Deriving, attacking and defending the GDOI protocol

Catherine Meadows
Code 5543
Naval Research Laboratory
Washington, DC 20375
`meadows@itd.nrl.navy.mil`

*Dusko Pavlovic**
Kestrel Institute
3260 Hillview Avenue
Palo Alto, CA 94304
`dusko@kestrel.edu`

Abstract

As a part of a continued effort towards a logical framework for incremental reasoning about security, we attempted a derivational reconstruction of GDOI, the protocol proposed in IETF RFC 3547 for authenticated key agreement in group communication over IPsec. The main advantage of the derivational approach to protocols is that it tracks the way they are designed: by refining and composing basic protocol components — but this time within a formal system, thus providing formal assurance of their security. Moreover, by distinguishing monotonic and compositional fragments of protocol designs, this approach facilitates tracking and formalizing of the incremental changes and modifications that complicate design process, increases reusability, and minimizes need for reverification.

While deriving GDOI, we have discovered that the simple logic, that we recently developed for deriving authentication, can also be used to prove its failures, and to generate attacks. By changing the proof goal, we have discovered an attack on GDOI that had not surfaced in the previous extensive analyses by other verification techniques. At the time of writing, we are working together with the Msec Working Group to develop a solution to this problem.

After a brief overview of the logic, we outline a derivation of GDOI, which displays its valid security properties, and the derivations of two attacks on it, which display its undesired properties. Finally, we discuss some modifications that eliminate these vulnerabilities. Their derivations suggest proofs of the desired authentication property.

1 Introduction

The formal analysis of cryptographic protocols has turned out to provide a useful way to find errors in security protocols, or to validate them. In a number of cases, formal tools and techniques have been used to discover problems in protocols that had already been designed and deployed. In principle, it is of course much preferable to catch and remove problems early in the design phase, and before deployment. The ultimate goal of formal methods is to assist in design and deployment of protocols, as well as analysis.

A key feature that is needed to support the integration of formal methods into cryptographic protocol design is composability. Most of the design of a working cryptographic protocol is incremental in nature. One starts with a simple pattern that gives the basic skeleton of the protocol. One then adds the particular functionality needed for the particular application in mind. Some of the added functions may be optional, leading to different versions of the protocol. Finally, if some of the added features require interaction between the principals, it may be necessary to compose the protocol with some other protocol or protocols. For example, if one wants a key distribution protocol to enforce perfect forward secrecy, one may want to compose it with the Diffie-Hellman protocol.

In a situation like this it would be ideal if one could verify as well as design incrementally. It should be possible to identify situations in which properties that have been proven to be true remained true even after the protocol was modified in certain constrained ways. Unfortunately, this is in general a very hard problem in formal methods. In the general case, even minor changes can fail to preserve properties that had previously held.

A solution, of course, is to restrict oneself to properties and transformations that *can* be reasoned about incrementally; and more generally, to develop techniques to recognize conditions under which the security properties of interest are preserved under refinement, composition, or transformations at large. One such technique, in the framework of protocol derivations, has been studied in [9]. In the context of general theory of protocols, the issue of compositionality has been widely investigated, in various abstract process models. Some of the recent references are [6, 2, 17]. While the general problem of compositionality and monotonicity of security properties remains open, particular applications do allow useful deployment of incremental methods, informally used

* Supported by ONR N00014-03-C-0237 and by NSF CCR-0345397.

in many protocol development efforts and publications. We believe that such practices can and should be formalized.

With this in mind, we have been developing a monotone epistemic logic that provides a straightforward way of composing derivations of properties. In this logic, all statements express agents' knowledge about concrete situations and events such as the possession of keys and the sending and receipt of messages. These statements can then be composed to prove the desired conclusions about the sequences of events that must occur in a protocol. The current version, addressing only authenticity, however, does not involve composition of knowledge modalities, so that the epistemic nature of this logic remains on a rather rudimentary level. While it resembles BAN logic [5] in this restriction to authentication, the present logic is much simpler, with the order of actions as its only non-logical primitive.

Most importantly, the logic proceeds in much the same way as a protocol is designed. One starts with some simple patterns, for which some basic properties are established in advance. These patterns are then composed into the basic protocol. The properties add up in so far as they preserve each other's assumptions [9]. The next step is to add specific features that the protocol must provide: these would include, for example, the actual data that the protocol is intended to distribute securely. At this step, the protocol can also be composed with other, auxiliary, patterns.

An interesting and useful feature of the logic is that the same approach can be used to derive attacks on insecure protocols as to prove security properties of sound ones. This is often done by lifting a simple attack on a simple protocol component to a more subtle attack on a more complex protocol. The attack on a component C is expressed as a process on its own, say $\widetilde{C}$, corresponding to a logical statement of an undesired property. If the protocol P has been derived by using C in a derivation Δ, then replacing C by $\widetilde{C}$ in Δ will yield a derivation $\widetilde{\Delta}$ of an attack $\widetilde{P}$ on P, whenever the relevant undesired property is preserved. The result is that we are able to take advantage of our knowledge of $\widetilde{C}$ to derive the attack $\widetilde{P}$.

Other attacks on protocols, of course, may not arise from attacks on their components, but may emerge, e.g., from insecure composition of secure components. Such attacks can still be derived, just like counterexamples are derived in logic, by changing the derivation goals. Since attacks, like protocols, are often based on simple patterns, attack derivations have the potential to be a useful feature for parlaying knowledge about basic attacks, and about propagating insecurities, into understanding of the vulnerabilities of complex protocols, just like protocol derivations parlay knowledge about basic protocols, and about preservation of security properties.

The logic used in our derivations draws upon the ideas of earlier derivational formalisms, developed in [11, 8]. The crucial difference is again that the statements of the new logic are couched entirely in terms of partial orders (distributed traces) of actions, as observed by each agent, or derived from her observations and the common axioms. One consequence of this is that we are now less likely to encounter some of the problems that epistemic logics at large have had in the past, where it was sometimes difficult to determine from the conclusions of an analysis what actual behavior was expected of a protocol. Another consequence is that our derivation system has a smaller syntax and simpler semantics than its predecessors. The space nevertheless permits only a broad overview; but the formalism is this time so simple that it will hopefully suffice to show it at work. A more detailed presentation is in preparation [19].

The present paper is concerned with an application to the Group Domain of Interpretation (GDOI) [3] protocol, developed by the Internet Engineering Task Force (IETF). This protocol is not only of great practical interest, because of its wide applications in secure multicast, and in secure group communication at large [3, sec. 1.1], but also of particular conceptual interest, because of the previous detailed analyses using the NRL Protocol Analyzer (NPA)[18]. The NPA is a model checker that, like the logic described in this paper, can be used to both provide security proofs and discover attacks, but does not support incremental or compositional verification. Interestingly, a failed composition involving a portion of the protocol called the "Proof of Possession" pointed up a potential problem with an optional means for providing authorization, which, because of a misunderstanding of the requirement, had been missed in the NPA analysis. The attack presented in this paper has arisen from an attempt to derive the GDOI protocol with the Proof of Possession option: the analysis of the step composing the core GDOI with the subprotocol underlying Proof of Possession has shown that the insufficient binding between the two components allowed deriving attacks, rather than the desired security property.

