A formalization of credit and responsibility within the GNDC schema

Roberto Gorrieri [a] Fabio Martinelli [b] Marinella Petrocchi [b]

[a] *Dipartimento di Scienze dell'Informazione*
Università di Bologna, Italy
gorrieri@cs.unibo.it

[b] *Istituto di Informatica e Telematica, CNR*
Area della Ricerca di Pisa, Italy
fabio.martinelli@iit.cnr.it, marinella.petrocchi@iit.cnr.it

Abstract

We formally define a notion of credit and responsibility within the Generalized Non Deducibility on Compositions framework. We investigate the validity of our definitions through some examples discussed in the literature.

Key words: Formal methods, authentication, Generalized Non Deducibility on Composition

1 Overview

Formal methods and tools have been successfully applied for the analysis of security properties of communication protocols. The protocol under investigation is described in a given language, then a formal specification of the security property to be analyzed is defined. Whether or not the security property is fulfilled is investigated by formally analyzing the protocol in a hostile environment, i.e., considering the presence of an adversary running in parallel with the honest participants.

Both the property's formal specification and its informal definition are crucial steps for the analysis. Indeed, even a common notion as authentication is usually considered a slippery security property (see [8]). Several definitions have been proposed in the security community. Henceforth, to define a formal model for authentication is a tricky task and many different formalisms actually exist [2,3,5,8,11,13,14,15]. Their expressiveness, usability and accuracy are often questioned, e.g. [8], and some effort has been devoted to compare them. In [4] three different authentication properties are compared, and the comparison is carried out within the Generalized Non Deducibility on Compositions schema (GNDC, [7,5]). GNDC has been proposed for defining and

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

analyzing security properties and it is based on the notion of non-interference.

In this paper, we try to formally capture the meaning of two different flavors of authentication, *credit* and *responsibility*. Abadi first focused his attention on these two properties in [1], arguing about two possible uses of an authenticated message m. Suppose that a principal A sends the message to a principal B. Then, as far as responsibility is concerned, "B may believe that m is being supported by A's authority". Also, as far as credit is concerned, "B may attribute credit for m to A". Abadi expresses the concepts of credit and responsibility through several examples, by discussing their distinction in the design and analysis of protocols. Different cryptographic operators (digital signatures, encryptions, decryptions) may lead either to responsibility or credit. Our investigation starts from the intuitions of Abadi by rephrasing them: we suggest that responsibility supports orders, i.e., public messages which speak for some principal, while credit is related to a principal for secret messages known in advance only by that principal, that wants to be acknowledged as the holder of the secret messages.

The paper is organized as follows. The next section recalls the GNDC schema used for modeling security properties. Section 3 defines credit and responsibility within GNDC. Section 4 considers some of the protocols analyzed by Abadi in [1]. It is checked if these protocols meet our formal definitions. Section 5 provides a sketch of a formal comparison between the two properties. Section 6 concludes the paper. The interested reader will find in the Appendix the syntax and semantics of the language used for specifying cryptographic protocols.

2 A General Schema for Security Properties

In this section we present the general schema GNDC for the definition of security properties given in [7,5]. We assume some familiarity with process algebras. Further, throughout the paper we specify protocols and properties through Crypto-CCS, [7], basically process algebra CCS with cryptographic modeling constructs. The syntax and semantics of the language can be found in the Appendix.

Informally, the GNDC schema states that a system specification P satisfies property $GNDC^{\alpha(P)}_{\lhd}$ if the behaviour of P, despite the presence of a hostile environment $\mathcal{E}^{\phi}_{C}$ that can interact with P only through a set of channels in C, *appears* to be same (w.r.t. a behavioral relation $\lhd$ of observational equivalence) as the behaviour of a modified version $\alpha(P)$ of P that represents the *expected* (correct) behaviour of P. The GNDC schema thus has the form

$$P \in GNDC^{\alpha(P)}_{\lhd} \text{ iff } \forall X \in \mathcal{E}^{\phi}_{C} : (P \parallel X)\backslash C \lhd \alpha(P), \tag{1}$$

where $(P \parallel X)\backslash C$ denotes the parallel composition of processes P and X restricted to communication over channels other than C. By varying parame-

ters $\triangleleft$ and α a variety of security properties can be formulated in the GNDC schema, [6].

In (1) there is an additional constraint that is required in the specific context of analyzing cryptographic protocols: the *static* (initial) knowledge of the hostile environment must be bound to a specific set of messages. This limitation is needed to avoid a too strong hostile environment that would be able to corrupt any secret (as it would know all cryptographic keys, etc.). Formally, $\mathcal{E}_C^\phi$ just represents all processes communicating through channels in C and having an initial knowledge bound by ϕ. We consider as hostile processes only the ones belonging to $\mathcal{E}_C^\phi$.

