\mathcal{H}$, which takes as arguments a *credential set request* (CSR) and a column number and returns true if and only if this column should be presented.

The rest of this Section assumes familiarity with [4].

The Issuing Sub-Protocol The issuing sub-protocol is ran once for each disclosed attribute required. $\mathtt{I_k}$ represents the issuer of the $\mathtt{k}^{\text{th}}$ disclosed attribute; A represents the issuers' client, i.e. the entity that wishes to acquire a selective disclosure credential. Note that the first message passing must be completed with each issuer before proceeding to the remaining message passings.

$$
\begin{array}{lcll}
\mathtt{I_k} & \rightarrow & \mathtt{A} & : \ \mathtt{\_session_k} \\
\mathtt{A} & \rightarrow & \mathtt{I_k} & : \ \mathtt{CSR}, \bar{\mathtt{h}}\mathtt{cred}_{1..n}, \bar{\mathtt{h}}\mathtt{credrnd}_{1..n}, \\
 & & & \quad \bar{\mathtt{h}}\mathtt{factor}_{1..n}, \bar{\mathtt{h}}\mathtt{cert}_{1..n,1..m}, \\
 & & & \quad \bar{\mathtt{h}}\mathtt{attrib}_{1..m}, \bar{\mathtt{h}}\mathtt{random}_{1..n,1..m} \\
\mathtt{I_k} & \rightarrow & \mathtt{A} & : \ \mathtt{VLD_k} \ ? \ \mathtt{SGN_k}
\end{array}
$$

where:

$$\mathtt{CSR} \equiv \mathtt{session}_{0..m}, \mathtt{obcred}_{1..n}, \mathtt{blindcert}_{1..n,1..m}$$

$$\mathtt{SGN_k} \equiv \mathtt{sign}(\textstyle\prod_{i=1;\neg\mathcal{H}(\mathtt{CSR},i)}^{n} \mathtt{blindcert}_{i,k}, \mathtt{key}_{\mathtt{I_k}}^{-1})$$

$$
\begin{aligned}
\mathtt{VLD_k} \equiv\ & \mathtt{I_k}\ \textit{deduces}\ \mathtt{CND_k} \\
& \wedge \forall\, \mathtt{i} \in [\mathtt{1..n}]\,.\ \mathcal{H}(\mathtt{CSR}, \mathtt{i}) \Rightarrow \\
& \qquad (\forall\, \mathtt{j} \in [\mathtt{1..m}]\,.\ \mathtt{I_k}\ \textit{identifies}\ \mathtt{C}(\bar{\mathtt{h}}\mathtt{cert}_{i,j}))
\end{aligned}
$$

$$
\begin{aligned}
\mathtt{CND_k} \equiv\ & \mathtt{session_k} = \mathtt{\_session_k} \\
& \wedge \forall\, \mathtt{i} \in [\mathtt{1..n}]\,.\ \mathcal{H}(\mathtt{CSR}, \mathtt{i}) \Rightarrow \\
& \quad (\mathtt{I}(\bar{\mathtt{h}}\mathtt{cert}_{i,k}) = \mathtt{I_k} \wedge \forall\, \mathtt{j} \in [\mathtt{1..m}]\,. \\
& \qquad [\mathtt{A}(\bar{\mathtt{h}}\mathtt{cert}_{i,j}) = \mathtt{oneway}(\bar{\mathtt{h}}\mathtt{attrib_j} \oplus \bar{\mathtt{h}}\mathtt{random}_{i,j}) \\
& \qquad\ \wedge \mathtt{blindcert}_{i,j} = \mathtt{blind}(\bar{\mathtt{h}}\mathtt{cert}_{i,j}, \bar{\mathtt{h}}\mathtt{factor_k}, \mathtt{key}_{\mathtt{I_k}})])
\end{aligned}
$$

The Showing Sub-Protocol The showing sub-protocol is run only once. V represents the verifier of the selective disclosure credential.

$$
\begin{array}{lcll}
\mathtt{A} & \rightarrow & \mathtt{V} & : \ \mathtt{cert}'_{1..n',1..m'}, \mathtt{attrib}'_{1..m'}, \\
 & & & \quad \mathtt{random}'_{1..n',1..m'}, \mathtt{sgn}_{1..m'}
\end{array}
$$

Initial State Assumptions In the case of a successful execution of the protocol, the set of initial assumptions $\mathcal{T}_0$ should contain the following:

1. Expressions that define $\bar{\mathtt{h}}\mathtt{cred}$, $\bar{\mathtt{h}}\mathtt{credrnd}$, $\bar{\mathtt{h}}\mathtt{factor}$, $\bar{\mathtt{h}}\mathtt{cert}$, $\bar{\mathtt{h}}\mathtt{attrib}$ and $\bar{\mathtt{h}}\mathtt{random}$—representing the information disclosed by the noninteractive cut and choose protocol—in terms of cred, credrnd, factor, cert, attrib and random—representing the full information.