The rest of the paper is organized as follows. In section 2 we describe the GDOI protocol. In sec. 3 we give a brief overview of the logic. In sec. 4 we describe the derivation of the Core GDOI protocol. In sec. 5 we describe the derivation of the Proof of Possession protocol and its composition with the core GDOI protocol. In sec. 6 we discuss the derivation of the attack and some suggestions for fixing the protocol. In sec. 7 we compare our results with the earlier NPA analysis and conclude the paper.

2 The GDOI Protocol

In this section we describe the relevant portions of the GDOI protocol that are discussed in this paper.

GDOI actually consists of two main protocols: the GROUPKEY-PULL protocol, which is used when a new member joins the group, and the GROUPKEY-PUSH datagram, which is used to distribute keys to current members. In this paper we are interested in the GROUPKEY-PULL protocol.

The GROUPKEY-PULL protocol takes place between a Group Controller/Key Server (GCKS) and a member who wants to join the group. Authentication and secrecy are provided by a key that was previously exchanged using the Internet Key Exchange (IKE) protocol [14]. The purpose of IKE is to provide secure key distribution between two principals. Keys are distributed by IKE in two phases: long term phase 1 keys, which in turn are used to distribute shorter-term Phase 2 keys. GDOI makes use of IKE Phase 1 only; GDOI can be thought of as taking the place of IKE Phase 2 for groups.

The GROUPKEY-PULL protocol serves two purposes: one is to distribute the current group key to the member, the other to provide mutual authentication and authorization between the member and GCKS. Furthermore, the latter purpose can be realized in two ways:

(i) by using the Phase 1 key for authentication, and storing the authorization information with principal's Phase 1 identity, where it can be readily looked up,

(ii) by storing the authorization information in a certificate that contains principal's public key, allowing it to be authenticated by a signature with the corresponding private key.

In the latter case, known as the Proof of Possession (PoP), it is not the purpose of a certificate, as usual, to allow the verification of a signature, but it rather it is the purpose of the signature to authenticate the certificate. A principal uses the PoP to prove that he possesses the key in the certificate by using it to sign the other principal's nonce.

An interesting feature of the above options, specified in the GDOI RFC [3] is that the identity, contained in the certificate in (ii), can be, and is expected to be, *different* from the Phase 1 identity, used in (i).[1] Allowing multiple identities can be useful for security associations, has been used in the recent versions of IKE, envisioned for Phase 1 of GDOI, and is not insecure in itself. In this case, however, it does cause problems, as we shall soon see.

The GDOI message flows, relevant for our analysis, are given below. The messages are passed along a secure channel where authentication and secrecy are provided by the key passed in Phase 1 IKE, and which is identified by an IKE header and Message ID. Since in this paper we are considering authentication issues alone, we do not specify the encryption and identification functions explicitly. We also leave off an optional Diffie-Hellman exchange, since it is not relevant to our analysis of authentication properties of the protocol.

Let A be a group member and B a GCKS.

(i) $A \to B : H^{AB}(m, id), m, ID$

Here, m is A's nonce, id is the ID of the group, and H^{BA} denotes computation of a hash using the Phase 1 key shared between A and B.

[1]This is explicit for the group member, and left open for the GCKS.

(ii) $B \to A : H^{BA}(n, m, sa), n, sa$

Here n is B's nonce and sa stands for the security association associated with the key. Note that the hash keyed hash here is denoted H^{BA} instead of H^{AB}. This is to reflect the requirement that the input to the hashes be formatted in such a way that a message from an initiator be distinguishable from a message from a responder. This is not an issue for this specification, but will become so later for various partial specifications of the protocol.

(iii) $A \to B : H^{AB}(n, m, C^{A'}, S^{A'}(n, m)),$
$C^{A'}, S^{A'}(n, m).$

Here $C^{A'}$ denotes a certificate pertaining to A's new identity, A'. This certificate contains a public key, and $S^{A'}$ denotes a signature by the corresponding private key.

(iv) $B \to A : H^{BA}(n, m, C^{B'}, sq, k, S^{B'}(n, m)),$
$sq, k, S^{B'}(n, m)$

Here k is the actual keying material, and sq is a sequence number indicating the current GROUPKEY-PUSH message.

We leave out irrelevant information where appropriate. For example, in our analysis of GDOI we will usually leave out the inclusion of n in the hash of the last message, since it does not contribute to the analysis. We will also often leave out some or all of the actual information (keying material and sequence number) that is passed to the group member.

3 Brief Overview of Challenge Response Logic

The logic describes *actions* performed by *agents*. Actions consist of sending, receiving, and rewriting data, and generating random values. Agents constitute processes in the underlying process calculus; in this case, they can be construed as strands [13], or as cords [12]. Roles and principals can then be modeled as classes of agents, sharing data or information: keys and names in the case of principals, or actions in the case of roles. The other way around, an agent may be thought of as a special instance of a role, or of a principal.

The logic is built out of simple axioms describing the conclusions that a principal can draw from her observations of protocol actions, using the known axioms. These axioms are usually of the form: "If an agent performs certain sequence of actions, then she can conclude that some other sequence of actions by other parties also occurred". For instance, if A receives a message, then she can conclude that someone must have sent that message; if that message contains a term that only B can form, then she knows that B must have been on the path of the message.

The notational conventions are as follows. A language of terms t, which can be sent in messages, is assumed to be given. It includes variables for unknown terms or agents, and sufficient typing. The expressions $\langle t\rangle_A$, resp. $(t)_A$, denote the statements that the agent A has sent, resp. received the term t. The expressions $\langle\langle t\rangle\rangle_A$ resp. $((t))_A$, denote the statements that A has sent, resp. received a message *containing* t. An agent asserting such a containment statement may not be able to extract t, but must be able to establish its presence (e.g., as in the case of hashing). When the source and the destination of a message are relevant, we use the verbose forms $\langle\langle t : A \to B\rangle\rangle_C$ and $((t : A \to B))_C$, where A and B are respectively the purported source and destination fields. Like the the "To" and the "From" fields, they can be spoofed, whereas the subscripts C name the agent who actually performs the action. A further convenient abbreviation is $\langle\langle t\rangle\rangle_{C<}$, which means that C is the *originator* of the first message containing t.[2] In general, t may contain subterms generated by others, yet $\langle\langle t\rangle\rangle_{C<}$ asserts that no one before C had sent t itself. The expression (νm) describes the generation of a fresh nonce m. As usually in process calculus, it binds m to the right.

Atomic statements are generated over actions in one of the following forms:

- a — *"the action a has occurred"*,
- $a < b$ — *"the action a has occurred before b"*, and
- $a = b$ — *"the actions a and b are identical"*.

The conditional precedance in the form *"if b occurs, then a must have occurred before"* is often used, so we abbreviate it as

- $a \prec b \iff b \Rightarrow a < b$ —

When authenticating each other, agents reason from partial descriptions and unknown order of actions, towards total descriptions and order. The names a and b thus usually denote only partially determined actions. Thus, for instance

- $\langle t\rangle_A < (x)_Y$ — means that *some* action in the form $\langle t\rangle_A$ precedes *some* action in the form $(x)_Y$,
- $a = \langle t\rangle_A$ — means that the action denoted by a must be in the form $\langle t\rangle_A$; note that in the same session there may be $b \neq a$ with $b = \langle t\rangle_A$;
- $\langle U(t)\rangle_A = \langle V(t)\rangle_B$, where $U(t)$ and $V(t)$ are undetermined messages containing t — means that $U(t) = V(t)$ and $A = B$.