Obviously, with a specific formal framework in mind, e.g. a CCS-like process algebra, all symbols in (1) need to be instantiated. Consider the instance

$$P \in GNDC^{\alpha(P)}_{\leq_{trace}} \text{ iff } \forall X \in \mathcal{E}_C^\phi : (P \parallel X)\backslash C \leq_{trace} \alpha(P) \tag{2}$$

of GNDC. Here, the trace inclusion relation has been instantiated as a behavioral relation $\triangleleft$ of observational equivalence. Trace inclusion is defined as follows: we say that the traces of P are included in the traces of Q ($P \leq_{trace} Q$) iff whenever P can move from the state P to the state P' through a sequence of actions γ, then also Q can do the same, apart from internal actions of P and Q. The trace inclusion relation is commonly used for the analysis of safety properties. When $\leq_{trace}$ is considered, there exists a sufficient criterion for the static characterization (i.e., not involving the universal predicate $\forall$) of $GNDC^{\alpha(P)}_{\triangleleft}$ (for further details, see [7,5]).

3 Defining Responsibility and Credit within the GNDC schema

According to Abadi [1], "Authentication can serve both for assigning responsibility and for giving credit". He concludes that some authentication protocols are adequate for applications requiring responsibility but not necessarily for applications requiring credit and *vice versa*. Hence, he claims the need to better clarify whether the properties of an authentication protocol suffice for establishing responsibility and/or credit.

We believe that responsibility and credit can be defined similarly to the characterization of the correspondance property given in [15] (note that correspondance was called agreement in [11], upon being formalized in the CSP process algebra, [13]). What is technically done in the characterization of the agreement property is to give each principal the possibility to perform *control actions*, expressing the current local view of the computation. These control actions are inserted in the specification of a protocol only for verification purposes and the principals cannot exploit them during the protocol run.

Note that the use of the correspondance actions and, consequently, the one

of the agreement actions, are considered a well-founded mechanism to specify and correctly evaluate authentication properties (see, *e.g.,* , [9]).

3.1 Assigning responsibility

We formally define here the responsibility property. Assigning responsibility to a principal A for a public message M [1] specifies that M is supported by A's authority. We give the following intuitive explanation of the property:

> "A principal B may assign *responsibility* to a principal A for a public message M *iff* A explicitly sent M to B.

Our informal interpretation finds its motivation in everyday life, where who actually claims something is generally considered responsible for what he claimed.

To formally model the property we exploit a pair of control actions. We define control actions corresponding to *accepting responsibility* and, symmetrically, *assigning responsibility*. We shall write $\overline{accept\_resp}\,(A, B, M)$ meaning that "A accepts responsibility from B for message M", and $\overline{assign\_resp}\,(B, A, M)$ meaning that "B assigns responsibility to A for message M".

The property is specified as follows:

$$\alpha_{AR}(P) = \|_{(x,y,m)\in D(P)}\, \overline{accept\_resp}\,(x, y, m).\overline{assign\_resp}\,(y, x, m)$$

where P is a protocol instance and $D(P)$ is the set of all possible deliveries one wishes to consider. A delivery is a triple (x, y, m), where x is the sender of message m and y the receiver. $\alpha_{AR}(P)$ is specified in Crypto-CCS. If Q is a term in the algebra, then $\overline{ch}\,(msg).Q$ is the process that sends message msg on channel ch and then behaves as Q.

Example 3.1 Consider a protocol instance P where we have a honest user A, a malicious one E and a bank B. Assume that A (E) sends a message M (M') to B, then the set $D(P)$ is $\{(A, B, M), (E, B, M')\}$. ■

We formulate responsibility within the GNDC schema as follows:

Definition 3.2 P satisfies Assigning Responsibility iff P is $GNDC^{\alpha_{AR}(P)}_{\leq_{trace}}$. ■

Thus, a protocol enjoys *Assigning Responsibility* iff, in each computation, the act of giving each responsibility is preceded by the accepting of such a responsibility. We remark that, since we consider a trace based semantics, accepting responsibility without the corresponding assignment is not considered an attack.

[1] A public message is a message possibly known by all the principals.

3.2 Giving credit

We formally define here the credit property. Giving credit to a principal A for a message M specifies that A asks credit for a particular piece of information. In a nutshell, we believe that this notion is similar to responsibility where only secret messages are taken into account [2]. Therefore, we give the following intuitive explanation of the property:

> "A principal B may give *credit* to a principal A for a secret message M *iff* only A initially knows M and A wants credit for M from B".

Namely, in order to get credit from B, A should convince B that message M is possessed by A itself. As an example, suppose B rules a scientific committee and A is a scientist that discovers a treatment for a disease considered hard to cure up to now. Accordingly, A sends to B a message whose meaning may be roughly expressed as: "I discover the treatment for X. The treatment is Y. I require the paternity for such a discovery." First, B should authenticate the message as coming from A and being actually originated by A. Then, B should give credit to A (i.e., B should give A the paternity for the medical discovery). Y can be something that B does not know in advance, but that can be easily verified by B. As another example, suppose indeed that B rules a quiz show. Some external trusted third party, e.g. a notary, can previously give to B the digest of the right answer. Upon receiving the answer from A, B verifies its correctness by comparing the digests. If they are equal, B may give credit to A for the answer.