   Here is the definition of $\bar{\mathtt{h}}\mathtt{cert}$; the others are defined in a similar way:

   $$\bar{\mathtt{h}}\mathtt{cert}_{i,j} = \begin{cases} \mathtt{cert}_{i,j} & \text{if } \mathcal{H}(\mathtt{CSR}, \mathtt{i}) \\ \epsilon & \text{otherwise} \end{cases}$$

2. Each issuer owns his private key, issuers' client has obtained the corresponding public keys, and the verifier trusts the issuers:

   $\mathtt{I_k}\ \textit{has}\ \mathtt{key}_{\mathtt{I_k}}^{-1} \wedge \mathtt{A}\ \textit{has}\ \mathtt{key}_{\mathtt{I_k}} \wedge \mathtt{V}\ \textit{trusts}\ \mathtt{I_k}$

3. Each issuer generates a fresh number and is able to identify it:

   $\mathtt{I_k}\ \textit{has}\ \mathtt{session_k} \wedge \mathtt{I_k}\ \textit{identifies}\ \mathtt{session_k}$

4. The same applies for the issuers' client:

   $\mathtt{A}\ \textit{has}\ \mathtt{session}_0 \wedge \mathtt{A}\ \textit{identifies}\ \mathtt{session}_0$

5. If the issuers' client is given all the fresh values, he is able to send to each issuer the information required by the protocol:

   $$
   \begin{aligned}
   & \mathtt{A}\ \textit{has}\ \mathtt{session}_0 \wedge \mathtt{A}\ \textit{has}\ \mathtt{session}_{1..m} \Rightarrow \\
   & \mathtt{A}\ \textit{has}\ (\mathtt{CSR}, \bar{\mathtt{h}}\mathtt{cred}_{1..n}, \bar{\mathtt{h}}\mathtt{credrnd}_{1..n}, \bar{\mathtt{h}}\mathtt{factor}_{1..n}, \\
   & \qquad \bar{\mathtt{h}}\mathtt{cert}_{1..n,1..m}, \bar{\mathtt{h}}\mathtt{attrib}_{1..m}, \bar{\mathtt{h}}\mathtt{random}_{1..n,1..m})
   \end{aligned}
   $$

6. The issuer's client will generate the information required by the protocol correctly (to be examined together with the definitions of $\mathtt{VLD_k}$ and $\mathtt{CND_k}$, and with theorem T3):

   $$
   \begin{aligned}
   & (\forall\, \mathtt{k} \in [\mathtt{1..m}]\,.\ \mathtt{CND_k}) \\
   & \wedge\ (\forall\, \mathtt{i} \in [\mathtt{1..n}]\,.\ \mathcal{H}(\mathtt{CSR}, \mathtt{i}) \Rightarrow \\
   & \qquad \forall\, \mathtt{j} \in [\mathtt{1..m}]\,.\ [\,\mathtt{C}(\bar{\mathtt{h}}\mathtt{cert}_{i,j}) = \\
   & \qquad\quad \mathtt{oneway}(\mathtt{id} \oplus (\oplus_{j=0}^{n}\mathtt{session_j}) \oplus \bar{\mathtt{h}}\mathtt{credrnd_i}) \\
   & \qquad\quad \wedge\ \mathtt{C}(\bar{\mathtt{h}}\mathtt{cert}_{i,j}) = \bar{\mathtt{h}}\mathtt{cred_i} \\
   & \qquad\quad \wedge\ \mathtt{obcred_i} = \mathtt{oneway}(\bar{\mathtt{h}}\mathtt{cred_i})])
   \end{aligned}
   $$