The state of each agent consists of what she has *seen* and *recorded*. In principle, she only sees her own actions. She can thus record (some of) the terms that she has sent, received, or computed, and the order of actions that she has performed. At each new state, an agent can draw new conclusions, applying the axioms, which constitute common knowledge, to the data seen or recorded. Each such derivation thus consists of three fields:

- "A sees:…" — displaying A's state,
- "A knows:…" — displaying axioms and the previously derived facts,
- "A concludes:…" — displaying the new conclusions, following from the above.

We omit "sees", "knows", and "concludes" whenever confusion seems unlikely.

There are two basic axioms that express semantics of actions. All principals are assumed to know them.

$$(t) \Longrightarrow \exists a.\ a = \langle t\rangle \wedge a < (t) \qquad \text{(rcv)}$$

$$\begin{aligned}(\nu m)_M \Longrightarrow \forall a_A.\ \big(&a = \langle\langle m\rangle\rangle \vee a = ((m))\\ &\Rightarrow (\nu m) < a \ \wedge\ A \neq M\\ &\Rightarrow (\nu m)_M < \langle\langle m\rangle\rangle_M <\\ &\quad ((m))_A \leq a_A\big)\end{aligned} \qquad \text{(new)}$$

The (rcv) axiom says that if a message is received, it must have been sent. The (new) axiom says that, if a fresh value is generated, then any action involving that fresh value must occur after its generation; moreover, if some principal other than the originator receives a message containing the fresh value, then the originator of the value must have sent a message containing it.

Axiom (cr) supports the reasoning of the initiator of a challenge-response protocol. It is formalized as follows:

$$\begin{aligned}A:\ (\nu m)_A\Big(&\langle\langle c^{AB}m\rangle\rangle_A < ((r^{AB}m))_A\\ &\Longrightarrow \langle\langle c^{AB}m\rangle\rangle_A <\\ &\quad ((c^{AB}m))_B < \langle\langle r^{AB}m\rangle\rangle_{B<} <\\ &\quad ((r^{AB}m))_A\Big)\end{aligned} \qquad \text{(cr)}$$

The expression $c^{AB}m$ denotes a challenge function applied to m, while the expression $r^{AB}m$ denotes a response function applied to m. The axiom can be viewed as a specification of the requirement defining these two functions. It tells that A can be sure that if she issues a message containing a challenge $c^{AB}m$, and receives response containing $r^{AB}m$, then B must be the originator of that response. In other words, B is the only agent who could have transformed $c^{AB}m$ to $r^{AB}m$, given the A's own observed actions.

In the various instances of axiom (cr), functions c and r satisfying the above specification, can be implemented in various ways, e.g. taking B's signature as the response, or B's public key encryption as the challenge. In each case, it will need to be proved that the particular implementation satisfies the specified requirement.

[2] Formally, $\langle\langle t\rangle\rangle_{C<}$ abbreviates $\exists c.\ c = \langle\langle t\rangle\rangle_C \wedge \forall b.\ b = \langle\langle t\rangle\rangle_B \Rightarrow b \leq c$.

The logic also contains axioms for composing, refining and transforming protocols. A transformation or refinement usually adds a new property or functionality to the protocol, in which case it comes annotated with an axiom, leading to new conclusions. In authentication protocols, such axioms may expand principal's knowledge about the events in the run of the protocol that he is participating. For example, in the basic challenge-response axiom there is no indication that B *intended* its message as a response to A's particular challenge. This would need to be supplied by some refinement introducing a specific integrity token, such as computing a MAC.

While many formal details of our logical derivation of GDOI will have to be omitted, the axioms and the derivation steps do yield to a simple diagrammatic presentation without an essential loss of precision or insight. As usually, messages are represented by horizontal arrows from one principal to another. A vertical line corresponds to principal's internal change of state. If the principal creates a new value m, this is represented by putting νm next to the appropriate vertical line. Below, we describe a derivation of a simple challenge and response protocol, which we use to form the core of GDOI.

There are several properties of protocols that will be of interest here. One, known as *matching histories*, due to Diffie, van Oorschot, and Wiener [10], says that after two principals complete a protocol successfully, then they both should have the same history of messages sent. Another, due to Lowe [16], known as *agreement*, says that the principals should agree not only about the message histories, but also about the source and destination of each message.

Assumptions. A principal can be honest, and follow the protocol, or dishonest, and perform arbitrary actions. However, it is assumed that neither an honest nor a dishonest principal can compromise the private data used to authenticate him. This means that the response function in a challenge response protocol cannot be delegated. One way to interpret this is that the identity is reduced to possession of authenticating data: whoever has the data, is recognized as a legitimate carrier of the identity.[3] More innocently, the same assumption can be construed as a simplifying convention, introduced to avoid carrying explicit conditions in proofs; but they can be added when needed.

We also tacitly assume strong typing. If a principal, for example, attempts to use data of one type in place of data of another type (e.g. a key in the place of a nonce), this will be detected and the message will be rejected. This again is a relatively strong assumption, but has been shown, at least for one formatting scheme, to lead to no loss of analytical power, under certain reasonable provisos [15]). We suspect these results could be extended to the sorts of typing schemes used by real protocols such as GDOI, and leave this as a problem for future work.

These assumptions are a matter of convenience, which will undoubtedly be modified in future work. For example, one of the properties of interest for GDOI and many other protocols is perfect forward secrecy, which describes the behavior of the protocol after a master key is compromised.

[3] For instance, no principal can pass his fingerprints or handwriting to others. In cryptographic authentication, this means that no agent can disclose the master keys, used to authenticate him: they are strictly bound to his identity. Finally, there are protocols for which such assumptions do not lead to any loss of the ability to reason about security (the "Machiavellian" protocols of Cervesato et al., [7]), but we do not necessarily limit ourselves to these.

4 Deriving Core GDOI

In this section we derive the conclusions the principals can draw as a result of participating in Core GDOI, without Proof of Possession. This is done by first constructing a mutual hash-based challenge-response protocol, and then inserting key distribution. For reasons of space, we will only give a detailed presentation of the derivation of A's conclusions as the result of participating in the Hash-based Challenge-Response, while giving a broad overview of the rest. We hope that this is enough to give a flavor of the logic.