To formally model the credit property we introduce a new pair of control actions. We represent a credit request from A to B by the action $\overline{ask\_cre}$, and a credit giving by the action $\overline{give\_cre}$. The protocol should ensure that B gives credit to A only if A required it. This requirement is captured by the following specification:

$$\alpha_{GC}(P) = \|_{(x,y,m)\in D(P)} \overline{ask\_cre}\,(x,y,m).\overline{give\_cre}\,(y,x,m)$$

A particular characterization of the set $D(P)$ is crucial in the above definition of the credit property. With respect to the definition of responsibility given in Subsection 3.1, we put a constraint on $D(P)$ to model the fact that we do not consider public values. We assume that it is not possible to have two (or more) triples in $D(P)$ with the same message m. By means of this constraint we assume that only one principal is in the position to ask credit for a certain message. We call this assumption "the message uniqueness" condition.

When modeling a protocol, we usually describe only the honest agents running the protocol. However, in this particular framework, the set $D(P)$ of correct deliveries should also take into account the deliveries of possible malicious users. In particular, such deliveries must consider each possible message

[2] A message is secret for a principal A when it is not known by other principals in a protocol except A.

belonging to the knowledge of a malicious user (and apart from private messages of other users). We call this assumption "the correct sending capability" condition.

Definition 3.3 Assume that $D(P)$ satisfies the message uniqueness condition and the correct sending capability condition. Then, P satisfies *Giving Credit* iff P is $GNDC^{\alpha_{GC}(P)}_{\leq_{trace}}$. ■

In terms of traces, if an action $give\_cre(B, A, M)$ exists in the trace, then an action $ask\_cre(A, B, M)$ must previously occur within the same trace.

In the following section, we consider some of the protocols analyzed by Abadi in [1] and we check if these protocols meet our formal definitions for credit and responsibility.

4 Examples

4.1 First Example

The first protocol analyzed in [1] shows how to send a short-term public key K from a principal A to a principal B (Message 1). The short-term public key is signed with the long-term private key of principal A, K_A^{-1} . T is a timestamp. The short-term secret key K^{-1} is then used in subsequent messages (*e.g.,* Message 2) for signing further messages, *e.g., M*. The protocol assumes that B knows A's public key K_A.

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & A, B, \{K, A, B, T\}_{K_A^{-1}} \\ \text{Message 2} & A \rightarrow B: & A, B, \{\{M\}_{K^{-1}}\}_{K_B} \end{array}$$

As far as credit is concerned, we wonder about the correctness of the following interpretation of Message 1: principal A is asking principal B credit for messages signed with K^{-1}.

In Tab. 1, the protocol is expressed in our process algebra. The expected control actions ask_cre and $give\_cre$ are opportunely inserted.

We require that principal A actually possesses M, in order to prevent man in the middle attacks. Indeed, let us consider the attack presented in [1] (the behaviour of the attacker is formally defined in Tab. 2):

$$\begin{array}{lll} \text{Message 1} & A \rightarrow E(B): & A, B, \{K, A, B, T\}_{K_A^{-1}} \\ \text{Message 1'} & E \rightarrow B: & E, B, \{K, E, B, T\}_{K_E^{-1}} \\ \text{Message 2} & A \rightarrow E(B): & A, B, \{\{M\}_{K^{-1}}\}_{K_B} \\ \text{Message 2'} & E \rightarrow B: & E, B, \{\{M\}_{K^{-1}}\}_{K_B} \end{array}$$

We denote an attacker pretending to be B as *E(B)*. The attacker intercepts the messages from A to B, replacing them with its own messages. In messages 1' and 2' the attacker impersonates itself. In particular, E makes a credit request using K, and therefore E could benefit from A's request. Indeed, when B receives Message 1' from E, B may think that E is asking credit for

$$A(m,k) \stackrel{\text{def}}{=} [\langle (k,A,B,T), k_A^{-1}\rangle \vdash_{enc} x]\overline{ask\_cre}\,(A,B,m)\,.$$
$$\overline{c_1}\,((A,B),x)\,.[\langle m, k^{-1}\rangle \vdash_{enc} y][\langle y, k_B\rangle \vdash_{enc} z]\overline{c_2}\,((A,B),z)$$
$$B(k) \stackrel{\text{def}}{=} c_1(y).[y \vdash_{snd} z][\langle z,k\rangle \vdash_{dec} t][t \vdash_{fst} u][u \vdash_{snd} b][b = B][u \vdash_{fst} r]$$
$$[r \vdash_{snd} a][r \vdash_{fst} k']c_2(w)\,.[w \vdash_{snd} h][\langle h, k_B^{-1}\rangle \vdash_{dec} j]$$
$$[\langle j, k'\rangle \vdash_{dec} g]\overline{give\_cre}\,(b,a,g)$$
$$P \stackrel{\text{def}}{=} A(M,K) \parallel B(K_A)$$

Table 1
First example.

$E \stackrel{\text{def}}{=} c_1(x)[x \vdash_{snd} y]$	% E intercepts A's message 1
$[\langle y, k_A\rangle \vdash_{dec} z]$	% E decrypts $\{K,A,B,T\}_{K_A^{-1}}$ and
$[\langle z, k_E^{-1}\rangle \vdash_{enc} w]$	% E encrypts the message under K_E^{-1}
$\overline{c_1}\,((E,B),w)\,.$	% E replaces A with E and sends it
$c_2(j)\,.\,[j \vdash_{fst} k]$	% E intercepts A's message 2 and
$\overline{c_2}\,((E,B),k)$	% replaces A with E. Then it sends to B
	% the message $E, B, \{\{M\}_{K^{-1}}\}_{K_B}$

Table 2
Credit attack on the first example.

subsequent messages signed with a private key correspondent to K. (E, and not A, is actually talking with B). In practice, E behaves badly by intercepting messages but then it executes correctly the steps of the protocol, by playing the role of itself. Note that E cannot decrypt message $\{\{M\}_{K^{-1}}\}_{K_B}$, because it does not know K_B^{-1}. Hence, E does not possess M.