7. Each issuer will sign his credential:

   $$\forall\, \mathtt{k} \in [\mathtt{1..m}]\,. \quad \mathtt{SGN_k} = \mathtt{sign}\Big(\textstyle\prod_{i=1;\neg\mathcal{H}(\mathtt{CSR},i)}^{n} \mathtt{blindcert}_{i,k}, \mathtt{key}_{\mathtt{I_k}}\Big)$$

8. After deciding which attributes A wishes to show to V, we need to include:

- The expressions that define $\texttt{cert}'$, $\texttt{attrib}'$ and $\texttt{random}'$—representing a subset of the information *not* disclosed by the noninteractive cut and choose protocol—in terms of `cert`, `attrib` and `random`—representing the full information.
- The expression that defines `sgn`—representing the signatures of the certificates whose attribute was selected for disclosure—in terms of `SGN`—representing all available signatures.

4.3. Protocol Properties

We can now prove desirable properties, such as the one that follows (to be examined together with rule R18):

$$\lceil \mathcal{T} \rceil \vdash \texttt{V}\ \textit{deduces}\ \left(\exists\, \texttt{x}_{1..\texttt{m}}\,.\,\texttt{attrib}(\texttt{cert}'_{1..\texttt{n}',1..\texttt{m}'}) = \texttt{x}_{1..\texttt{m}}\right)$$

This says that the verifier is convinced that the real attributes of the certificates' owner are indeed the ones he claims.

We can also attempt to prove undesirable properties in order to find attacks against the our protocol. Consider the following:

$$\lceil \mathcal{T} \rceil \vdash \exists\, \texttt{k} \in [\texttt{1..m}]\,.\,\exists\, \texttt{i} \in [\texttt{1..n}]\,.\\ \neg\mathcal{H}(\texttt{CSR}, \texttt{i}) \wedge \exists\, \texttt{j} \in [\texttt{1..m}]\,.\,\texttt{I}_\texttt{k}\ \textit{identifies}\ \texttt{C}(\texttt{cert}_{\texttt{i},\texttt{j}})$$

This says that there exists an issuer that can identify a certificate that was not chosen to be disclosed by the noninteractive cut-and-choose protocol. If we could prove it, we would expose a serious protocol weakness, because the owner's identity would be exposed as soon as he used the certificate.

5. Conclusions and Future Work

This work is a first attempt to reason about privacy-preserving selective disclosure credential protocols. We believe that our logic has captured and formalized the primitive methods and relationships that such a protocol will make use of. After formalizing the running example protocol, we are able to prove that properly constructed certificates signed by trusted issuers should convince a verifier. However, we are not able to prove privacy; we are only able to provide a framework on which mechanized attacks may be attempted by an automatic theorem prover.

Future work may include the following:

- Doing research on how privacy can be proved.
- Expanding the logic so that we are able to reason about the issuing protocol in more depth.
- Expressing the logic and the example protocol in an automatic theorem prover and proving desirable or undesirable properties.

We hope that our work will encourage further research into the area of privacy-preserving security systems, and into formal methods for reasoning about them.

References

[1] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates—Building in Privacy*. MIT Press, 2000.

[2] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, Feb 1990.

[3] S. Gritzalis, D. Spinellis, and P. Georgiadis. Security protocols over open networks and distributed systems: Formal methods for their analysis, design, and verification. In *Computer Communications Journal*, volume 22, pages 697–709. Elsevier Science, May 1999.

[4] J. E. Holt and K. E. Seamons. Selective disclosure credential sets. *http://citeseer.nj.nec.com/541329.html*, 2002.

[5] C. Liu, M. A. Ozols, and T. Cant. An axiomatic basis for reasoning about trust in PKIs. In *Proceedings of the 6th Australasian Conference on Information Security and Privacy (ACISP 2001)*, volume 2119 of *Lecture Notes in Computer Science*. Springer, 2001.

[6] C. Liu, M. A. Ozols, M. Henderson, and T. Cant. A state-based model for certificate management systems. In *Proceedings of the 3rd International Workshop on Practise and Theory in Public Key Cryptosystems (PKC 2000)*, volume 1751 of *Lecture Notes in Computer Science*. Springer, 2000.

[7] C. A. Meadows. Formal verification of cryptographic protocols: A survey. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1994.

[8] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[9] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[10] P. Syverson and I. Cervesato. The logic of authentication protocols. In *Foundations of Security Analysis and Design*, volume 2171 of *Tutorial Lectures*. Springer, 2001.