4.1 Hash-based Challenge-Response

The derivation of GDOI begins with the basic protocol functionality, which is mutual authentication through hash-based challenge-response. It is obtained by composing and binding two copies of the challenge-response protocol described above, with the challenge and response functions instantiated

$$\begin{aligned} c^{AB}m &= m \\ r^{AB}m &= H^{BA}m \end{aligned}$$

Here, the hash H^{AB} is axiomatized by

$$H^{AB}s = H^{AB}t \implies s = t \qquad \text{(hash1)}$$

$$\langle\langle H^{AB}t\rangle\rangle_{X<} \implies X = A \vee X = B \qquad \text{(hash2)}$$

$$H^{AB}t = H^{BA}t \implies A = B \qquad \text{(hash3)}$$

The idea is that $H^{AB}m = h \circ q(\sigma^{AB}, A, B, m)$, where h is a given pseudorandom function, σ^{AB} a secret shared by A and B, and q a convenient projection, perhaps eliminating one of the identifiers. The axioms capture enough of this intended meaning, to ensure that the above instantiation of c^{AB} and r^{AB} validates axiom (cr), so that we can prove

$$\begin{aligned} A: \ (\nu m)_A \Big(& \langle\langle m\rangle\rangle_A < ((H^{AB}\overline{m}))_A \\ & \implies \langle\langle m\rangle\rangle_A < ((m))_B < \langle\langle H^{AB}\overline{m}\rangle\rangle_{B<} < \\ & ((H^{AB}\overline{m}))_A \Big) \end{aligned} \qquad \text{(crh)}$$

where $\overline{m}$ denotes a term containing[4] m. The proof makes use of (rcv) to conclude, if the message $H^{AB}\overline{m}$ was received, it must have been sent by somebody, and (hash2) to conclude that the sender must have been B. It then makes use of (new) to conclude that m must have been created and sent by A and received by B previous to B's sending the hash.

We now use (crh) to derive A's conclusions.

$$\begin{array}{ll} A \text{ sees :} & (\nu m)_A < \langle m\rangle_A < (H^{BA}m)_A \\ \quad \text{knows (crh) :} & (\nu m)_A \Big(\langle\langle m\rangle\rangle_A < ((H^{BA}m))_A \\ & \quad \Longrightarrow \langle\langle m\rangle\rangle_A < \\ & \quad ((m))_B < \langle\langle H^{BA}m\rangle\rangle_{B<} < \\ & \quad ((H^{BA}m))_A \Big) \\ \hline \quad \text{concludes :} & (\nu m)_A < \langle m\rangle_A < \\ & ((m))_B < \langle\langle H^{BA}m\rangle\rangle_{B<} < \\ & (H^{BA}m)_A \end{array}$$

In words, keyed hash can be used for ping authentication.

But furthermore, it turns out that composing and binding two copies of such hash-based authentication allows both principals to derive the exact order of all of their joint actions, and thus arrive at matching records of the conversation.

To derive this, we begin from the simple hash-based challenge response, the first diagram in fig. 1. The responder B learns little (only that someone has sent a message), but A learns that B received her challenge and responded to it. The second diagram is obtained by sequential composition of two copies of the first one. We only display the first copy. The second one is symmetric, with B as the initiator and A as responder. The conclusions that A and B can draw are the union of the conclusions that they could draw as initiator and responder of two independent protocols: A knows that it receives a response to B, and B knows that it received a response from A, but they are not able to derive much more than that. The reasoning for A is as follows:

$$\begin{array}{ll} A \text{ sees :} & (\nu m)_A < \langle m\rangle_A < (n, H^{BA}m)_A < \\ & \qquad \langle H^{AB}n\rangle_A \\ \quad \text{(crh)} & (\nu m)_A \Big(\langle\langle m\rangle\rangle_A < ((H^{BA}m))_A \\ & \quad \Longrightarrow \langle\langle m\rangle\rangle_A < \\ & \quad ((m))_B < \langle\langle H^{BA}m\rangle\rangle_{B<} < \\ & \quad ((H^{BA}m))_A \Big) \\ \quad \text{(rcv)} & (t) \Longrightarrow \exists a.\, a = \langle t\rangle \wedge a < (t) \\ \hline A \quad : & (\nu m)_A < \langle m\rangle_A < \\ & ((m))_B < \langle\langle H^{BA}m\rangle\rangle_{B<} < \\ & (n, H^{BA}m)_A < \langle H^{AB}n\rangle_A \wedge \\ & \exists Y.\, \langle\langle n : B \to A\rangle\rangle_Y < \\ & \qquad (n, H^{BA}m : B \to A)_A \end{array}$$

[4]Like before, the agent asserging containment may not be able to extract m, but must be able to verify its presence.

The third protocol is obtained by binding the two one-way authentications by a simple protocol transformation, introducing responder's challenge into his response. In the logic, this is accompanied by the protocol specific definition of B's honesty:

$$\begin{array}{lll} A : & B \text{ honest} \iff & (x)_B \prec (\nu y)_B \prec \\ & & \langle y, H^{BA}(x,y)\rangle_B \prec \\ & & (H^{AB}y)_B \end{array}$$

where A is an agent from the initiator role.

This axiom implies that, if A knows that any of the above sequence of events have occurred, then A knows that all of the preceding events must have occurred as well, in the described order — provided that B is honest, and acts according to the protocol. The reasoning now proceeds as follows.

$$\begin{array}{ll} A \text{ sees :} & (\nu m)_A < \langle m\rangle_A < \\ & (n, H(m,n))_A < \langle Hn\rangle_A \\ \text{(crh)} & (\nu m)_A \Big(\langle\langle m\rangle\rangle_A < ((H\overline{m}))_A \\ & \quad \Longrightarrow \langle\langle m\rangle\rangle_A < \\ & \quad ((m))_B < \langle\langle H\overline{m}\rangle\rangle_{B<} < \\ & \quad ((H\overline{m}))_A \Big) \\ \text{(rcv)} & (t) \Longrightarrow \exists a.\, a = \langle t\rangle \wedge a < (t) \\ A \ : & B \text{ honest} \iff (x)_B \prec (\nu y)_B \prec \\ & \qquad \langle y, H(x,y)\rangle_B \prec (Hy)_B \end{array}$$

Just as in the one-way authentications, the conclusion is

$$\begin{array}{ll} A \text{ (i) :} & (\nu m)_A < \langle m\rangle_A < \\ & ((m))_B < \langle\langle H(m,n)\rangle\rangle_{B<} < \\ & (n, H(m,n))_A \end{array}$$

On the other hand, from the honesty assumption

$$\begin{array}{ll} A \text{ (ii) :} & B \text{ honest} \wedge \langle\langle H\overline{x}\rangle\rangle_B \\ & \Longrightarrow (x)_B < (\nu y)_B < \\ & < \langle y, H(x,y)\rangle_{B<} = \langle\langle H\overline{m}\rangle\rangle_B \end{array}$$

where, as we recall, $\overline{x}$ denotes a term containing x. Instantiating $x = m$ and $\overline{x} = (m, n)$, we get the antecedens that $\langle\langle H(m,n)\rangle\rangle_B$ has occurred. The consequens now tells that the action $\langle\langle H(m,n)\rangle\rangle_{B<}$ of (i) must be $\langle y, H(m,y)\rangle_B$ of (ii), with y fresh. From axiom (hash1), A derives that $n = y$ and thus

$$\begin{array}{ll} A \ : & B \text{ honest} \Longrightarrow (\nu m)_A < \langle m\rangle_A < \\ & (m)_B < (\nu n)_B < \langle n, H^{BA}(m,n)\rangle_B < \\ & (n, H^{BA}(m,n))_A < \langle H^{AB}n\rangle_A \end{array}$$

By similar reasoning, B reaches the same conclusion, just extended by $(H^{AB}n)_B$ at the end. The hash-based challenge-response thus yields the matching conversations authentication.