Let us consider the process $P' \doteq (P \parallel E) \setminus \{c_1, c_2\}$, c_1 and c_2 being the channels over which messages 1,1' and 2,2' are exchanged, respectively. There exists a trace for this process that is not a trace for $\alpha_{GC}(P)$. Indeed, if the above mentioned attack is successfully launched, we can find a trace in P' where a $give\_cre(B,E,M)$ action is preceded by a $ask\_cre(A,B,M)$ action. Thus, B may give credit to E for message M even if A has previously required credit for that message. On the contrary, given the restriction on *D(P)*, we

cannot have two credit requests related to deliveries dealing with the same message, i.e., A, B, M and E, B, M. Consequently, we cannot have a trace in which $give\_cre(B, E, M)$ is performed after $ask\_cre(A, B, M)$ in $\alpha_{GC}(P)$. Hence, $P \notin GNDC^{\alpha_{GC}(P)}_{\leq_{trace}}$. As Abadi, we conclude that the protocol is not adequate to give credit.

We believe that responsibility is concerned with public messages, as message $\{M\}_{K^{-1}}$ encrypted with K_B. Hence, it would make sense assigning responsibility to E for messages it sends to B. Unfortunately, our formalization of responsibility does not agree with Abadi's (and our) intuitive interpretation. This formally lies in the fact that A emits an $accept\_resp(A, B, \{\{M\}_{K^{-1}}\}_{K_B})$ action (message m in Table 1 must be opportunely instantiated) whereas B must perform an $assign\_resp(B, E, \{\{M\}_{K^{-1}}\}_{K_B})$ (again, with an opportune instance of the variables in Table 1). By allowing also attackers to issue control actions, many apparent attacks against responsibility would disappear. If agent E behaves correctly in issuing control actions, then E emits the action $accept\_resp(E, B, \{\{M\}_{K^{-1}}\}_{K_B})$ and the correspondance with the control action $assign\_resp(B, E, \{\{M\}_{K^{-1}}\}_{K_B})$ is established.

We require that the emission of control actions by an adversary happens in a fair way, i.e., whenever the adversary acts as a honest participant, then it emits the corresponding control action. Note that, contrary to the previous literature, here we allow adversaries to output control actions. Thus, we relax the GNDC requirement confining the adversary to interact only through a set of channels C. Here, we also allow the adversary to output control actions over special control channels, only for verification purposes. Basically, we augment the $Sort$ of actions that an adversary can do. To this aim, we introduce the operator $\downarrow_X$ s.t., given a parallel system S and a sequential process X, if $S \stackrel{\gamma}{\Longrightarrow} S'$ then $\gamma \downarrow_X$ denotes the sequence of actions performed by X during the computation γ.

Assume that an adversary can act either as a malicious user or as a honest one. Then, its honest behavior is described by a process R_X (that is a CryptoCCS term) prescribing its role. R_X represents the role of X. We define the predicate $Hon^X_{R_X,AR}(\gamma)$ that tells us whether or not X behaves correctly w.r.t. *accept_resp* control actions. Informally, X behaves correctly w.r.t. *accept_resp* control actions iff, whenever the first action of its role is performed, then this first action has been preceded by an opportune *accept_resp* action [3].

When malicious agent can perform only *accept_resp* control actions, the predicate $Hon^X_{R_X,AR}(\gamma)$ can be defined as follows:

[3] It is not necessary to study the correct behavior of the malicious user acting as the responder in a protocol, since the absence of *assign_resp* actions is not significant.

$$Hon^X_{R_X,AR}(\gamma)$$
$$\textit{iff}$$
$$\exists y, m \text{ s.t.}$$
$$\left(\begin{array}{l}\gamma \downarrow_X = \alpha_1 \ldots \alpha_n \wedge \\ \exists j. 2 \leq j \leq n : \alpha_j = FST(R_X, y, m) \wedge \\ m \in \mathcal{D}(\phi \cup msgs(\alpha_1, \ldots, \alpha_{j-1}))\end{array}\right) \rightarrow \alpha_{j-1} = accept\_resp(X, y, m)$$

where $FST(R_X, y, m)$ is the first action X performs during computation γ, following its role R_X and involving a receiver y and a message m.

A process can issue a control action involving a certain message only if this message belongs to its knowledge after the sequence $\alpha_1, \ldots, \alpha_{j-1}$. This requirement is essential for our different characterization of credit and responsibility. Broadly speaking, our idea is the following. A third party can be considered responsible for some public message it actually sends, but, whenever it acts according to a man in the middle scheme, it cannot have any credit assigned for forwarding messages that it actually does not know.

We define a relation between processes which is a refinement of the classical trace inclusion relation.

Definition 4.1 The relation between processes $\leq_{R_X,AR}$ is defined as follows:

$$S_1 \leq_{R_X,AR} S_2 \text{ iff } \{\gamma \mid S_1 \stackrel{\gamma}{\Longrightarrow} \wedge Hon^X_{R_X,AR}(\gamma)\} \subseteq \{\gamma \mid S_2 \stackrel{\gamma}{\Longrightarrow}\}$$

■

Thus, responsibility within the GNDC schema can finally be defined:

Definition 4.2 P satisfies Assigning Responsibility iff P is $GNDC^{\alpha_{AR}(P)}_{\leq_{R_X,AR}}$. ■

Similarly, we can define the predicate $Hon^X_{R_X,GC}$ and a relation $\leq_{R_X,GC}$ for credit properties.

Definition 4.3 Assume that $D(P)$ satisfies the message uniqueness condition and the correct sending capability condition. Then, P satisfies Giving Credit iff P is $GNDC^{\alpha_{GC}(P)}_{\leq_{R_X,GC}}$. ■

These slightly different notions allow us to identify those possible adversaries that correctly follow the steps in a protocol. Let us consider again the man in the middle attack where the enemy plays the role of itself while sending messages to B. Now E can issue control actions. Since the encrypted communication in Message 2 prevents learning message M and due to the restriction on $D(P)$, there is still a credit attack (E is indeed not allowed to ask for credit for a message that it does not know).

On the contrary, E is no longer a hostile attacker talking about responsibility. Indeed, since $\{\{M\}_{K^{-1}}\}_{K_B}$ is assumed to be a public message, E may issue the control action $accept\_resp(E, B, \{\{M\}_{K^{-1}}\}_{K_B})$. This action matches the corresponding $assign\_resp(B, E, \{\{M\}_{K^{-1}}\}_{K_B})$ raised by B. Thus, there is no responsibility attack.

Our latter formalization for credit and responsibility agrees with the intuition provided by Abadi in [1].

4.2 Second Example

In Abadi's second example, A transmits a session key K to B along with A's identity (Message 1). The message is encrypted under B's public key K_B. The session key may then be used for further messages (e.g., Message 2 and 3). The protocol is specified as follows:

Message 1	$A \rightarrow B:$	$\{A, K\}_{K_B}$
Message 2	$A \rightarrow B:$	$\{M\}_K$
Message 3	$B \rightarrow A:$	$\{M'\}_K$

We agree with Abadi's intuition that the protocol is adequate for applications requiring responsibility for B. Only B can retrieve K from Message 1, then A can reasonably hold B responsible for a message encrypted under K (unless A did not generate the encrypted message itself. The protocol assumes that both principals can recognize their own messages). We note also that the presence of A's identity in Message 1 allows B to know the principal who will hold B responsible for subsequent messages.

On the contrary, B cannot hold A responsible for messages encrypted under K, since Message 1 could be generated by others, pretending to be A. Indeed, the possible issue of an $assign\_resp(A, B, M')$ action from B will never be preceded by the issue of the correspondent $accept\_resp(B, A, M')$ from A.

Let us consider the credit property. Abadi suggests that this protocol seems to give a form of "unqualified" credit to A. By including its name in Message 1, A could claim credit for messages encrypted under K. Suppose now that A chooses K incompetently, or maliciously. In this case, a third party C can send a message $\{M''\}_K$ to B. As a consequence C gives "unqualified" credit to A for M", since B may give credit to A for M" even if A has not send that message.

This scenario contrasts with our intuition that messages for which someone gives us credit must be consciously sent by ourselves. According to our definition of credit, a principal is allowed to receive credit only for "asking credit" actions that (s)he has been able to perform. Thus, we rule out protocols giving forms of "unqualified" credit. Formally, it is easy to verify that the protocol under investigation does not guarantee credit to A. Since Message 1 does not provide A's authentication, whatever adversary can attribute messages encrypted under K to A. However, A cannot issue the control action $ask\_cre(A, B, M)$ if it does not possess M. In case, there would be an

action $give\_cre(B, A, M)$ not preceded by the correspondent ask_cre action and, consequently, a credit attack.

4.3 Third Example

In his paper, Abadi gives a third example, a protocol used as a component of Krawczyk's SKEME protocol [10]. The protocol aims at obtaining a shared key from two random quantities (J_A and J_B) that principal A and principal B respectively invent. A and B send each other these random quantities, encrypted with the public key of the receiver. They finally compute a shared key K by applying a one-way hash function to the concatenation of J_A and J_B. The shared key can be used in subsequent communication between A and B.

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & \{J_A\}_{K_B} \\ \text{Message 2} & B \rightarrow A: & \{J_B\}_{K_A} \\ & K = H(J_A, J_B) & \end{array}$$

Let us consider the following on-line attack:

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & \{J_A\}_{K_B} \\ \text{Message 1'} & B \rightarrow C: & \{J_A\}_{K_C} \\ \text{Message 2} & C \rightarrow A: & \{J_C\}_{K_A} \end{array}$$

In this case A initiates the protocol with a malicious principal B, who forwards A's half-key to C. Both A and C compute the same shared key K (K will not be known by B), but A mistakenly believes that it shares K with B, and C believes that it shares K with A. This attack supposes that B makes C believe that Message 1' comes from A. Further, A believes Message 2 comes from B.