In fact, with the same assumptions, A can derive essentially more: not only that B has indeed sent the response that

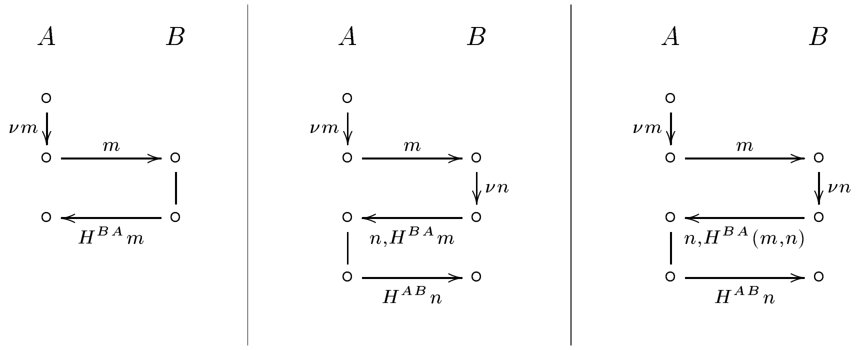

Figure 1: Hash-based Challenge-Response: from one-way to mutual authentication

she has received, and generated the challenge that she has responded to — but also that B has *intended* his response and his challenge for her. More precisely, the above conclusion of A's can be extended by the desired source and destination of each message, $A \to B$, or $B \to A$. Indeed, assuming that B is honest,

- he must have received $(m : A \to B)_B$, because he would never form $H^{BA}(m \ldots)$ otherwise, and then
- he must have generated fresh n and sent $\langle n, H^{BA}(m,n) : B \to A\rangle_B$, again because the protocol and his honesty say so.

Formalizing this, A can first prove that the above definition of B's honesty is equivalent to a stronger formula:

$$\begin{aligned} A:\quad & B \text{ honest} \iff (x : A \to B)_B \prec (\nu y)_B \prec \\ & \langle y, H^{BA}(x,y) : B \to A\rangle_B \prec \\ & (H^{AB}y : A \to B)_B \end{aligned}$$

and then strengthen the rest of her reasoning. Mutatis mutandis, the same holds for B. The principals thus agree not only about the order of their joint actions, but also about the intended sources and destinations of their messages, and about each other's identity. This stronger form of authentication is called *agreement* in Lowe's hierarchy [16].

While matching conversations authentication suffices for some purposes, we shall see in the sequel how it can lead to misidentification even in combination with agreement.

4.2 Towards GDOI: Hash-based authenticated key distribution

Towards authenticated key distribution, the hash-based mutual authentication protocol should now be composed with the hash-based key distribution protocol, identifying initiator's nonces used in the two components.

In the first diagram in fig. 2, we start with a protocol pattern in which hash-based challenge-response is used to guarantee authentication of a key (or any other piece of data). As a result of this protocol, A can conclude that B sent the key in response to her challenge. In the second diagram, we compose the key distribution with the challenge-response diagram from fig. 1. Now A can conclude that B has responded to her challenge with challenge of his own, and later with a key. B can conclude that A was still participating in the protocol at the time he received the challenge (that is, that the first message that it received from A was not a replay). The last diagram in fig. 2 is a simple refinement that authenticates the initial challenge[5]. This step is independent of the other steps, and can been introduced at any point in the derivation. In any case, it is not hard to prove that the authenticity properties, achieved in the hash-based challenge response, are preserved under the last two derivation steps. The same messages that appear in the hash-based challenge response also appear in the hash-based key distribution in the same order. Thus the same proof strategy that worked before works again.

Since the distributed key is also authenticated by hash, the resulting protocol — the core of GDOI — thus realizes the agreement authentication again. This means that each principal can exactly derive the order of all actions (except that the sender of the very last message cannot know that it is received) — *including* the correct source and the intended destination of each message.

5 Adding second authentication

In this section we describe the second way of authorizing group membership and leadership. This is done by passing a signed public key certificate with a new identity and additional authorizations. This can be done for either the group member, or the GCKS, or both. A principal is intended to prove possession of the private key corresponding to the public key contained in the certificate by using it to sign the two nonces. Thus we can think of GDOI with proof-of-possession (PoP) as the composition of two protocols: the core GDOI protocol and the PoP protocol.

[5] Omitting this would expose B to a denial-of-service.

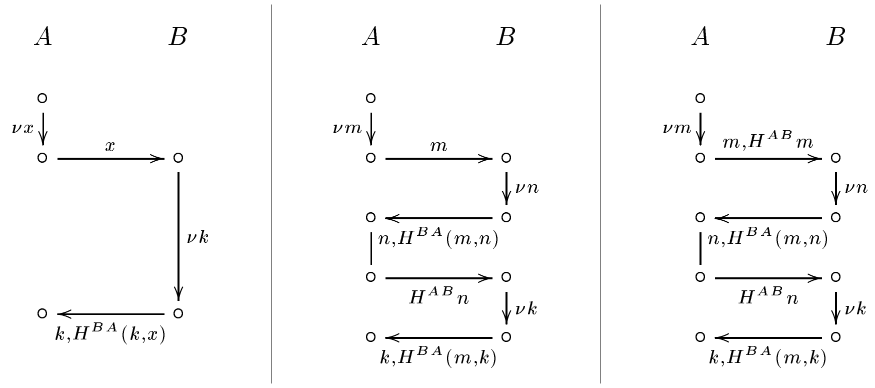

Figure 2: Composition with hash-based key distribution

5.1 Towards PoP: Signature-based Challenge-Response

Besides by keyed hashes, the abstract challenge response template from axiom (cr) can be implemented using signatures:

$$\begin{aligned} c^{AB}m &= m \\ r^{AB}m &= C^B, S^B m \end{aligned}$$

where S^B is B's signature, axiomatized by

$$S^B t = S^B u \Longrightarrow \; t = u \qquad \text{(sig1)}$$

$$\langle\langle S^B t\rangle\rangle_{X<} \Longrightarrow \; X = B \qquad \text{(sig2)}$$

$$V^B(y,t) \iff \; y = S^B t \qquad \text{(sig3)}$$

whereas C^B is B's certificate, with her identity bound to the signature verification algorithm V^B, and possibly containing additional authorization payload, used in GDOI. As usually, the integrity of this binding is assured by certifying authority.

The derivation proceeds similarly as for the hash-based authentication. The difference is that the second diagram in fig. 3 is a nested composition of two copies of the first, rather than a sequential composition, as in fig. 1. We only display the inner copy, while the outer one is symmetric, with A as the initiator, and B as responder. Like before, the third diagram is obtained by binding transformations, i.e. introducing each principal's own challenge into his response. The result is what we call the PoP protocol.

Properties and attacks

The proof of the matching conversation authenticity for the signature-based challenge response protocol closely follows the proof for the hash-based protocol, presented in sec. 4.1. However, while that proof readily extends to a proof of agreement, by extending the messages by $A \to B$ and $B \to A$, this one does not.

The reason is that the messages in the hash-based protocol carry enough information to ensure that an honest principal, say A, will send the messages to the correct destination B —whereas in the signature-based protocol they do not. More concretely, in the hash-based protocol, the assertion that A is honest and sends and receives the messages in correct order implies, as we have seen, that the sources and the destinations of the messages are also correct. In the simplest signature-based challenge response protocol, B's assumption that A is honest

$$B : \quad A \text{ honest} \iff \quad (n)_A \prec \langle\langle C^A, S^A n\rangle\rangle_A$$

cannot be extended to $\langle C^A, S^A n \,:\, A \to B\rangle_A$. While the response $H^{AB}n$ must be for B if A is honest, the response $C^A, S^A n$ does not tell who A may have intended it for, honestly or not.

The signature-based challenge response protocols thus realize matching conversations authentication, but they do not realize agreement, and do allow identity confusion. Indeed, by spoofing the source of B's challenge, and the destination of A's response, an intruder I can convince B that he has authenticated a, while A believes that she has been authenticated by I, and knows nothing of B. This is illustrated in in the first diagram in fig. 4. This attack validates e.g. the following statement

$$\begin{aligned} B : \quad & A \text{ honest} \Longrightarrow (\nu n)_B < \langle n\rangle_B < \\ & (n)_A < \langle C^A, S^A n\rangle_A < \\ & (C^A, S^A n)_B \\ \wedge\neg\Big(B : \quad & A \text{ honest} \Longrightarrow \langle C^A, S^A n \,:\, B \to A\rangle_A\Big) \end{aligned}$$

which shows that it cannot validate agreement authentication.

Now recall the derivation pattern displayed in fig. 3, used to derive a mutual authentication protocol by nested composition and binding of two copies of one way authentication. Applying this derivation pattern not to the one way authentication protocol itself, but to the attack on it — yields an attack to the resulting mutual authentication protocol. This

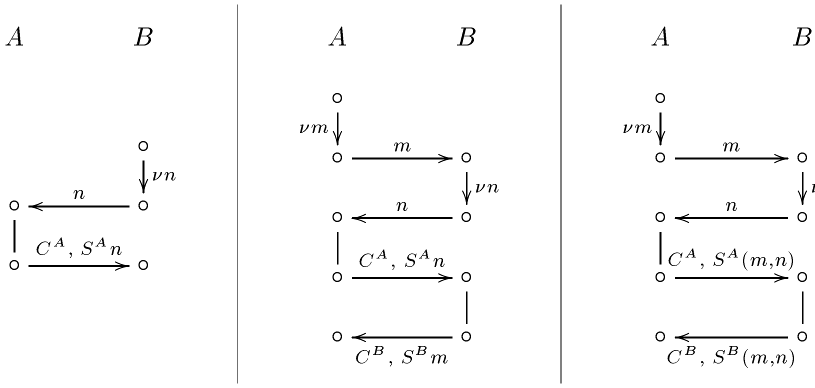

Figure 3: Signature-based Challenge-Response: from one-way to mutual authentication

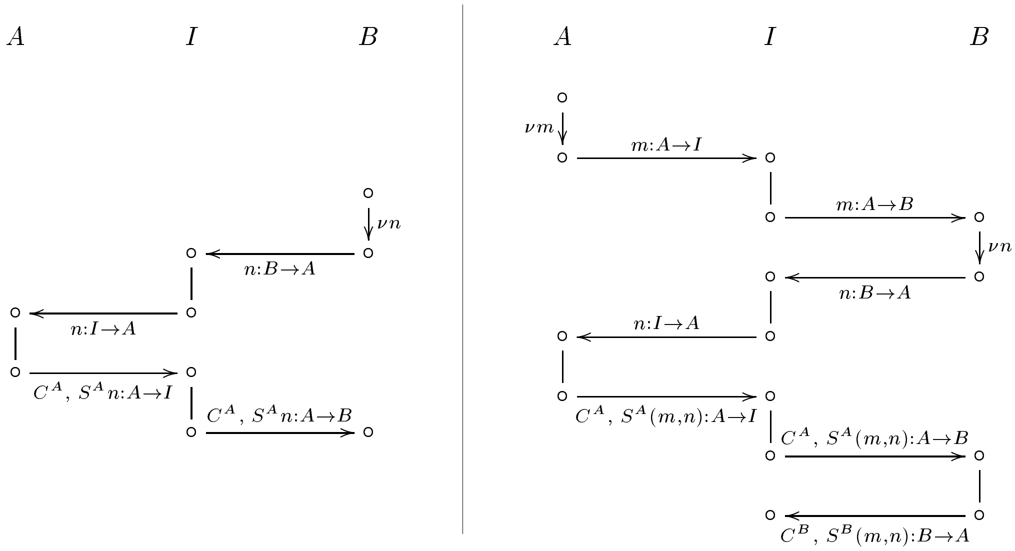

Figure 4: Attacks Against Signature-Based Challenge-Response

attack is illustrated in the second diagram of fig. 4. A formula contradicting the agreement authentication, but asserting the matching conversation can be extracted just as above.

To remove the attacks, one would need to provide grounds for the reasoning pattern that led to establishing the agreement authentication in sec. 4.1. Like there, the response function should thus be extended by peer's information, which would allow the extending honesty assertion by the source and destination fields. When anonymity of the initiator is not required, this can be achieved by taking

$$r^{AB}m \quad = \quad C^B, S^B(A, m)$$

5.2 Composing core GDOI with the PoP option

We are now ready to compose core GDOI, derived in sec. 4, with the PoP protocol, derived above. This is shown in fig. 5, where we abbreviate $\Sigma^X = S^X(m, n)$.

In the first diagram, we compose the hash-based protocol from sec. 4.2 with the signature-based protocol from sec. 5.1, identifying the fresh data. In the second diagram, we bind the two components using hashes. Note that the principal A claims both A and A' as her identities: one as the source field of her message, the other through the certificate, which she proves as hers by the signature. The principal B similarly claims both B and B'.

6 Attacks on GDOI with PoP and its defenses

When we attempted to prove the security of the composition of the core GDOI and POP, we found that we had no way of doing so. This was because, since the two protocols used two different identities, there was no way of using the propositions we could prove about GDOI and POP to prove any results about their composition. Closer examination showed that this was not the results of any deficiency in the logic; rather it was the result of an actual deficiency in the protocol, which we were able to demonstrate by finding a man-in-the-middle attack.

Subsequently, we found something even more interesting: it is possible to use the compositional logic as a means of deriving attacks on compositions of protocols. This can be done in two ways. The first method works in the case in which there is an attack on one of the component protocols. One specifies that attack as if it were a logical proposition. One then attempts to compose the attack with the propositions governing the other protocol, although in this case one attempts to discover what statements can be formed from the composition, rather than just the ones that are necessarily implied by it. If one of the statements describes an attack on the protocols, then we have pinpointed the problem. The second way can work even if there are no attacks on the component protocols. One attempts to form statements using the propositions proved about both protocols. Again, if one of the statements describes an attack, then we have pinpointed a problem arising from the emergent behavior of the composition. In this section we will show how both methods of deriving attacks can be used to derive attacks on the composition of core GDOI and POP.