According to this computation, C can claim credit to A for messages encrypted under K. This leads to a credit attack. Suppose indeed that communications take place over channels c_{ab}, c_{bc} and c_{ca} respectively and consider the process $P' \doteq (A \parallel B \parallel C) \backslash \{c_{ab}, c_{bc}, c_{ca}\}$. In our analysis framework, we can obtain a trace in P' s.t. an action $ask\_cre(C, A, M')$ is observed (provided that C possesses M'), and then a $give\_cre(A, B, M')$ action is executed.

Similarly, there is a responsibility attack on B. Indeed, each message from C will be assigned to B (i.e., A may think that the message is supported by B's authority). However, since B behaved badly, it is questionable if we have to study responsibility attacks on B. It is worthy to investigate such attacks when B's behavior causes some harm to an honest agent C. As an example, consider the case in which the message sent by C contains a password to enter C's secret data. A may believe that such a credential is supported by B's authority and consequently that B can enter C's secret data. The message sent from C to A could be "put on my directory the files in the common repository protected with password p".

Thus, in our interpretation, we have both a credit and a responsibility

attack on the responder.

5 Comparison

Informally, upon analyzing the previous examples, it would seem that giving credit is at least as difficult as assigning responsibility. In particular, if we restrict to consider AR only over processes s.t. $D(P)$ enjoys the restriction for credit, then clearly the two notions are equal. Indeed, our definition for credit could be imagined as a requirement of responsibility provided that the messages for which we require responsibility are secret.

The GNDC analysis framework is suitable for formally comparing different security properties. Actually, since responsibility is defined for public messages while credit for secret ones, a formal comparison of the two properties is not possible. In order to make it feasible, we need to drop the assumption about the secrecy of messages in the credit specification. Though maintaining coherence with the previous formulation, another characterization could be the following:

> We have a credit attack whenever a user B gives credit to A for a message M and either A did not ask credit from B for that message or there is another user C who asked credit for M from B before A.

This expresses a form of race condition for asking credit. Provided a correct environment, whenever a user is willing to ask credit for a message M, M should be known only by itself[4]. Thus, if the protocol is correct, no one else could try to ask credit for that message. Roughly, this means that if a protocol satisfies our previous definition of credit, then it satisfies also the current one.

The above-mentioned characterization does not mention secret values. A possible formalization could be the following:

$$\begin{aligned} P'_m = {} & \textstyle\sum_{(x,y,m)\in D(P)_m} \overline{ask\_cre}\,(x,y,m). \\ & (\overline{give\_cre}\,(y,x,m). \\ & \|(\|_{(x',y',m)\in D(P)_m\setminus\{(x,y,m)\}} \overline{ask\_cre}\,(x',y',m))) \\ \alpha_{GC'}(P) = {} & \|_{m\in M_D} P'_m \end{aligned}$$

where $M_D = \{m \mid (x,y,m) \in D(P)\}$ are all the possible messages that occur in $D(P)$ and $D(P)_m = \{(x,y,m) \in D(P)\}$ are the triples in $D(P)$ containing message m. Given a message m, only the first request asking credit for m will be served.

Definition 5.1 P satisfies *Giving Credit* iff E is $GNDC^{\alpha_{GC'}(P)}_{\leq_{R_X},GC}$. ∎

[4] In particular circumstances, we may allow the recipient to know M too. More likely, the recipient could simply have a way to validate the message, e.g. by having the message digest.

We can now perform a comparison with the *responsibility* property.

Proposition 5.2 *If P enjoys* credit *(Def. 5.1) then $P[f]$ enjoys* responsibility *(Def. 4.2), where $f(ask\_cre) = accept\_resp$ and $f(give\_cre) = assign\_resp$.* ■

Basically, the proof is done by noticing that, up to renaming of actions by f, we have that $\alpha_{GC'}(P) \leq_{trace} \alpha_{AR}(P)$.

6 Conclusions

We have defined a possible notion of the credit and responsibility properties within a well-established schema for modeling and analyzing security properties. We have also considered some of the protocols analyzed by Abadi in [1] and we checked if these protocols meet our formal definitions. We have sketched a formal comparison between the two properties.

We conclude that our definitions of credit and responsibility almost always adhere to Abadi's intuitions. However, as highlighted from our studies, we argue that credit should be considered somehow a stronger property than responsibility, whereas [1] gives special cases where this seems not completely reasonable. Indeed, the current work and the work in [1] differ for the notion of "unqualified credit". Abadi allows a user C to speak for a user A. As a consequence, A can get credit for a statement A did not make. On the contrary, we force the user A to get credit only for messages originated by itself.

Some aspects of these two properties have been transversally treated. Indeed, we also put our definitions at work, and let them provide a formal analysis for the secure emission of digital certificates in [12].

The reader should note that what have been given in this paper is only a possible interpretation of the two properties. Thus, further studies could be actuated, in order to investigate if other flavors of the properties are coherent with respect to Abadi's intuitions. We remark that the choice here actuated is arbitrary and other directions could be followed. As a future work, we aim at continue these investigations, due to the relevance of the treated topics.

Finally, we actually dealt with direct responsibility and direct credit, rather than with delegation. It could be worthy to study how delegation can be interpreted in our (and other) framework.