6.1 Lifting the attack from PoP

We recall from fig. 4 that there is an attack on the PoP protocol that allows the responder to be confused about who the initiator is actually trying to initiate contact with. We can obtain this attack with the specification of the core GDOI protocol in sec. 4.2 — in the same way as the protocol in sec. 5.1 was composed the one in with sec. 4.2, to yield GDOI. This is illustrated in fig. 6. We compose two copies of the GDOI protocol, one between A as initiator and I as responder, and the other between I as initiator and B as responder, with a copy of the attack on the signature protocol in sec. 5.1, in which I claims to B that A' is his identity, and proves this by certificate and signature. The notation $X_{A'}$ refers to the fact that the agent X claims the identity of A' (in this case by proving possession of the certificate belonging with this identity). The upshot of this attack is that a rogue GCKS I, who does not have the credentials to join the group managed by GCKS B, could hijack the credentials belonging to A under identity A' that she presents to I when joining I's group.

A solution? The simplest way to eliminate this composition is to eliminate the attack in fig. 4, i.e. to strengthen the signature-based authentication from matching conversations to agreement. As pointed out in sec. 5.1, this can be done by introducing the peer's identity under the signature, responding to the challenge, i.e. to replace $\Sigma^{A'}$ by

$$\Sigma^{A'}_{B'} \quad = \quad S^{A'}(B', m, n)$$

We do the same for $\Sigma^{B'}$. However, as we shall see, in this case eliminating the attack on the component protocol is not enough to solve the problem.

6.2 Emergent attack

Even if the PoP protocol is modified as recommended in the previous section, so that there are no attacks on the components, an attack can still emerge in composition. To see how this happens, consider the modified version of GDOI with PoP, where $\Sigma^{A'}$ is replaced by $\Sigma^{A'}_{B'}$, and $\Sigma^{B'}$ by $\Sigma^{B'}_{A'}$. In order to allow A' to introduce the identity B' under her signature in the third message, this transformation must be enabled by moving the certificate $C^{B'}$ from the last message to the second one. The attack that nevertheless arises is presented in fig. 7.

The attack still allows a correct hash-based mutual authentication between A and I (obtained by removing certificates

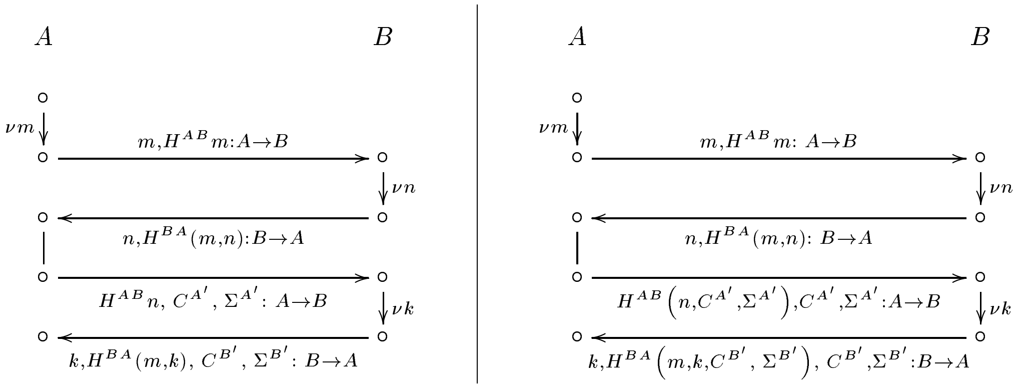

Figure 5: Composition of Core GDOI with PoP

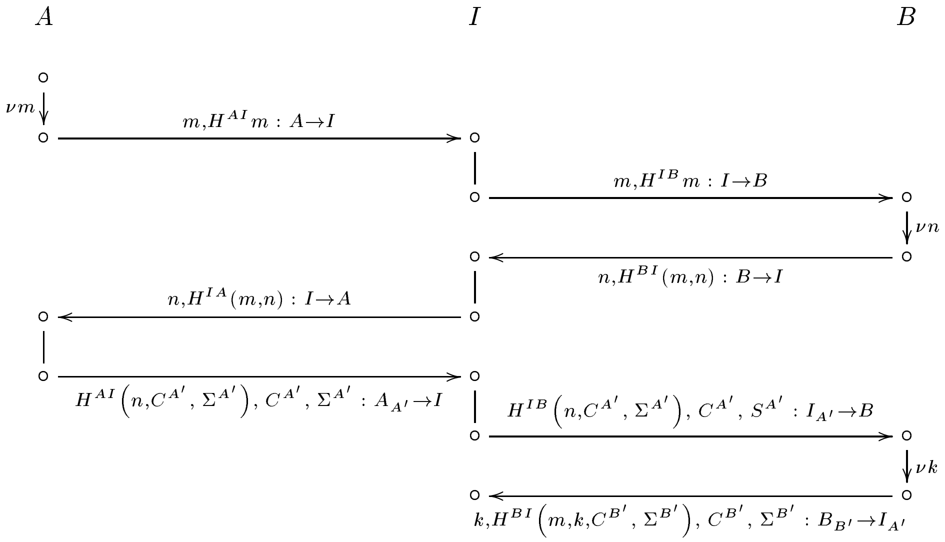

Figure 6: Lifted attack on GDOI with PoP

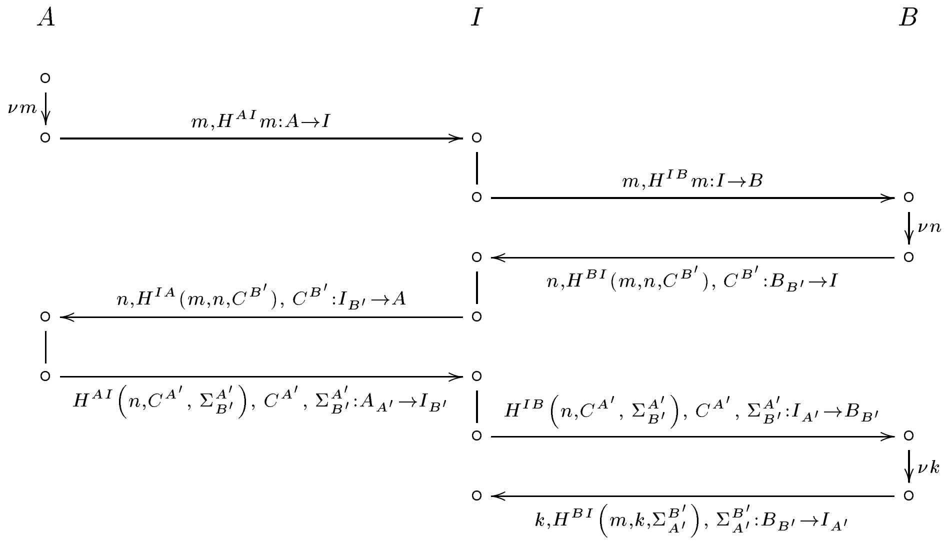

Figure 7: Emergent Attack on modified GDOI

and signatures), and a correct signature-based mutual authentication between A' and B' (obtained by removing hashes) yet putting these two authentications together leaves A believing that the identifiers I and B' belong to the same principal, and B believing that the identifiers I and A' belong to the same principal.