References

[1] Abadi, M., *Two Facets of Authentication*, Proc. of CSFW'98, IEEE, 1998, 27-33.

[2] Abadi, M., *Security Protocols and Specifications*, Proc. of FOSSACS'99, Springer, 1999, LNCS 1578, 1-13.

[3] Burrows, M., Abadi, M., and Needham, R., *A Logic of Authentication*, Technical Report 39, 1989, DEC Systems Research CENTER.

[4] Focardi, R, Gorrieri, and Martinelli, F., *A Comparison of Three Authentication Properties*, TCS **291, 3** (2003), 285-327.

[5] Focardi, R, and Martinelli, F., *A Uniform Approach for the Definition of Security Properties*, Proc. of FM'99, Springer, 1999, LNCS 1708, 794-813.

[6] Focardi, R, Gorrieri, and Martinelli, F., *Classification of Security Properties – Part II: Network security*, Proc. of FOSAD'01/'02, Springer, 2004, LNCS 2946, 139-185.

[7] Focardi, R, Gorrieri, and Martinelli, F., *Non Interference for the Analysis of Cryptographic Protocols*, Proc. of ICALP'00, Springer, 2000, LNCS 1853, 354-372.

[8] Gollman, D., *On the Verification of Cryptographic Protocols - a Tale of Two Committees*, Electronic Notes in Theoretical Computer Science **32** (2000).

[9] Gollmann, D., *Facets of Security*. Proc. of Global Computing'03, Springer, 2003, LNCS 2874, 192-202.

[10] Krawczyk, H., *SKEME: A Versatile Secure Key Exchange Mechanism for Internet*, Proc. of NDSS'96, ISOC, 1996.

[11] Lowe, G., *A Hierarchy of Authentication Specifications*, Proc. of CSCW'97, IEEE, 1997, 31-44.

[12] Martinelli, F., Petrocchi, M., Vaccarelli, A., *A Formal Analysis of Two Secure Procedures for Certificate Delivery*, accepted for publication STVR **16**, **1** (2006).

[13] Ryan, P., Schneider, S., Goldsmith, M., Lowe, G., and Roscoe, B., *The Modelling and Analysis of Security Protocols: the CSP Approach*, Addison-Wesley, 2000.

[14] Thayer J., Herzog J., and Guttman, J., *Honest Ideals on Strand Spaces*, Proc. of CSCW'98, IEEE, 1998, 66-78.

[15] Woo, T.Y.C., and Lam, S., *A Semantic Model for Authentication Protocols*, Proc. of SS&P'93, IEEE, 1993.

A Crypto-CCS

The model of the language consists of sequential agents able to communicate by exchanging messages.

The data handling part of the language consists of messages and inference systems. Messages are the data manipulated by agents, they form a set *Msgs* of terms possibly containing variables. The set *Msgs* is defined by the grammar:

$$m ::= x \mid b \mid F^1(m_1, \ldots, m_{k_1}) \mid \ldots \mid F^l(m_1, \ldots, m_{k_l})$$

where F^i (for $1 \leq i \leq l$) are the constructors for messages, $x \in V$ is a countable set of variables, $b \in B$ is a collection of basic messages and k_i, for $1 \leq i \leq l$, gives the number of arguments of the constructor F^i. Messages without variables are *closed* messages.

Inference systems model the possible operations on messages. They consist of a set of rules r, *e.g.,* :

$$r = \frac{m_1 \quad \dots \quad m_n}{m_0}$$

where $\{m_1, \dots, m_n\}$ is a set of premises (possibly empty) and m_0 is the conclusion. An instance of the application of rule r to closed messages m_i is denoted as $m_1 \quad \dots \quad m_n \vdash_r m_0$. Given an inference system, a *deduction function* $\mathcal{D}$ is defined such that, if ϕ is a finite set of closed messages, then $\mathcal{D}(\phi)$ is the set of closed messages that can be deduced starting from ϕ by applying instances of the rules in the system. The syntax and semantics of Crypto-CCS are parametric with respect to a given inference system. Example inference systems suitable to model specific cryptographic protocols will be shown in the following sections.

The control part of the language consists of compound systems, i.e., sequential agents running in parallel. The language syntax is as follows:

$$\begin{array}{ll} \textsc{Compound systems:} & S ::= (S_1 \parallel S_2) \mid S \setminus C \mid A_\phi \\ \textsc{Sequential agents:} & A ::= \mathbf{0} \mid p.A \mid A_1 + A_2 \mid [m_1 \dots m_n \vdash_r x]A_1; A_2 \\ & \mid [m = m']A_1; A_2 \mid E(m_1, \dots, m_n) \\ \textsc{Prefix constructs:} & p ::= c!m \mid c?x \end{array}$$

where $m, m', m_1, \dots, m_n$ are *closed* messages or variables, x is a variable, $c \in Ch$ (a finite set of channels) ϕ is a finite set of *closed* messages, C is a subset of Ch.

$\mathbf{0}$ is the process that does nothing.

$p.A$ is the process that can perform an action according to the particular prefix construct p and then behaves as A. In particular,

- $c!m$ denotes a message m sent on channel c;
- $c?x$ denotes the receiving of a message m on channel c. The received message replaces the variable x.

$A_1 + A_2$ represents the non deterministic choice between A and A_1.

$[m_1 \dots m_n \vdash_r x]A_1; A_2$ is the inference construct. If, by applying an instance of rule r, with premises $m_1 \dots m_n$, a message m can be inferred, then the process behaves as A_1 (where m replaces x), otherwise it behaves as A_2.

$[m = m']A_1; A_2$ is the match construct, to check message equality. If $m = m'$ then the system behaves as A_1, otherwise it behaves as A_2.

A compound system $S_1 \parallel S_2$ denotes the parallel execution of S_1 and S_2. $S_1 \parallel S_2$ performs an action p if one of its sub-components performs p. A synchronization, or internal action, denoted by τ, may take place whenever

$$(!)\frac{}{(c!m.A)_\phi \xrightarrow{c!m} (A)_\phi} \qquad (\|_1)\frac{S \xrightarrow{a} S'}{S \,\|\, S_1 \xrightarrow{a} S' \,\|\, S_1}$$

$$(?)\frac{m \in Msgs}{(c?x.A)_\phi \xrightarrow{c?m} (A[m/x])_{\phi\cup\{m\}}} \qquad (\|_2)\frac{S \xrightarrow{c!m} S' \quad S_1 \xrightarrow{c?m} S_1'}{S \,\|\, S_1 \xrightarrow{\tau} S' \,\|\, S_1'}$$

$$(\mathcal{D})\frac{m_1 \ldots m_n \vdash_r m \quad (A[m/x])_{\phi\cup\{m\}} \xrightarrow{a} (A')_{\phi'}}{([m_1 \ldots m_n \vdash_r x]A; A_1)_\phi \xrightarrow{a} (A')_{\phi'}} \qquad (\backslash_1)\frac{S \xrightarrow{c!m} S' \quad c \notin L}{S \setminus L \xrightarrow{c!m} S' \setminus L}$$

$$(+_2)\frac{S \xrightarrow{a} S'}{S + S_1 \xrightarrow{a} S'}$$

$$(\mathcal{D}_1)\frac{\nexists m \text{ s.t. } m_1 \ldots m_n \vdash_r m \quad (A_1)_\phi \xrightarrow{a} (A_1')_{\phi'}}{([m_1 \ldots m_n \vdash_r x]A; A_1)_\phi \xrightarrow{a} (A_1')_{\phi'}}$$

$$(=)\frac{m = m' \quad (A)_\phi \xrightarrow{a} (A')_{\phi'}}{([m = m']A; A_1)_\phi \xrightarrow{a} (A')_{\phi'}}$$

$$(=_1)\frac{m \neq m' \quad (A_1)_\phi \xrightarrow{a} (A_1')_{\phi'}}{([m = m']A; A_1)_\phi \xrightarrow{a} (A_1')_{\phi'}}$$

$$(Const)\frac{E(x_1, \ldots, x_n) =_{def} A \quad A[m_1/x_1, \ldots, m_n/x_n] \xrightarrow{a} A_1}{E(m_1, \ldots, m_n) \xrightarrow{a} A_1}$$

Fig. A.1. Operational semantics of Crypto-CCS.

S_1 and S_2 are able to perform two complementary actions, i.e., send -receive actions on the same channel.

A compound system $S \setminus C$ allows only visible actions whose channels are not in C. (Internal action τ being the invisible action).

The term A_ϕ is a single sequential agent whose knowledge, i.e., the set of messages which occur in its term, is described by ϕ. The knowledge of an agent increases either when it receives messages (see rule (?) in Fig. A.1) or it infers new messages from the messages it knows (see rule $\mathcal{D}$ in Fig. A.1). For every sequential agent A_ϕ, it is required that all the closed messages that appear in A_ϕ belong to its knowledge ϕ.

The activities of the agents are described by the actions that they can perform. The set Act of actions which may be performed by a compound system ranges over by a and it is defined as: $Act = \{c?m, c!m, \tau \mid c \in C, m \in Msgs, m \text{ closed}\}$. $\mathcal{P}$ is the set of all the Crypto-CCS *closed* terms (i.e., with no free variables). *sort(P)* is the set of all the channels that syntactically occur in the term P.

The operational semantics of a Crypto-CCS term is described by means of the *labeled transition system* (*lts*, for short) $\langle \mathcal{P}, Act, \{\xrightarrow{a}\}_{a\in Act}\rangle$, where $\{\xrightarrow{a}\}_{a\in Act}$ is the least relation between Crypto-CCS processes induced by the axioms and inference rules of Fig. A.1 (in that figure the symmetric rules for $\|_1, \|_2, \backslash_1, +_2$ are omitted).

The expression $S \xrightarrow{a} S'$ means that the system can move from the state S to the state S' through the action a. The expression $S \Longrightarrow S'$ denotes that S and S' belong to the reflexive and transitive closure of $\xrightarrow{\tau}$; let $\gamma = a_1 \dots a_n \in (Act \backslash \{\tau\})^*$ be a sequence of actions. Then, $S \stackrel{\gamma}{\Longrightarrow} S'$ if $S \Longrightarrow \xrightarrow{a_1} \Longrightarrow \dots \Longrightarrow \xrightarrow{a_n} \Longrightarrow S'$.