Suggested solution A solution to this problem that we are currently discussing with the designers of GDOI is to use in the protocol in fig. 5

$$\Sigma^{A'}_{AB} = S^{A'}(\sigma^{AB}, m, n)$$

instead of $\Sigma^{A'} = S^{A'}(m, n)$, and symmetrically $\Sigma^{B'}_{AB}$ instead of $\Sigma^{B'}$. As before, σ^{AB} is the shared secret, generated in Phase 1, and used for keying the hashes H^{AB} and H^{BA} (as explained in sec 4.1). Other identifying information, such as A's Phase 1 identity, could be used in place of σ^{AB},

The security argument, from B's point of view, boils down to proving that, if either A or A' is honest, then $A = A'$. If A' is honest, then she will only sign the σ^{AB} belonging to her, so B can conclude that $A' = A$. On the other hand, if A is honest, then, since $\Sigma^{A'}_{AB}$ appears in a hash computed with A's key σ^{AB} , B can conclude that A would only have included the signature if she herself had produced it using the private key of A'. Therefore, B can again conclude that $A = A'$. A similar proof works for A's conclusions about B and B'.

One thing that B cannot conclude is that $A = A'$ if both A and A' are dishonest, even if A and A' are not willing to share their long-term keys with each other. Indeed, we can produce a counterexample by refining our definition of digital signature. Note that in our axiomitization of digital signature, we assume message recovery is not allowed. One refinement of this assumption, and indeed a recommended one, is to take a one-way hash over data before signing it. If that is so, then A can simply pass the hash of (σ^{AB}, m, n) to A' to sign without revealing any information about its long-term key. After A' computes the signature, she can pass $\Sigma^{A'}_{AB} = S^{A'}(\sigma^{AB}, m, n)$ to A, who can then include it in her hashed message. Even if we could avoid this instance of collusion by signing without hashing, it is not considered cryptographically sound to do so, and would probably open up more risks than the one offered by collusion (which could probably be accomplished by other means). We consider the problem of preventing collusion in this case as an open one.

7 Conclusion

In this paper we described some of the features of a logic for cryptographic protocol analysis that we are developing. The logic is designed to be compositional, so that one can prove results of components separately, and then compose them. It is also designed to facilitate incremental analysis. With some modifications, the logic also has the potential of being used to find attacks on protocols. We showed how this logic was used to discover a compositional flaw in the GDOI group key exchange protocol, how it also could be used to discover attacks directly, and various ways in which the protocol could be fixed and proved correct. We are currently discussing the best way of solving this problem with the GDOI authors and the Msec working group.

One might ask, how could this problem happen in the first place, since GDOI had already undergone a formal analysis with another tool, the NRL Protocol Analyzer? Indeed, the flaw that we found is very much of the type that the NPA can find. The answer is that the fact that certificate identities could be different from Phase 1 identities was missed by the authors of [18] when they were eliciting requirements. But, even if it had been caught, it would have not been trivial to go back and reverify the protocol with the NPA. With the derivational approach, we are able to verify only the parts that had changed.

Indeed, we note also that the approach of this logic addresses a growing problem in the design of cryptographic protocols: the problem of securely composing two or more different protocols. As Asokan et al. point out in [1], deploying a new protocol is expensive, and there is considerable pressure to reuse security protocols and security context databases by composing them with other protocols. However, the problem of securely reusing these protocols by in new contexts is not well understood. Indeed, a man-in-the middle attack of the sort described in [1] was found on the IETF's Extendible Authentication Protocol [4], and has much in common with the attack we found on Proof of Possession. Thus our methodology has the potential of being of great use in the study of secure protocol reuse.

In conclusion, we believe that our approach offers the possibility of greatly facilitating the formal methods to cryptographic protocol analysis. Not only should it be possible to provide a complete verification of a protocol at one stage of its life, but to reverify the protocol as modifications are suggested and incorporated. Moreover, it facilitates the study of the growing problem of composition of protocols and protocol reuse. We believe that should increase enormously the role that formal methods can play in assisting the development of protocol standards.

8 Acknowledgements

We would like to thank Mark Baugher, Ran Canetti, Thomas Hardjono, and Brian Weis for helpful conversations about GDOI, and Iliano Cervesato for helpful suggestions about the logic and its presentation.

References

[1] N. Asokan, V. Niemi, and K. Nyberg. Man-in-the-middle in tunnelled authentication protocols. In *2003 Cambridge Security Protocol Workshop*, April 2-4 2003.

[2] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. Cryptology ePrint Archive, Report 2003/015, 2003. URL `http://eprint.iacr.org/`.

[3] M. Baugher, B. Weis, T. Hardjono, and H. Harney. The group domain of interpretation. Network Working Group, Internet Engineering Task Force. RFC 3547, July 2003.

[4] L. Blunk, J. Vollbrecht, B. Aboba, J. Carlson, and H. Levkowetz. Extensible authentication protocol (eap). EAP Working Group, Internet Engineering Task Force. RFC 2284bis, November 27 2003. `<draft-ietf-eap-rfc2284bis-07.txt>`.

[5] Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. *ACM Transactions in Computer Systems*, 8(1):18–36, February 1990.

[6] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on the Foundations of Computer Science*. IEEE Computer Society Press, 2001.

[7] I. Cervesato, S. Meadows, and P. Syverson. Dolev-Yao is no better than Machiavelli. In *First Workshop on Issues in the Theory of Security: WITS'00*, July 8-9 2000.

[8] A. Datta, A. Derek, J.C. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *IEEE Computer Security Foundations Workshop*, pages 109–125, Pacific Grove, CA, June 2003. IEEE Computer Society Press.

[9] A. Datta, A. Derek, J.C. Mitchell, and D. Pavlovic. Secure protocol composition. In *Proceedings of ACM Workshop on Formal Methods in Computer Security 2003*, pages 109–125, Washington, DC, October 2003. ACM.

[10] W. Diffie, P. C. van Oorschot, and M.l J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes, and Cryptography*, 2:107–125, 1992.

[11] N.A. Durgin, J.C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4):667–721, 2003.

[12] Nancy Durgin, John C. Mitchell, and Dusko Pavlovic. A compositional logic for protocol correctness. In Steve Schneider, editor, *Proceedings of CSFW 2001*, pages 241–255. IEEE, 2001.

[13] F. J. Thayer Fabr}´ega, J. Herzog, and J. Guttman. Strand Spaces: What Makes a Security Protocol Correct? *Journal of Computer Security*, 7:191–230, 1999.

[14] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). Network Working Group, Internet Engineering Task Force. RFC 2409, November 1998.

[15] J. Heather, S. Schneider, and G. Lowe. How to prevent type flaw attacks on security protocols. In *Proceedings*

of the 13th IEEE Computer Security Foundations Workshop. IEEE Computer Society Press, June 2000.

[16] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, pages 31–43. IEEE Computer Society Press, 1997.

[17] P. Mateus, J.C. Mitchell, and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. In *Proceedings of 14-th International Conference on Concurrency Theory 2003*, volume 2761 of *Lecture Notes in Computer Science*, pages 327–349, Marseille, September 2003. Springer-Verlag.

[18] C. Meadows, P. Syverson, and I. Cervesato. Formal specification and analysis of the Group Domain of Interpretation Protocol using NPATRL and the NRL Protocol Analyzer. *Journal of Computer Security*. To appear, currently available at `http://chacs.nrl.navy.mil/publications/CHACS/2003/2003meadows-gdoi.pdf`.

[19] D. Pavlovic and C. Meadows. Deriving authenticity and integrity as order of actions. Technical report, Kestrel Institute technical report, January 2004.